

Sahaja Pinnu and Anjana Pisupati
14 December 2020

SI 206 Final Project

<https://github.com/sahajap/Sahanjana>

Initial Goal:

Our original goal was to compare fluctuations in COVID-19 cases with the popularity of various workout channels on YouTube to see if the increase and decrease in subscribers correlated with the increase and decrease of COVID-19 cases. We planned on using a COVID-19 database to track the number of cases each month, and the YouTube API to track the change in number of subscribers per channel over time.


Achieved Goal:

We were able to gather data from the YouTube API, a COVID-19 API, and a website called SocialBlade. Using the COVID-19 API, we found the number of cases from March to October of 2020 and calculated the daily growth rate. From SocialBlade, we found the subscriber count for the most popular cooking and workout channels and calculated their growth over 2019 and 2020. With the YouTube API, we then found the top 50 videos in each category and narrowed this down to the top 5, for which we found the channel title, number of videos, total views, total likes, subscriber count, and calculated the proportion of likes to views. Using all this data, we analyzed the change in people's YouTube channel and category preferences in correlation to the number of COVID-19 cases, and ultimately found that workout channels are more successful than cooking channels as COVID-19 cases rise.

Problems Faced:

We faced several roadblocks and difficulties in our journey. We had to become satisfied with less data than preferred, and we had to change the direction of our project. We originally planned to use the Youtube API; however we found that YouTube only gives the current subscriber count and some channels block users from accessing even that information. Given that we wanted to analyze the trend of COVID-19 cases with the trend of 4 different popular channels, using only the current number of subscribers wouldn't be helpful. Therefore, we chose to use Social Blade. Social Blade has information on the growth rate of a channel's subscriber count and even the total number of subscribers at a given point of time, so we felt like we'd found a goldmine. However, the live social blade website blocked us from web scraping data from it. Therefore, we had to get crafty and found that there is a separate social blade archive website, but this was limited to a snapshot of the number of subscribers in one month in one year. These months weren't even consistent. For example, Chloe Ting's snapshot in 2019 was from April 25th to May 4th (Figure 1), whereas Fitness Blender's was June 6th to July 7th in 2019. Once we looked past the limited data and decided to use BeautifulSoup to web scrape the website, we realized that the 2019 data was different from the 2020 data. As you can see below,

the 2020 subscribers were formatted like “7.27M”, whereas the 2019 subscribers were formatted like “1,194,501”. We fixed this by removing the M from 2020 data and used `(float(item.replace(',','')))/1000000` to reformat the 2019 data. Storing only 25 items at once was also a struggle, so we added the “input” element, since we were extracting data from workout and cooking channels, and it would be inefficient to copy the same code into 2 different files. Overall, this project taught us a lot about adapting and learning how to refocus your project when necessary.

 Chloe Ting UPLOADS 236 SUBSCRIBERS 1,292,824				2020-05-13	Wed	+90K	7.27M
				2020-05-14	Thu	+90K	7.36M
				2020-05-15	Fri	+80K	7.44M
				2020-05-16	Sat	+80K	7.52M
				2020-05-17	Sun	+90K	7.6M
				2020-05-18	Mon	+100K	7.7M
				2020-05-19	Tue	+100K	7.8M
				2020-05-20	Wed	+100K	7.9M
				2020-05-21	Thu	+100K	8M
				2020-05-22	Fri	+100K	8.1M
				2020-05-23	Sat	+90K	8.19M
				2020-05-24	Sun	+90K	8.28M
				2020-05-25	Mon	+140K	8.42M
				2020-05-26	Tue	+120K	8.54M
				2020-05-27	Wed	+110K	8.65M
				2020-05-28	Thu	+100K	8.75M
				2020-05-29	Fri	+110K	8.86M
DATE		SUBSCRIBERS					
2019-04-25	Thu	+1,639	1,194,501				
2019-04-26	Fri	+1,554	1,196,055				
2019-04-27	Sat	+1,562	1,197,617				
2019-04-28	Sun	+1,741	1,199,358				
2019-04-29	Mon	+2,046	1,201,404				
2019-04-30	Tue	+2,805	1,204,209				
2019-05-01	Wed	+5,659	1,209,868				
2019-05-02	Thu	+5,679	1,215,547				
2019-05-03	Fri	+5,898	1,221,445				
2019-05-04	Sat	+5,830	1,227,275				
2019-05-05	Sun	+5,841	1,233,116				
2019-05-06	Mon	+4,025	1,237,141				
2019-05-07	Tue	+4,114	1,241,255				

