Transport and Telecommunication Institute

Faculty of Engineering Science

**Big Data**

# Big Data Analysis and Implementation in Social Media

Student: Gonzalo, Gamez

Student's ID: st83446

Study Group: 4303MDA

**2024**

# Introduction

## 1.1 Brief Overview:

The topic chosen is big data in social media which focuses on videos and the engagement with users that watch said videos. With millions of gbs of data generated from user generated content, creators can take a data-driven approach to create content to improve metrics for engagement. Two data sets will be used, one from kaggle that has 19084 entries from TikTok with many 12 features and the other of 202 entries drawn using YouTube api with 9 features though will end up with more entries after exploration. Both TikTok and YouTube are social media platforms. The business problem is a TikTok influencer manager group is looking to expand into managing YouTube influencers or expanding to YouTube. They wish to see if the same trends apply in YouTube as they do in TikTok.

## 1.2 Objective:

The goal is to gather data from both structured and unstructured databases. Extract from both databases, extract into a usable form, and then load into a dataframe. The data should be able to provide insights and be made into a dashboard that can be accessed. The goal is to make a proof of concept if there is a way to increase user engagement such as views, likes, comments, shares by a variable influenced by creators in the different media platforms.

## 1.3 Scope and Limitations:

The boundaries are of data features from the kaggle dataset and possible features from YouTube api. Technologies used will be open source or free to use and on a local machine except for the deployment of the dashboard. Numerical data that is common between social media platforms will be used in the dashboard and categorical data will be excluded except in exploration. Exploration, ETL and the process of the development of graphs for the dashboard will be focused on. Database creation and deployment will not be covered. Limitations are no cloud services used, the limitations of features from TikTok and YouTube, still a small set of records from both, lack of parallel process involved, and no pipelines created to automate data flow from sources.

# Methodology

## 2.1 Data Selection:

### 2.1.1 Structured Data Source:

From the kaggle dataset (Yakhyojon, 2023) it has 19084 entries from TikTok with 12 features. The dataset is provided as a csv file though it will be inserted into a sqlite database. The data was chosen since the abundance of data that is created in social media and this data set had unique features that are not available on TikTok itself.

| Column Name | Type | Description |
|---|---|---|
| claim_status | obj | Whether the published video has been identified as an “opinion” or a “claim.” In this dataset, an “opinion” refers to an individual’s or group’s personal belief or thought. A “claim” refers to information that is either unsourced or from an unverified source. |
| video_duration_sec | int | How long the published video is measured in seconds. |
| transcription_length | obj | Length of transcribed text of the words spoken in the published video. |
| verified_status | obj | Indicates the status of the TikTok user who published the video in terms of their verification, either “verified” or “not verified.” |
| author_ban_status | obj | Indicates the status of the TikTok user who published the video in terms of their permissions: “active,” “under scrutiny,” or “banned.” |
| video_view_count | float | The total number of times the published video has been viewed. |
| video_like_count | float | The total number of times the published video has been liked by other users. |
| video_share_count | float | The total number of times the published video has been shared by other users. |
| video_download_count | float | The total number of times the published video has been downloaded by other users. |
| video_comment_count | float | The total number of comments on the published video. |

Table 1. Features descriptions for Tiktok dataset (Yakhyojon, 2023)

### 2.1.2 Unstructured/Semi-Structured Data Source (if applicable):

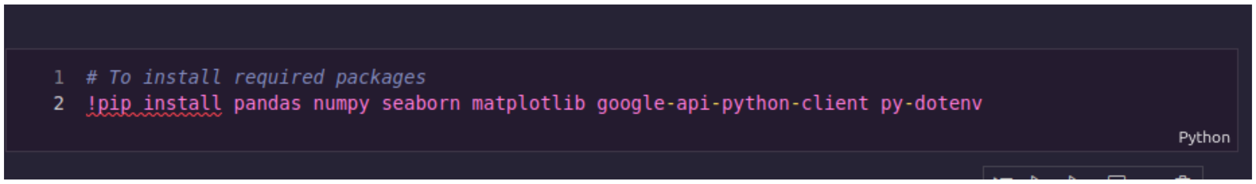
From the YouTube API that has 202 entries from TikTok with 9 features and will end up with 4500 in the full project. The dataset is provided in JSON though will be inserted into a Mongo database. The data was chosen since it practices using APIs and is able to scrape data from YouTube.

| Column Name | Data Type | Description |
|---|---|---|
| id | String | Unique identifier for the YouTube video. |
| title | String | Title of the YouTube video. |
| description | String | Detailed description of the YouTube video. |
| channelTitle | String | Name of the YouTube channel that published the video. |
| publishedAt | Date/Time | Date and time when the video was published. |
| viewCount | Integer | Total number of times the video has been viewed. |
| likeCount | Integer | Total number of likes received by the video. |
| commentCount | Integer | Total number of comments posted on the video. |
| topicCategories | List of Strings | List of URLs representing the video's topic categories. |

Table 2. Features descriptions for YouTube dataset (YouTube, 2020)

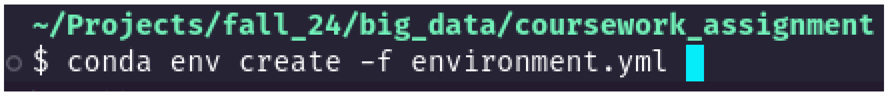
## 2.2 Setup in a Jupyter Notebook:

For the exploration notebook, the needed packages are installed in Figure 1. If using all notebooks to create the full project, it is recommended to create a conda environment from the yml in the terminal according to Figure 2. The minimum to interact with the exploration is having access to top_videos.json and tiktok_dataset.csv otherwise API Key for YouTube will need to be stored in a .env file to make top_videos.json.



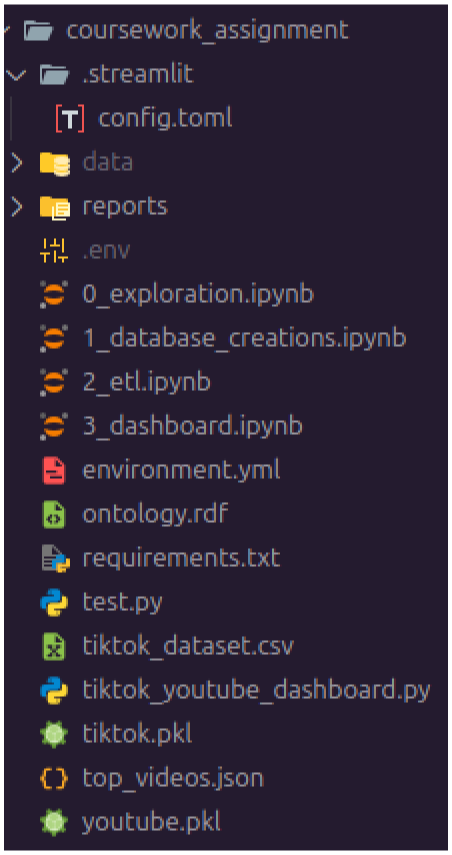
```
# To install required packages
!pip install pandas numpy seaborn matplotlib google-api-python-client py-dotenv
```

Figure 1. Code snippet to install required packages for 0_exploration.ipynb



```
~/Projects/fall_24/big_data/coursework_assignment
$ conda env create -f environment.yml
```

Figure 2. Code snippet to create environment with conda



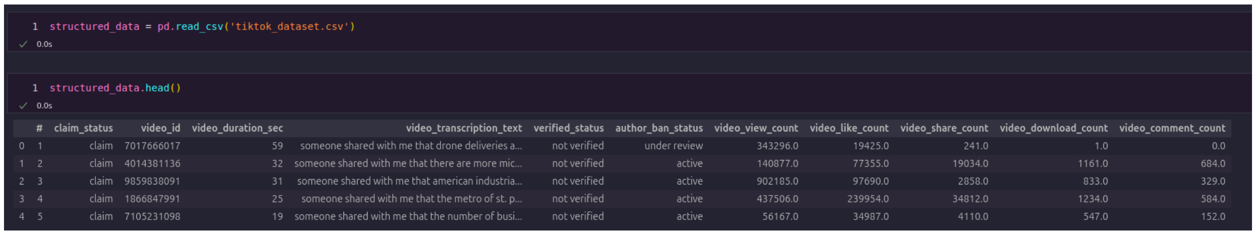
```
coursework_assignment
  .streamlit
    config.toml
  data
  reports
  .env
  0_exploration.ipynb
  1_database_creations.ipynb
  2_etl.ipynb
  3_dashboard.ipynb
  environment.yml
  ontology.rdf
  requirements.txt
  test.py
  tiktok_dataset.csv
  tiktok_youtube_dashboard.py
  tiktok.pkl
  top_videos.json
  youtube.pkl
```

Figure 3. Project files in coursework folder

## 2.3 Data Loading and Exploration:

### 2.3.1 Structured (TikTok) Data:

Loading the TikTok data from the csv and seeing a sample of a couple show how the data looks like in Figure 4. #, video_id, and video_transcription_text will be dropped and replaced with transcription_lenght to be easier to process. Figure 5 describes numerical data where the average duration is around 32 seconds, with a standard deviation of 16 seconds. Showing TikTok videos are short along with transcription length being short with 89 characters/words. According to Figure 6, there is a positive correlation with video views, likes, shares, downloads, and comments with no correlation with video duration and length. Looking at Figure 7, views disturbed over the categorical feature of claim status shows that videos that make claims are more likely to have more views than those with opinions.



```python
structured_data = pd.read_csv('tiktok_dataset.csv')
```

