Rousseau, Cassandra
Student ID:40177594
December 8th, 2023
CART451: Networked Media Studio

Final Project - *YT*

Project Analysis

YT changed completely from its initial design. However, the essence of the idea stayed the same. My initial idea was to display popular and niche *YouTube* videos of the past 18 years that are data-generated on a webpage. The data was supposed to be *YouTube* videos and comments from various channels from 2005 to 2023. I wanted to use pre-existing *YouTube* datasets that cover the years between 2005 (i.e., *YouTube*'s first year of existence) and 2023 to create a randomly generated list of videos. The page would generate new videos after a delay. The usual thumbnail icons would display videos instead of their usual images. Generated comments would be displayed. If the video does not interest the user, they could generate new videos by clicking the "skip" button. The skipping button would be used to create apophenia, which is the perception of patterns within random data. If it is pressed, the comments will start floating around on the webpage. Each time the user clicks on this button, more floating comments will be generated, and even generated videos will start floating around on the webpage. Comments and images will blend to create an abstract representation of the platform. The web page design was supposed to be like *YouTube*'s video layout. I wanted to use video and web development for this project because it represents *YouTube*'s essence. *Node.js, React.js*, *npm* packages like *node-ytdl-core* or *react-youtube, HTML* and *CSS* were the programming methods planned in the initial idea because I was new to data collection and its way of collecting it through programming, I wanted to use something that I was more familiar with. The goal was to loop sequences of generated *YouTube* videos on a webpage, create a skip interactive button and generate comments and videos on the page.

The idea of making an interactive webpage was completely removed from the project after the stage of the proposal. After a discussion with Sabine, the data visualization of the project changed. The discussed design was supposed to look more like Mary

Flanagan's *pile of secrets*, which is a series that displays video game footage on LCD screens. Flanagan collected video game footage from 1980 to 2011 to answer the question: "What makes a game a game?" The artist gathered common elements in video games into her database. So, instead of displaying videos on a webpage, the *YouTube* data will be displayed on multiple screens in a room. Each screen in this installation will be associated to a specific year. The videos will loop on the screens.

The prototype version of *YT* was four videos gathering videos on *YouTube* from four specific years: 2005 (i.e., year of *YouTube* creation), 2010, 2015 and 2020. Each video compiled 10 videos from their year and both the visuals and audio were layered on top of each other. I wanted to test a layering effect similar to Jason Salavon's *All the Ways (The Simpsons)*. This series gathers all the frames of every episode of *The Simpsons*' 26 seasons into a single episode. But during the video editing, I noticed that the various lengths of the videos created some gaps in the layering at the end of the dataset video. Instead, I balanced the number of layers on a time frame of approximately five minutes, so the current result of the videos is closer to the aesthetic found in another inspirational data project, *Particle*. *Particle* is a processed urban imagery that fluctuates between recognizable landscapes and abstract data-like patterns combined with dense sound textures. It deconstructs visual and audible urban landscapes. The images and sounds have been broken into fragments and then reconfigured. My prototype had a similar aesthetic. However, the final version of the project completely changed from its prototype version.

During the prototyping phase, I noticed that open-source datasets are often structured to provide the specific release date of the videos, which makes it hard to retrieve the data when looking globally at videos released within a year. Rare were the datasets that only provided the released year of videos. Regarding the data visualization, I was unsure about the outcome of the videos through the layering effect. It was very hard to notice and display all the videos by layering them. For the final version of the videos, decided to create data videos by categories. I noticed that they were multiple open-source datasets that provided categories for the videos, or even datatsets made for a specific video

category. By creating videos per category, it was easier for me to explore more datasets and add more data videos to my video collection.

Throughout the process of the final version, I faced a couple of technical and institutional limitations. I could not add more than 25 videos within my edits in the video software, so I had to stick to 25 collected videos to display for each category. I have also reached the limited amount of memory available on MongoDB free account, so I had to stop looking for new datasets and stick to those I had already found for the project. At first, I wanted to use four LCD screens to display the videos. However, I noticed that the screens offered by the *CDA* are all placed vertically (i.e., which is not the direction I want for the project), so I decided to use projectors instead. My intent was to project the four videos with their own projector on different walls, surrounding the room to create an immersive effect. But then I faced a second issue. The *CDA* does not allow students to rent more than one projector at a time. I had to quickly change the design of my project and present the videos through a video compilation combining the four categories together, creating a 22-minute-long video projected on a wall.

Regardless of the multiple changes made throughout the process, *YT* stayed a data visualization video project about *YouTube* culture. The project still explores its identity but instead of discovering how and why the platform resists and adapts to the changes and trends of the digital culture over the past 18 years, it tries to identify the visual and sound patterns of each video category existing on YouTube. Even though I completely changed the representation of the subject, I think the final version of the project results in a better apophenia effect. The grid of videos simultaneously playing with their audio provides a better way to perceive their patterns through their randomness. Viewers can identify that *YouTube* provides overstimulating content through saturated colours, cacophonous loud audio, and fast-paced sequences. It is possible to define some patterns of the platform and even the pattern differences between categories (e.g., the science category has mostly speeches audio while the arts & crafts category has mostly loud music.) I have achieved my goal of identifying some aspects of the essence of *YouTube* through this project. However, the platform contains so much content that it would require a deeper analysis to identify properly its identity.

Technologies

The data collected is displayed in a video series format. The videos were made using the software *Adobe Premiere Pro*. I decided to represent the data as a video series because it reminds the audience of the core of *YouTube*: a video-uploading platform.

For the installation aspect, I used the *AAXA M6 Pico Projector 1200 lumens*. The projector is used to project the data collection videos on the wall. I decided to use a projector because computer screens as a support limit the viewers' capacity to watch the videos. Projecting the videos on a larger scale allows one to view the data collected fully. To hear the audio, the viewers had to put a pair of headphones connected to the projector on their ears. Using headphones allow the viewers to not get distracted from the ambient sounds of the environment and immerse themselves completely into the projection.