Calculations File (calc.py):

```
import requests
import json
import requests
import re
import os
import csv
import unittest
import operator
import sqlite3

conn = sqlite3.connect('final.db')
cur = conn.cursor()

def covidCalculations(csvfile):
    cur.execute('SELECT * FROM covidData')
```

```

rows = cur.fetchall()
cases_list = []
date_list = []
for row in rows:
    date_list.append(row[0])
    cases_list.append(row[1])

covid_daily_growth = []
for i in range(1, len(cases_list)):
    percentincrease = (cases_list[i] + cases_list[i-1])/cases_list[i-1]
    covid_daily_growth.append(percentincrease)

with open(csvfile, mode='w') as csv_file:
    fieldnames = ['date', 'cases', 'growth_rate']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()

    for i in range(0, len(date_list)):
        date = date_list[i]
        num_cases = cases_list[i]
        try:
            growth = covid_daily_growth[i+1]
        except:
            break
        writer.writerow({'date': date, 'cases': num_cases, 'growth_rate': growth})

def socialBladeCalculations(csvfile, keyword):
    cur.execute('SELECT * FROM SOCIALBLADE WHERE user="{}"'.format(keyword))
    rows = cur.fetchall()
    sub_list = []
    date_list = []
    for row in rows:
        date_list.append(row[1])
        sub_list.append(float(row[2]))

    subs_growth = []
    for i in range(1, len(date_list)):
        percentincrease = (sub_list[i] + sub_list[i-1])/sub_list[i-1]
        if percentincrease < 3:
            subs_growth.append(percentincrease)
        else:
            subs_growth.append(2.0)

```

```

with open(csvfile, mode='w') as csv_file:
    fieldnames = ['channel_name', 'date', 'growth_rate']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()
    for i in range(0, len(date_list)):
        date = date_list[i]
        try:
            growth = subs_growth[i+1]
        except:
            break
        writer.writerow({'channel_name': keyword, 'date': date, 'growth_rate':
growth))

def socialBladeAverageCalculations(csvfile, keyword):
    cur.execute('SELECT * FROM SOCIALBLADE WHERE user="{}"'.format(keyword))
    rows = cur.fetchall()
    sub_list = []
    date_list = []
    for row in rows:
        date_list.append(row[1])
        sub_list.append(float(row[2]))

    subs_growth = []
    for i in range(1, len(date_list)):
        percentincrease = (sub_list[i] + sub_list[i-1])/sub_list[i-1]
        if percentincrease < 3:
            subs_growth.append(percentincrease)
        else:
            subs_growth.append(2.0)

    with open(csvfile, mode='w') as csv_file:
        fieldnames = ['channel_name', '2019average', '2020average']
        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
        writer.writeheader()
        total2019 = 0
        count2019 = 0
        total2020 = 0
        count2020 = 0
        for i in range(0, len(date_list)):
            year = date_list[i].split("-")[0]
            if year == "2019":

