```
!pip install google-api-python-client
```

```
Collecting google-api-python-client
  Downloading google_api_python_client-2.146.0-py2.py3-none-
any.whl.metadata (6.7 kB)
Collecting httplib2<1.dev0,>=0.19.0 (from google-api-python-client)
  Downloading httplib2-0.22.0-py3-none-any.whl.metadata (2.6 kB)
Collecting google-auth!=2.24.0,!=2.25.0,<3.0.0.dev0,>=1.32.0 (from
google-api-python-client)
  Downloading google_auth-2.35.0-py2.py3-none-any.whl.metadata (4.7
kB)
Collecting google-auth-httplib2<1.0.0,>=0.2.0 (from google-api-python-
client)
  Downloading google_auth_httplib2-0.2.0-py2.py3-none-any.whl.metadata
(2.2 kB)
Collecting google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!
=2.3.0,<3.0.0.dev0,>=1.31.5 (from google-api-python-client)
  Downloading google_api_core-2.20.0-py3-none-any.whl.metadata (2.7
kB)
Collecting uritemplate<5,>=3.0.1 (from google-api-python-client)
  Downloading uritemplate-4.1.1-py2.py3-none-any.whl.metadata (2.9 kB)
Collecting googleapis-common-protos<2.0.dev0,>=1.56.2 (from google-
api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-
api-python-client)
  Downloading googleapis_common_protos-1.65.0-py2.py3-none-
any.whl.metadata (1.5 kB)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!
=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0.dev0,>=3.19.5 in
c:\users\prasanna kumar\anaconda3\lib\site-packages (from google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-
python-client) (3.20.3)
Collecting proto-plus<2.0.0dev,>=1.22.3 (from google-api-core!=2.0.*,!
=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-client)
  Downloading proto_plus-1.24.0-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in c:\
users\prasanna kumar\anaconda3\lib\site-packages (from google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-
python-client) (2.31.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\
prasanna kumar\anaconda3\lib\site-packages (from google-auth!=2.24.0,!
=2.25.0,<3.0.0.dev0,>=1.32.0->google-api-python-client) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\
prasanna kumar\anaconda3\lib\site-packages (from google-auth!=2.24.0,!
=2.25.0,<3.0.0.dev0,>=1.32.0->google-api-python-client) (0.2.8)
Collecting rsa<5,>=3.1.4 (from google-auth!=2.24.0,!
=2.25.0,<3.0.0.dev0,>=1.32.0->google-api-python-client)
  Downloading rsa-4.9-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!
=3.0.3,<4,>=2.4.2 in c:\users\prasanna kumar\anaconda3\lib\site-
packages (from httplib2<1.dev0,>=0.19.0->google-api-python-client)
```

```
(3.0.9)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\
prasanna kumar\anaconda3\lib\site-packages (from pyasn1-
modules>=0.2.1->google-auth!=2.24.0,!=2.25.0,<3.0.0.dev0,>=1.32.0-
>google-api-python-client) (0.4.8)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
prasanna kumar\anaconda3\lib\site-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!
=2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-client) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\prasanna
kumar\anaconda3\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0-
>google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5-
>google-api-python-client) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\prasanna
kumar\anaconda3\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0-
>google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5-
>google-api-python-client) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\prasanna
kumar\anaconda3\lib\site-packages (from requests<3.0.0.dev0,>=2.18.0-
>google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0.dev0,>=1.31.5-
>google-api-python-client) (2024.2.2)
Downloading google_api_python_client-2.146.0-py2.py3-none-any.whl
(12.2 MB)
   ---------------------------------------- 0.0/12.2 MB ? eta -:--:--
   ---------------------------------------- 0.0/12.2 MB ? eta -:--:--
   ---------------------------------------- 0.0/12.2 MB 653.6 kB/s eta
0:00:19
   ---------------------------------------- 0.1/12.2 MB 1.2 MB/s eta
0:00:10
    --------------------------------------- 0.2/12.2 MB 1.2 MB/s eta
0:00:10
   - -------------------------------------- 0.3/12.2 MB 1.8 MB/s eta
0:00:07
   - -------------------------------------- 0.5/12.2 MB 1.9 MB/s eta
0:00:07
   -- ------------------------------------- 0.8/12.2 MB 2.7 MB/s eta
0:00:05
   ---- ----------------------------------- 1.3/12.2 MB 3.8 MB/s eta
0:00:03
   ---- ----------------------------------- 1.5/12.2 MB 3.8 MB/s eta
0:00:03
   ------- -------------------------------- 2.4/12.2 MB 5.6 MB/s eta
0:00:02
   ---------- ----------------------------- 3.1/12.2 MB 6.4 MB/s eta
0:00:02
   --------------- ------------------------ 4.9/12.2 MB 9.2 MB/s eta
0:00:01
   ---------------- ----------------------- 5.1/12.2 MB 8.9 MB/s eta
0:00:01
```

```
   --------------------------- ------------ 8.5/12.2 MB 13.5 MB/s eta
0:00:01
   ------------------------------- -------- 9.5/12.2 MB 14.1 MB/s eta
0:00:01
   ---------------------------------------  12.0/12.2 MB 32.8 MB/s eta
0:00:01
   ---------------------------------------- 12.2/12.2 MB 29.7 MB/s eta
0:00:00
Downloading google_api_core-2.20.0-py3-none-any.whl (142 kB)
   ---------------------------------------- 0.0/142.2 kB ? eta -:--:--
   ---------------------------------------- 142.2/142.2 kB 8.2 MB/s
eta 0:00:00
Downloading google_auth-2.35.0-py2.py3-none-any.whl (208 kB)
   ---------------------------------------- 0.0/209.0 kB ? eta -:--:--
   ---------------------------------------- 209.0/209.0 kB ? eta
0:00:00
Downloading google_auth_httplib2-0.2.0-py2.py3-none-any.whl (9.3 kB)
Downloading httplib2-0.22.0-py3-none-any.whl (96 kB)
   ---------------------------------------- 0.0/96.9 kB ? eta -:--:--
   ---------------------------------------- 96.9/96.9 kB 5.4 MB/s eta
0:00:00
Downloading uritemplate-4.1.1-py2.py3-none-any.whl (10 kB)
Downloading googleapis_common_protos-1.65.0-py2.py3-none-any.whl (220
kB)
   ---------------------------------------- 0.0/220.9 kB ? eta -:--:--
   ---------------------------------------- 220.9/220.9 kB 13.2 MB/s
eta 0:00:00
Downloading proto_plus-1.24.0-py3-none-any.whl (50 kB)
   ---------------------------------------- 0.0/50.1 kB ? eta -:--:--
   ---------------------------------------- 50.1/50.1 kB ? eta 0:00:00
Downloading rsa-4.9-py3-none-any.whl (34 kB)
Installing collected packages: uritemplate, rsa, proto-plus, httplib2,
googleapis-common-protos, google-auth, google-auth-httplib2, google-
api-core, google-api-python-client
Successfully installed google-api-core-2.20.0 google-api-python-
client-2.146.0 google-auth-2.35.0 google-auth-httplib2-0.2.0
googleapis-common-protos-1.65.0 httplib2-0.22.0 proto-plus-1.24.0 rsa-
4.9 uritemplate-4.1.1
```