The coding aspect of the project was mostly for data collection. I did not intend to add interactivity because I wanted to focus on the viewers having their full attention on the data by observing what is happening in front of their eyes and discovering parts of *YouTube* they were unaware of. To collect and store datasets, I have used the cloud database service *MongoDB* and its integrated service *MongoDB Atlas*, a developer data platform that helps build data. I have used *Node.js*, the *MongoDB* tool for *VS Code* that synchronizes the uploaded datasets with the code editor *VS Code* to retrieve data. All datasets used came from the open-source platform *Kaggle*. It was the platform with the most datasets related to *YouTube*, and by using open-sourced datasets, it shows the biases in the algorithms and the infinite number of videos on the platform. I used the dataset *Youtube Oldest Videos (2005) Dataset*, a dataset gathering videos released during the platform's first year, *All PewDiePie Videos,* which is a dataset gathering all the videos released by the famous YouTuber PewDiePie, *Youtube videos having more than 1 Billion views* that gather all the videos that reached over one billion views on *YouTube*, the *Top 14 Ever Most Viewed YouTube Videos*, which is a dataset gathering the 14 most-ever watched videos on YouTube, *Data of YouTube Videos* collecting videos over the year 2020, *TED Talk | Youtube* collecting videos from *TED*'s *YouTube* channel, *AI/ML*

Youtube Videos gathering educational videos about AI and ML on *YouTube*, *ChatGPT – Youtube Data* containing two datasets of videos released on the platform before and after the launch of *ChatGPT*, *Mr Beast Youtube Video Statistics* gathering MrBeast videos until 2021, *Data Science YouTube channels Video Metadata* gathering data science videos from 60 *YouTube* channels, and *YouTube Videos Dataset (~3400 videos)* which collects around 3400 videos from four different categories: travel vlogs, food, history, and art and music.

I used a GUI front-end for *youtube-dl* and *yt-dlp* named *TarTube* to download the collected videos. I decided to use this GUI because it was a good option to download good quality videos without facing any restrictions from *YouTube*.

Because the project was heavy and required a lot of memory, I bought memory and installed *Git LFS* (i.e., Large File Storage), which is a *GitHub* extension that replaces large files such as audio samples, videos, datasets and graphics with text pointers inside Git when the content is stored on GitHub.com.

Features

Because the project is a video installation, there is no interactivity involved for the viewers. They just put the headphones on their head and watch the video series. For the code, I have reused what I have built up for the first assignment of this course to apply to this project. The code was used to collect the data but not to visualize it; the datasets helped me discover the most popular videos from four categories (i.e., arts and crafts, music, science, and vlogs and entertainment) and use them to create my data visualization. Different functions were used depending on the dataset format or what I was looking for within a dataset. I have used the *.aggregate()*, *.sort()*, *.find()*, *.toArray()* and *.limit()* functions. Some datasets have been reused in more than one category.

Only a js file with the YouTube Videos Dataset (~3400 videos) dataset was created for the Arts and Crafts category. I looked for 100 videos within the category art and music. I used the aggregate function with this dataset.

For the Music category, five *js* files have been created with the *Youtube Oldest Videos (2005) Dataset, Youtube videos having more than 1 Billion views dataset, Top 14 Ever Most Viewed YouTube Videos, Data of YouTube Videos* datasets. For the *Youtube Oldest Videos (2005) Dataset* dataset, I looked for videos released in 2005 that now have views over 32,500,000. I used the find, sort, and limit functions with this dataset. I used the *Youtube videos having more than 1 Billion views dataset* dataset in four different ways, each extraction had their own *js* file. The first query I created two *js* files, one for each dataset used for that year. The first one was to find 15 videos released in 2015 that the highest number of views. I used the find, sort and limit functions. The second query was to find the four most viewed videos within the musical category. I used the find, sort and limit functions. The third query was to find the 10 most viewed released in 2010. I used the find, sort, and limit functions. The fourth query was to find the three most viewed videos released in 2020. I used the find, sort, and limit functions with this dataset. I looked for all the videos in the Top 14 Ever Most Viewed YouTube Videos dataset. I used the find and toArray functions. For the *Data of YouTube Videos*, the query was to find the ten most viewed videos over 100 000 views. I used the find, sort, and limit functions.

Five js files were created for the Science category with the *AI/ML Youtube Videos, ChatGPT – Youtube Data, Data Science YouTube channels Video Metadata* and *TED Talk | Youtube* datasets. For the *AI/ML Youtube Videos* dataset, I looked for five most viewed videos about AI and ML. I used the aggregate function with this dataset. For the *ChatGPT – Youtube Data* dataset, I have made two different *js* files for each dataset of this collection. For the dataset of videos released before the launch of ChatGPT, I have looked for the three most viewed videos. I used the aggregate function with this dataset. For the dataset of videos released after the launch of ChatGPT, I have looked for the three most viewed videos. I used the aggregate function with this dataset. For the *Data Science YouTube channels Video Metadata* dataset, I looked for seven most viewed videos within the Science and Technology category of the dataset. I used the aggregate function with this dataset. For the *TED Talk | Youtube* dataset, I looked for the seven most viewed TED Talks videos. I used the aggregate function with this dataset.

For the Vlogs and Entertainment category, six *js* files were created with the *Youtube Oldest Videos (2005) Dataset, Mr Beast Youtube Video Statistics, All PewDiePie Videos, Data of YouTube Videos, Youtube videos having more than 1 Billion views,* and *YouTube Videos Dataset (~3400 videos)* datasets. For the *Mr Beast Youtube Video Statistics* dataset, I looked for his ten most viewed videos on his channel. I used the aggregate function with this dataset. For the *All PewDiePie Videos* dataset, I looked for the 20 most viewed videos on his channel. I used the aggregate function. For the *Youtube Oldest Videos (2005) Dataset* dataset, I looked for videos released in 2005 that now have views over 32,500,000. I used the find, sort, and limit functions with this dataset. For the *Youtube videos having more than 1 Billion views dataset*, I looked for the most viewed videos within the non-musical category. I used the find and sort functions with this dataset. For the *Data of YouTube Videos*, the query was to find the ten most viewed videos over 100 000 views. I used the find, sort and limit functions. For the *YouTube Videos Dataset (~3400 videos)* dataset, I have looked for the five most viewed videos within the travel vlog category. I used the aggregate function.

After collecting the data, I downloaded the collected videos and created a data visualization video for each category through video editing. The intended results were somewhat achieved, except for the installation itself due to CDA's limited rental to one projector.

I wish I could work on finding a way to download YouTube videos by creating a downloading system instead of using *Tartube*'s GUI. It would have been possible by using some Node.js packages, but due to the lack of time (i.e., both due to personal issues that I had to deal with and other heavy final assignments for other courses), I had to give up on this idea to have the final version of my project.

Rousseau, Cassandra
Student ID:40177594

Screenshots

Figure 1 – This is a screenshot of the Arts & Crafts video introduction. Each category video has a little countdown of 8 seconds before the data collection is displayed; it is inspired by the autoplay feature on YouTube, where 8 seconds pass before starting the next video suggested by YouTube; I created a 'thumbnail' for each video by taking a screenshot of a moment during the video.

Figure 2- This is the Arts & Crafts video collection. 25 videos are playing simultaneously within each collection video. The subtitles displayed provide the video title, the YouTube channel, and the release date of each video in the collection. New video information appears in the subtitles every 12 seconds.

