

# ytchannel

May 11, 2025

```
[2]: pip install google-api-python-client pandas matplotlib seaborn textblob tqdm
```

```
Requirement already satisfied: google-api-python-client in  
c:\users\tvino\downloads\anaconda\lib\site-packages (2.169.0)  
Requirement already satisfied: pandas in  
c:\users\tvino\downloads\anaconda\lib\site-packages (2.2.2)  
Requirement already satisfied: matplotlib in  
c:\users\tvino\downloads\anaconda\lib\site-packages (3.9.2)  
Requirement already satisfied: seaborn in  
c:\users\tvino\downloads\anaconda\lib\site-packages (0.13.2)  
Requirement already satisfied: textblob in  
c:\users\tvino\downloads\anaconda\lib\site-packages (0.19.0)  
Requirement already satisfied: tqdm in  
c:\users\tvino\downloads\anaconda\lib\site-packages (4.66.5)  
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-python-  
client) (0.22.0)  
Requirement already satisfied: google-auth!=2.24.0,!>=2.25.0,<3.0.0,>=1.32.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-python-  
client) (2.40.1)  
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-python-  
client) (0.2.0)  
Requirement already satisfied: google-api-  
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-python-  
client) (2.24.2)  
Requirement already satisfied: uritemplate<5,>=3.0.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-python-  
client) (4.1.1)  
Requirement already satisfied: numpy>=1.26.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from pandas) (1.26.4)  
Requirement already satisfied: python-dateutil>=2.8.2 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from pandas) (2.9.0.post0)  
Requirement already satisfied: pytz>=2020.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from pandas) (2024.1)  
Requirement already satisfied: tzdata>=2022.7 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from pandas) (2023.3)
```

Requirement already satisfied: contourpy>=1.0.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (1.2.0)

Requirement already satisfied: cycycler>=0.10 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (4.51.0)

Requirement already satisfied: kiwisolver>=1.3.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (1.4.4)

Requirement already satisfied: packaging>=20.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (24.1)

Requirement already satisfied: pillow>=8 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (10.4.0)

Requirement already satisfied: pyparsing>=2.3.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from matplotlib) (3.1.2)

Requirement already satisfied: nltk>=3.9 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from textblob) (3.9.1)

Requirement already satisfied: colorama in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from tqdm) (0.4.6)

Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-  
core!=2.0.\*,!=2.1.\*,!=2.2.\*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)  
(1.70.0)

Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.2  
1.2,!=4.21.3,!=4.21.4,!=4.21.5,<7.0.0,>=3.19.5 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-  
core!=2.0.\*,!=2.1.\*,!=2.2.\*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)  
(4.25.3)

Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-  
core!=2.0.\*,!=2.1.\*,!=2.2.\*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)  
(1.26.1)

Requirement already satisfied: requests<3.0.0,>=2.18.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-api-  
core!=2.0.\*,!=2.1.\*,!=2.2.\*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)  
(2.32.3)

Requirement already satisfied: cachetools<6.0,>=2.0.0 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-  
auth!=2.24.0,!=2.25.0,<3.0.0,>=1.32.0->google-api-python-client) (5.3.3)

Requirement already satisfied: pyasn1-modules>=0.2.1 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-  
auth!=2.24.0,!=2.25.0,<3.0.0,>=1.32.0->google-api-python-client) (0.2.8)

Requirement already satisfied: rsa<5,>=3.1.4 in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from google-  
auth!=2.24.0,!=2.25.0,<3.0.0,>=1.32.0->google-api-python-client) (4.9.1)

Requirement already satisfied: click in  
c:\users\tvino\downloads\anaconda\lib\site-packages (from nltk>=3.9->textblob)  
(8.1.7)

Requirement already satisfied: joblib in

```

c:\users\tvino\downloads\anaconda\lib\site-packages (from nltk>=3.9->textblob)
(1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from nltk>=3.9->textblob)
(2024.9.11)
Requirement already satisfied: six>=1.5 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from
pyasn1-modules>=0.2.1->google-auth!=2.24.0,!2.25.0,<3.0.0,>=1.32.0->google-api-
python-client) (0.4.8)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from
requests<3.0.0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from
requests<3.0.0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from
requests<3.0.0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\tvino\downloads\anaconda\lib\site-packages (from
requests<3.0.0,>=2.18.0->google-api-
core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client)
(2024.8.30)
Note: you may need to restart the kernel to use updated packages.

