Rousseau, Cassandra
Student ID:40177594

Final Project Prototype

Project Description

YT is a data visualization video project about *YouTube* culture. In this project, I want to explore its identity and discover how and why the platform resists and adapts to the changes and trends of the digital culture over the past 18 years. 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 will 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 am 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.

After a discussion with Sabine, the data visualization of the project changed. The design will look more like Mary Flanagan's *pile of secrets*. *pile of secrets* 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 with a specific year. The videos will loop on the screens.

At first, I wanted to display the videos by looping them one after the other. But while pulling the data from their datasets, I thought of testing layering the videos like 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. I wanted to layer all the videos like in Salavon's project. However, while doing 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 project prototype has a similar aesthetic.

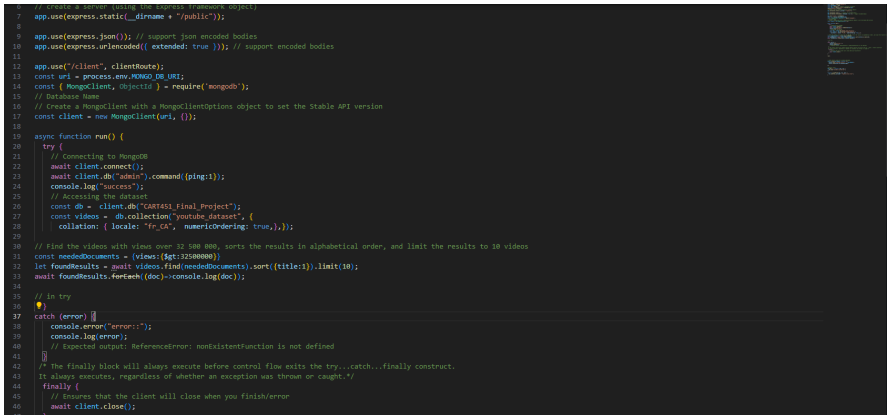
Artistic Process

My goal in this prototype was to select specific years, find datasets that give results fitting to these years, and download their associated videos. I decided to use a five-year gap, resulting in four collections. The years selected were 2005, 2010, 2015, and 2020. Each year, a total of ten videos are collected. All datasets used for this prototype come from the platform *Kaggle*. For the year 2005, I used the dataset *Youtube Oldest Videos (2005) Dataset*, which is a dataset gathering videos released during the first year of the platform. For 2010, I used the *All PewDiePie Videos* and *Youtube videos having more than 1 Billion views* datasets. *All PewDiePie Videos* is a dataset gathering all the videos released by the famous YouTuber PewDiePie. *Youtube videos having more than 1 Billion views* is a dataset gathering all the videos that reached over one billion views on *YouTube*. For 2015, I used the *Top 14 Ever Most Viewed YouTube Videos* and *Youtube videos having more than 1 Billion views* datasets. The *Top 14 Ever Most Viewed YouTube Videos* is a dataset gathering the 14 most-ever watched videos on YouTube. For 2020, I used the *Youtube videos having more than 1 Billion views, Cyberpunk 2077 YouTube Reception* and *Data of YouTube Videos*. *Cyberpunk 2077 YouTube Reception* is a dataset that collected all the reactions related to the controversial release of the video game *Cyberpunk 2077* in 2020. *Data of YouTube Videos* is a dataset collecting videos over the year 2020. For the code, I have reused what I have built up for the first assignment of this course to apply to this project. After collecting the data, I downloaded the collected videos and created a data video data visualization through video editing. I am currently satisfied with the resulting data visualization.

The steps missing in the processing are creating more yearly videos (i.e., the best would be to create all the years from 2005 to 2023), the number of screens to use to display the videos, and how to configure the space with the screens, and find the equipment for the installation.

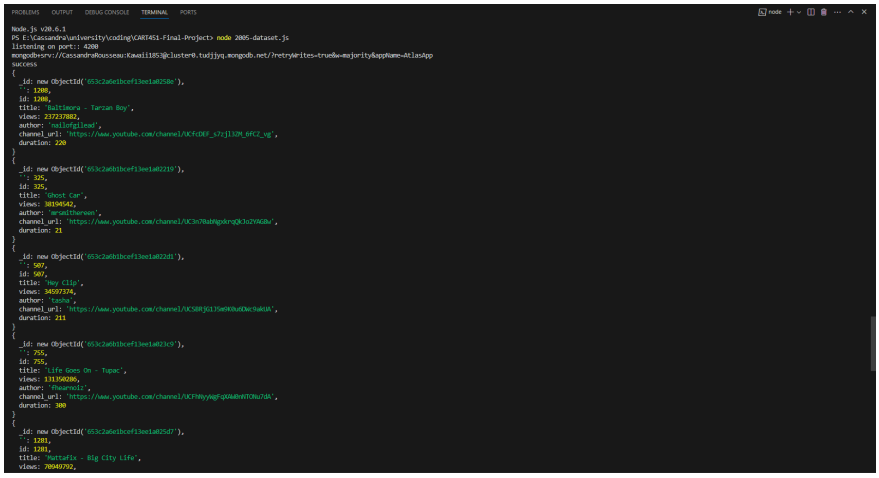
Images

2005 video creation



```
// Create a server (using the Express framework object)
app.use(express.static(__dirname + "/public"));

app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

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

// Find the videos with views over 32 500 000, sorts the results in alphabetical order, and limit the results to 10 videos
const neededDocuments = {views:{$gt:32500000}}
let foundResults = await videos.find(neededDocuments).sort({title:1}).limit(10);
await foundResults.forEach((doc)=>console.log(doc));

// in try
}
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
  /* The finally block will always execute before control flow exits the try...catch...finally construct.
  It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
```