Get the video details using the API

```python
import pandas as pd
from googleapiclient.discovery import build

# replace with your own API key
API_KEY = 'AIzaSyBfLPsSeHdvht-KiW9eZo6xFVkvYGpUQtE'

def get_trending_videos(api_key, max_results=200):
    # build the youtube service
    youtube = build('youtube', 'v3', developerKey=api_key)

    # initialize the list to hold video details
    videos = []

    # fetch the most popular videos
    request = youtube.videos().list(
        part='snippet,contentDetails,statistics',
        chart='mostPopular',
        regionCode='US',
        maxResults=50
    )

    # paginate through the results if max_results > 50
    while request and len(videos) < max_results:
        response = request.execute()
        for item in response['items']:
            video_details = {
                'video_id': item['id'],
                'title': item['snippet']['title'],
                'description': item['snippet']['description'],
                'published_at': item['snippet']['publishedAt'],
                'channel_id': item['snippet']['channelId'],
                'channel_title': item['snippet']['channelTitle'],
                'category_id': item['snippet']['categoryId'],
                'tags': item['snippet'].get('tags', []),
                'duration': item['contentDetails']['duration'],
                'definition': item['contentDetails']['definition'],
                'caption': item['contentDetails'].get('caption',
'false'),
                'view_count': item['statistics'].get('viewCount', 0),
                'like_count': item['statistics'].get('likeCount', 0),
                'dislike_count':
item['statistics'].get('dislikeCount', 0),
                'favorite_count':
item['statistics'].get('favoriteCount', 0),
                'comment_count':
item['statistics'].get('commentCount', 0)
            }
            videos.append(video_details)

        # get the next page token
        request = youtube.videos().list_next(request, response)

    return videos[:max_results]

def save_to_csv(data, filename):
    df = pd.DataFrame(data)
    df.to_csv(filename, index=False)
```

```
def main():
    trending_videos = get_trending_videos(API_KEY)
    filename = 'trending_videos.csv'
    save_to_csv(trending_videos, filename)
    print(f'Trending videos saved to {filename}')

if __name__ == '__main__':
    main()

Trending videos saved to trending_videos.csv
```

Import the data and read the file

```
import pandas as pd

trending_videos = pd.read_csv('trending_videos.csv')
print(trending_videos.head())

      video_id                                              title  \
0  qyP8arCDJk8                         They put me in a video game...
1  -jYfC4YYXIw  GREATEST GAME EVER?!? Shohei Ohtani goes 6-FOR...
2  rWjky-ibZIM  SIDEMEN AMONG US PROXIMITY CHAT: SHAPESHIFTER ...
3  On1mm8vWJ50                    BAD BUNNY - Una Velita (Visualizer)
4  1YHDGLqH1VM              Future - TOO FAST (Official Music Video)

                                         description
published_at  \
0  Thanks to AFK Journey for sponsoring this vide...  2024-09-
20T17:00:49Z
1  Is this the greatest game in baseball history?...  2024-09-
20T00:01:31Z
2  □: Order food NOW at: https://www.eatsides.com...  2024-09-
19T19:50:10Z
3  BAD BUNNY \nUna Velita  (Visualizer)\n\nLetra/...  2024-09-
20T00:00:01Z
4  Future - TOO FAST (Official Music Video)\n\n"M...  2024-09-
20T04:00:07Z

                 channel_id     channel_title  category_id  \
0  UCGwu0nbY2wSkW8N-cghnLpA  JaidenAnimations            1
1  UCoLrcjPV5PbUrUyXq5mjc_A               MLB           17
2  UCh5mLn90vUaB1PbRRx_AiaA      MoreSidemen           22
3  UCmBA_wu8xGg1OfOkfW13Q0Q         Bad Bunny           10
4  UCFNosi99Sp0_eLilBiXmmXA        FutureVEVO           10

                                                tags   duration
definition  \
0  ['jaiden', 'animations', 'jaidenanimation', 'j...     PT5M2S
hd
1  ['mlb', 'baseball', 'sports', 'mlb highlights'...    PT5M58S
```

```
hd
2  ['sidemen', 'moresidemen', 'miniminter', 'ksi'...  PT1H7M20S
hd
3  ['Bad', 'Bunny', 'Bad Bunny', 'YHLQMDLG', 'EUT...        PT4M
hd
4  ['future', 'metro boomin', 'like that', 'type ...    PT3M46S
hd

   caption  view_count  like_count  dislike_count  favorite_count  \
0    False      625570       84047              0               0
1    False     2127139       51527              0               0
2    False     3242856      167288              0               0
3    False     1476428      195765              0               0
4    False      580421       41664              0               0

   comment_count
0           4139
1           5018
2           7757
3          14720
4           2223
```

