# Yt_analysis

September 2, 2024

[17]:
```python
import pandas as pd
from googleapiclient.discovery import build

# replace with your own API key
API_KEY = 'Your API KEY'

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
```

[18]:
```
import pandas as pd

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

```
      video_id                                              title  \
0  UPQXWEGdrWc                          DRESS TO IMPRESS IN REAL LIFE
1  j_TvWRS2_Hw  50 CENT: MILLION DOLLAZ WORTH OF GAME EPISODE 289
2  fZSj_M2ay6s  USC Trojans vs. LSU Tigers | Full Game Highlig…
3  IZ4HOCld5nY                                   AI is here. What now?
4  3-NrCcr8Bcg  HELLUVA SHORTS 3 // MISSION: WEEABOO-BOO // HE…

                                         description          published_at  \
0  COMMENT DOWN WHO Y'ALL THINK WON & THUMBS UP F…  2024-09-01T21:27:31Z
1  50 CENT invited Million Dollaz Worth of Game d…  2024-09-01T23:30:07Z
2  Check out these highlights as the No. 23 USC T…  2024-09-02T03:26:05Z
3  Get 50% off your first order of CookUnity meal…  2024-09-01T15:36:53Z
4  IMP MISSION: WEEABOO-BOO\nWARNING: For cringe\…  2024-08-31T17:00:03Z

                  channel_id                 channel_title  category_id  \
0  UCt_DaLB_NDqPVxezyvcfRtg                        LARRAY           22
1  UC16Ne7V6Fe_bp1omKLiymFg  MILLION DOLLAZ WORTH OF GAME           24
2  UCzRWWsFjqHk1an4OnVPsl9g         ESPN College Football           17
3  UCuo9VyowIT-1jA5G2ZuC6Yw                  Eddy Burback           23
4  UCzfyYtgvkx5mLy8nlLlayYg                     Vivziepop            1

                                                tags  duration definition  \
0  ['LARRAY', 'DRESS TO IMPRESS', 'QUENLIN BLACKW…  PT17M41S         hd
1  ['Wallo', 'Wallo 267', 'Gillie', 'Gillie Da Ki…  PT58M58S         hd
2  ['college football espn', 'espn college footba…  PT18M31S         hd
3                                                 []  PT46M16S         hd
4                         ['Vivziepop', 'Zoophobia']   PT4M40S         hd

  caption  view_count  like_count  dislike_count  favorite_count  \
0   False      692774       74141              0               0
1   False      471303       17701              0               0
2   False      534498        8217              0               0
3   False      853182       61515              0               0
4    True     4022717      444594              0               0

  comment_count
0          3625
1          2563
2          1665
3          5683
4         37339
```

[19]:
```python
# check for missing values
missing_values = trending_videos.isnull().sum()

# display data types
data_types = trending_videos.dtypes

missing_values, data_types
```

[19]:
```
(video_id          0
 title             0
 description       6
 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)
```

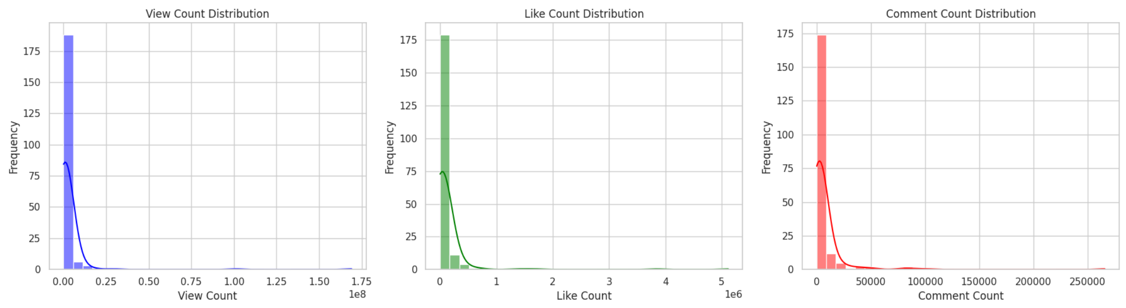
[ ]:
```
# fill missing descriptions with "No description"
trending_videos['description'].fillna('No description', inplace=True)