Figure 1 - Code sorting videos released in 2005 that now have views over 32,500,000



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    node

Node.js v20.6.1
PS E:\CassandraRu\university\coding\CART451-Final-Project> node 2005-dataset.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1185$@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('653c2a6e1bcef13ee1a025Be'),
  '': 1200,
  id: 1200,
  title: 'Baltimora - Tarzan Boy',
  views: 237237882,
  author: 'mailofgileaf',
  channel_url: 'https://www.youtube.com/channel/UCFcDEF_s7zjlJ2M_6fC2_vg',
  duration: 220
}
{
  _id: new ObjectId('653c2a6b1bcef13ee1a0221b'),
  '': 325,
  id: 325,
  title: 'Ghost Car',
  views: 30194542,
  author: 'mrsmithereen',
  channel_url: 'https://www.youtube.com/channel/UC3s76abNgskrqQk3o2YAGBw',
  duration: 21
}
{
  _id: new ObjectId('653c2a6b1bcef13ee1a022d1'),
  '': 507,
  id: 507,
  title: 'Hey Clip',
  views: 34597374,
  author: 'tasha',
  channel_url: 'https://www.youtube.com/channel/UC50RjQi1Ge9KUbuEXe9aKsA',
  duration: 211
}
{
  _id: new ObjectId('653c2a6b1bcef13ee1a023c9'),
  '': 755,
  id: 755,
  title: 'Life Goes On - Tupac',
  views: 131150286,
  author: 'thearnell',
  channel_url: 'https://www.youtube.com/channel/UCFhVyyAgFqUAAbhNTONu7dA',
  duration: 380
}
{
  _id: new ObjectId('653c2a6e1bcef13ee1a025d7'),
  '': 1281,
  id: 1281,
  title: 'Nattafix - Big City Life',
  views: 70949792,
```

Figure 2 - Videos sorted from the query

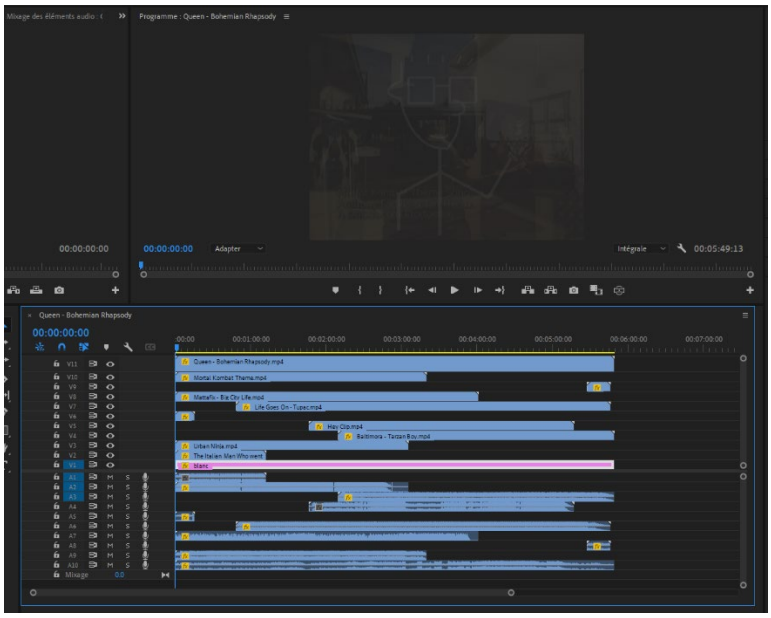


Figure 3 - Video editing for the 2005 dataset

Figure 4 - Screenshot of the dataset video

2010 video creation

```
// create a server (using the Express framework object)
app.use(express.static(__dirname + "/public"));

app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the dataset
    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 videos released in 2010 that have over a billion views,only the ten videos with the most views appear in the results
const neededDocuments = {UploadYear:2010}
let foundResults = await videos.find(neededDocuments).sort({'video views':1}).limit(10);
await foundResults.forEach((doc)=>console.log(doc));
// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
 /* The finally block will always execute before control flow exits the try...catch...finally construct.
 It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
```

Figure 5 - Code pulling data from videos released in 2010 with over one billion views

```
PS E:\Cassandra\university\coding\CART451-Final-Project> node 2010-dataset.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1853@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a7d'),
  Title: 'Katy Perry - Firework (Official Music Video)',
  Channel: 'KatyPerryVEVO',
  Published: '25-10-2010',
  UploadYear: 2010,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a7e'),
  Title: 'a-ha - Take On Me (Official Video) [Remastered in 4K]',
  Channel: 'a-ha',
  Published: '06-01-2010',
  UploadYear: 2010,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a79'),
  Title: 'Eminem - Not Afraid (Official Video)',
  Channel: 'EminemVEVO',
  Published: '05-06-2010',
  UploadYear: 2010,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a7c'),
  Title: 'Don Omar - Danza Kuduro ft. Lucenzo',
  Channel: 'DonOmarVEVO',
  Published: '15-09-2010',
  UploadYear: 2010,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a7a'),
  Title: 'Twinkle Twinkle Little Star',
  Channel: 'Super Simple Songs - Kids Songs',
  Published: '05-09-2010',
  UploadYear: 2010,
  Category: 'Non-Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3a78'),
  Title: 'Eminem - Love The Way You Lie ft. Rihanna',
  Channel: 'EminemVEVO',
  Published: '06-08-2010',
  UploadYear: 2010,
```

Figure 6 - Videos sorted from the first query