missing values and data types

```
# check for missing values
missing_values = trending_videos.isnull().sum()

# display data types
data_types = trending_videos.dtypes

missing_values, data_types

(video_id          0
 title             0
 description       1
 published_at      0
 channel_id        0
 channel_title     0
 category_id       0
 tags              0
 duration          0
 definition        0
 caption           0
 view_count        0
 like_count        0
 dislike_count     0
 favorite_count    0
 comment_count     0
 dtype: int64,
 video_id          object
 title                 object
 description           object
 published_at          object
 channel_id            object
 channel_title         object
 category_id            int64
 tags                  object
 duration              object
 definition            object
 caption                 bool
 view_count             int64
 like_count             int64
 dislike_count          int64
 favorite_count         int64
 comment_count          int64
 dtype: object)
```

```
# fill missing descriptions with "No description"
trending_videos['description'].fillna('No description', inplace=True)

# convert `published_at` to datetime
trending_videos['published_at'] =
pd.to_datetime(trending_videos['published_at'])

# convert tags from string representation of list to actual list
trending_videos['tags'] = trending_videos['tags'].apply(lambda x:
eval(x) if isinstance(x, str) else x)

# descriptive statistics
descriptive_stats = trending_videos[['view_count', 'like_count',
'dislike_count', 'comment_count']].describe()

descriptive_stats
```

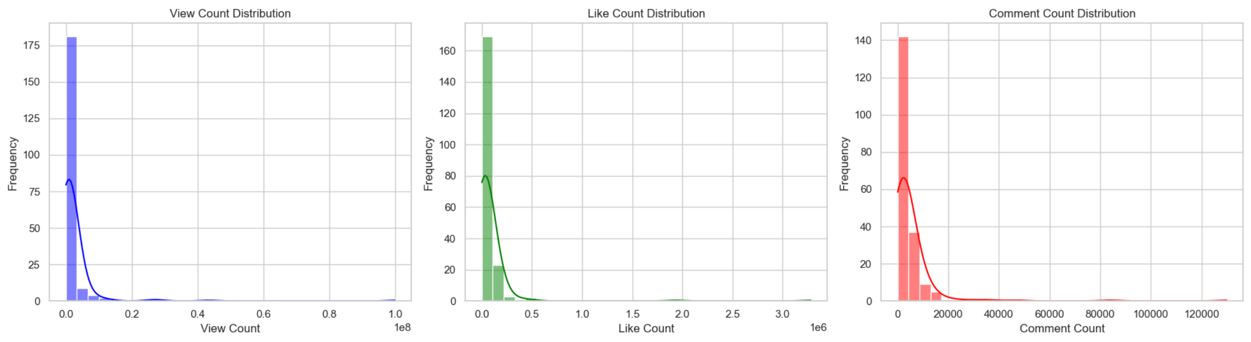
```
         view_count    like_count  dislike_count  comment_count
count  2.000000e+02  2.000000e+02          200.0     200.000000
mean   2.294847e+06  7.707470e+04            0.0    5041.545000
std    8.192936e+06  2.730284e+05            0.0   11987.030014
min    1.136000e+04  0.000000e+00            0.0       0.000000
25%    3.160382e+05  9.636750e+03            0.0     914.750000
50%    6.108145e+05  2.664450e+04            0.0    2147.000000
75%    1.624882e+06  6.223675e+04            0.0    4924.000000
max    9.998604e+07  3.279392e+06            0.0  129931.000000
```

distribution of views, likes and comments of all the videos

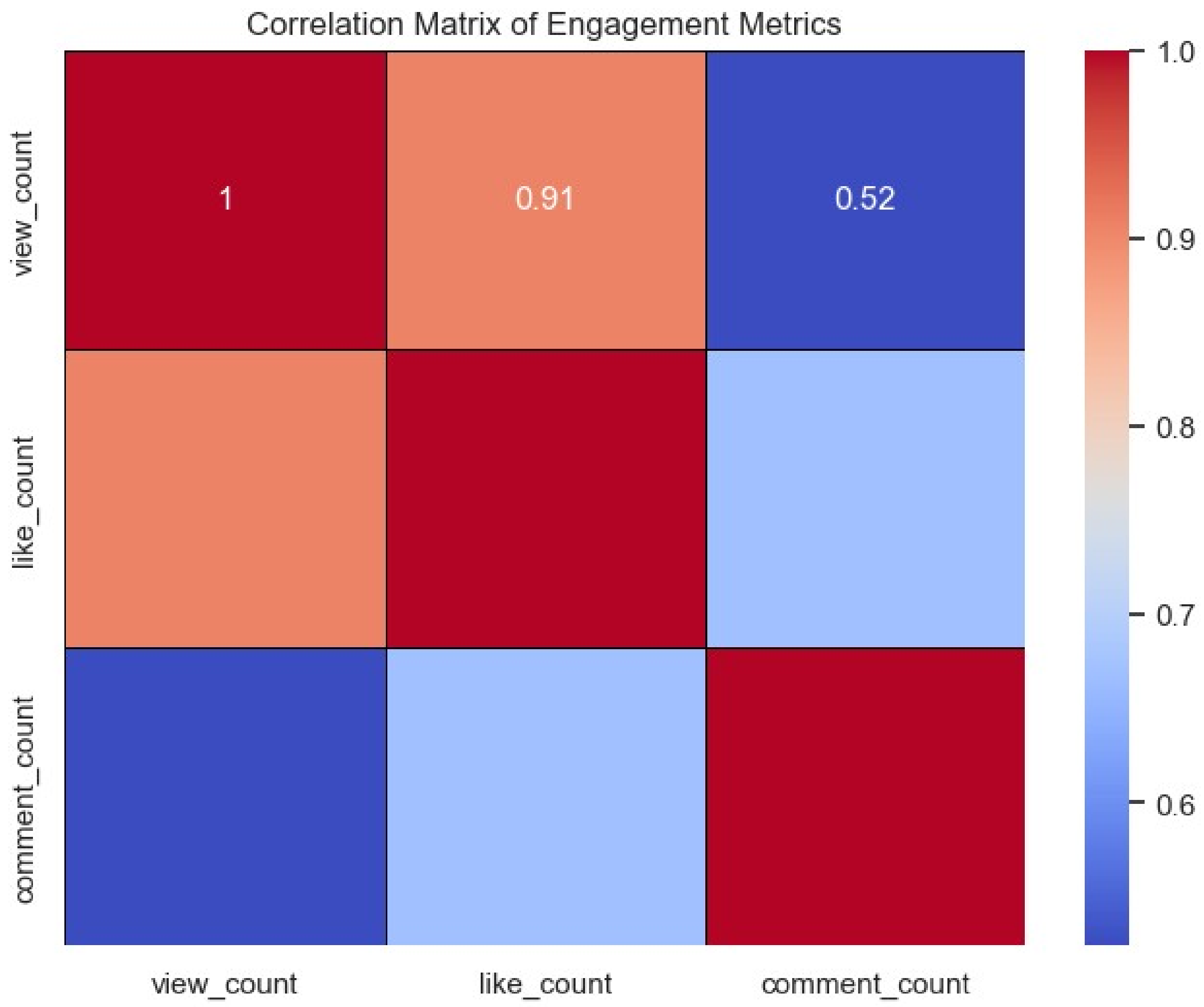
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# view count distribution
sns.histplot(trending_videos['view_count'], bins=30, kde=True,
ax=axes[0], color='blue')
axes[0].set_title('View Count Distribution')
axes[0].set_xlabel('View Count')
axes[0].set_ylabel('Frequency')