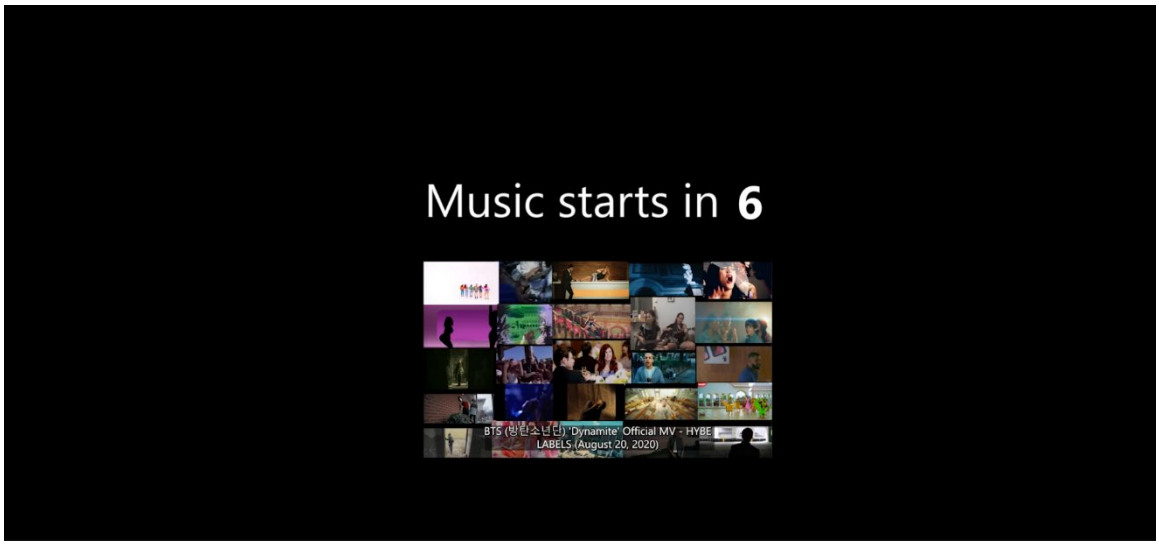

Figure 3 - This is a screenshot of the Music video introduction. The category videos are played alphabetically, one after the other in the compilation.

Figure 4 - This is the Music video collection. This video is the category that contains most clips used in the prototyping version. For the other categories, I had to retrieve new data and/or explore new datasets.

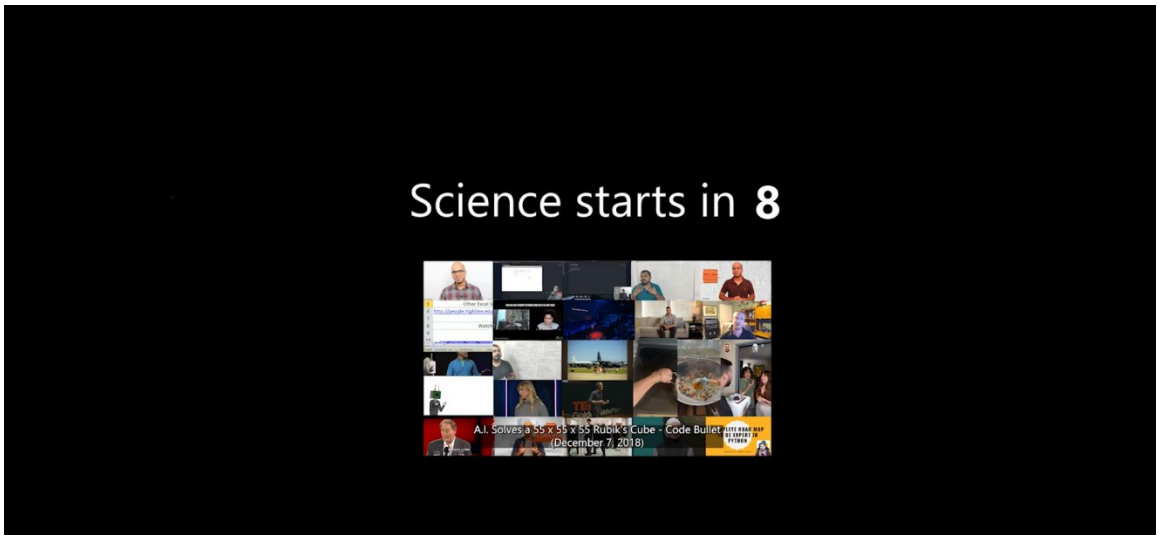

Figure 5 - This is a screenshot of the Science video introduction.

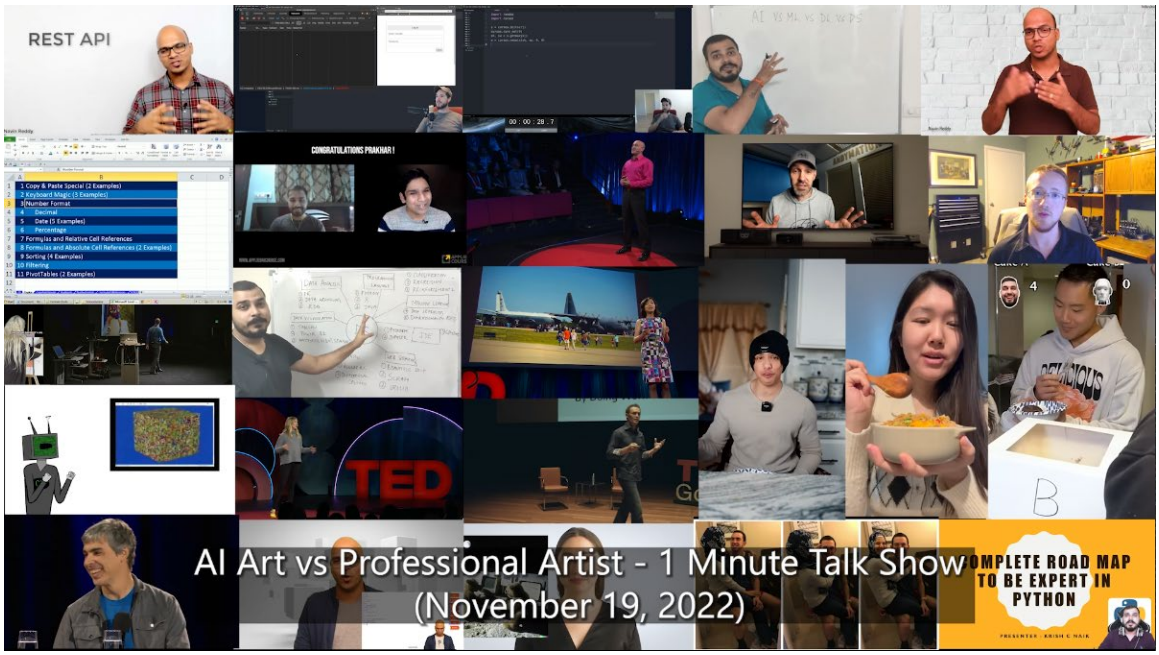

Figure 6 - This is the Science video collection.