```javascript
app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

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

  // Filter the videos by order of publication, only the 10 first videos published appear in the results
  const pipeline = [
      {$sort:{publishedAt:1}},
      { $limit : 10 },
];

let filteredResults = await videos.aggregate(pipeline)
for await (const doc of filteredResults){
console.log(doc);
};
// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
 /* The finally block will always execute before control flow exits the try...catch...finally construct.
 It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}

```

Figure 7 - Code pulling PewDiePie videos released in 2010

```
PS E:\Cassandra\university\coding\CART451-Final-Project> node 2010-dataset-part2.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1853@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('65430b8a19487874ba98d278'),
  etag: 'HQOv0St8oLZuXpg9r0m16uB-G6A',
  id: '9jeJbdVl2jI',
  publishedAt: '2010-10-03T00:45:31Z',
  channelId: 'UC-lHJZR3Gqxm24_Vd_AJ5Yw',
  title: 'Minecraft Multiplayer Fun',
  description: 'My friend was going to show me something inside the building, somehow a zombie had spawn waiting for us in his minecart.',
  channelTitle: 'PewDiePie',
  tags: 'yt:quality=high;minecraft;alpha;Minecraft;Alpha;notch;pewdie;pie;wooden;house;underwater;mansion;spider;spiders;skeleton;skeletons;zombie;zombies;lumber;wood;coal;stone;torch;torches;workbench;crafting;square;pickaxe;pick;tutori
al;guide;how;to;tips;tricks;sandbox;RPG',
  categoryId: 20,
  viewCount: 21298416,
  likeCount: 906204,
  commentCount: 161726
}
{
  _id: new ObjectId('65430b8a19487874ba98d277'),
  etag: 'NERT-uH3rpjAHDBGBopnGUdzFvY',
  id: '0U6ZgImZ5vo',
  publishedAt: '2010-12-10T20:03:23Z',
  channelId: 'UC-lHJZR3Gqxm24_Vd_AJ5Yw',
  title: 'Blacklight Tango Down: Team Deathmatch 38-4 (PC Gameplay/Commentary)',
  description: 'Thank YOU for subscribing and participating! :]\n' +
    'FUCK YEAH THEY PATCHED IT!\n' +
    "Back with black ops , tomorrow... I guess D: sry for the late upload busy day -.-'",
  channelTitle: 'PewDiePie',
  tags: 'yt:quality=high;blacklight;tango;down;commentary;Black;ops;bltd;multiplayer;montage;pewpewdie;PewDiePie;Blacklight;Triple;kill;two;double;overkill;streets;quad;spray;Multiplayer;live;video;games;xbox;360;ps3;PC;gaming;weapons;ga
me;fps;consoles',
  categoryId: 20,
  viewCount: 1860528,
  likeCount: 42612,
  commentCount: 5210
}
{
  _id: new ObjectId('65430b8a19487874ba98d276'),
  etag: 'OzcKTB8mVZ_IJibpMN8TslnGDRg',
  id: 'lUtoLyyeOwQ',
  publishedAt: '2010-12-16T22:48:09Z',
  channelId: 'UC-lHJZR3Gqxm24_Vd_AJ5Yw',
  title: 'Call of Duty: Black Ops: Wager Match: Gun Game',
  description: '100 SUBS THANK YOU! :D\n' +
    "Have these running in the background with different tabs! Cause you're awesome!\n" +
    'www.rainymood.com\n' +
    'www.infiniteyoutube.com/?v=HMnrl0tmd3k',
  channelTitle: 'PewDiePie',
```

Figure 8 - Videos sorted from the second query

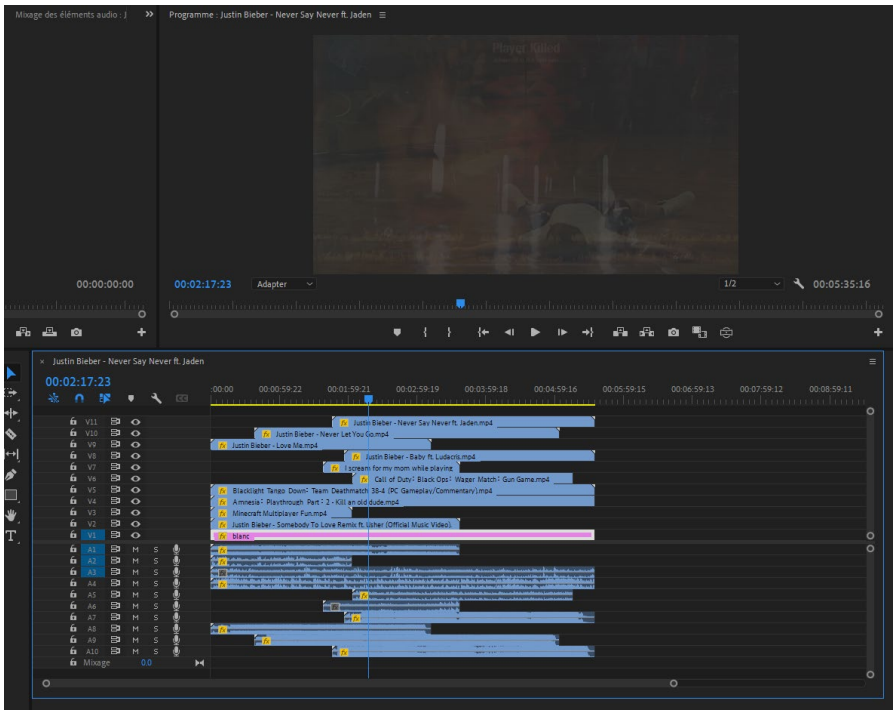


Figure 9 - Video editing for the 2010 dataset

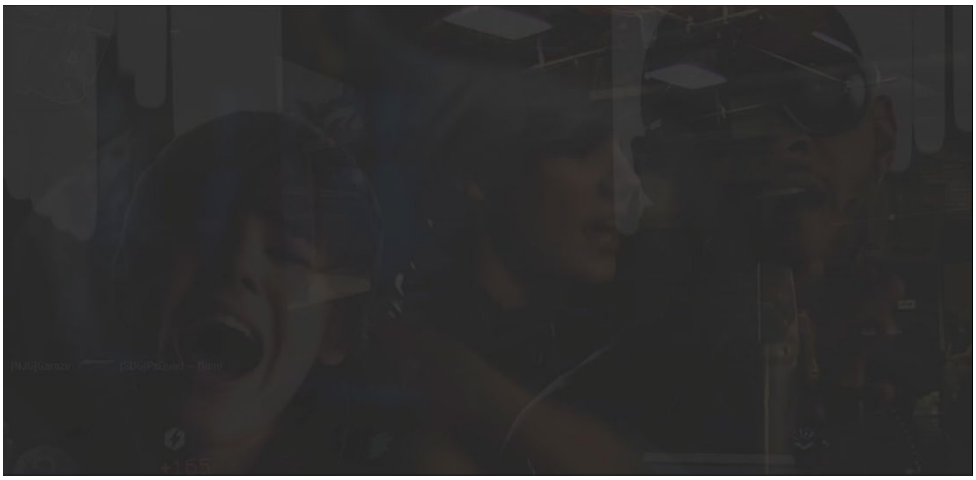


Figure 10 - Screenshot of the dataset video

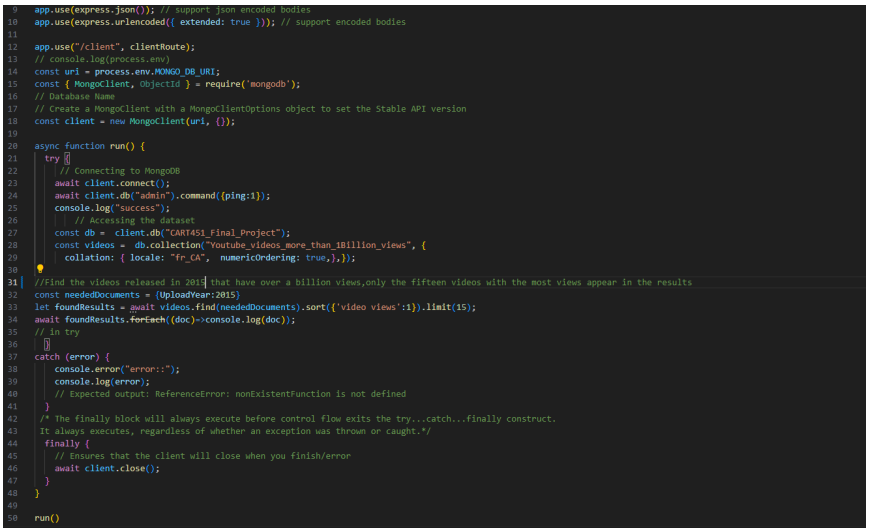
2015 video creation