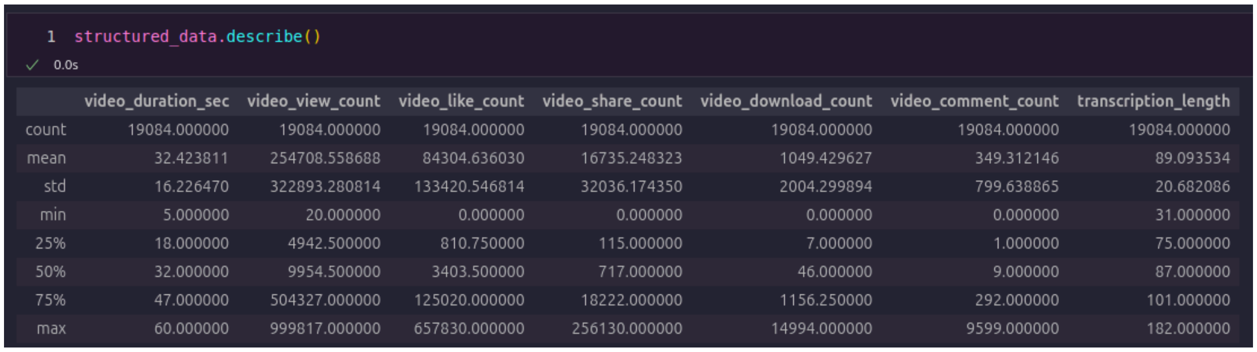
✓ 0.0s

```python
structured_data.head()
```

✓ 0.0s

| | # | claim_status | video_id | video_duration_sec | video_transcription_text | verified_status | author_ban_status | video_view_count | video_like_count | video_share_count | video_download_count | video_comment_count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | claim | 7017666017 | 59 | someone shared with me that drone deliveries a... | not verified | under review | 343296.0 | 19425.0 | 241.0 | 1.0 | 0.0 |
| 1 | 2 | claim | 4014381136 | 32 | someone shared with me that there are more mic... | not verified | active | 140877.0 | 77355.0 | 19034.0 | 1161.0 | 684.0 |
| 2 | 3 | claim | 9859838091 | 31 | someone shared with me that american industria... | not verified | active | 902185.0 | 97690.0 | 2858.0 | 833.0 | 329.0 |
| 3 | 4 | claim | 1866847991 | 25 | someone shared with me that the metro of st. p... | not verified | active | 437506.0 | 239954.0 | 34812.0 | 1234.0 | 584.0 |
| 4 | 5 | claim | 7105231098 | 19 | someone shared with me that the number of busi... | not verified | active | 56167.0 | 34987.0 | 4110.0 | 547.0 | 152.0 |

Figure 4. Loading and sample of structured data



```python
structured_data.describe()
```

✓ 0.0s

| | video_duration_sec | video_view_count | video_like_count | video_share_count | video_download_count | video_comment_count | transcription_length |
|---|---|---|---|---|---|---|---|
| count | 19084.000000 | 19084.000000 | 19084.000000 | 19084.000000 | 19084.000000 | 19084.000000 | 19084.000000 |
| mean | 32.423811 | 254708.558688 | 84304.636030 | 16735.248323 | 1049.429627 | 349.312146 | 89.093534 |
| std | 16.226470 | 322893.280814 | 133420.546814 | 32036.174350 | 2004.299894 | 799.638865 | 20.682086 |
| min | 5.000000 | 20.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 31.000000 |
| 25% | 18.000000 | 4942.500000 | 810.750000 | 115.000000 | 7.000000 | 1.000000 | 75.000000 |
| 50% | 32.000000 | 9954.500000 | 3403.500000 | 717.000000 | 46.000000 | 9.000000 | 87.000000 |
| 75% | 47.000000 | 504327.000000 | 125020.000000 | 18222.000000 | 1156.250000 | 292.000000 | 101.000000 |
| max | 60.000000 | 999817.000000 | 657830.000000 | 256130.000000 | 14994.000000 | 9599.000000 | 182.000000 |

Figure 5. Describing numerical features of structured data

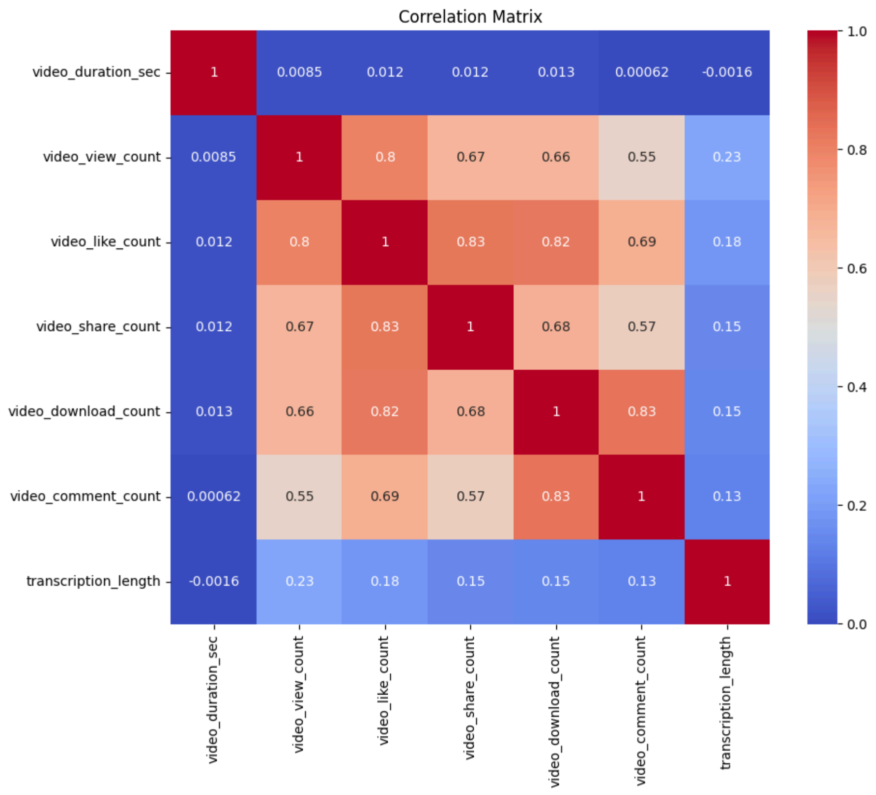


Figure 6. Correlation Matrix for numerical features

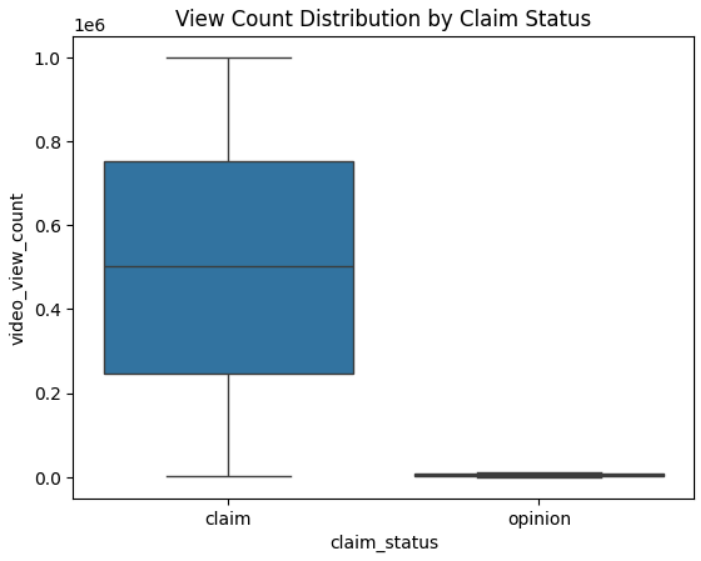
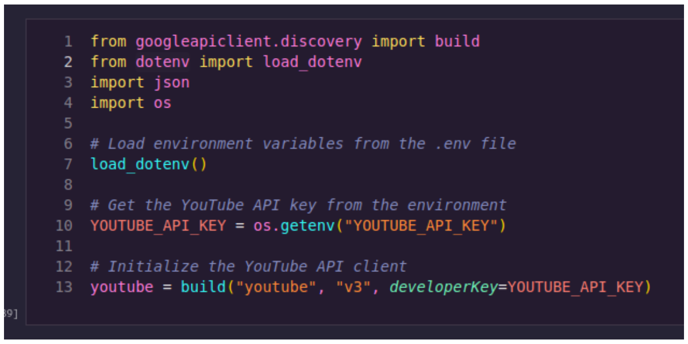