```

```

[26]: # Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob
from googleapiclient.discovery import build
from tqdm import tqdm

sns.set(style='whitegrid')
%matplotlib inline
# Replace with your own YouTube API Key
API_KEY = 'AIzaSyC1Yhq66YmNp-v4VCgI96gJf_-dR1NgCgY'
youtube = build('youtube', 'v3', developerKey=API_KEY)

```

```

def fetch_trending(country_code, max_results=50):
    request = youtube.videos().list(
        part='snippet,statistics',
        chart='mostPopular',
        regionCode=country_code,
        maxResults=max_results
    )
    response = request.execute()

    videos = []
    for item in response.get('items', []):
        video = {
            'video_id': item['id'],
            'title': item['snippet']['title'],
            'channel_title': item['snippet']['channelTitle'],
            'category_id': item['snippet']['categoryId'],
            'publish_time': item['snippet']['publishedAt'],
            'views': int(item['statistics'].get('viewCount', 0)),
            'likes': int(item['statistics'].get('likeCount', 0)),
            'dislikes': int(item['statistics'].get('dislikeCount', 0)), #
            # might be deprecated
            'comment_count': int(item['statistics'].get('commentCount', 0)),
            'country': country_code
        }
        videos.append(video)
    return videos

countries = ['US', 'IN', 'PK', 'RU', 'FR', 'IT', 'DE']
all_videos = []

for country in tqdm(countries):
    videos = fetch_trending(country, max_results=50)
    all_videos.extend(videos)

df = pd.DataFrame(all_videos)
df['publish_time'] = pd.to_datetime(df['publish_time'])
df['publish_day'] = df['publish_time'].dt.day_name()
df['publish_hour'] = df['publish_time'].dt.hour
df['like_ratio'] = df['likes'] / (df['likes'] + df['dislikes'] + 1)
df['comment_ratio'] = df['comment_count'] / (df['views'] + 1)
df['title_sentiment'] = df['title'].apply(lambda x: TextBlob(x).sentiment.
    # polarity)

plt.figure(figsize=(10,6))
sns.barplot(x='publish_day', y='views', data=df, estimator='mean')
plt.title("Average Views by Day of Week")
plt.xticks(rotation=45)
plt.show()
plt.figure(figsize=(10,6))