```
app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
      // Accessing the dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("Top-14-Ever-Most-Viewed-YouTube-Videos", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});

  // Display all the data from this dataset in the terminal
let results = (await videos.find({}).toArray());
console.log(results);

// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
 /* The finally block will always execute before control flow exits the try...catch...finally construct.
 It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
```

Figure 11 – Code pulling the list of the most viewed videos on YouTube

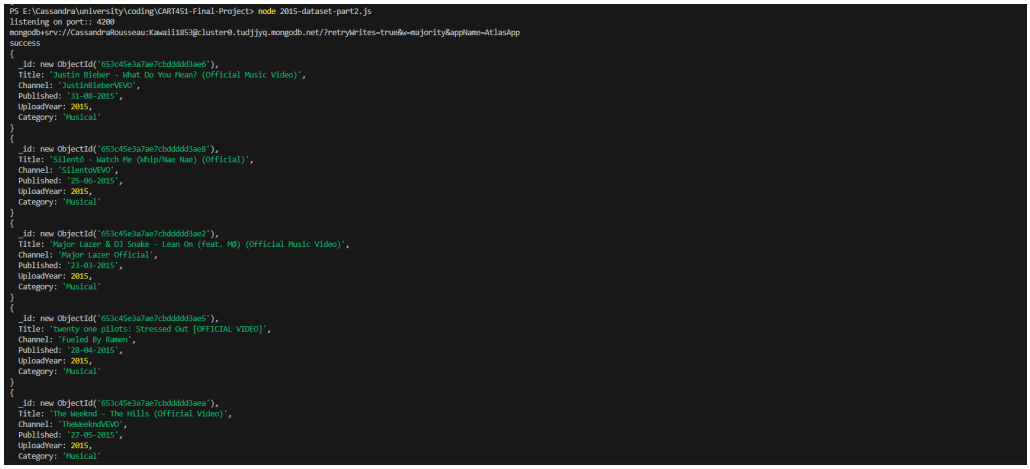
```
PS E:\Cassandra\university\coding\CART451-Final-Project> node 2015-dataset.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1053@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
[
  {
    _id: new ObjectId('653c208fee9c0f2d63e9edc5'),
    'Video Name': 'Masha and The Bear - Recipe For Disaster',
    'Channel Name': 'Masha and The Bear',
    Views: 5611185827,
    Likes: '2.1M',
    'Uploading Date': '14-Aug-15',
    Duration: '0:06:42',
    Description: 'Kids Cartoon'
  },
  {
    _id: new ObjectId('653c208fee9c0f2d63e9edc6'),
    'Video Name': 'Johny Johny Yes Papa - Great Songs for Children',
    'Channel Name': 'LooLoo Kids - Nursery Rhymes and Children's Songs',
    Views: 1020807584,
    Likes: '3.2M',
    'Uploading Date': '15-Jan-17',
    Duration: '0:55:18',
    Description: 'Kids Cartoon Song'
  },
  {
    _id: new ObjectId('653c208fee9c0f2d63e9edc7'),
    'Video Name': 'Taylor Swift - Shake It Off',
    'Channel Name': 'Taylor Swift',
    Views: 3210140604,
    Likes: '12M',
    'Uploading Date': '19-Aug-14',
    Duration: '0:04:01',
    Description: 'Song for Adults'
  },
  {
    _id: new ObjectId('653c208fee9c0f2d63e9edc8'),
    'Video Name': 'Justin Bieber - Sorry (PURPOSE : The Movement)',
    'Channel Name': 'Justin Bieber',
    Views: 3587835774,
    Likes: '15M',
    'Uploading Date': '23-Oct-15',
    Duration: '0:03:25',
    Description: 'Song for Adults'
  },
```

Figure 12 - Videos sorted from the first query