```

```

        try:
            total2019 = total2019 + subs_growth[i]
            count2019 = count2019 + 1
        except:
            continue
    if year == "2020":
        try:
            total2020 = total2020 + subs_growth[i]
            count2020 = count2020+1
        except:
            continue

    average2019 = total2019/count2019
    average2020 = total2020/count2020

    writer.writerow({'channel_name': keyword, '2019average': average2019,
'2020average':average2020})

# covidCalculations('covid.csv')
socialBladeCalculations('chloesaddictiongrowth.csv', 'chloesaddiction')
socialBladeAverageCalculations('chloesaddictionaveragegrowth.csv', 'chloesaddiction')
socialBladeCalculations('fitnessblendergrowth.csv', 'fitnessblender')
socialBladeAverageCalculations('fitnessblenderaveragegrowth.csv', 'fitnessblender')
socialBladeCalculations('bgfilmsgrowth.csv', 'bgfilms')
socialBladeAverageCalculations('bgfilmsaveragegrowth.csv', 'bgfilms')
socialBladeCalculations('bonappetitdotcomgrowth.csv', 'bonappetitdotcom')
socialBladeAverageCalculations('bonappetitdotcomaveragegrowth.csv',
'bonappetitdotcom')

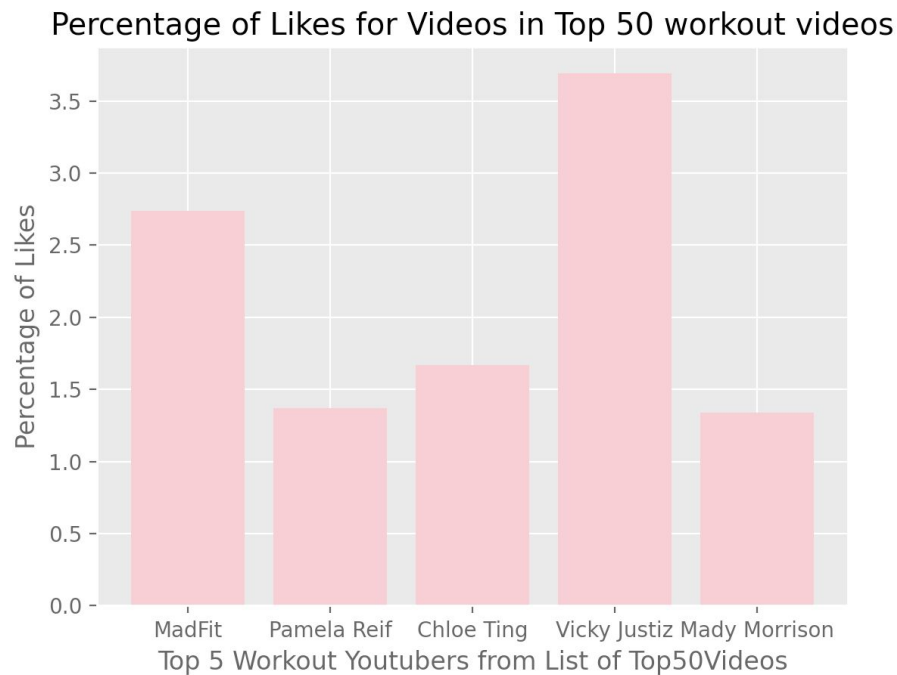
```

The calculation data is stored in the CSV files attached to the assignment. The covid.csv files contain the growth rate per day of covid cases. For example, on day 1 if there were 50 cases and day 2 there were 100, then in the csv file next to day 2 it would show 1, for a 100% increase. We also calculated the subscriber growth rate for each of the 4 channels are stored this data into chloesaddictiongrowth.csv, fitnessblendergrowth.csv, chloesaddictiongrowth.csv, bgfilmsgrowth.csv, and bonappetitdotcomgrowth.csv. Since these numbers seemed extremely similar, we ended up calculating the average of each of these growth rates for 2019 and 2020, and stored them in fitnessblenderaveragegrowth.csv, bgfilmsaveragegrowth.csv, bonappetitdotcomaveragegrowth.csv, bonappetitdotcomgrowth.csv. Finally, we wanted to see the average number of likes per videos for the top 5 channels (channels that have the most videos

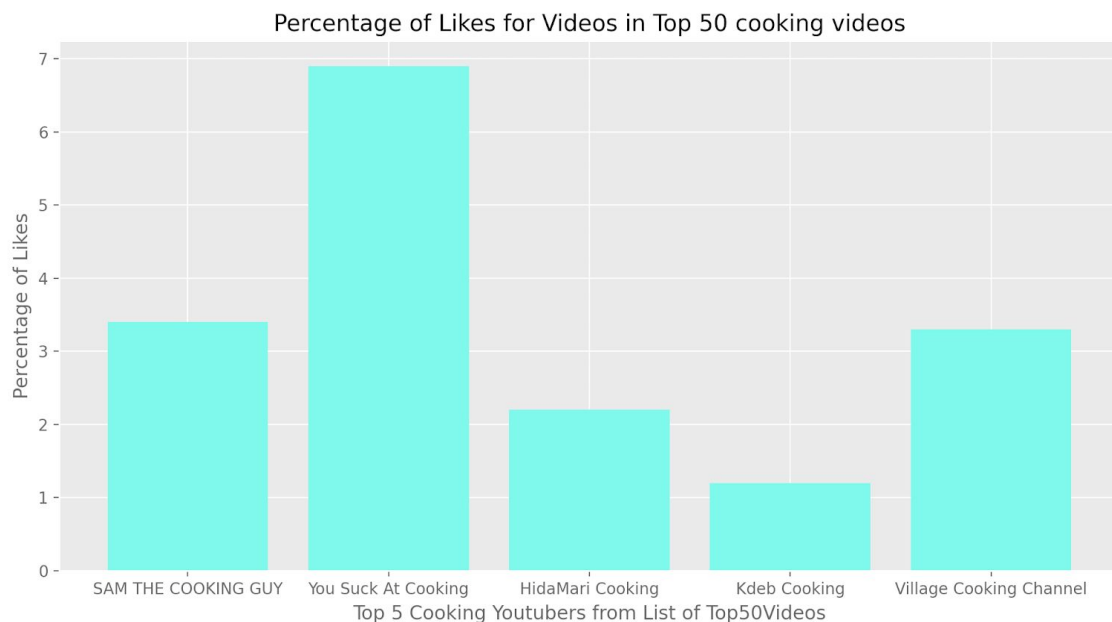
in the top 50) for both workout and cooking channels. We stored this data into top5cooking.csv and top5workout.csv.