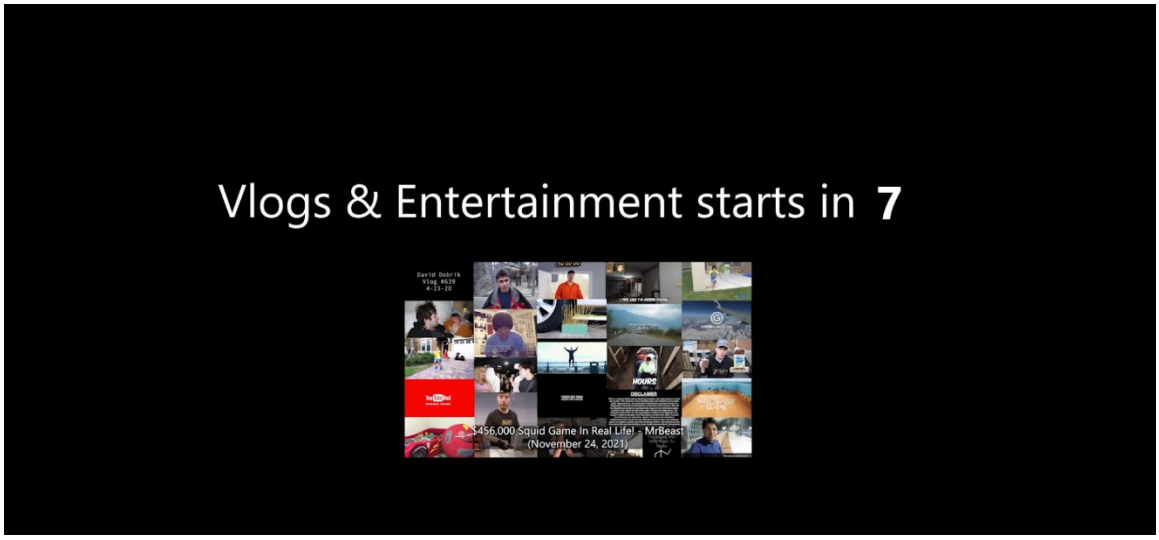

Figure 7 - This is a screenshot of the Vlogs & Entertainment video introduction.

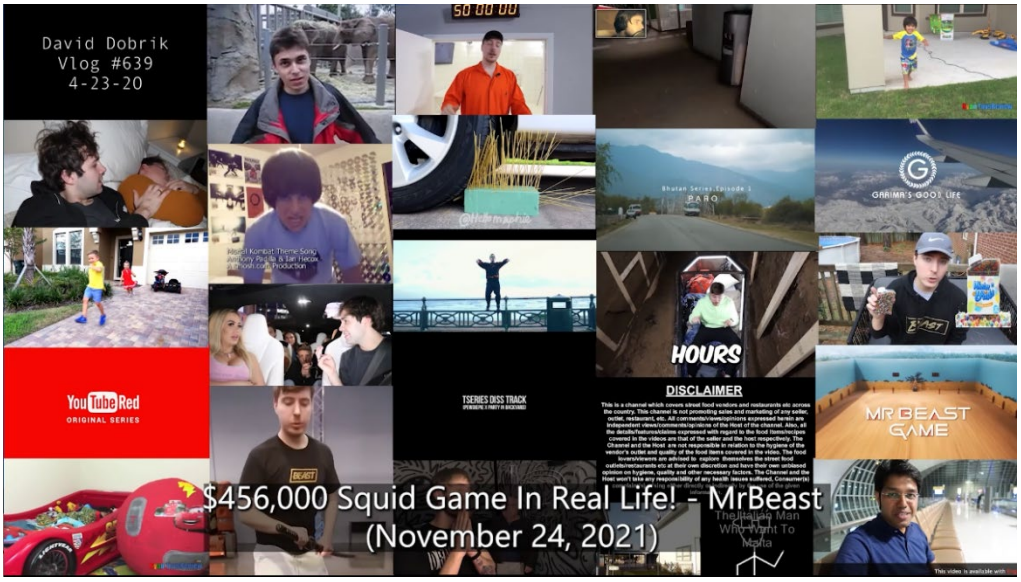

Figure 8 - This is the Vlogs & Entertainment video collection.

New Pieces of the Project

The coding process is the same as the prototype, extracting *YouTube* video data from datasets. However, because I have used new datasets for the final version of the project, I will share screenshots of the new code and extracted datasets.

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the dataset with 3400 YouTube videos
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("youtube", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      // Find 100 videos within the arts and music category
      const pipeline = [
        {$match:{category:'art_music'}},
        { $limit : 100 },
  ];

  let filteredResults = await videos.aggregate(pipeline)
  for await (const doc of filteredResults){
  console.log(doc);
  };
```

Figure 9 – Extracting data from the dataset YouTube Videos Dataset (~3400 videos)

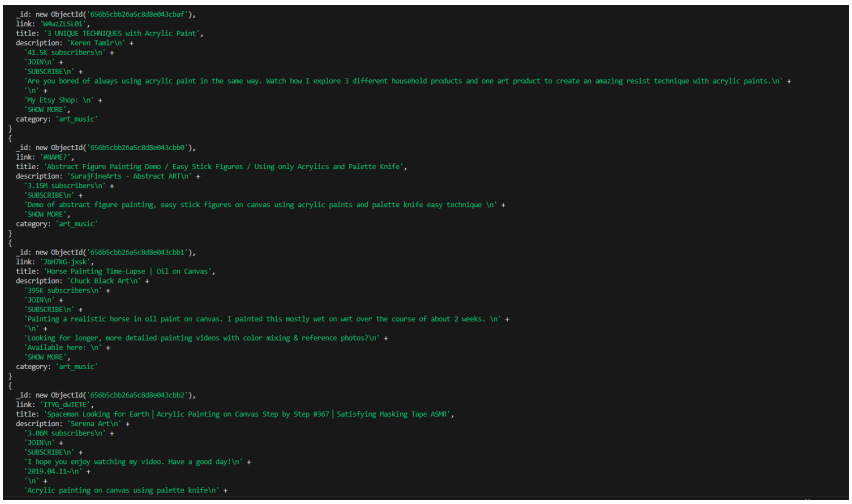

Figure 10 – Data Results from the datatset YouTube Videos Dataset (~3400 videos)

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the dataset gathering videos over a billion views
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("Youtube_videos_more_than_1Billion_views", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});

//Find the four videos that have the most views over a billion views within the muscial category
const neededDocuments = {Category:"Musical"}
let foundResults = await videos.find(neededDocuments).sort({'video views':1}).limit(4);
await foundResults.forEach((doc)=>console.log(doc));
```