```
app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the dataset
    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 videos released in 2015 that have over a billion views,only the fifteen videos with the most views appear in the results
const neededDocuments = {UploadYear:2015}
let foundResults = await videos.find(neededDocuments).sort({'video views':1}).limit(15);
await foundResults.forEach((doc)=>console.log(doc));
// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
 /* The finally block will always execute before control flow exits the try...catch...finally construct.
 It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}

run()
```

Figure 13 - Code pulling videos released in 2015 with over a billion views



```
PS E:\Cassandra\university\coding\CART451-Final-Project> node 2015-dataset-part2.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1853@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3ae6'),
  Title: 'Justin Bieber - What Do You Mean? (Official Music Video)',
  Channel: 'JustinBieberVEVO',
  Published: '31-08-2015',
  UploadYear: 2015,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3ae8'),
  Title: 'Silento - Watch Me (Whip/Nae Nae) (Official)',
  Channel: 'SilentoVEVO',
  Published: '25-06-2015',
  UploadYear: 2015,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3ae2'),
  Title: 'Major Lazer & DJ Snake - Lean On (feat. MØ) (Official Music Video)',
  Channel: 'Major Lazer Official',
  Published: '23-03-2015',
  UploadYear: 2015,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3ae5'),
  Title: 'twenty one pilots: Stressed Out [OFFICIAL VIDEO]',
  Channel: 'Fueled By Ramen',
  Published: '28-04-2015',
  UploadYear: 2015,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3aea'),
  Title: 'The Weeknd - The Hills (Official Video)',
  Channel: 'TheWeekndVEVO',
  Published: '27-05-2015',
  UploadYear: 2015,
  Category: 'Musical'
```

Figure 14 - Videos sorted from the second query

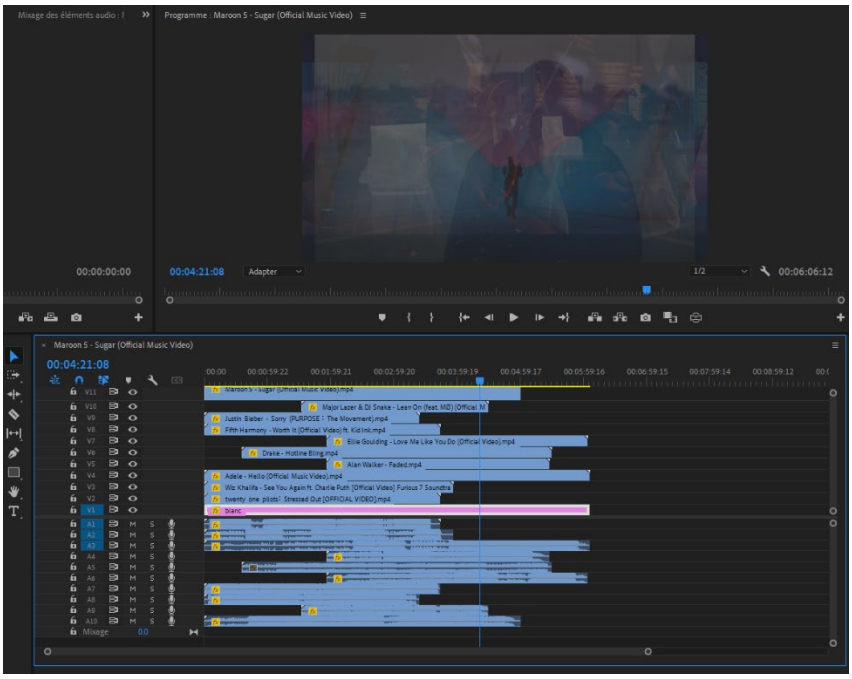


Figure 15 - Video editing for the 2015 dataset

Figure 16 - Screenshot of the dataset video

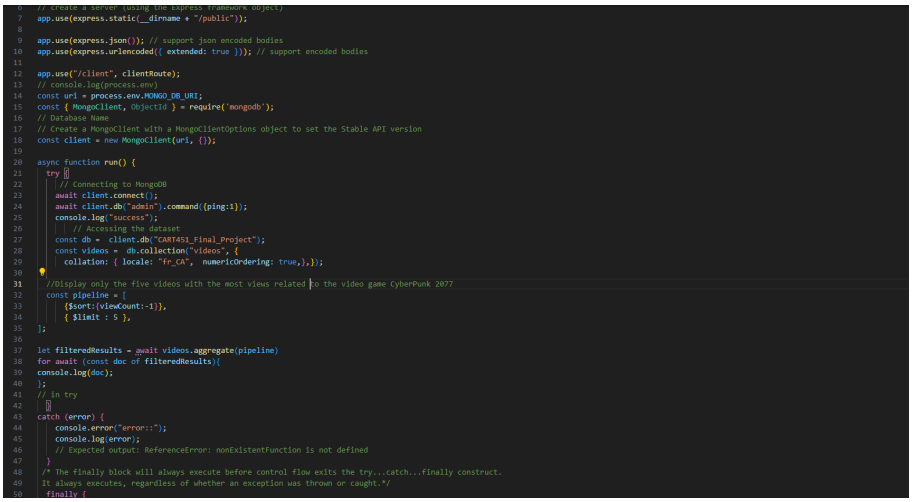
2020 video creation