# like count distribution
sns.histplot(trending_videos['like_count'], bins=30, kde=True,
ax=axes[1], color='green')
axes[1].set_title('Like Count Distribution')
axes[1].set_xlabel('Like Count')
axes[1].set_ylabel('Frequency')

# comment count distribution
sns.histplot(trending_videos['comment_count'], bins=30, kde=True,
ax=axes[2], color='red')
axes[2].set_title('Comment Count Distribution')
axes[2].set_xlabel('Comment Count')
axes[2].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```

```
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
# correlation matrix
correlation_matrix = trending_videos[['view_count', 'like_count',
'comment_count']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5, linecolor='black')
plt.title('Correlation Matrix of Engagement Metrics')
plt.show()
```



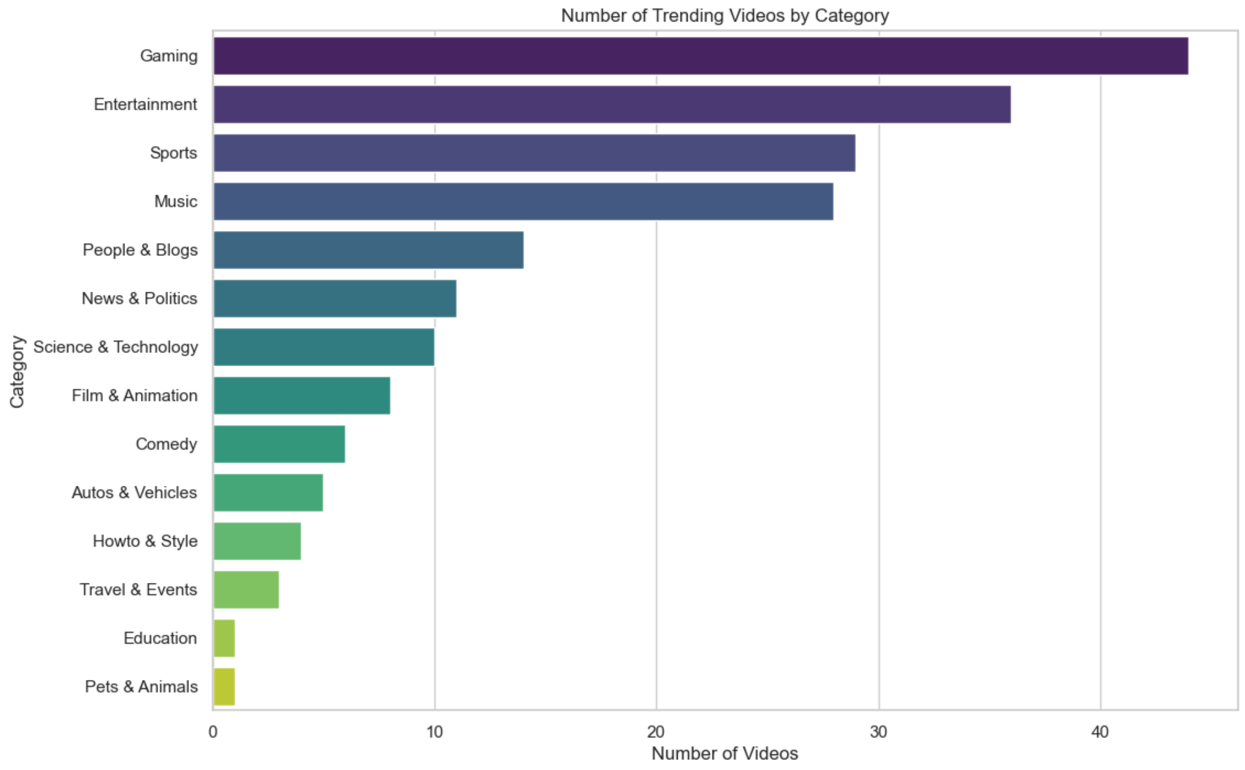
```python
from googleapiclient.discovery import build

API_KEY = 'AIzaSyBfLPsSeHdvht-KiW9eZo6xFVkvYGpUQtE'
youtube = build('youtube', 'v3', developerKey=API_KEY)

def get_category_mapping():
    request = youtube.videoCategories().list(
        part='snippet',
        regionCode='US'
    )
    response = request.execute()
    category_mapping = {}
    for item in response['items']:
        category_id = int(item['id'])
        category_name = item['snippet']['title']
        category_mapping[category_id] = category_name
    return category_mapping

# get the category mapping
```

```
category_mapping = get_category_mapping()
print(category_mapping)
```

```
{1: 'Film & Animation', 2: 'Autos & Vehicles', 10: 'Music', 15: 'Pets
& Animals', 17: 'Sports', 18: 'Short Movies', 19: 'Travel & Events',
20: 'Gaming', 21: 'Videoblogging', 22: 'People & Blogs', 23: 'Comedy',
24: 'Entertainment', 25: 'News & Politics', 26: 'Howto & Style', 27:
'Education', 28: 'Science & Technology', 29: 'Nonprofits & Activism',
30: 'Movies', 31: 'Anime/Animation', 32: 'Action/Adventure', 33:
'Classics', 34: 'Comedy', 35: 'Documentary', 36: 'Drama', 37:
'Family', 38: 'Foreign', 39: 'Horror', 40: 'Sci-Fi/Fantasy', 41:
'Thriller', 42: 'Shorts', 43: 'Shows', 44: 'Trailers'}
```