# 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   2.939143e+06  1.178149e+05            0.0    6915.560000
std    1.407951e+07  4.783872e+05            0.0   22333.334868
min    5.926900e+04  0.000000e+00            0.0       0.000000
25%    3.856502e+05  1.007775e+04            0.0     975.250000
50%    6.854300e+05  2.778300e+04            0.0    2004.500000
```

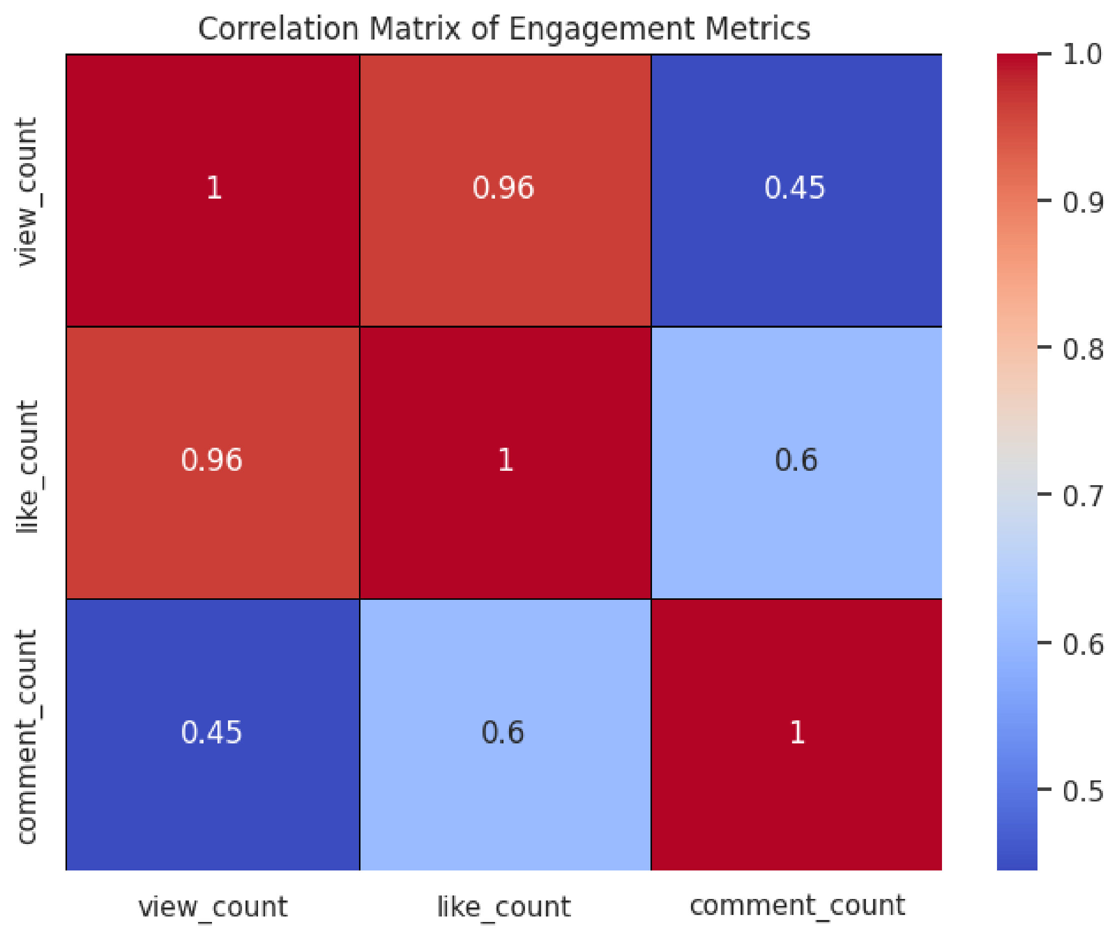
```
75%    1.500596e+06  6.270925e+04            0.0    4481.750000
max    1.691344e+08  5.116290e+06            0.0  265807.000000
```

[21]:
```
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()
```



[22]:
```
# 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()
```

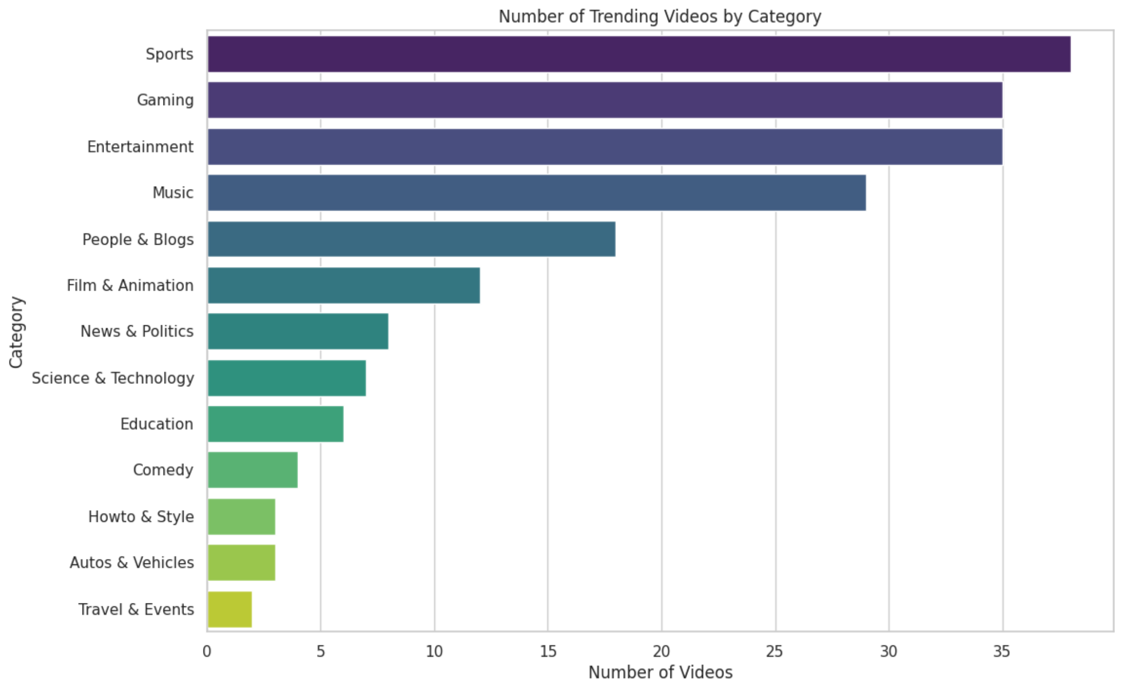


[23]:
```
from googleapiclient.discovery import build