Visualizations:

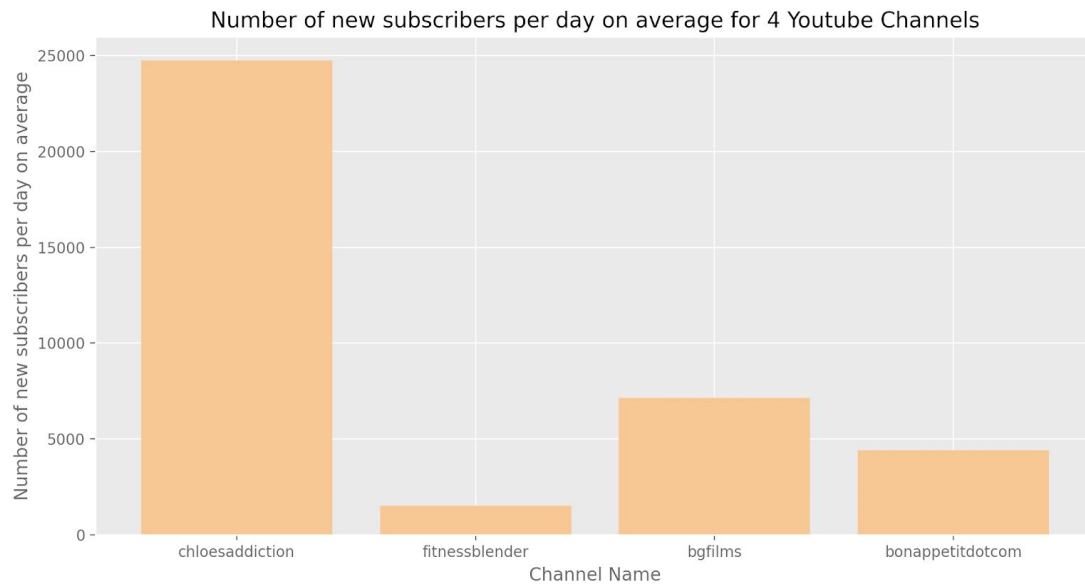
1. Using the YouTube API, we found the top 5 most popular YouTubers based on the Top 50 workout videos.