Figure 11 - Extracting data from the dataset Youtube videos having more than 1 Billion views

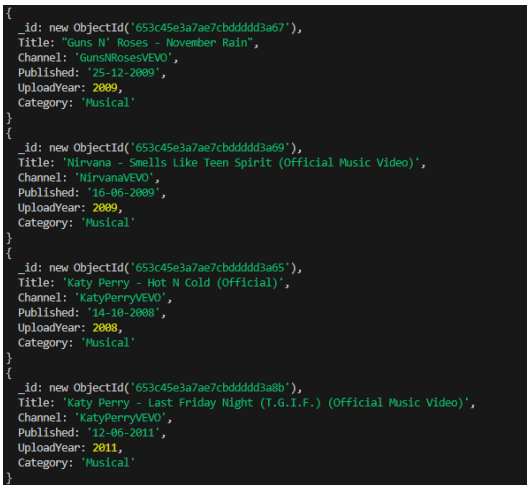

Figure 12 – Data Results from the dataset Youtube videos having more than 1 Billion views

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the ChatGPT dataset with all the videos released about AI before its launch
    const db =  client.db("CART451_Final_Project");
    // Filtering the three most viewed videos about AI before the existence of ChatGPT
    const videos =  db.collection("vidsDataBefore", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      const pipeline = [
        {$sort:{views: -1}},
        { $limit : 3 },
  ];

  let filteredResults = await videos.aggregate(pipeline)
  for await (const doc of filteredResults){
  console.log(doc);
  };
```

Figure 13 – Extracting data from the dataset ChatGPT – Youtube Data

Figure 14 – Data Results from the dataset ChatGPT – Youtube Data

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
     // Accessing the ChatGPT dataset with all the videos released about AI after its launch
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("vidsDataAfter", {
     // Filtering the three most viewed videos about AI after the lauch of ChatGPT
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      const pipeline = [
        {$sort:{views: -1}},
        { $limit : 3 },
  ];

  let filteredResults = await videos.aggregate(pipeline)
  for await (const doc of filteredResults){
  console.log(doc);
  };
```

Figure 15 – Extracting data from the dataset ChatGPT – Youtube Data


```
{
  _id: new ObjectId('656b6fe24aaf352a7f8dda9b'),
  '': 0,
  videoId: '2xrz3O0nINo',
  publishedAt: '2022-12-12T17:06:29Z',
  channelID: 'UC_3ShRJlT4PEmvFCcU6l_3Q',
  title: 'Robot Vs Human Bake Off #shorts #artificialintelligence #cake #chocolate #chatgpt',
  channelTitle: 'Patrick Zeinali',
  viewCount: 28494006,
  channelType: "['/m/02vbm', '/m/019_rr']"
}
{
  _id: new ObjectId('656b6fe24aaf352a7f8dda9c'),
  '': 1,
  videoId: '7sP2WyLln0A',
  publishedAt: '2023-01-12T21:58:19Z',
  channelID: 'UCeU_MLvxddi_yg0UttG8IJkQ',
  title: 'AI vs. Uncle Roger, who makes better egg fried rice? #shorts',
  channelTitle: 'HashSlingers',
  viewCount: 5235162,
  channelType: "['/m/019_rr', '/m/02vbm']"
}
{
  _id: new ObjectId('656b6fe24aaf352a7f8dda9d'),
  '': 2,
  videoId: 'U0bSzTSOvX0',
  publishedAt: '2023-02-05T01:05:11Z',
  channelID: 'UCPG-KbkVpjZWFAJeCW54cQw',
  title: 'Gordon Ramsay VS AI Scrambled Eggs!',
  channelTitle: 'Blatant Reviews',
  viewCount: 3193822,
  channelType: "['/m/02vbm', '/m/019_rr']"
}
```


Figure 16 – Data Results from the dataset ChatGPT – Youtube Data

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
       // Accessing the AI and ML dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("AI_ML_YT_Videos", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
     // Filtering the 5 most viewed videos about AI and ML
      const pipeline = [
        {$sort:{Views: -1}},
        { $limit : 5 },
  ];

  let filteredResults = await videos.aggregate(pipeline)
  for await (const doc of filteredResults){
  console.log(doc);
  };
```

Figure 17 – Extracting data from the dataset AI/ML Youtube Videos

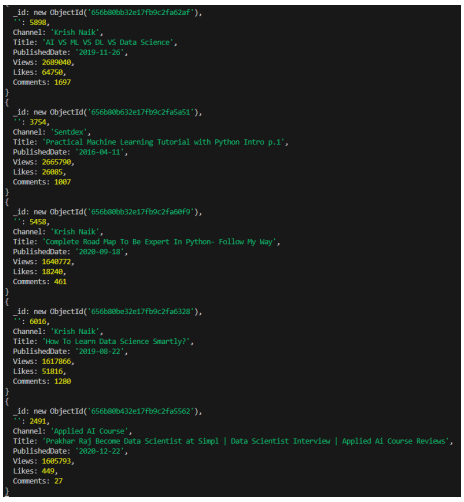

Figure 18 – Data Results from the dataset AI/ML Youtube Videos

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the data science videos dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("data-science-youtube-channel-videos-metadata", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      // Filtering the 7 most viewed videos within the science and technology category
      const pipeline = [
        {$match:{videoCategoryLabel:'Science & Technology'}},
        {$sort:{viewCount: -1}},
        { $limit : 7 },
    ];

    let filteredResults = await videos.aggregate(pipeline)
    for await (const doc of filteredResults){
    console.log(doc);
    };
```

Figure 19 – Extracting data from the dataset Data Science YouTube channels Video Metadata

Figure 20 – Data Results from the dataset Data Science YouTube channels Video Metadata

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
      // Accessing the TED Talks dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("TED", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      // Filtering the 7 most viewed TED Talks videos
      const pipeline = [
        {$sort:{views: -1}},
        { $limit : 7 },
    ];

    let filteredResults = await videos.aggregate(pipeline)
    for await (const doc of filteredResults){
    console.log(doc);
    };
```

Figure 21 – Extracting data from the dataset TED Talk | Youtube

Figure 22 – Data Results from the dataset TED Talk | Youtube

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
      // Accessing the MrBeast dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("MrBeast_youtube_stats", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      // Find the ten most viewed videos on MrBeast channel
      const pipeline = [
        {$sort:{viewCount:-1}},
        { $limit : 10 },
    ];

    let filteredResults = await videos.aggregate(pipeline)
    for await (const doc of filteredResults){
    console.log(doc);
    };
```