Figure 7. Boxplot for tik tok view count distributed by claim status

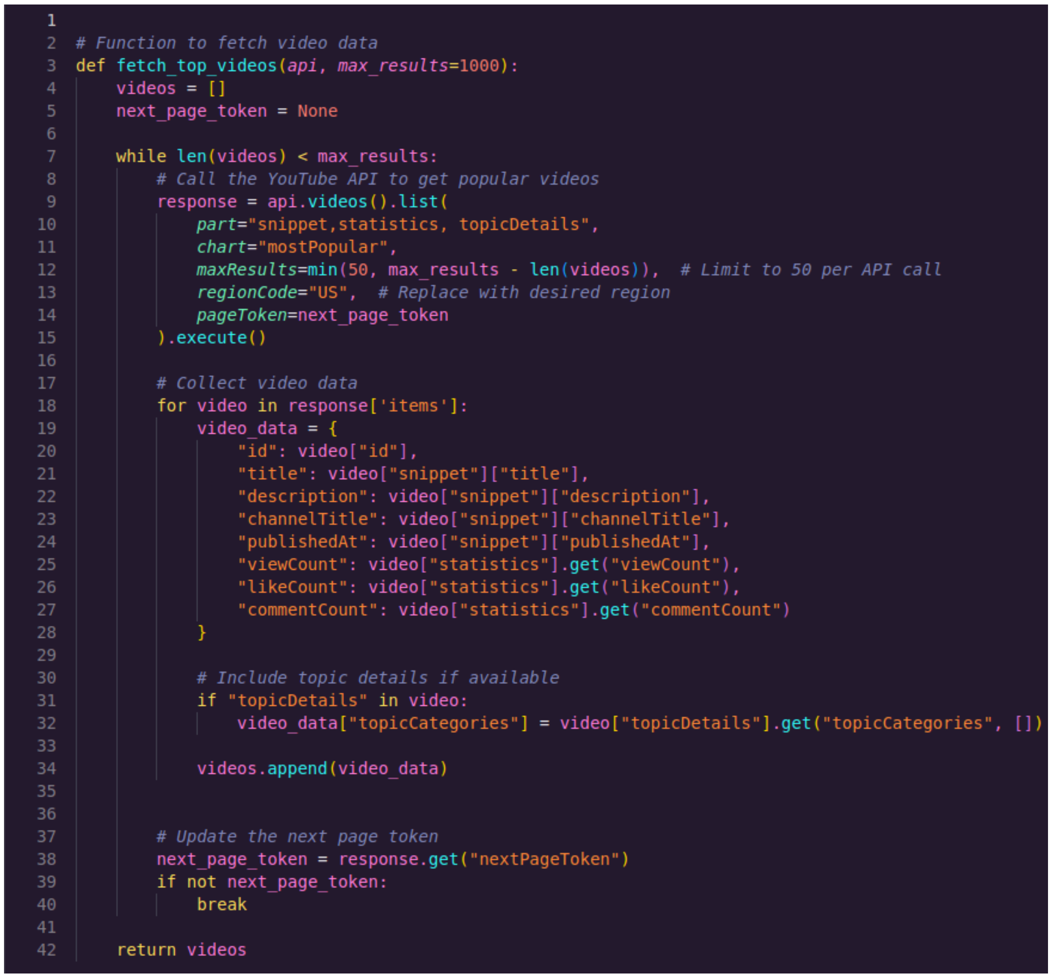
### 2.3.2 Unstructured (YouTube) Data:

With Youtube (2010) documentation and ChatGPT (2024), Figure 8, 9, and 10 is fetching 1000 entries of video stats into a JSON file though only 202 were fetched. Loading the TikTok youtube from the csv and seeing a sample of a couple show how the data looks like in Figure 11. Figure 12 shows a significant disparity in views, likes, and comments across videos, likely influenced by outliers with exceptionally high counts, while comment counts remain relatively low compared to views and likes. There is positive correlation between views, likes and comments according to Figure 13. Figure 14 shows music, sports, and film have high average view counts if grouped by category.



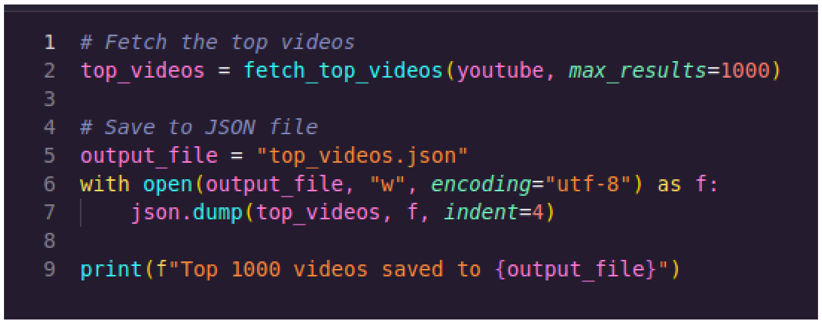
```python
from googleapiclient.discovery import build
from dotenv import load_dotenv
import json
import os

# Load environment variables from the .env file
load_dotenv()

# Get the YouTube API key from the environment
YOUTUBE_API_KEY = os.getenv("YOUTUBE_API_KEY")

# Initialize the YouTube API client
youtube = build("youtube", "v3", developerKey=YOUTUBE_API_KEY)
```

Figure 8. Loading api keys from .env



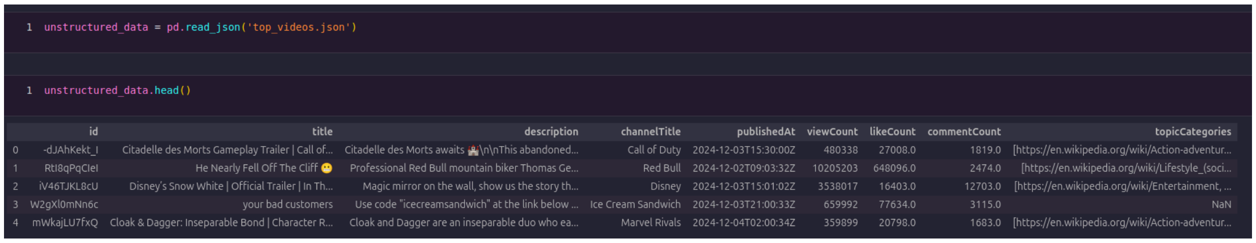
```python

# Function to fetch video data
def fetch_top_videos(api, max_results=1000):
    videos = []
    next_page_token = None

    while len(videos) < max_results:
        # Call the YouTube API to get popular videos
        response = api.videos().list(
            part="snippet,statistics, topicDetails",
            chart="mostPopular",
            maxResults=min(50, max_results - len(videos)),  # Limit to 50 per API call
            regionCode="US",  # Replace with desired region
            pageToken=next_page_token
        ).execute()

        # Collect video data
        for video in response['items']:
            video_data = {
                "id": video["id"],
                "title": video["snippet"]["title"],
                "description": video["snippet"]["description"],
                "channelTitle": video["snippet"]["channelTitle"],
                "publishedAt": video["snippet"]["publishedAt"],
                "viewCount": video["statistics"].get("viewCount"),
                "likeCount": video["statistics"].get("likeCount"),
                "commentCount": video["statistics"].get("commentCount")
            }

            # Include topic details if available
            if "topicDetails" in video:
                video_data["topicCategories"] = video["topicDetails"].get("topicCategories", [])

            videos.append(video_data)


        # Update the next page token
        next_page_token = response.get("nextPageToken")
        if not next_page_token:
            break

    return videos
```

Figure 9. Function from ChatGPT (2024) to get stats



```python
# Fetch the top videos
top_videos = fetch_top_videos(youtube, max_results=1000)

# Save to JSON file
output_file = "top_videos.json"
with open(output_file, "w", encoding="utf-8") as f:
    json.dump(top_videos, f, indent=4)

print(f"Top 1000 videos saved to {output_file}")
```

Figure 10. Fetching top 1000 videos on youtube dumping into a JSON file

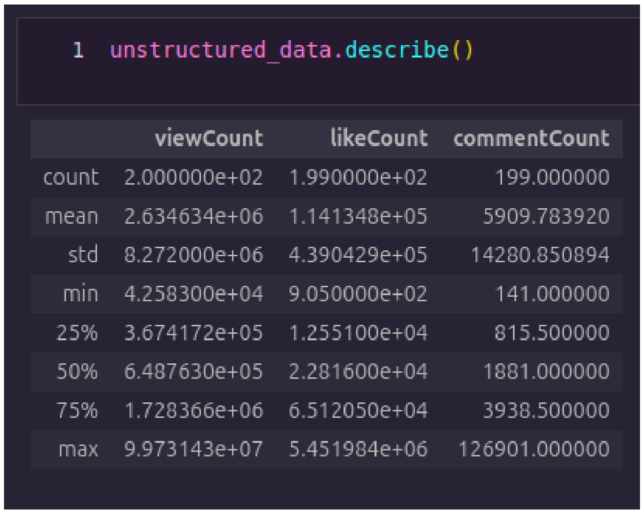


```python
unstructured_data = pd.read_json('top_videos.json')
```

```python
unstructured_data.head()
```

```
            id                                           title                                         description        channelTitle           publishedAt  viewCount  likeCount  commentCount                                     topicCategories
0  -dJAhKekt_I    Citadelle des Morts Gameplay Trailer | Call of...  Citadelle des Morts awaits 🏰\n\nThis abandoned...         Call of Duty  2024-12-03T15:30:00Z     480338    27008.0        1819.0  [https://en.wikipedia.org/wiki/Action-adventur...
1  RtI8qPqCIeI                       He Nearly Fell Off The Cliff 😬  Professional Red Bull mountain biker Thomas Ge...             Red Bull  2024-12-02T09:03:32Z   10205203   648096.0        2474.0   [https://en.wikipedia.org/wiki/Lifestyle_(soci...
2  iV46TJKL8cU    Disney's Snow White | Official Trailer | In Th...   Magic mirror on the wall, show us the story th...               Disney  2024-12-03T15:01:02Z    3538017    16403.0       12703.0  [https://en.wikipedia.org/wiki/Entertainment, ...
3  W2gXl0mNn6c                               your bad customers  Use code "icecreamsandwich" at the link below ...  Ice Cream Sandwich  2024-12-03T21:00:33Z     659992    77634.0        3115.0                                                 NaN
4  mWkajLU7fxQ  Cloak & Dagger: Inseparable Bond | Character R...  Cloak and Dagger are an inseparable duo who ea...        Marvel Rivals  2024-12-04T02:00:34Z     359899    20798.0        1683.0  [https://en.wikipedia.org/wiki/Action-adventur...
```