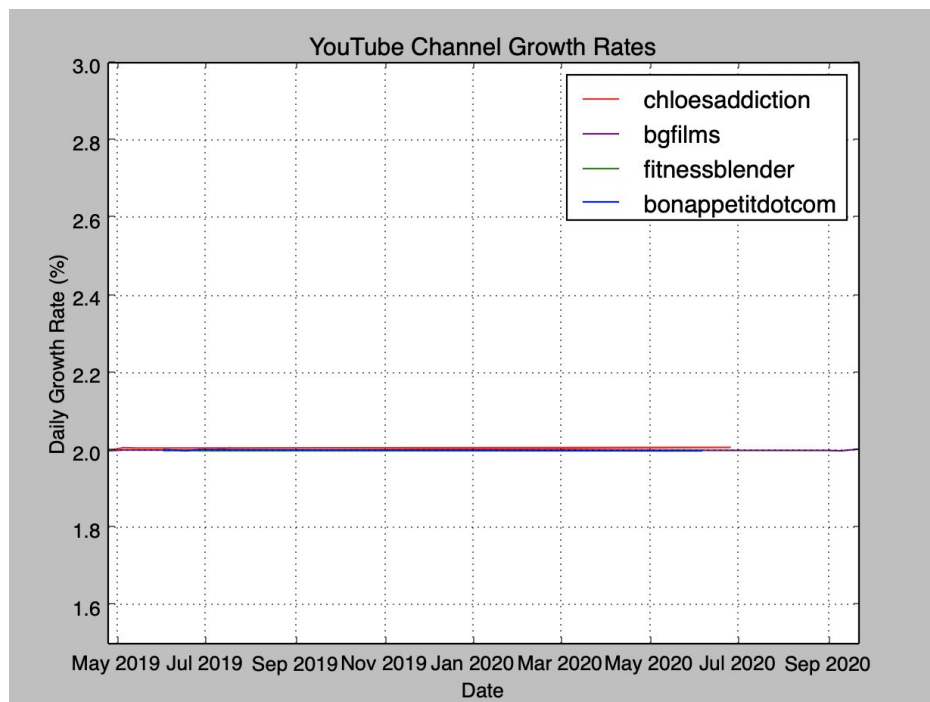
2. Using the YouTube API, we calculated the percentage of viewers who also liked the top 50 cooking videos.



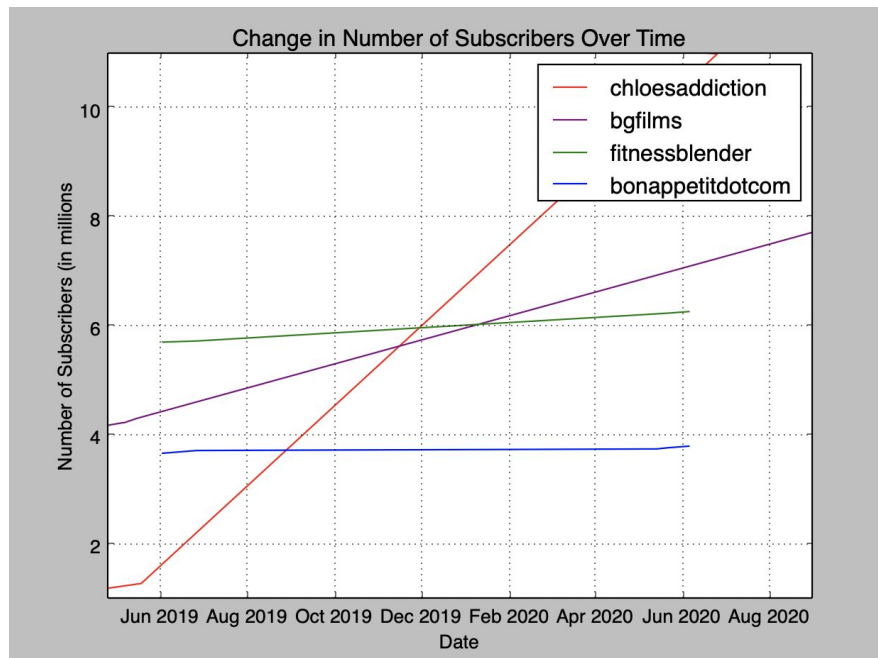
3. Using data from Social Blade, we calculated the number of new daily subscribers for the two most popular workout channels and two most popular cooking channels.