```

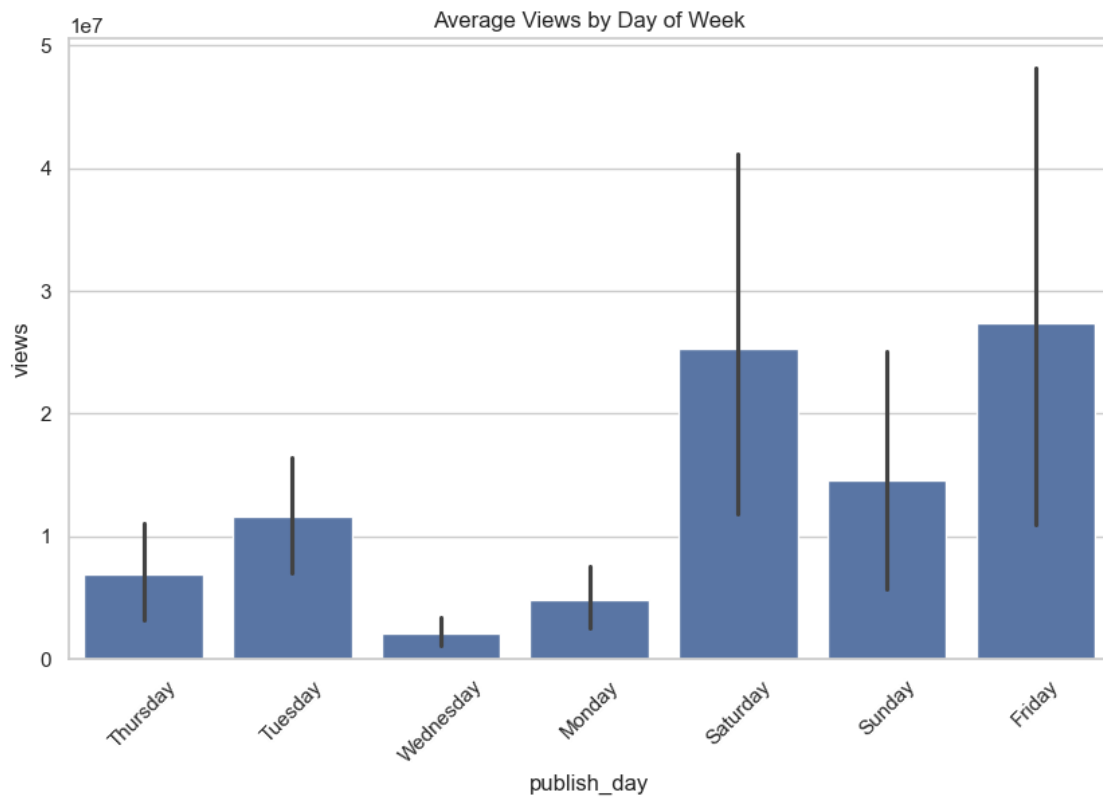
```

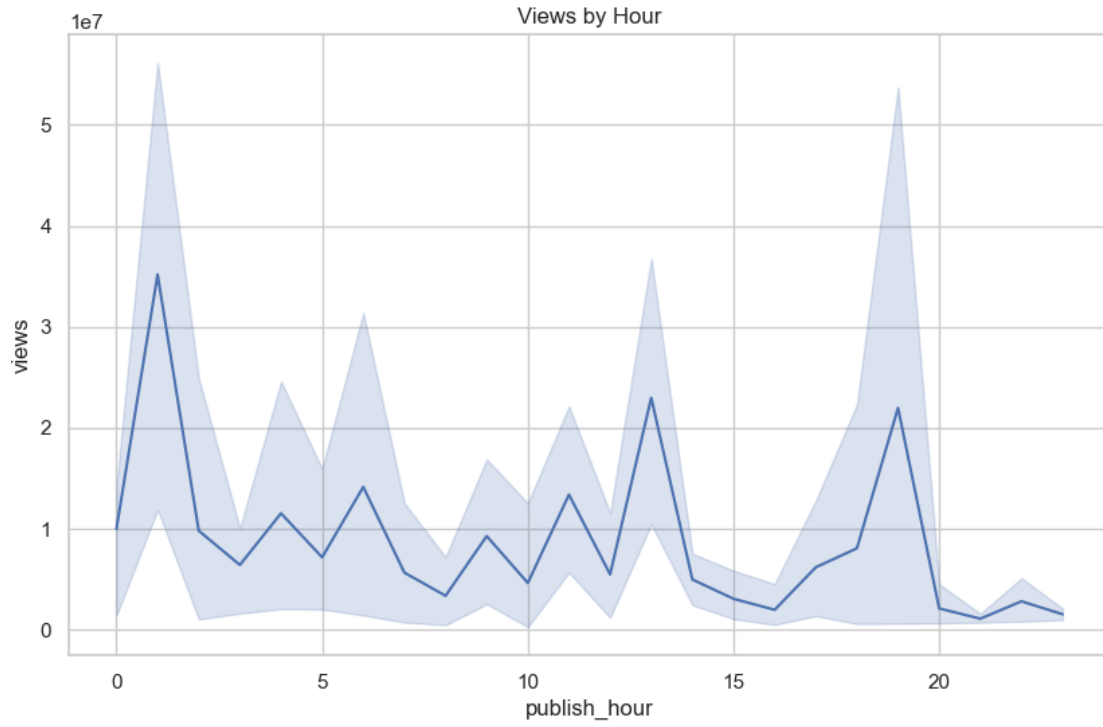
sns.lineplot(x='publish_hour', y='views', data=df)
plt.title("Views by Hour")
plt.grid(True)
plt.show()