Figure 23 – Extracting data from the dataset Mr Beast Youtube Video Statistics

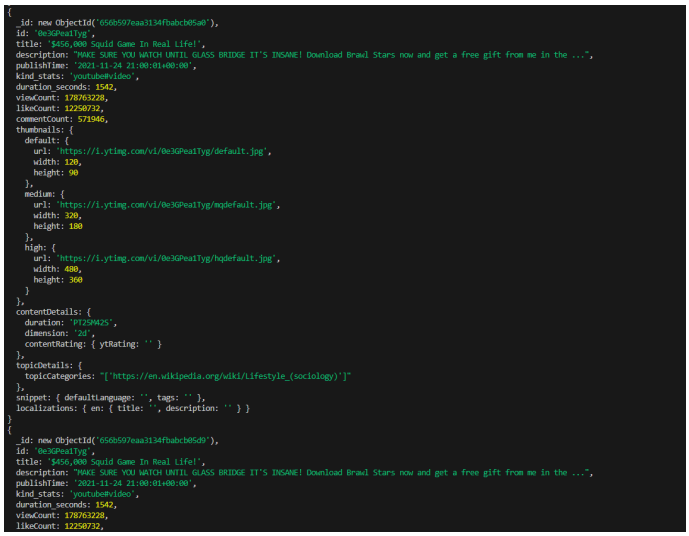

Figure 24 – Data Results from the dataset Mr Beast Youtube Video Statistics

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the PewDiePie dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("pewdiepie", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});

  // Filter the 20 most viewed videos on PewDiePie channel
  const pipeline = [
      {$sort:{viewCount:-1}},
      { $limit : 20 },
];

let filteredResults = await videos.aggregate(pipeline)
for await (const doc of filteredResults){
console.log(doc);
};
// in try
```

Figure 25 – Extracting data from the dataset All PewDiePie Videos

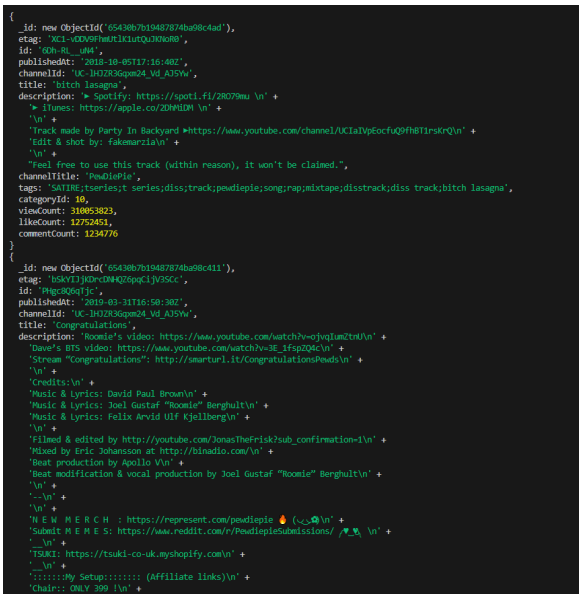

Figure 26 – Data Results from the dataset All PewDiePie Videos

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
    // Accessing the dataset gathering videos over a billion views
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("Youtube_videos_more_than_1Billion_views", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});

//Find the most viewed videos in the non-musical category
const neededDocuments = {Category:'Non-Musical'}
let foundResults = await videos.find(neededDocuments).sort({'video views':1});
await foundResults.forEach((doc)=>console.log(doc));
// in try
```

Figure 27 – Extracting data from the dataset Youtube videos having more than 1 Billion views

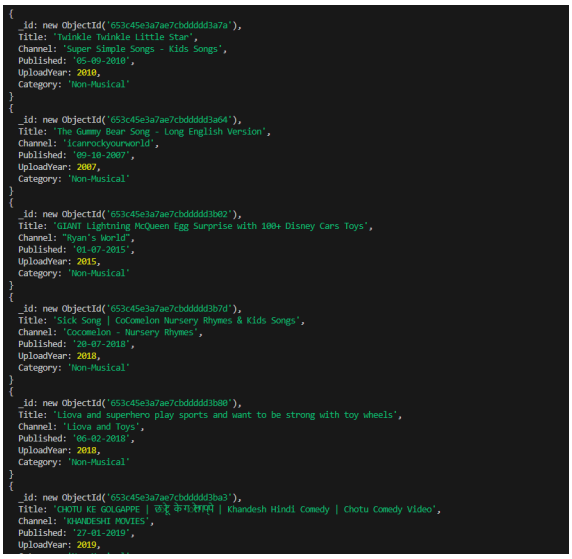

Figure 28 – Data Results from the dataset Youtube videos having more than 1 Billion views

```
async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the 3400 YouTube videos dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("youtube", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
      // Filter videos in the travel category and limit to 5 results
      const pipeline = [
        {$match:{category:'travel'}},
        { $limit : 5 },
  ];

  let filteredResults = await videos.aggregate(pipeline)
  for await (const doc of filteredResults){
  console.log(doc);
  };
```

Figure 29 – Extracting data from the dataset YouTube Videos Dataset (~3400 videos)

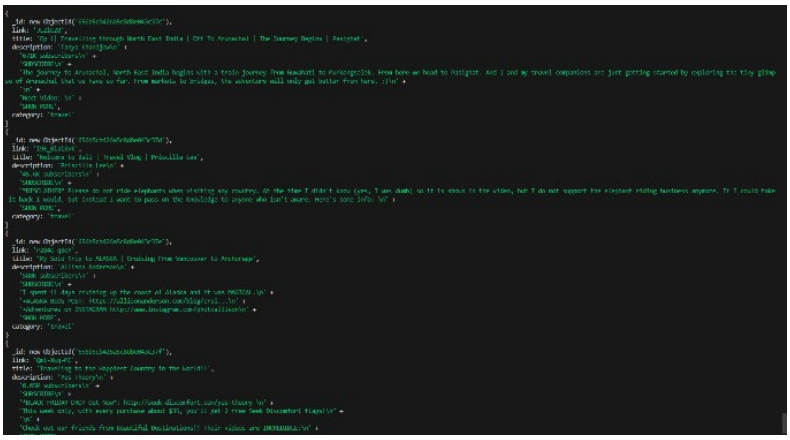

Figure 30 – Data Results from the datatset YouTube Videos Dataset (~3400 videos)

Also, the way YouTube videos are displayed in the video series is different. Here's how the video editing files look like.