```
const express = require("express");
const portNumber = 4200;
const app = express(); //make an instance of express
const server = require("http").createServer(app);
require("dotenv").config();
// create a server (using the Express framework object)
app.use(express.static(__dirname + "/public"));

app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
        // Accessing the dataset
    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 videos released in 2020 that have over a billion views, only the three videos with the most views appear in the results
const neededDocuments = {UploadYear:2020}
let foundResults = await videos.find(neededDocuments).sort({'video views':1}).limit(3);
await foundResults.forEach((doc)=>console.log(doc));

// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
/* The finally block will always execute before control flow exits the try...catch...finally construct.
It always executes, regardless of whether an exception was thrown or caught.*/
```

Figure 17 - Code pulling a list of videos released in 2020 that have more than a billion views

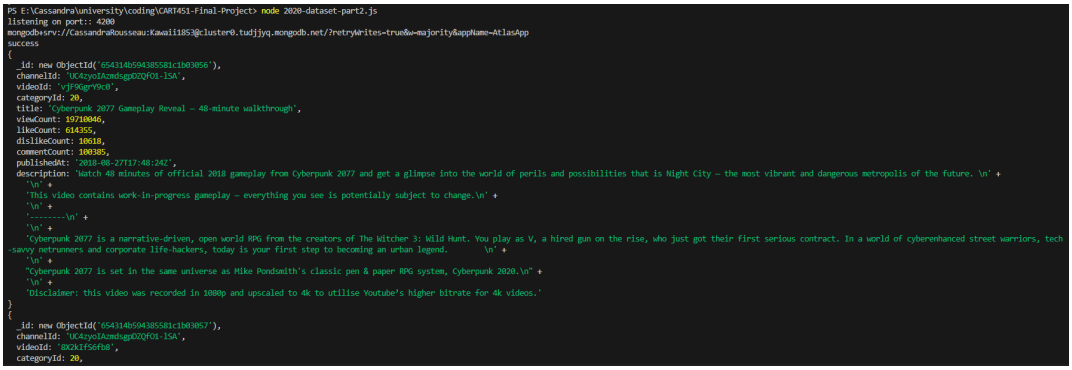
```
PS E:\Cassandra\university\coding\CART451-Final-Project> node 2020-dataset.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1853@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3bae'),
  Title: 'Future - Life is good (Official Music Video) ft Drake',
  Channel: 'Future',
  Published: '09-01-2020',
  UploadYear: 2020,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3baf'),
  Title: "BTS 'Dynamite' Official MV",
  Channel: 'HYBE LABELS',
  Published: '20-08-2020',
  UploadYear: 2020,
  Category: 'Musical'
}
{
  _id: new ObjectId('653c45e3a7ae7cbdddddd3bb0'),
  Title: '52 Gaj Ka Daman',
  Channel: 'Desi Records',
  Published: '01-10-2020',
  UploadYear: 2020,
  Category: 'Musical'
}
```

Figure 18 - Videos sorted from the first query



```
// create a server (using the Express framework object)
app.use(express.static(__dirname + "/public"));

app.use(express.json()); // support json encoded bodies
app.use(express.urlencoded({ extended: true })); // support encoded bodies

app.use("/client", clientRoute);
// console.log(process.env)
const uri = process.env.MONGO_DB_URI;
const { MongoClient, ObjectId } = require('mongodb');
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

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

  //Display only the five videos with the most views related to the video game CyberPunk 2077
  const pipeline = [
      {$sort:{viewCount:-1}},
      { $limit : 5 },
];

let filteredResults = await videos.aggregate(pipeline)
for await (const doc of filteredResults){
console.log(doc);
};
// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
/* The finally block will always execute before control flow exits the try...catch...finally construct.
It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
```

Figure 19 - Pulls a list of the most viewed videos in 2020