corr = df[['views', 'likes', 'dislikes', 'comment_count', 'like_ratio', 'comment_ratio']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
df.to_csv('youtube_trending_live.csv', index=False)

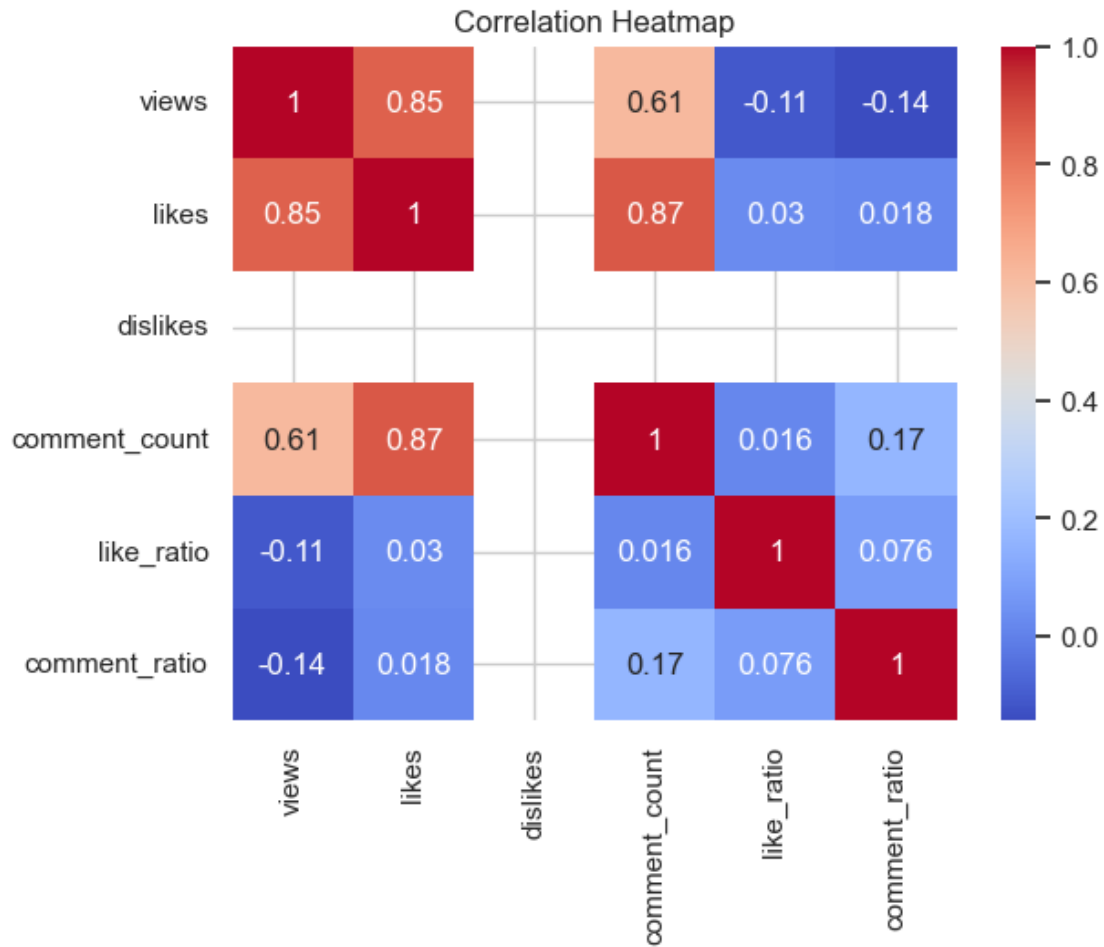
```