API_KEY = 'YOUR API KEY'
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'}
```

[ ]:
```
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()
```

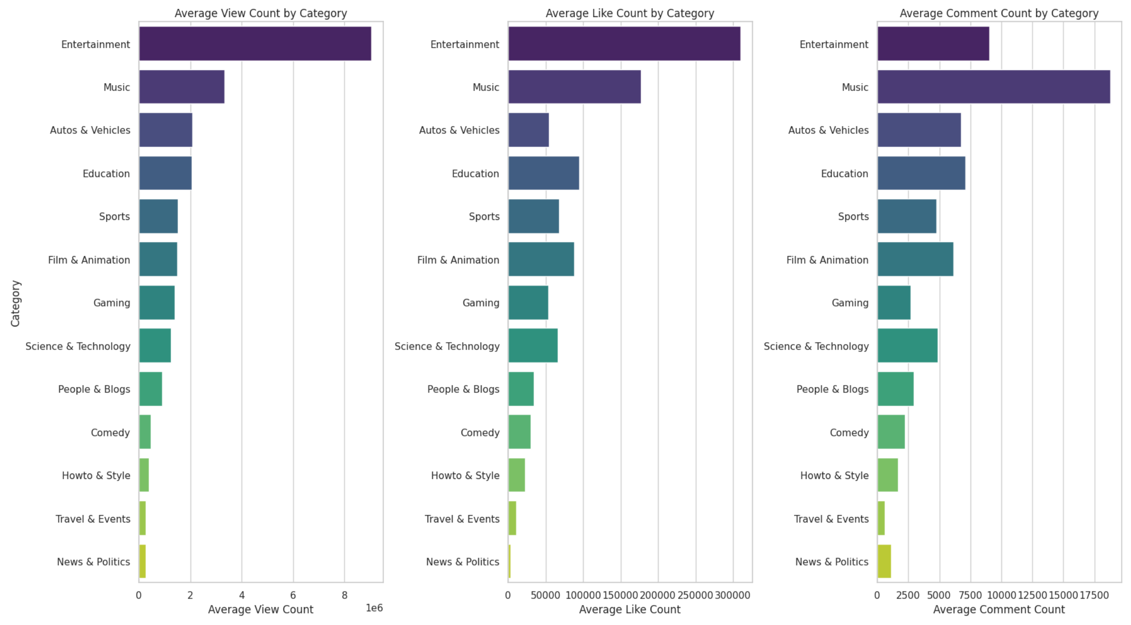
```
<ipython-input-10-20e3e73c616f>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(y=trending_videos['category_name'],
order=trending_videos['category_name'].value_counts().index, palette='viridis')
```



[ ]:
```
# 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()
```