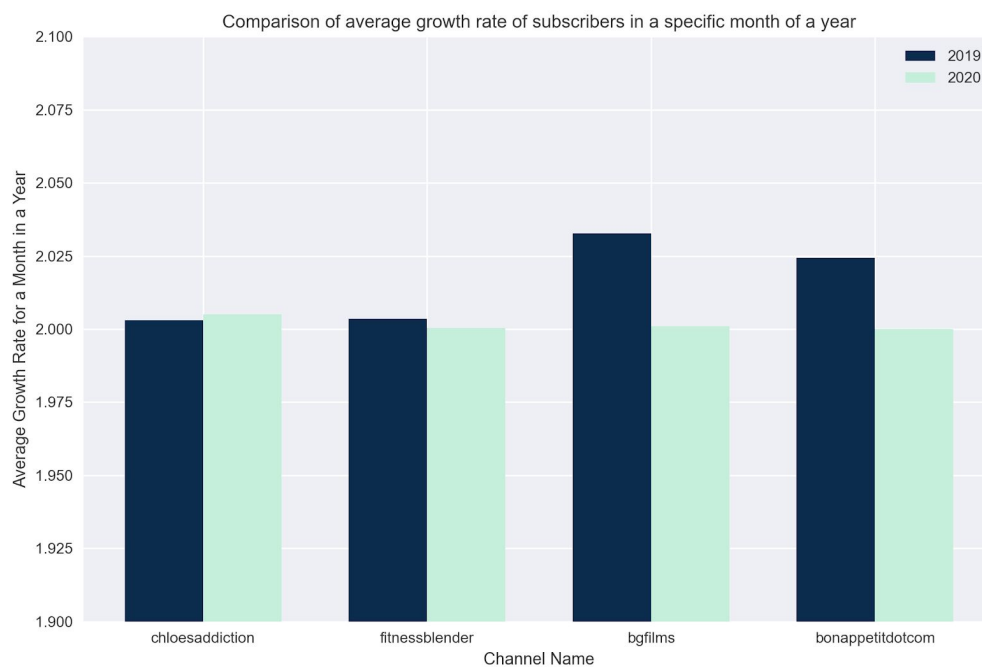
4. Using data from Social Blade, we calculated the growth rate for each of the YouTube channels. However, we found that Social Blade did not have data from certain parts of each year, including the early spring of 2020 when COVID-19 hit the United States. Therefore, the results of this graph do not tell us much.



6. We felt that a line graph showing the increase in number of subscribers might be more fitting, and found that Chloe Ting was the most successful by a wide margin.



7. As our final comparison, we used Social Blade to compare the growth rate of subscribers from 2019 to 2020.



Instructions for Running Our Code:

Due to the limitation of uploading 25 data points at a time, the process for running our code might seem slightly tedious. We apologize for this inconvenience, but we did our best given the guidelines.

*

socialblade.py

- Input 'chloesaddiction', press enter, then input 2019 then press Enter
- Input 'chloesaddiction', press enter, then input 2020 then press Enter
- Input 'fitnessblender', press enter, then 2019 then press Enter
- Input 'fitnessblender', press enter, then 2019 then press Enter
- Input 'bgfilms', press enter, then 2019 then press Enter
- Input 'bgfilms', press enter, then 2019 then press Enter
- Input 'bonappetitdotcom', press enter, then 2019 then press Enter
- Input 'bonappetitdotcom', press enter, then 2019 then press Enter

covid.py

- Run this file 10 times

topVideos.py

- Input 'workout' press enter (Do this twice)
- Input 'cooking' press enter (Do this twice)

videoData.py

- Input 'workout' press enter (Do this twice)
- Input 'cooking' press enter (Do this twice)

channelData.py

- Input 'workout' press enter (Do this twice)
- Input 'cooking' press enter (Do this twice)

calc.py

- Run this program once

Function Documentation:

covid.py

```
def collect_covid_data():
```

```
    # Utilizes COVID-19 API
```

```
    # Stores the date and number of cases from 03-10 to 11-10 (in 2020)
```

```
    # Returns a list of tuples of (date, cases)
```

```
def test_create_covid_table(covid_data):
```

```
    # Takes in a list of tuples in the format (date, cases) and creates a table in SQLite
```

```
    # doesn't return anything
```

topVideos.py

```
def topVideos(keyword):  
    # Utilizes YoutubeAPI  
    # Collects the top 50 videos for a specific channel_type(keyword) and outputs a list of tuples  
    # in format of  
    # (unique tag, channel_type, videoid, channelid, title of channel, and publishTime)  
def create_topVideos_table(keyword):  
    # Takes in the channel_type and creates a table in the database with the tuple of  
    # (unique tag, channel_type, videoid, channelid, title of channel, and publishTime)  
    # for the top 50 channels of the specific keyword
```

videoData.py

```
def video_stats(video_id, keyword):  
    # Utilizes Youtube API;  
    # Takes in a video_id and channel_type and outputs a list of tuples in format of  
    # (video_id, channel_type, total view count, total like count)  
def create_video_stats_table(keyword):  
    # Takes in the channel_type and creates a table in the database with the tuple of  
    # (video_id, channel_type, total view count, total like count)  
    # for all the videos collected in topVideos table
```