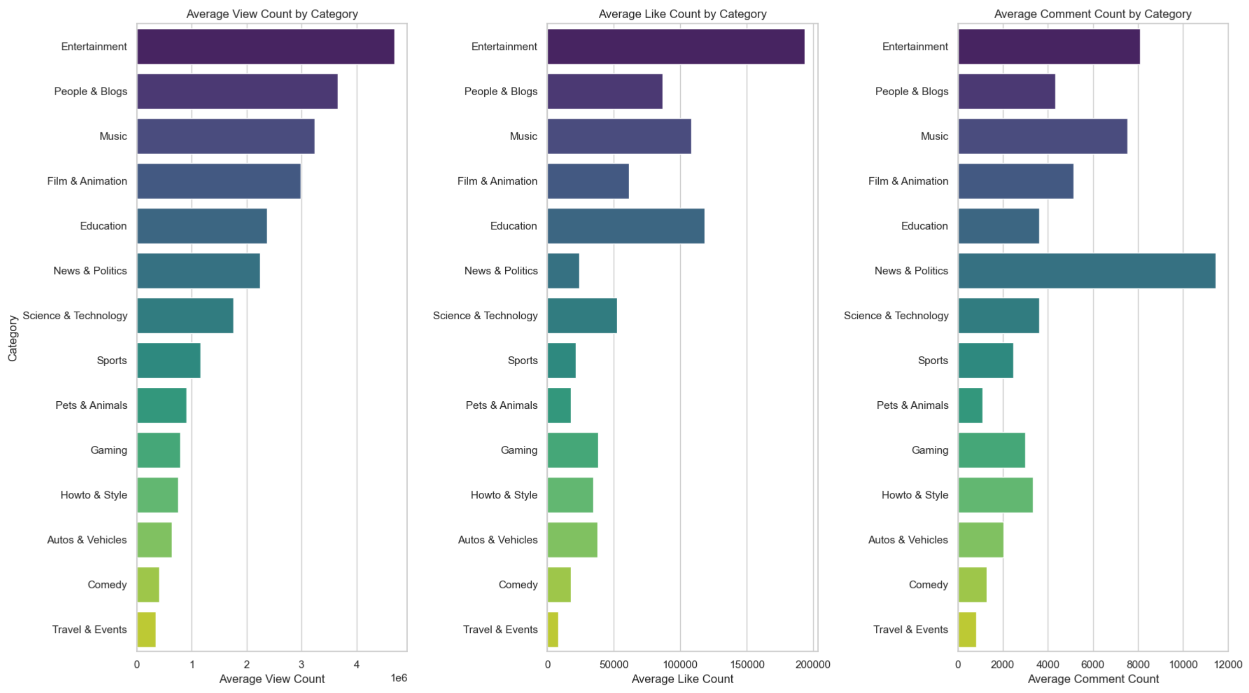
number of trending videos by category

```
trending_videos['category_name'] =
trending_videos['category_id'].map(category_mapping)

# Bar chart for category counts
plt.figure(figsize=(12, 8))
sns.countplot(y=trending_videos['category_name'],
order=trending_videos['category_name'].value_counts().index,
palette='viridis')
plt.title('Number of Trending Videos by Category')
plt.xlabel('Number of Videos')
plt.ylabel('Category')
plt.show()
```



Gaming, Entertainment, Sports, and Music categories have the highest number of trending videos

```
# average engagement metrics by category
category_engagement = trending_videos.groupby('category_name')
[['view_count', 'like_count',
'comment_count']].mean().sort_values(by='view_count', ascending=False)

fig, axes = plt.subplots(1, 3, figsize=(18, 10))

# view count by category
sns.barplot(y=category_engagement.index,
x=category_engagement['view_count'], ax=axes[0], palette='viridis')
axes[0].set_title('Average View Count by Category')
axes[0].set_xlabel('Average View Count')
axes[0].set_ylabel('Category')

# like count by category
sns.barplot(y=category_engagement.index,
x=category_engagement['like_count'], ax=axes[1], palette='viridis')
axes[1].set_title('Average Like Count by Category')
axes[1].set_xlabel('Average Like Count')
axes[1].set_ylabel('')

# comment count by category
sns.barplot(y=category_engagement.index,
x=category_engagement['comment_count'], ax=axes[2], palette='viridis')
axes[2].set_title('Average Comment Count by Category')
axes[2].set_xlabel('Average Comment Count')
axes[2].set_ylabel('')

plt.tight_layout()
plt.show()
```