```
<ipython-input-11-6df855744d52>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_engagement.index, x=category_engagement['view_count'],
ax=axes[0], palette='viridis')
<ipython-input-11-6df855744d52>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_engagement.index, x=category_engagement['like_count'],
ax=axes[1], palette='viridis')
<ipython-input-11-6df855744d52>:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_engagement.index,
x=category_engagement['comment_count'], ax=axes[2], palette='viridis')
```

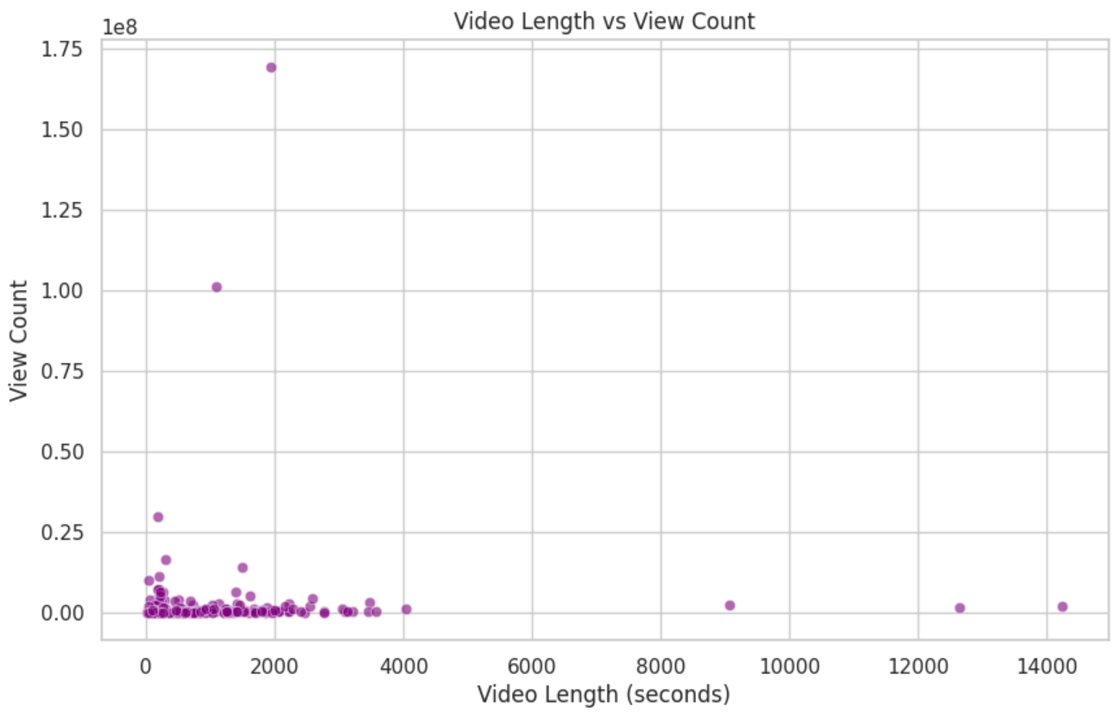