Figure 11. Loading and sample of unstructured data



```python
unstructured_data.describe()
```

```
          viewCount     likeCount   commentCount
count  2.000000e+02  1.990000e+02     199.000000
mean   2.634634e+06  1.141348e+05    5909.783920
std    8.272000e+06  4.390429e+05   14280.850894
min    4.258300e+04  9.050000e+02     141.000000
25%    3.674172e+05  1.255100e+04     815.500000
50%    6.487630e+05  2.281600e+04    1881.000000
75%    1.728366e+06  6.512050e+04    3938.500000
max    9.973143e+07  5.451984e+06  126901.000000
```

Figure 12. Describing numerical features of unstructured data

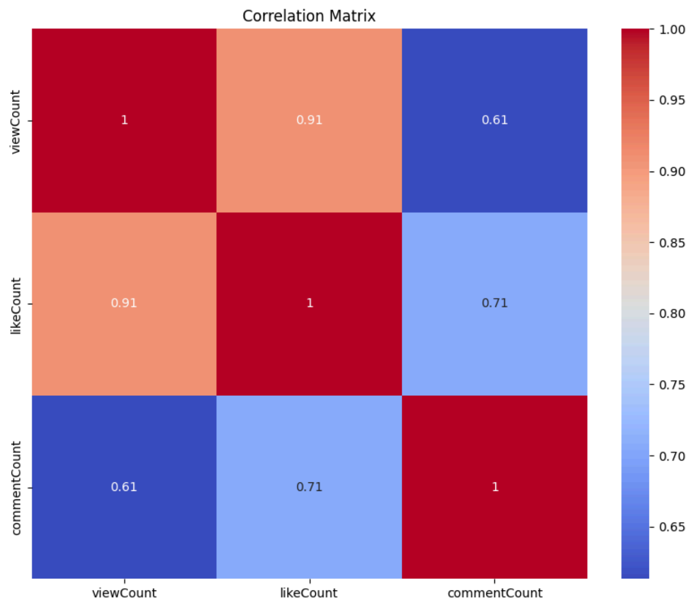


Figure 13.  Correlation Matrix for numerical features

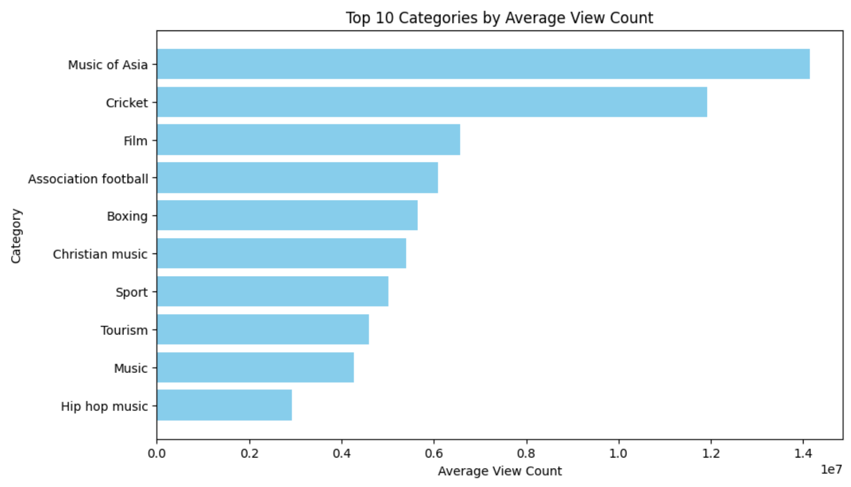
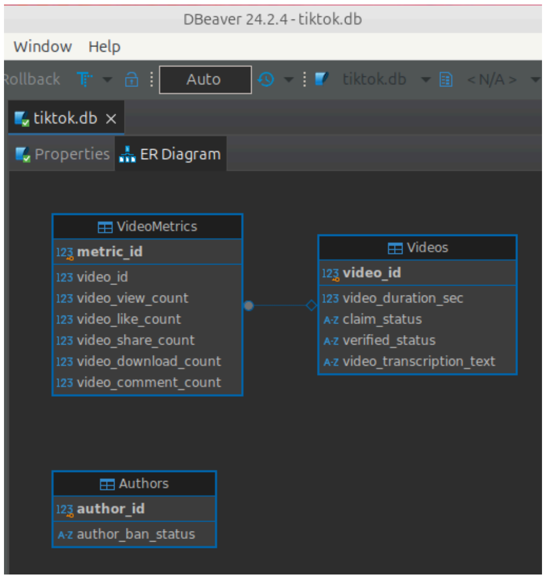


Figure 14. Barplot of top 10 categories sorted by average view count

# Analysis (MO3 & MO5)

## 3.1 Business Requirements Understanding:

The business problem is a TikTok influencer manager group that helps manage a variety of TikTok creators. In Figure 15 currently they have an sqlite database with information of the influencers they manage. In Figure 16 they have set up a MongoDB to gather data from YouTube. For a proof of concept (POC), they wish to move data to the cloud though doing it locally first to understand which services they should use afterwards according to Figure 17. The manager group is able to negotiate better deals with sponsorships if they show engagement metrics such as comments, shares, likes, and downloads from videos.

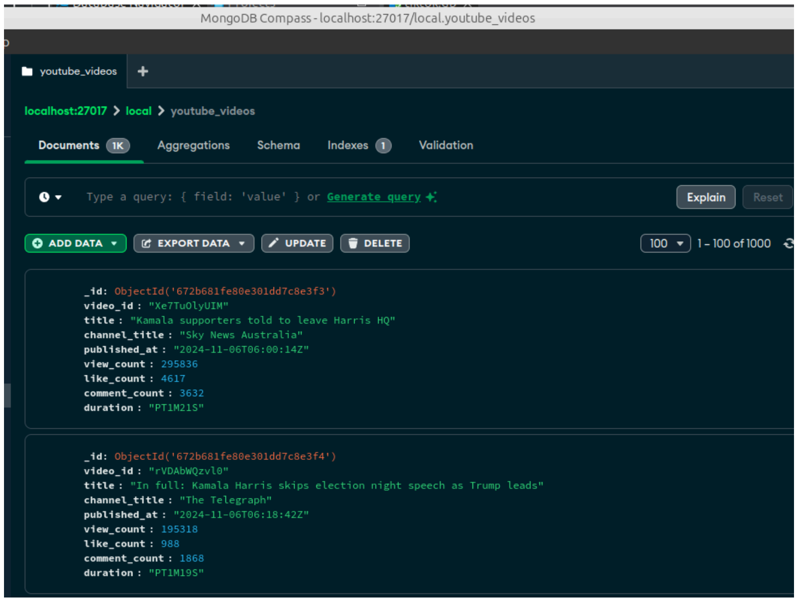


VideoMetrics
- metric_id
- video_id
- video_view_count
- video_like_count
- video_share_count
- video_download_count
- video_comment_count

Videos
- video_id
- video_duration_sec
- claim_status
- verified_status
- video_transcription_text

Authors
- author_id
- author_ban_status

Figure 15. ERD for TikTok sqlite


MongoDB Compass - localhost:27017/local.youtube_videos
youtube_videos
localhost:27017 > local > youtube_videos
Documents 1K
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
ADD DATA
EXPORT DATA
UPDATE
DELETE
100
1 – 100 of 1000

```
_id: ObjectId('672b681fe80e301dd7c8e3f3')
video_id : "Xe7TuOlyUIM"
title : "Kamala supporters told to leave Harris HQ"
channel_title : "Sky News Australia"
published_at : "2024-11-06T06:00:14Z"
view_count : 295836
like_count : 4617
comment_count : 3632
duration : "PT1M21S"
```

```
_id: ObjectId('672b681fe80e301dd7c8e3f4')
video_id : "rVDAbWQzvl0"
title : "In full: Kamala Harris skips election night speech as Trump leads"
channel_title : "The Telegraph"
published_at : "2024-11-06T06:18:42Z"
view_count : 195318
like_count : 988
comment_count : 1868
duration : "PT1M19S"
```