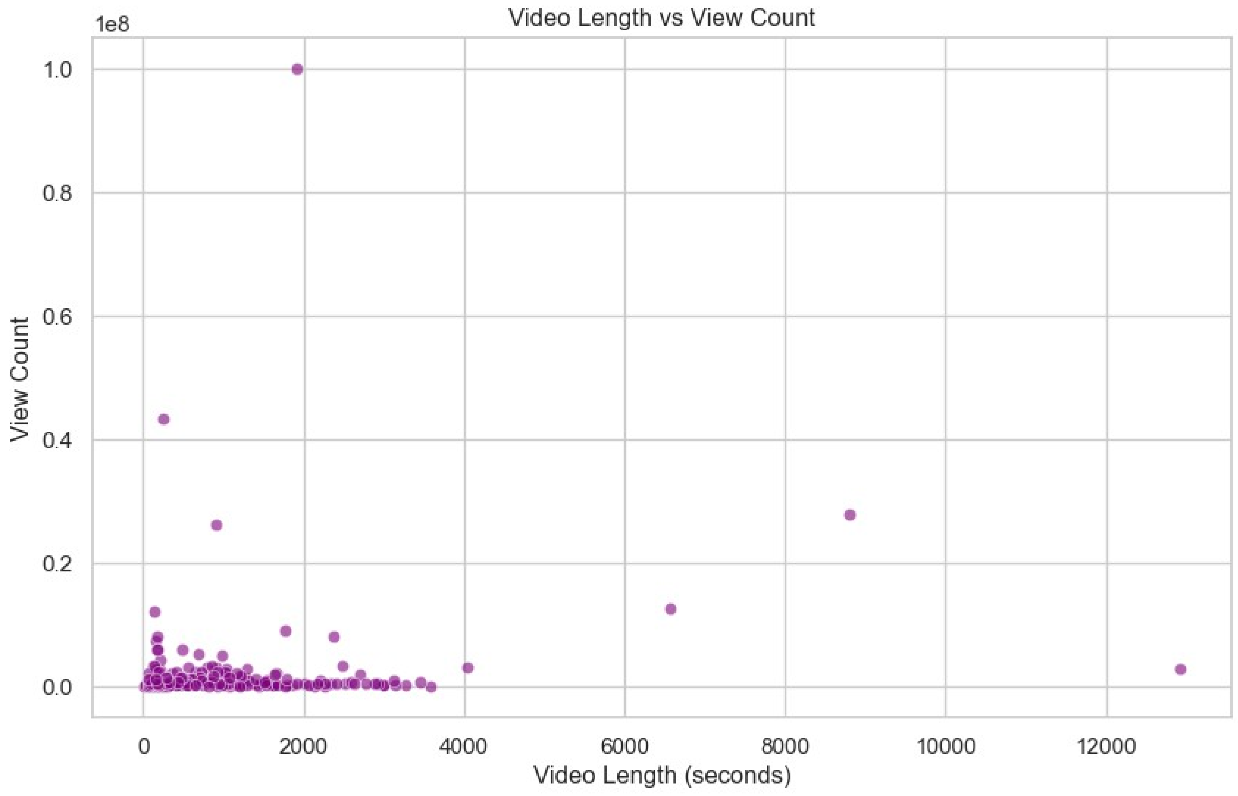
we are using the isodate library to convert the duration of each video from the ISO 8601 format to seconds, which allows for numerical analysis. After converting the durations, we are categorizing the videos into different duration ranges (0-5 minutes, 5-10 minutes, 10-20 minutes, 20-60 minutes, and 60-120 minutes) by creating a new column called duration_range. This categorization enables us to analyze and compare the engagement metrics of videos within specific length intervals, providing insights into how video length influences viewer behaviour and video performance.

```
!pip install isodate
import isodate

# convert ISO 8601 duration to seconds
trending_videos['duration_seconds'] =
trending_videos['duration'].apply(lambda x:
isodate.parse_duration(x).total_seconds())

trending_videos['duration_range'] =
pd.cut(trending_videos['duration_seconds'], bins=[0, 300, 600, 1200,
3600, 7200], labels=['0-5 min', '5-10 min', '10-20 min', '20-60 min',
'60-120 min'])
```

```
Collecting isodate
  Downloading isodate-0.6.1-py2.py3-none-any.whl.metadata (9.6 kB)
Requirement already satisfied: six in c:\users\prasanna kumar\
anaconda3\lib\site-packages (from isodate) (1.16.0)
Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
   ---------------------------------------- 0.0/41.7 kB ? eta -:--:--
   ---------------------------------------- 0.0/41.7 kB ? eta -:--:--
   ----------------------------- ---------- 30.7/41.7 kB 660.6 kB/s
```

```
eta 0:00:01
   ---------------------------------------- 41.7/41.7 kB 507.8 kB/s
eta 0:00:00
Installing collected packages: isodate
Successfully installed isodate-0.6.1
```

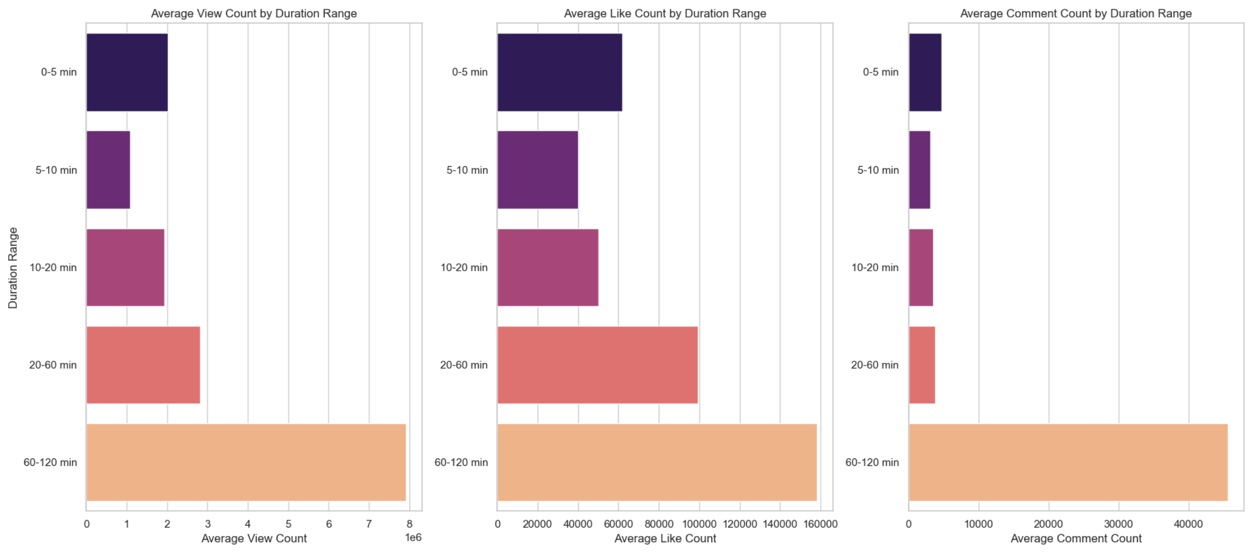
```
# scatter plot for video length vs view count
plt.figure(figsize=(10, 6))
sns.scatterplot(x='duration_seconds', y='view_count',
data=trending_videos, alpha=0.6, color='purple')
plt.title('Video Length vs View Count')
plt.xlabel('Video Length (seconds)')
plt.ylabel('View Count')
plt.show()
```



```
# bar chart for engagement metrics by duration range
length_engagement = trending_videos.groupby('duration_range')
[['view_count', 'like_count', 'comment_count']].mean()

fig, axes = plt.subplots(1, 3, figsize=(18, 8))

# view count by duration range
sns.barplot(y=length_engagement.index,
x=length_engagement['view_count'], ax=axes[0], palette='magma')
axes[0].set_title('Average View Count by Duration Range')
```