```
PS E:\Cassandra\uniVersity\coding\CART451-Final-Project> node 2020-dataset-part2.js
listening on port:: 4200
mongodb+srv://CassandraRousseau:Kawaii1853@cluster0.tudjjyq.mongodb.net/?retryWrites=true&w=majority&appName=AtlasApp
success
{
  _id: new ObjectId('654314b594385581c1b03056'),
  channelId: 'UC4zyoIAzmdsgpDZQfO1-lSA',
  videoId: 'vjF9GgrvY9c0',
  categoryId: 20,
  title: 'Cyberpunk 2077 Gameplay Reveal – 48-minute walkthrough',
  viewCount: 19710046,
  likeCount: 614355,
  dislikeCount: 10618,
  commentCount: 100385,
  publishedAt: '2018-08-27T17:48:24Z',
  description: 'Watch 48 minutes of official 2018 gameplay from Cyberpunk 2077 and get a glimpse into the world of perils and possibilities that is Night City – the most vibrant and dangerous metropolis of the future. \n' +
    '\n' +
    'This video contains work-in-progress gameplay – everything you see is potentially subject to change.\n' +
    '\n' +
    '--------\n' +
    '\n' +
    'Cyberpunk 2077 is a narrative-driven, open world RPG from the creators of The Witcher 3: Wild Hunt. You play as V, a hired gun on the rise, who just got their first serious contract. In a world of cyberenhanced street warriors, tech
-savvy netrunners and corporate life-hackers, today is your first step to becoming an urban legend.       \n' +
    '\n' +
    "Cyberpunk 2077 is set in the same universe as Mike Pondsmith's classic pen & paper RPG system, Cyberpunk 2020.\n" +
    '\n' +
    'Disclaimer: this video was recorded in 1080p and upscaled to 4k to utilise Youtube's higher bitrate for 4k videos.'
}
{
  _id: new ObjectId('654314b594385581c1b03057'),
  channelId: 'UC4zyoIAzmdsgpDZQfO1-lSA',
  videoId: '0X2kIF56fb8',
  categoryId: 20,
```

Figure 20 - Videos sorted from the second query

```
// Database Name
// Create a MongoClient with a MongoClientOptions object to set the Stable API version
const client = new MongoClient(uri, {});

async function run() {
  try {
    // Connecting to MongoDB
    await client.connect();
    await client.db("admin").command({ping:1});
    console.log("success");
      // Accessing the dataset
    const db =  client.db("CART451_Final_Project");
    const videos =  db.collection("2020", {
      collation: { locale: "fr_CA",  numericOrdering: true,},});
  //Find ten videos with views over 100 000 released in 2020
const neededDocuments = {view:{$gt:100000}}
let foundResults = await videos.find(neededDocuments).sort({view:-1}).limit(10);
await foundResults.forEach((doc)=>console.log(doc));
// in try
  }
catch (error) {
    console.error("error::");
    console.log(error);
    // Expected output: ReferenceError: nonExistentFunction is not defined
  }
 /* The finally block will always execute before control flow exits the try...catch...finally construct.
 It always executes, regardless of whether an exception was thrown or caught.*/
  finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
 }

 run()



 // make server listen for incoming messages
 server.listen(portNumber, function () {
   console.log("listening on port:: " + portNumber);
   console.log(process.env.Mongo_DB_URI);