```
[ ]: !pip install isodate
     import isodate

     # 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 /usr/local/lib/python3.10/dist-packages
(from isodate) (1.16.0)
Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
                         41.7/41.7 kB
655.5 kB/s eta 0:00:00
Installing collected packages: isodate
Successfully installed isodate-0.6.1
```

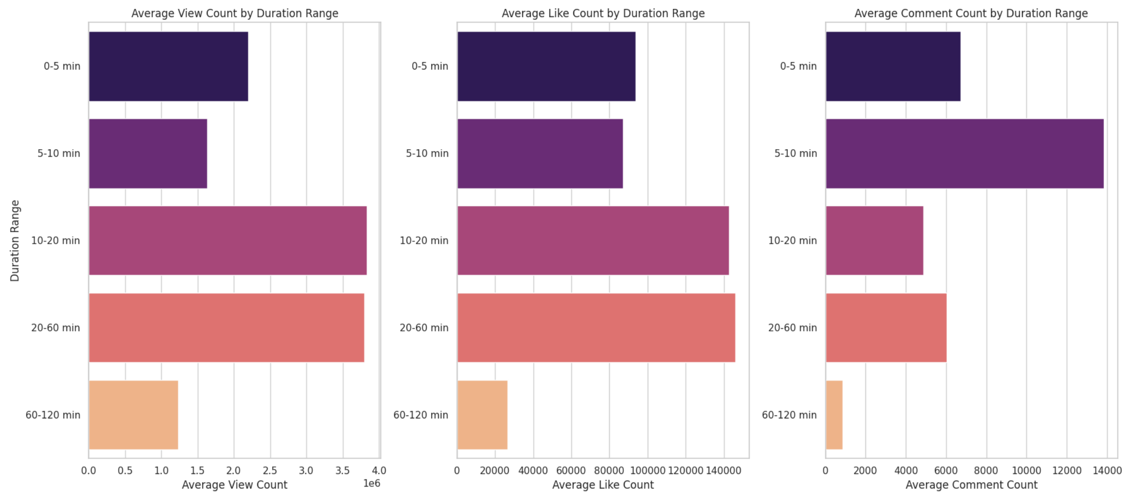
```
[ ]: # scatter plot for video length vs view count
     plt.figure(figsize=(10, 6))
     sns.scatterplot(x='duration_seconds', y='view_count', data=trending_videos,␣
       ↪alpha=0.6, color='purple')
     plt.title('Video Length vs View Count')
     plt.xlabel('Video Length (seconds)')
```