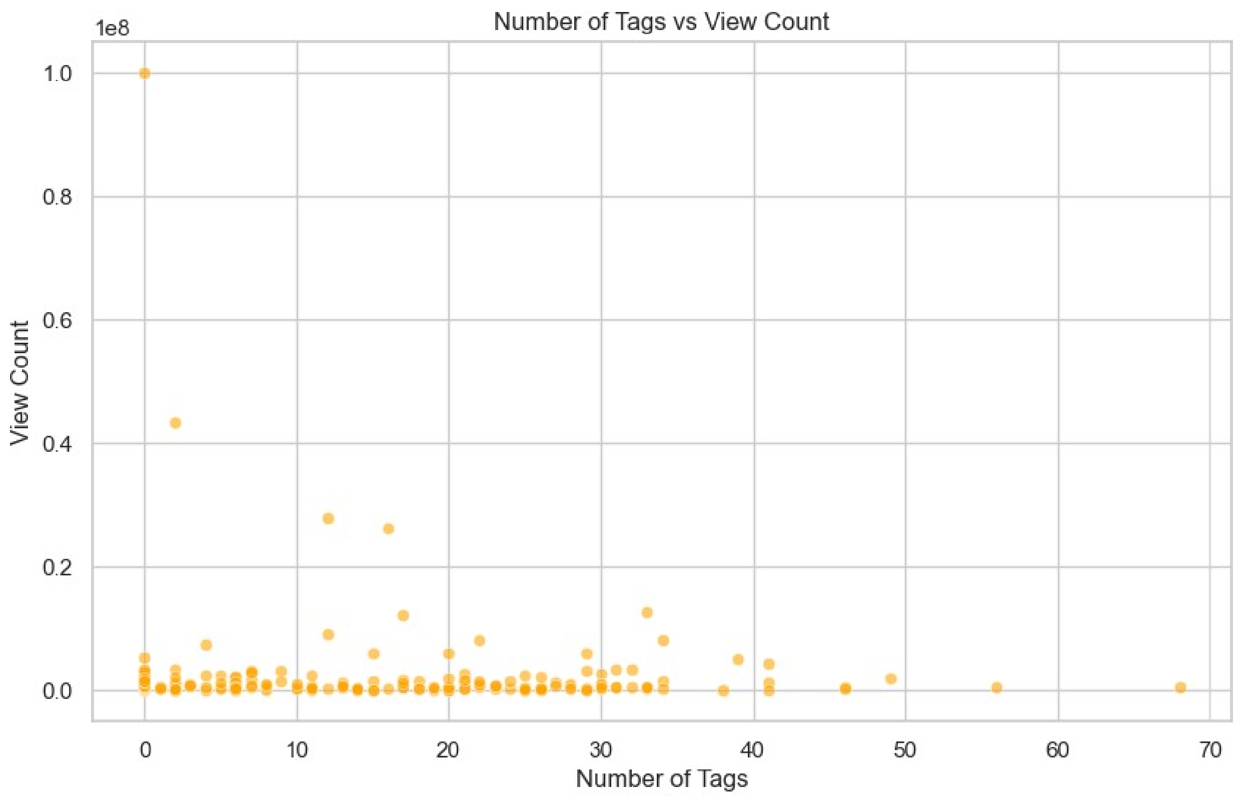
```
axes[0].set_xlabel('Average View Count')
axes[0].set_ylabel('Duration Range')

# like count by duration range
sns.barplot(y=length_engagement.index,
x=length_engagement['like_count'], ax=axes[1], palette='magma')
axes[1].set_title('Average Like Count by Duration Range')
axes[1].set_xlabel('Average Like Count')
axes[1].set_ylabel('')

# comment count by duration range
sns.barplot(y=length_engagement.index,
x=length_engagement['comment_count'], ax=axes[2], palette='magma')
axes[2].set_title('Average Comment Count by Duration Range')
axes[2].set_xlabel('Average Comment Count')
axes[2].set_ylabel('')

plt.tight_layout()
plt.show()
```

```
C:\Users\PRASANNA KUMAR\AppData\Local\Temp\
ipykernel_14424\865213030.py:2: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  length_engagement = trending_videos.groupby('duration_range')
[['view_count', 'like_count', 'comment_count']].mean()
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
categorical.py:641: FutureWarning: The default of observed=False is
deprecated and will be changed to True in a future version of pandas.
Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
categorical.py:641: FutureWarning: The default of observed=False is
deprecated and will be changed to True in a future version of pandas.
Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
C:\Users\PRASANNA KUMAR\anaconda3\Lib\site-packages\seaborn\
categorical.py:641: FutureWarning: The default of observed=False is
deprecated and will be changed to True in a future version of pandas.
Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



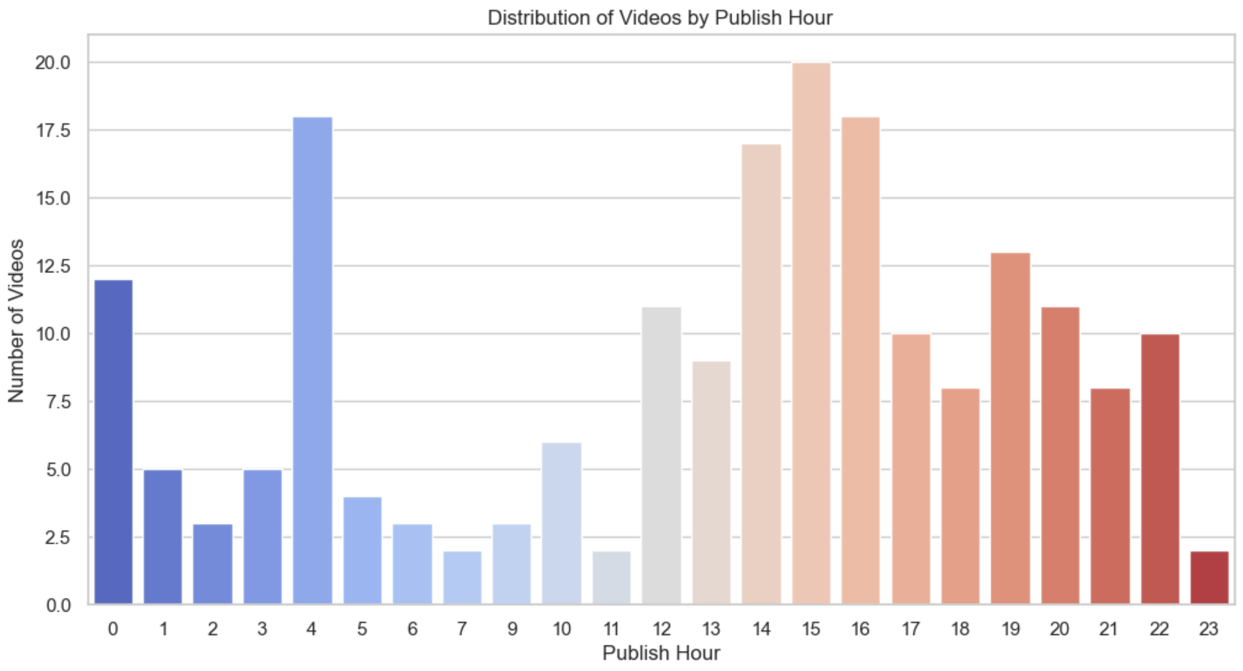
relationship between views and number of tags used in the video

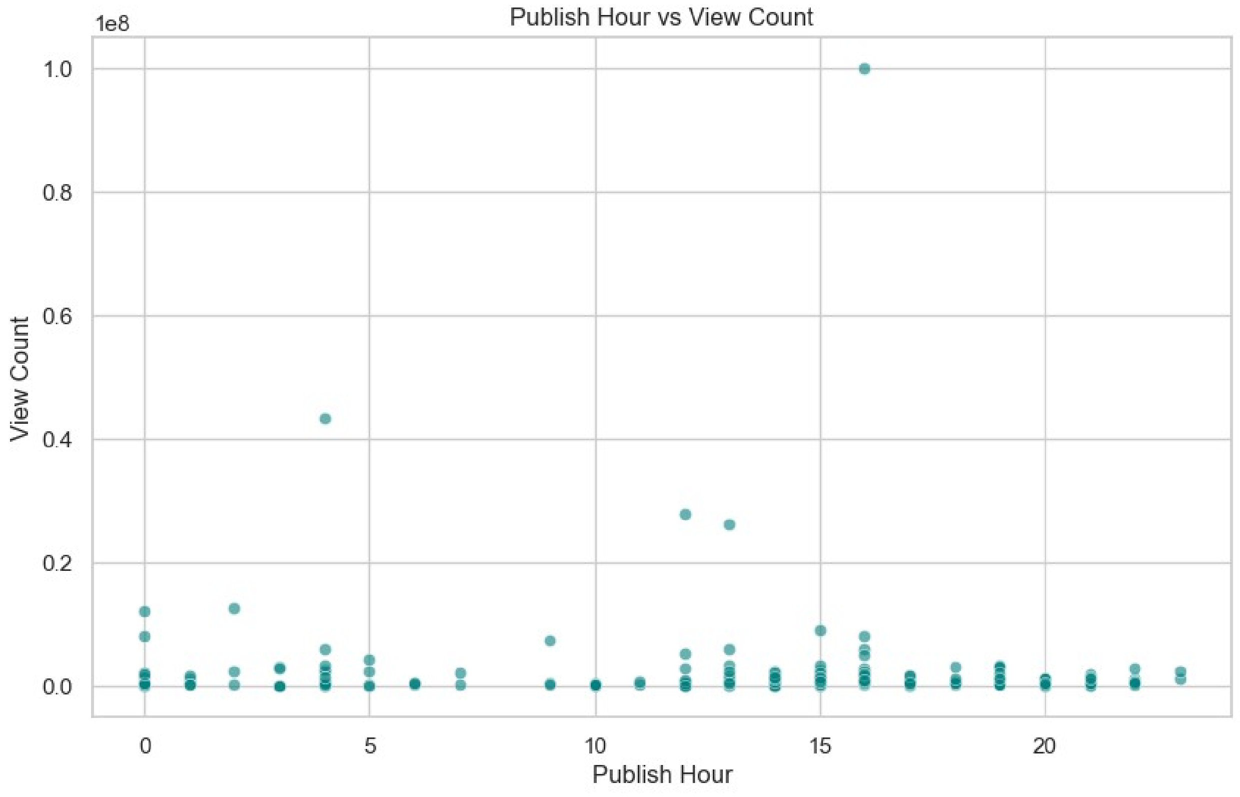
```
# calculate the number of tags for each video
trending_videos['tag_count'] = trending_videos['tags'].apply(len)

# scatter plot for number of tags vs view count
plt.figure(figsize=(10, 6))
sns.scatterplot(x='tag_count', y='view_count', data=trending_videos,
alpha=0.6, color='orange')
plt.title('Number of Tags vs View Count')
plt.xlabel('Number of Tags')
plt.ylabel('View Count')
plt.show()
```



impact of the time a video is posted on its views

```
# extract hour of publication
trending_videos['publish_hour'] =
trending_videos['published_at'].dt.hour

# bar chart for publish hour distribution
plt.figure(figsize=(12, 6))
sns.countplot(x='publish_hour', data=trending_videos,
palette='coolwarm')
plt.title('Distribution of Videos by Publish Hour')
plt.xlabel('Publish Hour')
plt.ylabel('Number of Videos')
plt.show()

# scatter plot for publish hour vs view count
plt.figure(figsize=(10, 6))
sns.scatterplot(x='publish_hour', y='view_count',
data=trending_videos, alpha=0.6, color='teal')
plt.title('Publish Hour vs View Count')
plt.xlabel('Publish Hour')
plt.ylabel('View Count')
plt.show()
```


Distribution of Videos by Publish Hour
Number of Videos
Publish Hour
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23


Publish Hour vs View Count
1e8
View Count
Publish Hour
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20