Figure 16. MongoDB compass screenshot of data being gathered

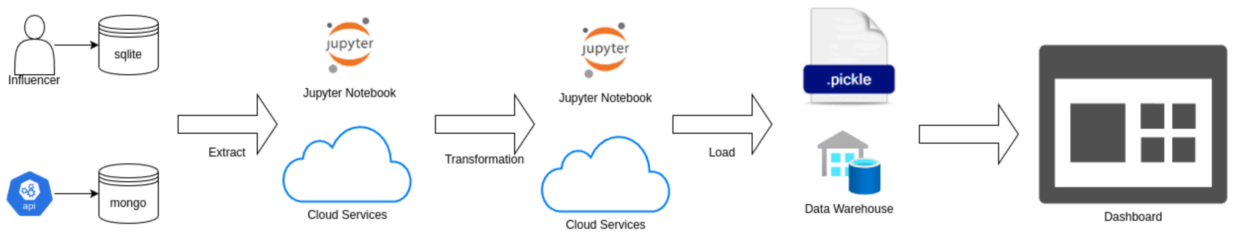
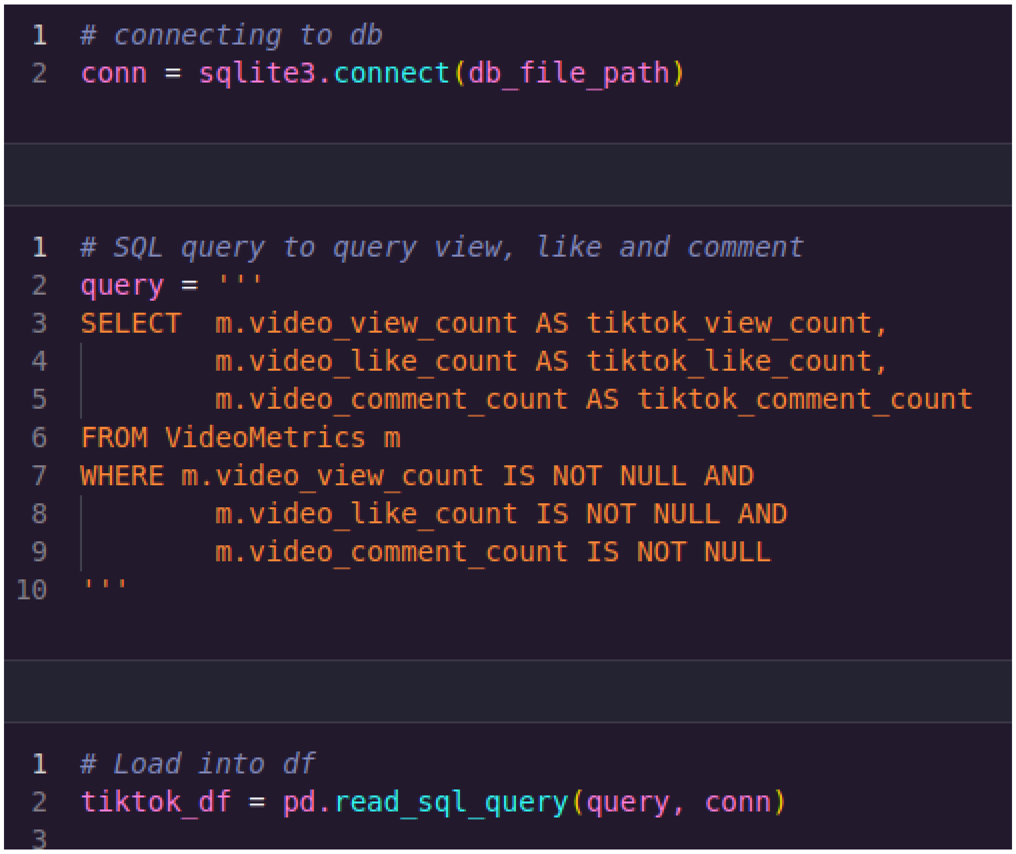


Figure 17. ETL process for POC

## 3.2 Data Extraction Techniques:

In a jupyter notebook, we will only gather views, likes and comments since they are available on both platforms. Connecting into the sqlite database and then querying non-null views, likes, and comments in TikTok then loading into a dataframe according to Figure 18. Figure 19 is similar to connecting to a mongo database then using a cursor to gather views, likes, and comments in YouTube then loading into a dataframe.

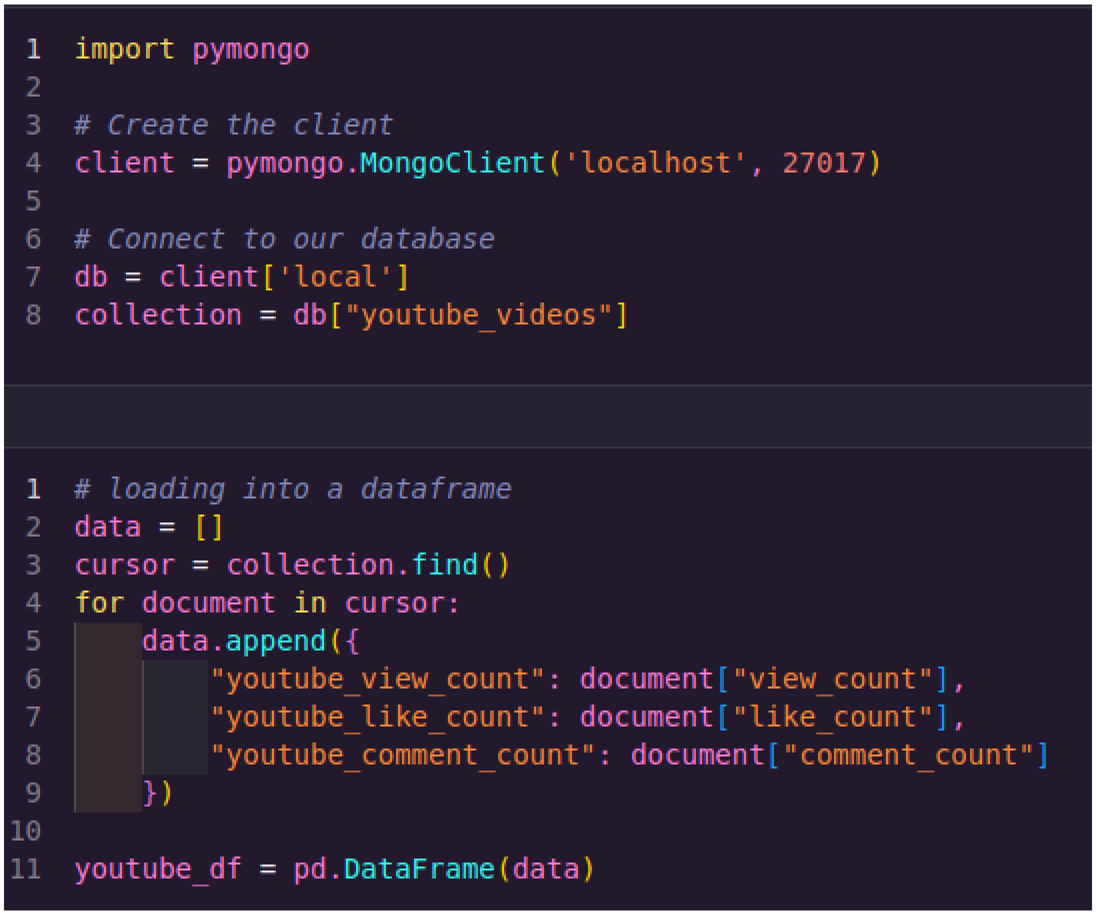


```
# connecting to db
conn = sqlite3.connect(db_file_path)
```

```
# SQL query to query view, like and comment
query = '''
SELECT  m.video_view_count AS tiktok_view_count,
        m.video_like_count AS tiktok_like_count,
        m.video_comment_count AS tiktok_comment_count
FROM VideoMetrics m
WHERE m.video_view_count IS NOT NULL AND
        m.video_like_count IS NOT NULL AND
        m.video_comment_count IS NOT NULL
'''
```

```
# Load into df
tiktok_df = pd.read_sql_query(query, conn)

```

Figure 18. Connecting to sqlite DB then loading query into a dataframe

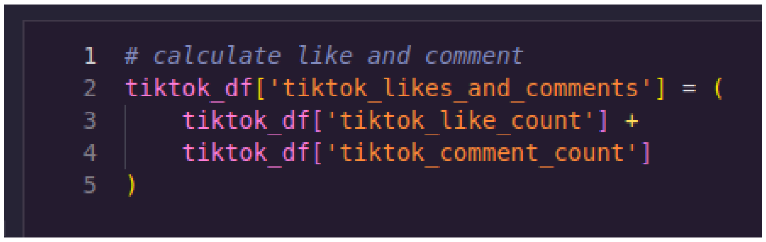


```python
import pymongo

# Create the client
client = pymongo.MongoClient('localhost', 27017)

# Connect to our database
db = client['local']
collection = db["youtube_videos"]
```

```python
# loading into a dataframe
data = []
cursor = collection.find()
for document in cursor:
    data.append({
        "youtube_view_count": document["view_count"],
        "youtube_like_count": document["like_count"],
        "youtube_comment_count": document["comment_count"]
    })

youtube_df = pd.DataFrame(data)
```

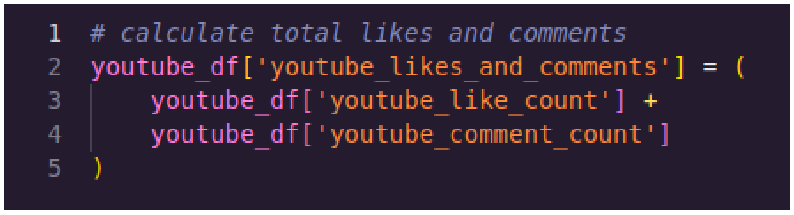
Figure 19. Connecting to mongoDB and loading into a dataframe

Originally a feature was created of total likes and comments as seen in figure 20 and 21. Though looking at correlation matrix in Figure 22 and 23, correlation is similar to like count so was not included.



```python
# calculate like and comment
tiktok_df['tiktok_likes_and_comments'] = (
    tiktok_df['tiktok_like_count'] +
    tiktok_df['tiktok_comment_count']
)
```

Figure 20. Creating a likes and comment feature for TikTok



```
# calculate total likes and comments
youtube_df['youtube_likes_and_comments'] = (
    youtube_df['youtube_like_count'] +
    youtube_df['youtube_comment_count']
)
```