 });
```

Figure 21 - Code pulling the most popular videos related to the video game CyberPunk 2077 in 2020

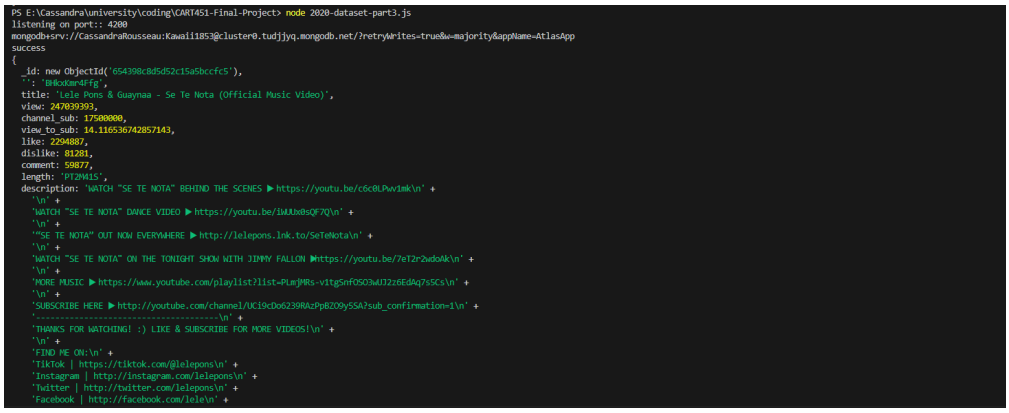


Figure 22 - Videos sorted from the third query

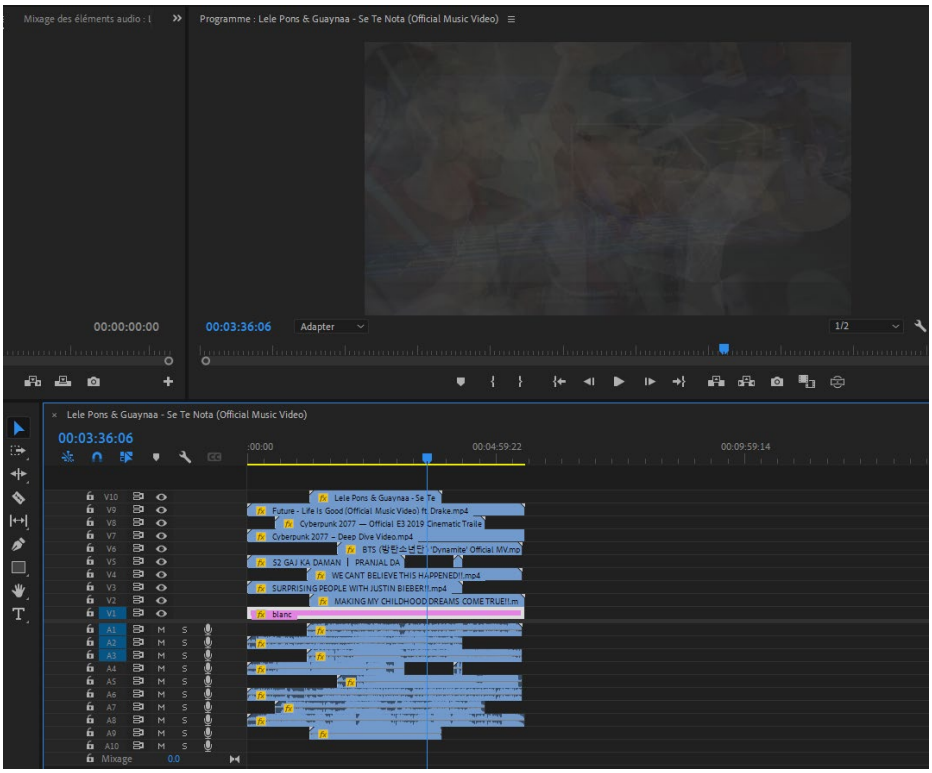


Figure 23 - Video editing for the 2020 dataset

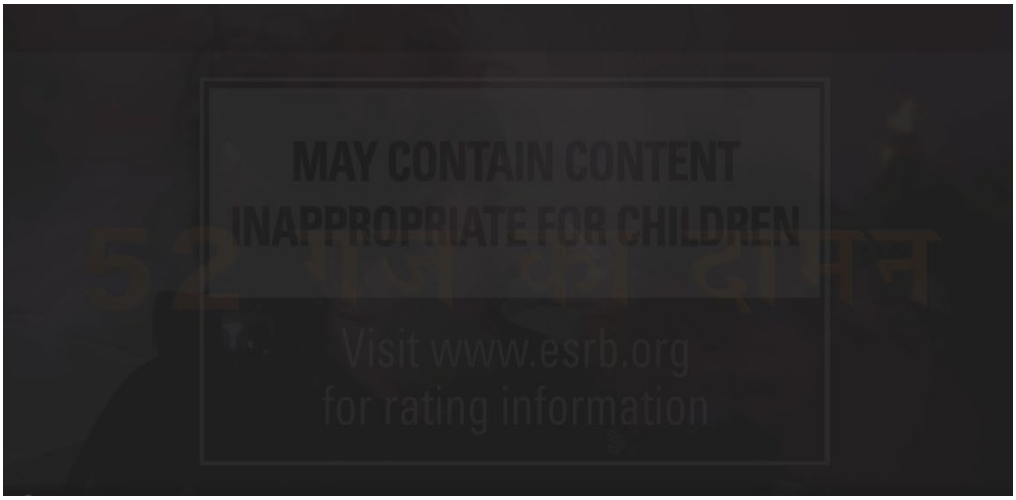


Figure 24- Screenshot of the dataset video

Code

The code was used to collect the data but not to visualize it; the datasets served me to discover the most influential or popular videos over the selected years to download them and create my data visualization. I created a code prototype through the first assignment to collect video data I reused for the final project prototype. However, 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. For the 2005 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 2010, I created two *js* files, one for each dataset used for that year. The first one was to pull data from videos released in 2010 with over one billion views. I used the find, sort, and limit functions with this dataset. The second one served to pull PewDiePie videos released in 2010 (i.e., the year his channel was created). I used the aggregation function for this dataset. For 2015, two *js* files were created. The first is pulling the list of the most viewed videos on *YouTube*. I used the find and toArray functions for this dataset. The second one is pulling videos released in 2015 with over a billion views. I used the find, sort, and limit functions with this dataset. Three datasets have been pulled for the year 2020. The first one is pulling a list of videos released in 2020 that have more than a billion views. I used the find, sort, and limit functions with this dataset. The second pulls a list of the most viewed videos in 2020. I used the aggregation function for this dataset. The third one pulls a list of the most popular videos related to the release of the video game *CyberPunk 2077* in 2020. I used the find, sort, and limit functions with this dataset. The results from the code are the videos that I've used for the data visualization. The only exception to the rule is the music videos of Justin Bieber from the 2010 video because only his title, *Baby,* was part of the results. Still, due to the phenomenon he brought in 2010, I decided to put his most viewed videos in 2010 in the data visualization.

Challenges & Issues

The most challenging part of the project is finding the right datasets or the right way to pull the data from them to get the results needed for the project. Most datasets I have found so far do not have a column within their collections telling just the year of video creation. It often provides the very specific date and hour the video was published. This accuracy is not helpful to pull a group of videos because the code requires a specific date to pull its data. Otherwise, I receive no results in the terminal. Suppose I can't find a dataset describing the year of publication. In that case, I must pull data by the highest number of views or find datasets of a subject that happened specifically during the selected year.

Regarding data visualization, I'll need to find a way to make the video layers opaque without making them hide each other. For now, I feel the opacity is still low and makes the videos a bit too dark to watch, but it is very hard to intensify their opacity without hiding the other layers. I tested some colour editing, but so far, I barely see any improvement by doing that.

Another challenge was downloading good-quality videos for the data. *YouTube* restricted the downloads of videos by adding their paid subscription to *YouTube Premium*. For the sake of the project, I decided to get a free month's trial. However, I got fooled by the platform; the downloading function is not allowing you to download the videos on your local computer, it stays on the digital platform. I tried to find an alternative to download the videos, and I found *TarTube*, a GUI front-end for *youtube-dl* and *yt-dlp*. This app allows you to download videos with the quality of your choice. This GUI was very useful, and I could download the videos I needed for my data collection and video editing.

References & Resources

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

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

"Cyberpunk 2077 Youtube Reception." Accessed November 4, 2023.
https://www.kaggle.com/datasets/andrewmvd/cyberpunk-2077.

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

"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.

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/.

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