Members:

Jack Van Holland (jvanhol1), Soo-Yeon Kim (skim403)

Modifications:

We rewrote many of our questions from Part A due to database changes. One change was not including “trending” data, because it was difficult to find this in the past (is not in the YouTube API). In addition, we included more information about channels and comments than we initially assumed, and included more questions about these. See revised questions below.

In addition, the files and setup scripts have minor changes, fixing the delimiters so that all of the data is actually entered into the database.

1.1. What is the average ratio of number of comments to number of likes?
1.2. How many of the top 10 videos with the most comments are also the top 10 videos with the most likes?
1.3. What is the most common video category among the top 10 videos with the most subscribers?
1.4. What is the average duration of videos in each video category?
1.5. On average, do the number of likes on a video correlate to the number of comments of the video?
1.6. What is the average length of channel descriptions that are in all-caps?
1.7. What are the countries of the top 5 channels with the most views?
1.8. What is the ratio between likes and dislikes for the top 10 most viewed videos?
1.9. How many subscribers do the 10 oldest channels have compared to the 10 newest channels?
1.10. How many comments contain the word “Coronavirus”?
1.11. What are the number of comments that are replies?
1.12. What is the average number of likes of comments that are posted within the first five minutes after the video was published?
1.13. What is the title and category of the video that has the most dislikes and how many dislikes are there?
1.14. How many videos (and what percentage is it) have titles that are in all-caps and what is the average number of views on these videos?
1.15. How many comments contain a red heart emoji?

Process (no changes):

<u>Generating the videos</u>

Using Youtube API to Find 250 videos (half most viewed, half random-ish): will need to download the Google Client library and run the following python programs to generate the videos.

Top Videos: https://drive.google.com/open?id=1XPWAD_CioLL5TuyasjQdpbd95LlIUJIR

Random Videos: https://drive.google.com/open?id=1p2ma1ZBqx2sY3K9x17JUaPBWkR5KNia9

The video and channel ids generated when we ran these are listed in the following files, separated by commas: https://drive.google.com/file/d/1Tj1PxQNNt7kYy3YoVj7I7W5GZxpWnFIk

Downloading the data

We used Bernhard Rieder's YouTube data tools to download information and comments. With the comma-separated list of channels, we used the following tool to get channel info, downloading the .tab file generated. We renamed this to random125Channels.tab and top125Channels.tab.
https://tools.digitalmethods.net/netvizz/youtube/mod_channel_info.php

Then, with the comma-separated list of videos, we used the following tool to get video information (with the Manual Selection option), downloading the .tab file generated. We renamed this to random125Videos.tab and top125Videos.tab.
https://tools.digitalmethods.net/netvizz/youtube/mod_videos_list.php

Then, for each video ID, we used the following tool to download all of the comments, downloading for each, the .tab file of the comments. (each video id is inside the filename automatically)
https://tools.digitalmethods.net/netvizz/youtube/mod_video_info.php

We then renamed these all from .tab to .csv. We copied all of the file names of the video comments files into a file called files.txt.

Creating the relations

Note that YouTube comments, titles, etc. can have almost any characters but not tabs, so these text files are formatted somewhat strangely, with double tabs starting and ending tuples, and tabs between values.

Using the files.txt, we used the following Java program to generate comment.txt and reply.txt with all of the individual comment files:
https://drive.google.com/open?id=1xcvf3nVsCi3zvSFG74JLOAYQqgvK85ox

Using the above downloaded and renaming video and channel files, we created channel.txt and video.txt with the following Java program:
https://drive.google.com/open?id=1BdHTgXJd9oDoh8PZVYSc16tAFI7DOGO2

Then, we need to update the channel relation in a few ways. First, the fields were enclosed in quotation marks that were difficult to differentiate from quotation marks inside of the strings, so these were removed by renaming channel.txt to channelOld.txt and running the following java program: https://drive.google.com/open?id=1eNCrqjHgstU0mQ_APHAbtoVeQFumtybe

Also, the times were formatted as YYYY-MM-DDTHH:mm:ss.000Z, and needed to be changed to YYYY-MM-DD HH:mm:ss. Once again, we renamed channel.txt to channelOld.txt and ran the following Java program:
https://drive.google.com/open?id=19PRitACjMZpWxUzeSq-s167eqXxZkqDq

Finally, there were duplicate tuples in the channel relation, so these were removed by again renaming channel.txt to channelOld.txt and running the following Java program: https://drive.google.com/open?id=13h2L1Q6zeW7FNAgdzsnbHOp0KFk8bkJ-

Successes:

While we didn't get special characters to work in the final stage, being able to write the database to handle them took a lot of troubleshooting and was a significant accomplishment. Another success was being able to create conditional queries, where the data selected, the filters chosen, and the ordering chosen would affect the query. We had to learn how to use case statements within the elements of a query to make this possible. One final success was gathering the data for our database. We didn't use a pre-made dataset, we actually found specific videos to use and downloaded the information about these videos, channels, and comments, so our database is unique and the data cannot be found somewhere else without recollecting the data.

Known Issues:

While we figured out how to get the database to store and recognize special characters, we could not get this to work with php/html/DBvisualizer, so this only works on command line queries. So special characters are displayed as "?" in results and using special characters in "Contains" fields will not produce results.

Also, we ran in some "Internal Server Error 500" when running the Comments queries without filtering the data enough. We think this is just because the queries are too slow and are causing a timeout for the page. For instance, in testing, Comments containing "a" returned an error, while "ab" did not, indicating that if the comments are filtered enough it should not run into a problem. This even seems to change during different times of day, being able to return larger results late at night when the server is less busy. Sometimes the server seemed slow that only a single comment could be returned without an error. No similar errors were found with the small testing files, so this shows this is just a consequence of size, but that our code seems correct.

Extensions:

1. One of the questions we came up with in Phase A was "On average, how many comments contain a substring of its video's title?". We spent a lot of time trying to figure out how to work this query out but we weren't able to find a decent solution in the end. We would've liked to add this functionality to our project if we had more time.
2. A log-in functionality with query storage space for users to see their past queries would have been a nice feature to add in.
3. Ability to download query results (as excel/csv files) rather than only being able to see them on the webpage