Figure 21. Creating a likes and comment feature for YouTube

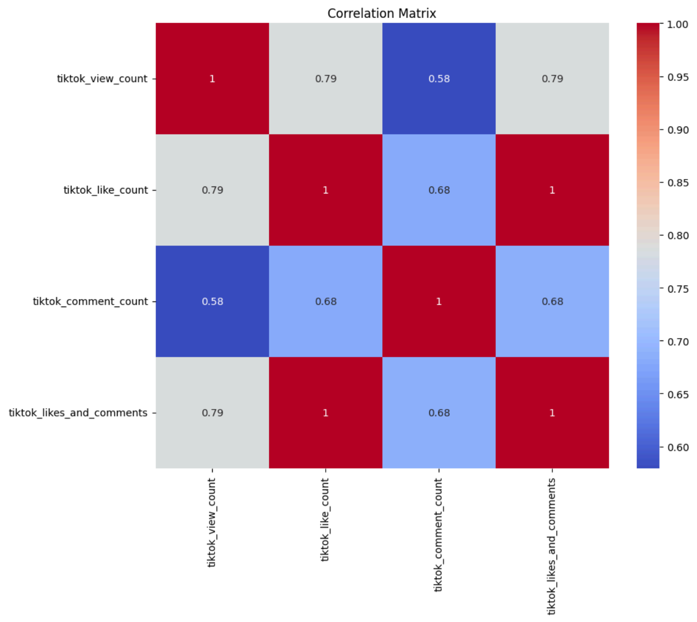


Figure 22. Correlation Matrix for TikTok

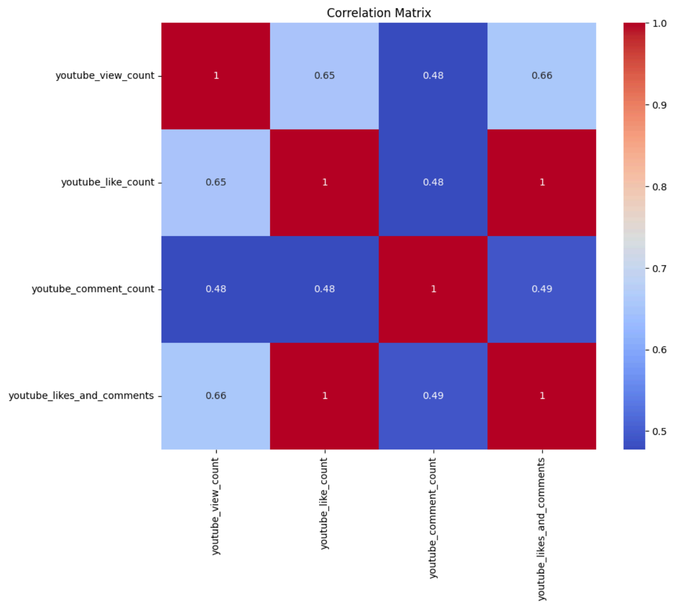


Figure 23. Correlation Matrix for YouTube

Figure 24 and 25 scatterplots with a linear regression. The graphs suggest that while both platforms have a positive relationship between views and likes, the relationship is stronger on TikTok than on YouTube. Similar graphs were made for comment counts on the dashboard.

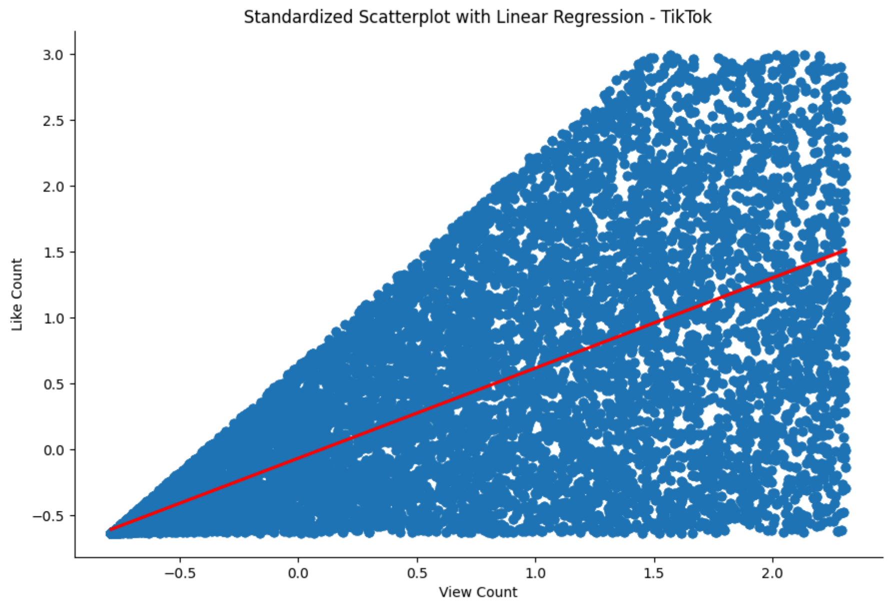


Figure 24. Standardized Scatterplot with Linear Regression for TikTok

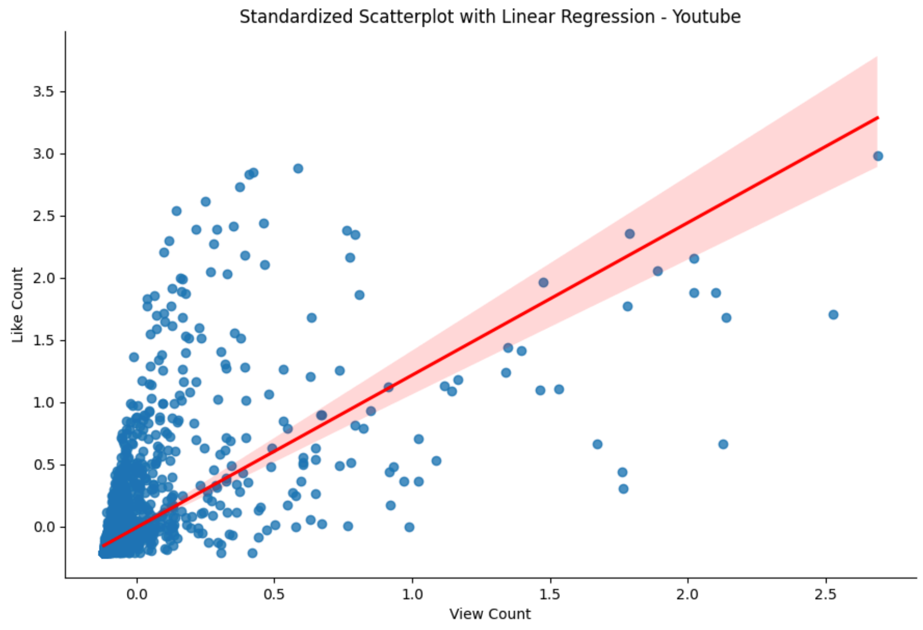
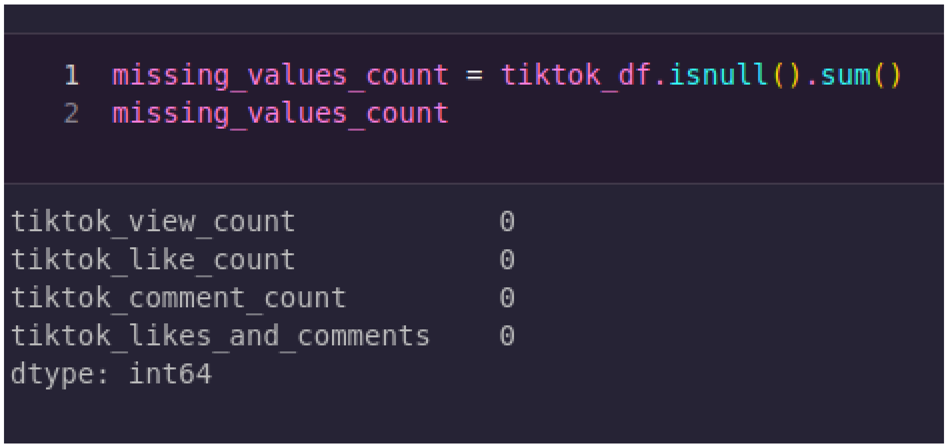


Figure 25. Standardized Scatterplot with Linear Regression for YouTube

# Design (MO6 & MO7)

## 4.1 Data Quality Improvement:

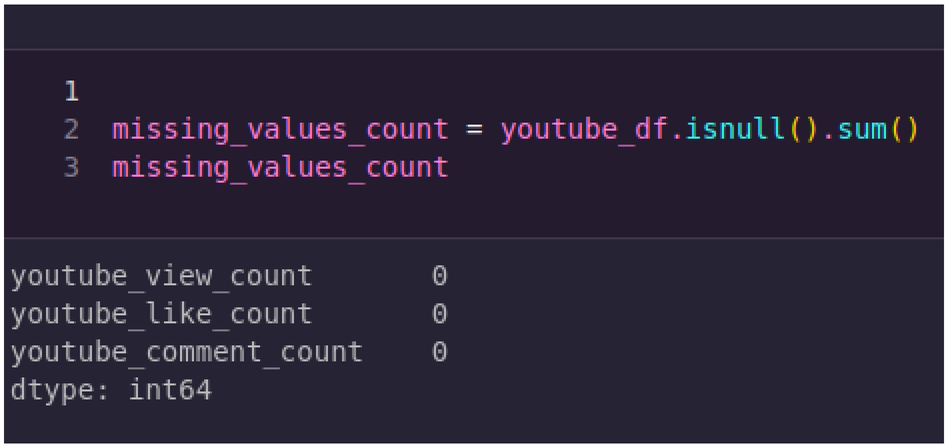
To deal with null values entries were excluded, the process can be done at extraction such as Figure 18 with a sql query to check if NOT NULL. To double check values are not null, Figure 26 and 27 show there are no null values. In figure 28, the data was standardized and outliers were removed using 3 standard deviation as a threshold.



```
missing_values_count = tiktok_df.isnull().sum()
missing_values_count
```

```
tiktok_view_count          0
tiktok_like_count          0
tiktok_comment_count       0
tiktok_likes_and_comments  0
dtype: int64
```