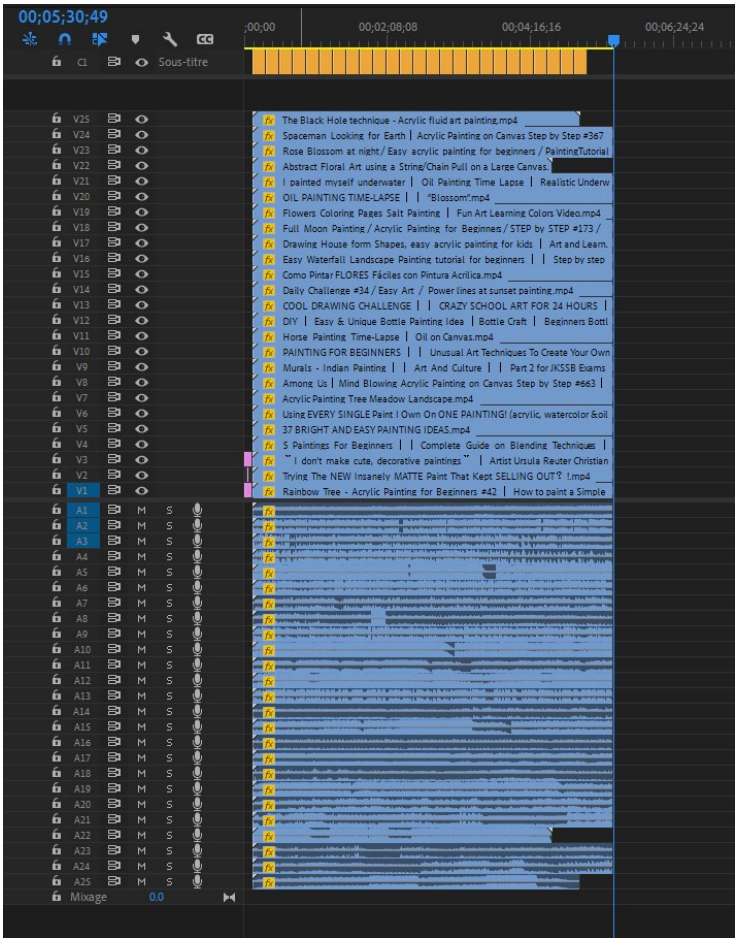

Figure 31- The video editing layers for the category Arts & Crafts

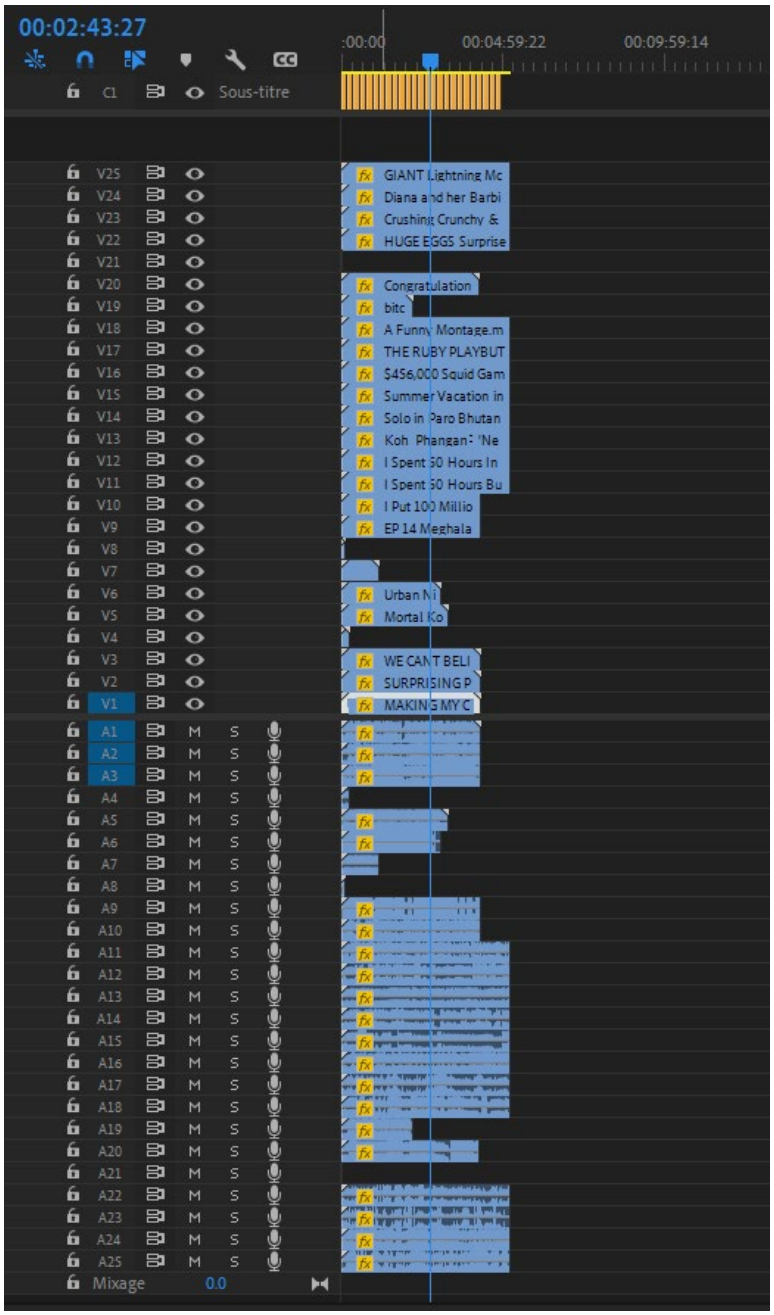

Figure 32 - The video editing layers for the category – Vlogs & Entertainment

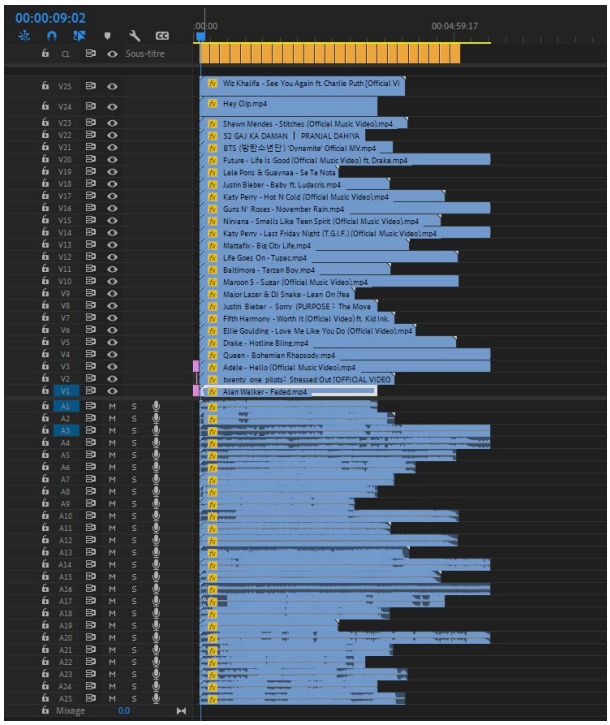

Figure 33 - The video editing layers for the category Music

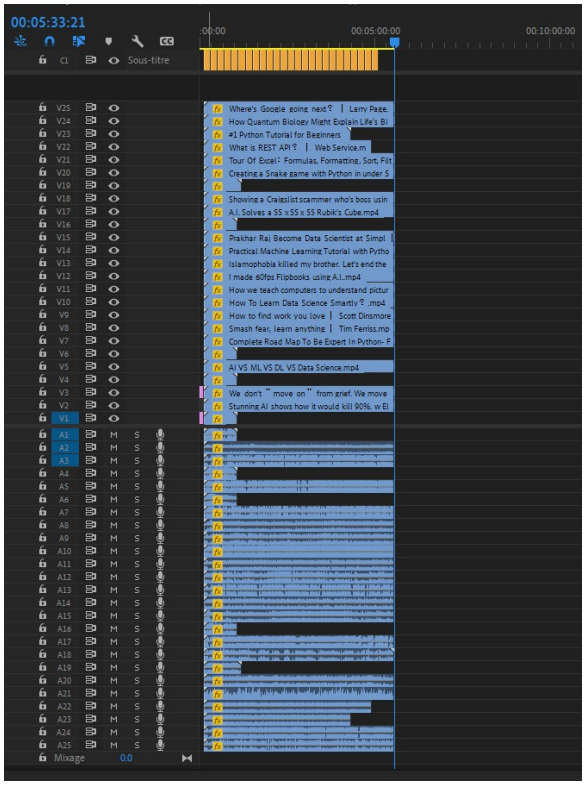

Figure 34 - The video editing layers for the category Science

Inspirations

The outcome of my final project is different from my previous artistic inspirations. However, the new version of *YT* looks similar to another project from Jason Salavon called *Little Infinity*. Salavon's project is a site-specific 72-foot-long wallpaper commissioned by the Museum of Fine Arts in Houston. The wallpaper consists of a grid of 304,656 images sampled from the entirety of the full *ImageNet* dataset. Each ¾" square tile is taken from one of the over 20k categories and arranged in a short gradient within that category. The meaning behind his project is to show the flood of imagery inundating our daily lives, questioning their nature, and showing the weird matches between some images and categories or the missing images within those categories in the *ImageNet* dataset. The data visualization of my project is very similar to Salavon's, displaying the collected *YouTube* videos within a video related to their associated category. Naturally, my video installation has also shown some anomalies within the datasets and their categories (e.g., some cooking videos were found in the results of the *ChatGPT* dataset because the recipes were made through AI). The only difference is that Salavon's project is image-based, while my project is video-based.

Live Action Recording

Because the project is already videos, I will not screen record the project. I do not think it is relevant to the format of this project. The videos for each category are found here:

https://github.com/CassandraRousseau/CART451-Final-Project/tree/main/videos/FINAL_VERSION

References & Resources

"(8) YouTube Premium - YouTube." Accessed November 4, 2023. https://www.youtube.com/premium.

"AI/ML Youtube Videos." Accessed December 6, 2023. https://www.kaggle.com/datasets/asmaahadir/aiml-youtube-channels-content-2018-2019.

"All PewDiePie Videos." Accessed November 4, 2023. https://www.kaggle.com/datasets/arusouza/all-pewdiepie-videos.

"ChatGPT - Youtube Data." Accessed December 6, 2023. https://www.kaggle.com/datasets/dekomorisanae09/chatgpt-youtube-analysis-data.