channelData.py

```
def channel_stats(channel_id, keyword):  
    # Utilizes Youtube API;  
    # Takes in a channel_id and channel_type and outputs a list of tuples in format of  
    # (channel_id, channel_type, total view count, total subscriber count, and total video count)  
def create_channel_stats_table(keyword):  
    # Takes in the channel_type and creates a table in the database with the tuple of  
    # (channel_id, channel_type, total view count, total subscriber count, and total video count)  
    # for all the channels collected in topVideos table
```

socialblade.py

```
def create_date_list(year, user):  
    # Web scrapes data from the social blade archives  
    # Takes in a year and the username for a channel
```

```

    # Outputs a list of dates for a given year and user
def create_subscriber_list(year, user):
    # Web scrapes data from the social blade archives
    # Input a year and the username for a channel
    # Outputs a list of total subscribers for a given date and name
def create_table(year, user):
    # Takes in a year and user
    # Creates a list of dates and totalSubscriber count for those dates
    # Creates a table in the SQLite database in the form of (username, date, number of subscribers)

```

calc.py

```

def covidCalculations(csvfile):
    # Takes in the name of csvfile and writes to that file
    # Calculates the percent increase of covid cases per day using the data from covidData from
the SQLite database
def socialBladeCalculations(csvfile, keyword):
    # Takes in the name of csvfile and writes to that file
    # Calculates the growth rate of subscribers from the dates provided in the SOCIALBLADE
table
    # Takes in a channelname
def socialBladeAverageCalculations(csvfile, keyword):
    # Takes in the name of csvfile and writes to that file
    # Calculates the average growth rate of subscribers in 2019 and 2020
    # Takes in a channelname
def top5(keyword):
    # Takes in a channel_type
    # Outputs a list of top 5 channels (most frequent channels) for a given channel_type
def JOINfunc(keyword, csvfile):
    # Takes in a channel type and csvfile name and writes to the csvfile
    # Join the channelData, videoData, and top50videos tables
    # Calculates total number of likes and views for a specific channel and calculates the
proportion of views to likes

```

```
# Creates a csv_file with the format below for a specific channel
# ('Channel Title', 'Number of Videos', 'Total Views', 'Total Likes', 'Proportion of Likes',
'Subscriber Count')
```

Resources:

Date	Issue Description	Source	Did it resolve?
12/03/2020	No virtual env, Youtube API	https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/	yes
12/04/2020	NoneType for youtube object (Youtube API)	https://www.youtube.com/watch?v=5qtC-tsQ-wE	yes
12/05/2020	Replacing floats in a string with a number	https://stackoverflow.com/questions/43077843/replacing-floats-in-a-string-with-a-number-python	yes
12/12/2020	ValueError: I/O operation on closed file.	https://stackoverflow.com/questions/18952716/valueerror-i-o-operation-on-closed-file	yes
12/12/2020	Storing items in dictionary	https://www.geeksforgeeks.org/python-program-to-sort-a-list-of-tuples-by-second-item/	no
12/12/2020	Sort list of tuples	https://www.pythoncentral.io/how-to-sort-a-list-tuple-or-object-with-sorted-in-python/	yes

12/12/2020	Storing items in dictionary by value	https://stackoverflow.com/questions/613183/how-do-i-sort-a-dictionary-by-value	yes
12/12/2020	Concatenating a string within quotes within a string	https://www.w3schools.com/python/ref_string_format.asp	Kind of - mix of this and guess and check
12/13/2020	How to do multi-bar graph	https://stackoverflow.com/questions/14270391/python-matplotlib-multiple-bars	Yes
12/13/2020	How to do single-bar graph	https://cmdlinetips.com/2020/03/ggplot2-2-3-0-is-here-two-new-features-you-must-know/	Yes
12/13/2020	Plotly was not importing properly	https://plotly.com/python/getting-started/	No
12/13/2020	More issues importing plotly	https://stackoverflow.com/questions/31615322/zsh-conda-pip-in-stalls-command-not-found	No
12/13/2020	Stylesheet reference for background of graphs	https://matplotlib.org/3.1.1/gallery/style_sheets/style_sheets_reference.html	Yes
12/13/20	Formatting dates in chronological order on matplotlib	https://saralgyaan.com/posts/plot-time-series-in-python-matplotlib-tutorial-chapter-8/	Yes