```python
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()
```

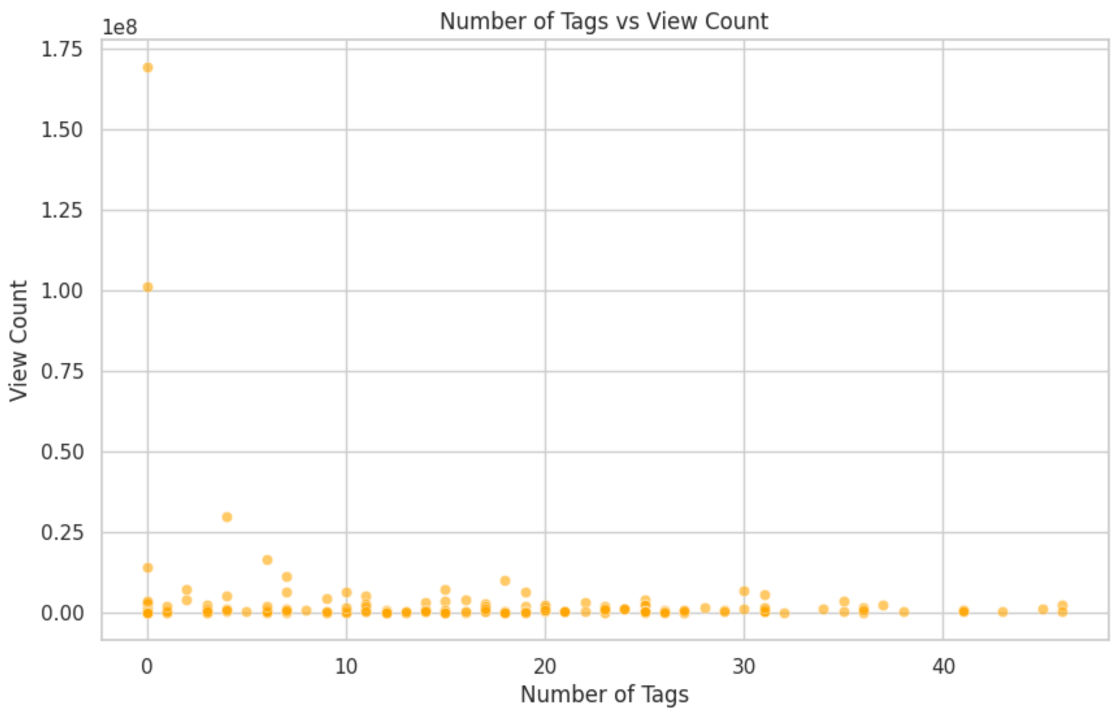