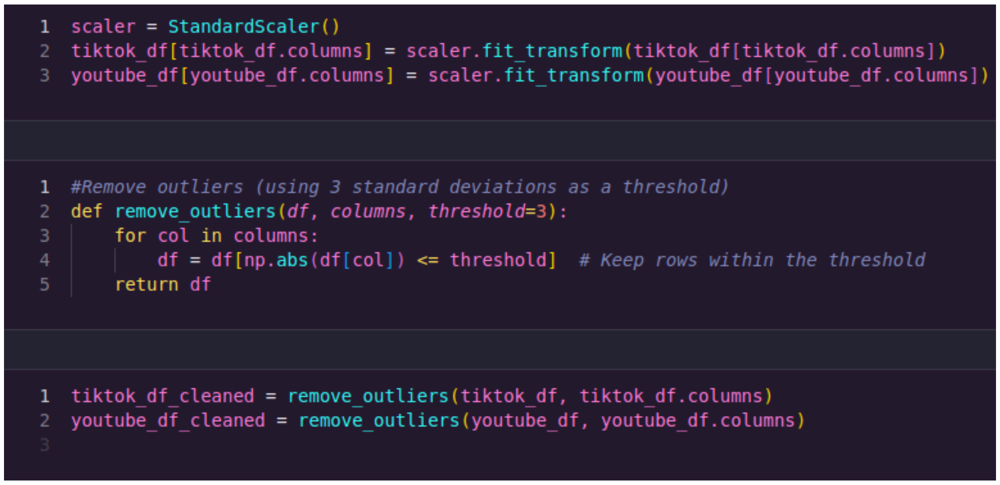
Figure 26. Checking nulls in TikTok dataframe



```

missing_values_count = youtube_df.isnull().sum()
missing_values_count
```

```
youtube_view_count       0
youtube_like_count       0
youtube_comment_count    0
dtype: int64
```

Figure 27. Checking nulls in YouTube dataframe



```python
scaler = StandardScaler()
tiktok_df[tiktok_df.columns] = scaler.fit_transform(tiktok_df[tiktok_df.columns])
youtube_df[youtube_df.columns] = scaler.fit_transform(youtube_df[youtube_df.columns])
```

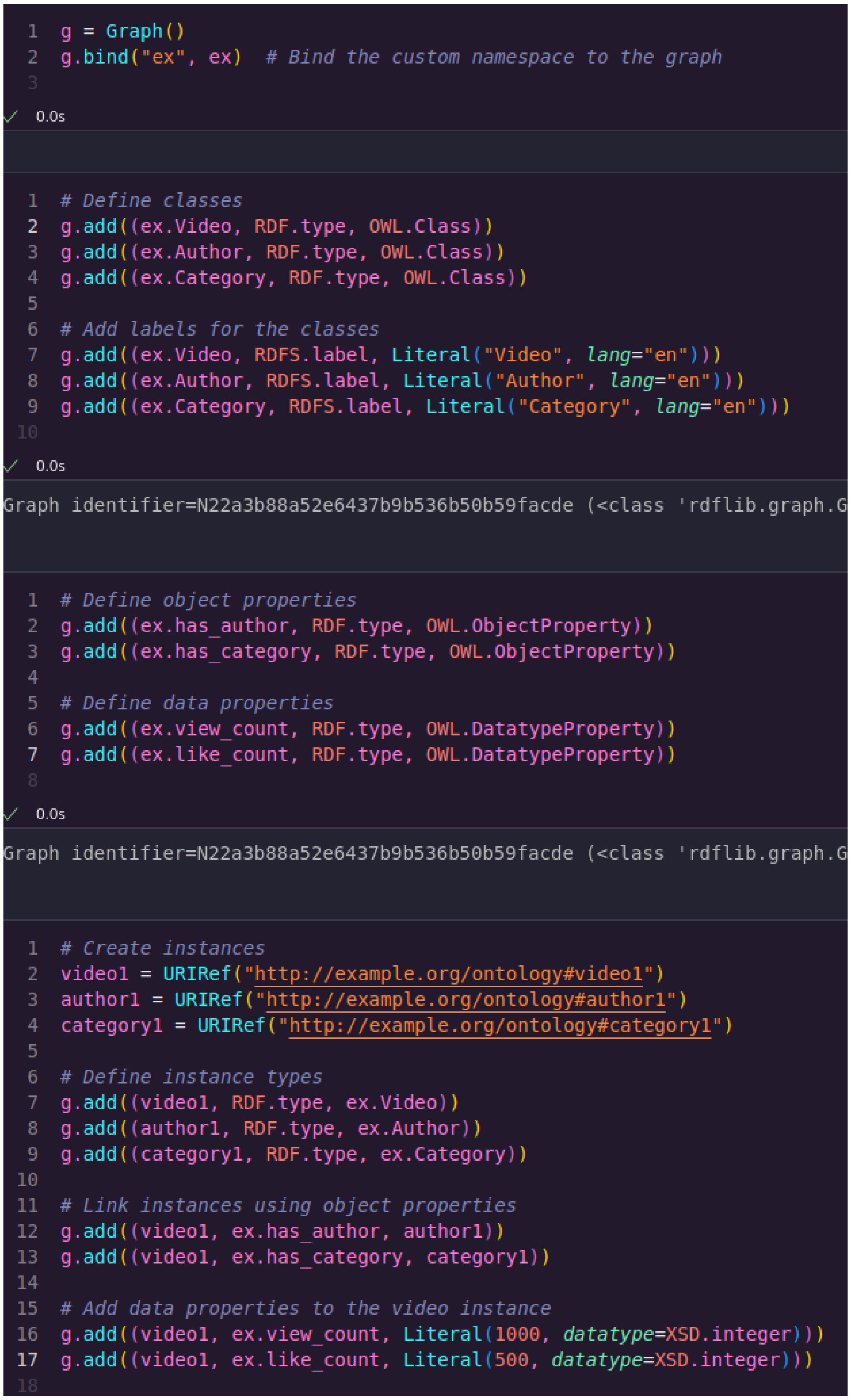
```python
#Remove outliers (using 3 standard deviations as a threshold)
def remove_outliers(df, columns, threshold=3):
    for col in columns:
        df = df[np.abs(df[col]) <= threshold]  # Keep rows within the threshold
    return df
```

```python
tiktok_df_cleaned = remove_outliers(tiktok_df, tiktok_df.columns)
youtube_df_cleaned = remove_outliers(youtube_df, youtube_df.columns)

```

Figure 28. Standardizing data and removing outliers

## 4.2 Ontology Generation:

Figure 29 has python code using rdflib which defines three classes: Video, Author, and Category. It also defines two object properties (has_author, has_category) to relate instances of these classes and two data properties (view_count, like_count) to associate numerical values with video instances. The code then creates instances of each class and assigns types and relationships between them. For example, it creates an instance of Video, links it to an instance of Author using the hasAuthor property, and assigns values to its view_count and like_count properties.



```
g = Graph()
g.bind("ex", ex)  # Bind the custom namespace to the graph

✓ 0.0s
```

```
# Define classes
g.add((ex.Video, RDF.type, OWL.Class))
g.add((ex.Author, RDF.type, OWL.Class))
g.add((ex.Category, RDF.type, OWL.Class))

# Add labels for the classes
g.add((ex.Video, RDFS.label, Literal("Video", lang="en")))
g.add((ex.Author, RDFS.label, Literal("Author", lang="en")))
g.add((ex.Category, RDFS.label, Literal("Category", lang="en")))

✓ 0.0s
Graph identifier=N22a3b88a52e6437b9b536b50b59facde (<class 'rdflib.graph.G
```

```
# Define object properties
g.add((ex.has_author, RDF.type, OWL.ObjectProperty))
g.add((ex.has_category, RDF.type, OWL.ObjectProperty))

# Define data properties
g.add((ex.view_count, RDF.type, OWL.DatatypeProperty))
g.add((ex.like_count, RDF.type, OWL.DatatypeProperty))

✓ 0.0s
Graph identifier=N22a3b88a52e6437b9b536b50b59facde (<class 'rdflib.graph.G
```