Git Large File Storage. "Git Large File Storage." Accessed December 8, 2023. https://git-lfs.com/.

"Data of YouTube Videos." Accessed November 4, 2023. https://www.kaggle.com/datasets/wchaktse/data-of-5132-youtube-videos.

"Data Science YouTube Channels Video Metadata." Accessed December 6, 2023. https://www.kaggle.com/datasets/themlphdstudent/data-science-youtube-video-meta-data.

"Jason Salavon | All the Ways (The Simpsons)." Accessed September 30, 2023. http://salavon.com/work/all-the-ways-video/.

JASON SALAVON - ALL THE WAYS Feb 25 - Apr 9, 2016 @ Mark Moore Gallery, 2016. https://www.youtube.com/watch?v=ja1c1HbQdKQ.

"Jason Salavon | Little Infinity (v. MFAH)." Accessed December 8, 2023. http://salavon.com/work/little-infinity-v-mfah/.

"Jason Salavon | Little Infinity (v.MFAH)." Accessed December 8, 2023. http://salavon.com/work/little-infinity-v-mfahmap/.

MongoDB. "MongoDB Atlas Database | Multi-Cloud Database Service." Accessed December 8, 2023. https://www.mongodb.com/atlas/database.

"MOST LIKED COMMENTS ON YOUTUBE." Accessed September 30, 2023. https://www.kaggle.com/datasets/nipunarora8/most-liked-comments-on-youtube.

"Mr Beast Youtube Video Statistics." Accessed December 6, 2023. https://www.kaggle.com/datasets/robikscube/mr-beast-youtube-video-statistics.

npm. "React-Youtube," November 22, 2022. https://www.npmjs.com/package/react-youtube.

npm. “Ytdl-Core,” July 14, 2023. https://www.npmjs.com/package/ytdl-core.

“Particle.” Accessed September 30, 2023. http://www.dfuse.com/particle.html.

“Tartube - The Easy Way To Watch And Download Videos.” Accessed November 4, 2023. https://tartube.sourceforge.io/.

“TED Talk | Youtube.” Accessed December 6, 2023. https://www.kaggle.com/datasets/ashishjangra27/ted-talk-youtube.

“Top 14 Ever Most Viewed YouTube Videos.” Accessed November 4, 2023. https://www.kaggle.com/datasets/moazzimalibhatti/top-14-ever-most-viewed-youtube-videos.

user, mf default. “[Pile of Secrets].” *Mary Flanagan* (blog), July 3, 2011. https://maryflanagan.com/pile-of-secrets/.

“Youtube Oldest Videos(2005) Dataset.” Accessed November 4, 2023. https://www.kaggle.com/datasets/demko1/youtube-oldest-videos2005-dataset.

“Youtube Videos Dataset (~3400 Videos).” Accessed September 30, 2023. https://www.kaggle.com/datasets/rajatrc1705/youtube-videos-dataset.

“Youtube Videos Having More than 1 Billion Views.” Accessed November 4, 2023. https://www.kaggle.com/datasets/jkanthony/youtube-videos-having-more-than-1-billion-views.