```
<ipython-input-14-071672337da8>:10: FutureWarning: The default of observed=False
is deprecated and will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to adopt the future
default and silence this warning.
  length_engagement = trending_videos.groupby('duration_range')[['view_count',
'like_count', 'comment_count']].mean()
<ipython-input-14-071672337da8>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=length_engagement.index, x=length_engagement['view_count'],
ax=axes[0], palette='magma')
<ipython-input-14-071672337da8>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=length_engagement.index, x=length_engagement['like_count'],
ax=axes[1], palette='magma')
<ipython-input-14-071672337da8>:27: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=length_engagement.index, x=length_engagement['comment_count'],
ax=axes[2], palette='magma')
```



[ ]:
```python
# 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()
```

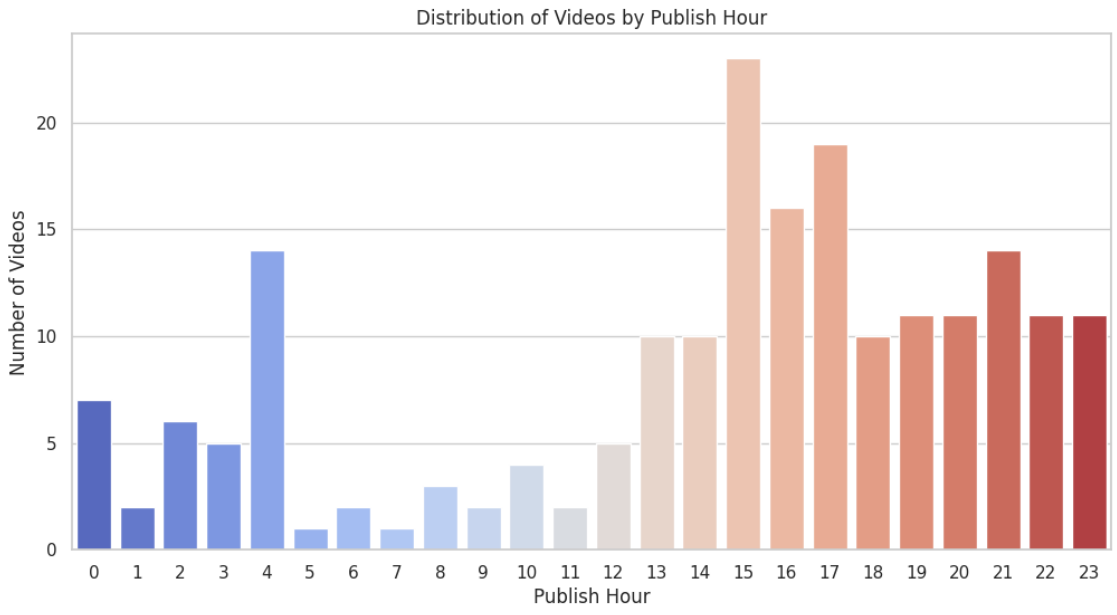


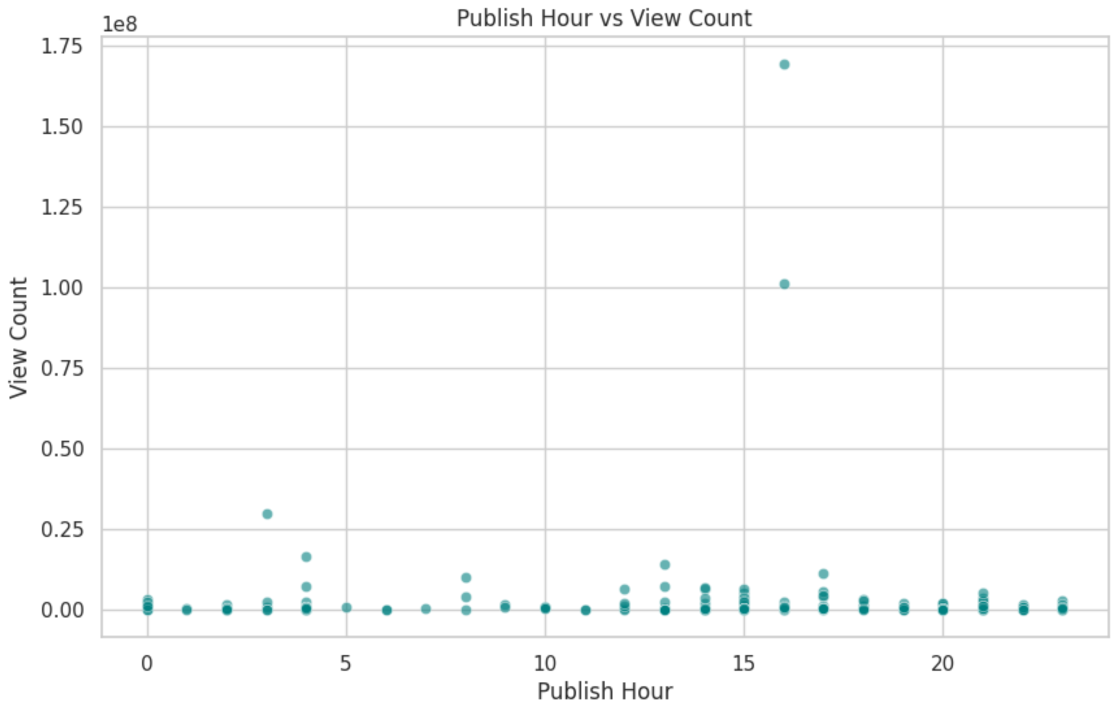
```
[ ]: # extract hour of publication
trending_videos['publish_hour'] = trending_videos['published_at'].dt.hour

# 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()
```

```
<ipython-input-16-ccbdff83c60f>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
```

effect.

```
sns.countplot(x='publish_hour', data=trending_videos, palette='coolwarm')
```


Distribution of Videos by Publish Hour
Number of Videos
Publish Hour


Publish Hour vs View Count
1e8
View Count
Publish Hour