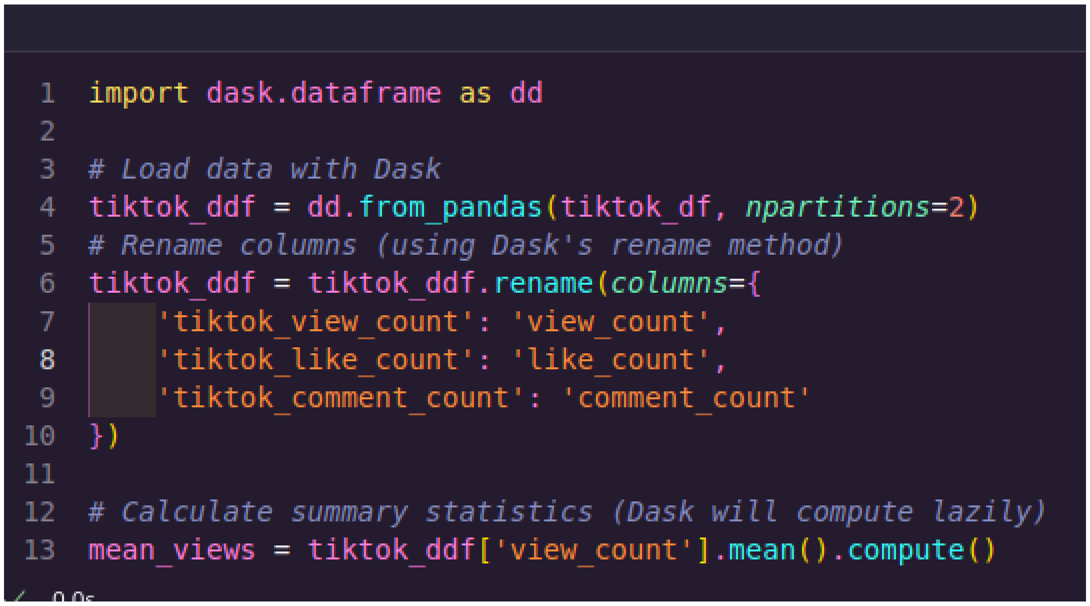
```
# Create instances
video1 = URIRef("http://example.org/ontology#video1")
author1 = URIRef("http://example.org/ontology#author1")
category1 = URIRef("http://example.org/ontology#category1")

# Define instance types
g.add((video1, RDF.type, ex.Video))
g.add((author1, RDF.type, ex.Author))
g.add((category1, RDF.type, ex.Category))

# Link instances using object properties
g.add((video1, ex.has_author, author1))
g.add((video1, ex.has_category, category1))

# Add data properties to the video instance
g.add((video1, ex.view_count, Literal(1000, datatype=XSD.integer)))
g.add((video1, ex.like_count, Literal(500, datatype=XSD.integer)))
```

Fig 29. Creating ontology

# Implementation (MO8)

## 5.1 Big Data Management with Dask:

Though in the project pandas was used, Figure 30 shows that code could be refactored to use Dask instead of pandas to deal with large amounts of data. According to Dask.org (2014), Dask is lazily evaluated where a computation isn't computed until asked such with compute as the mean_views. Plotting could be GPU accelerated with a GPU's cuda cores using dask_cudf also.



```python
import dask.dataframe as dd

# Load data with Dask
tiktok_ddf = dd.from_pandas(tiktok_df, npartitions=2)
# Rename columns (using Dask's rename method)
tiktok_ddf = tiktok_ddf.rename(columns={
    'tiktok_view_count': 'view_count',
    'tiktok_like_count': 'like_count',
    'tiktok_comment_count': 'comment_count'
})

# Calculate summary statistics (Dask will compute lazily)
mean_views = tiktok_ddf['view_count'].mean().compute()
```

Fig 30. Using Dask.

## 5.2 Cloud Computing Integration:

If the MongoDB cluster was deployed, pyspark could have been utilized to do groupings on several collections which could provide data that could be used in Figure 14. Figure 17 gives the intentions of storing in a data warehouse which would involve Online analytical process (OLAP) which could be structured for faster access to the data since it would not be directly connected to the online transition process (OLTP) of daily use. OLAP would also be used to give historical data too since snapshots of the past.

## 5.3 Addressing Privacy and Trust:

For the project itself, secrets were stored in an .env file with it added to the .gitignore. Ids were not extracted and only available in csv and JSON files which makes exploration possible. The databases are only accessible locally though would be implemented online only with a secure connection. To use cloud services, using standards like AES-256 should be employed. With secure protocols such as TLS so data is not intercepted and encrypting sensitive fields such as user ids within datasets.

# Results and Discussion

## 6.1 Findings:

Figure 24, 25, 31, 32 show that while both platforms have a positive relationship between views and likes, the relationship is stronger on TikTok than on YouTube. The relationship is stronger for comments on YouTube for comments.

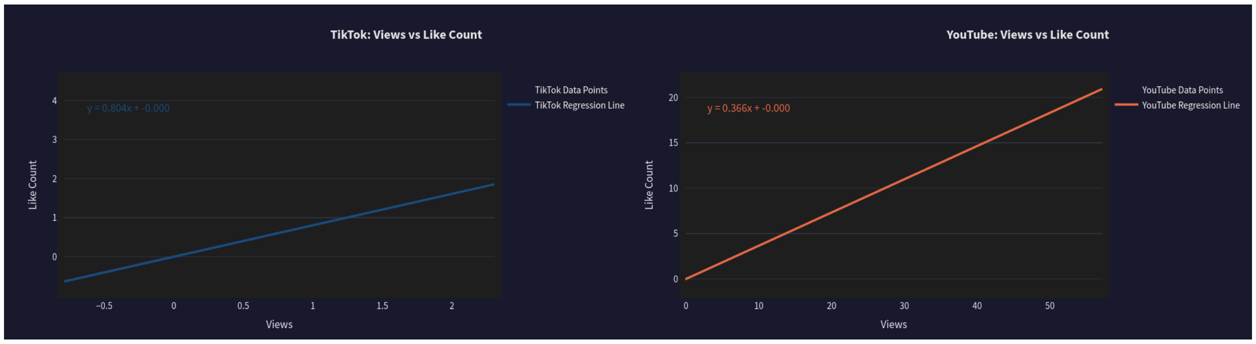


Fig 31. Dashboard for views vs likes from data created assisted by ChatGPT(2024)

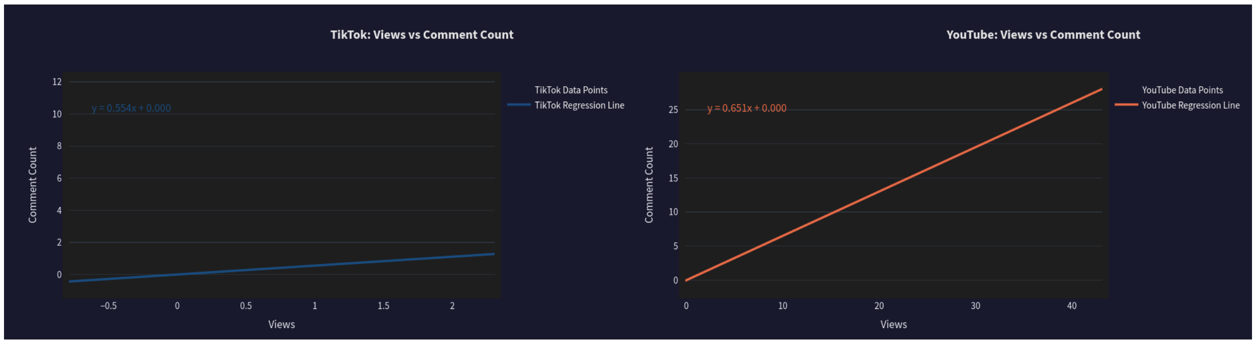


Fig 32. Dashboard for views vs comments from data created assisted by ChatGPT(2024)

## 6.2 Discussion:

In the context of the business problem, the more views the better for likes and comments. That in either platform, more views is what a creator should want. It is an industry standard to want more views for influencers. A closer look at categorical data is needed to measure against views since Figure 7 and 14 show that depending on topic or if a view makes a claim or opinion it can influence views.

## 6.3 Limitations:

The data set for TikTok does not provide when content was created and how the data was gathered in 2023. YouTube exploration was 202 entries of top videos and the data used in the full project was from random query words to gather 4500 entries in fall of 2024. Data was from the US, so might only pertain to the US. Full project only focused on numerical data and data that was shared by both platforms of views, likes, and comments. The dashboard data did not correctly standardize the data for YouTube. Pipelines were not created for the process to be automated to be ongoing with data.

# Conclusion

## 7.1 Summary:

This project analyzed video engagement on TikTok and YouTube, focusing on the relationship between views, likes, and comments. Key findings include a strong positive correlation between views and engagement on both platforms, with a slightly stronger relationship between views and likes on TikTok and between views and comments on YouTube. These findings show the role of view count in driving engagement for influencers on both platforms. While the analysis provides valuable insights, limitations such as limited data size and scope necessitate further research, including in-depth categorical analysis, time series analysis, and the development of predictive models. This project serves as a foundation for more sophisticated analyses and data-driven influencer marketing strategies.

## 7.2 Final Thoughts:

Understanding the process made it clear since I had originally wanted to do metrics against video duration. Working through the whole process locally also gives the idea what I would need to achieve in the cloud. The deployment of the dashboard even with the help of ChatGPT was a challenging experience but had to deploy to finish. The dashboard visuals can also do better. As mentioned, encryption, parallel processes, pipelines and cloud would be future work along with categorical data in analysis along with date creation.

# Project Links

Dashboard

Project Repo

## Reference list

ChatGPT. (2024). *ChatGPT - YouTube API JSON Loading*. [online] Available at: https://chatgpt.com/share/6755b642-9580-8008-9f99-e550471a7b68 [Accessed 8 Nov. 2024].

Chatgpt.com. (2024). *ChatGPT*. [online] Available at: https://chatgpt.com/c/6754484a-5b2c-8008-8878-2a2dcf920e80 [Accessed 5 Dec. 2024]. (I would like to make a scatter plot with a linear regression to see correlation as view count as the independent value).

Dask.org. (2014). *Dask — Dask documentation*. [online] Available at: https://docs.dask.org/.

Yakhyojon (2023). *TikTok User Engagement Data*. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/yakhyojon/tiktok/data [Accessed 8 Nov. 2024].

YouTube (2020). *Videos | YouTube Data API | Google for Developers*. [online] Google for Developers. Available at: https://developers.google.com/youtube/v3/docs/videos [Accessed 8 Nov. 2024].