100%

| 7/7 [00:02<00:00, 3.11it/s]







```
[30]: # Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob
from googleapiclient.discovery import build
from tqdm import tqdm

sns.set(style='whitegrid')
%matplotlib inline

# Replace with your own YouTube API Key
API_KEY = 'AIzaSyC1Yhq66YmNp-v4VCgI96gJf_-dR1NgCgY'
youtube = build('youtube', 'v3', developerKey=API_KEY)

# Fetch trending videos function
def fetch_trending(country_code, max_results=50):
```

```

request = youtube.videos().list(
    part='snippet,statistics',
    chart='mostPopular',
    regionCode=country_code,
    maxResults=max_results
)
response = request.execute()

videos = []
for item in response.get('items', []):
    video = {
        'video_id': item['id'],
        'title': item['snippet']['title'],
        'channel_title': item['snippet']['channelTitle'],
        'category_id': item['snippet']['categoryId'],
        'publish_time': item['snippet']['publishedAt'],
        'views': int(item['statistics'].get('viewCount', 0)),
        'likes': int(item['statistics'].get('likeCount', 0)),
        'dislikes': int(item['statistics'].get('dislikeCount', 0)), #_
        # might be deprecated
        'comment_count': int(item['statistics'].get('commentCount', 0)),
        'country': country_code
    }
    videos.append(video)
return videos

# Collect from multiple countries
countries = ['US', 'IN', 'PK', 'RU', 'FR', 'IT', 'DE']
all_videos = []

for country in tqdm(countries):
    videos = fetch_trending(country, max_results=100)
    all_videos.extend(videos)

# Create DataFrame
df = pd.DataFrame(all_videos)

# Feature engineering
df['publish_time'] = pd.to_datetime(df['publish_time'])
df['publish_day'] = df['publish_time'].dt.day_name()
df['publish_hour'] = df['publish_time'].dt.hour
df['like_ratio'] = df['likes'] / (df['likes'] + df['dislikes'] + 1)
df['comment_ratio'] = df['comment_count'] / (df['views'] + 1)
df['title_sentiment'] = df['title'].apply(lambda x: TextBlob(x).sentiment.
    # polarity)

# Reorder and format columns for clarity

```



```

df = df[[
    'video_id', 'title', 'channel_title', 'country', 'category_id',
    'publish_time', 'publish_day', 'publish_hour',
    'views', 'likes', 'dislikes', 'comment_count',
    'like_ratio', 'comment_ratio', 'title_sentiment'
]]

# Sort videos by views
df_sorted = df.sort_values(by='views', ascending=False)

# Show top 10 videos by views
print(" Top 10 Trending Videos by Views:\n")
display(df_sorted[['title', 'channel_title', 'country', 'views', 'likes', 'comment_count']].head(10))

# Visualization 1: Views by Day
plt.figure(figsize=(10,6))
sns.barplot(x='publish_day', y='views', data=df, estimator='mean')
plt.title("Average Views by Day of Week")
plt.xticks(rotation=45)
plt.show()

# Visualization 2: Views by Hour
plt.figure(figsize=(10,6))
sns.lineplot(x='publish_hour', y='views', data=df)
plt.title("Views by Hour")
plt.grid(True)
plt.show()

# Correlation heatmap
corr = df[['views', 'likes', 'dislikes', 'comment_count', 'like_ratio', 'comment_ratio']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Save to CSV
df_sorted.to_csv('youtube_trending_live.csv', index=False)
print(f" Saved {len(df_sorted)} trending videos to 'youtube_trending_live.csv'")

```

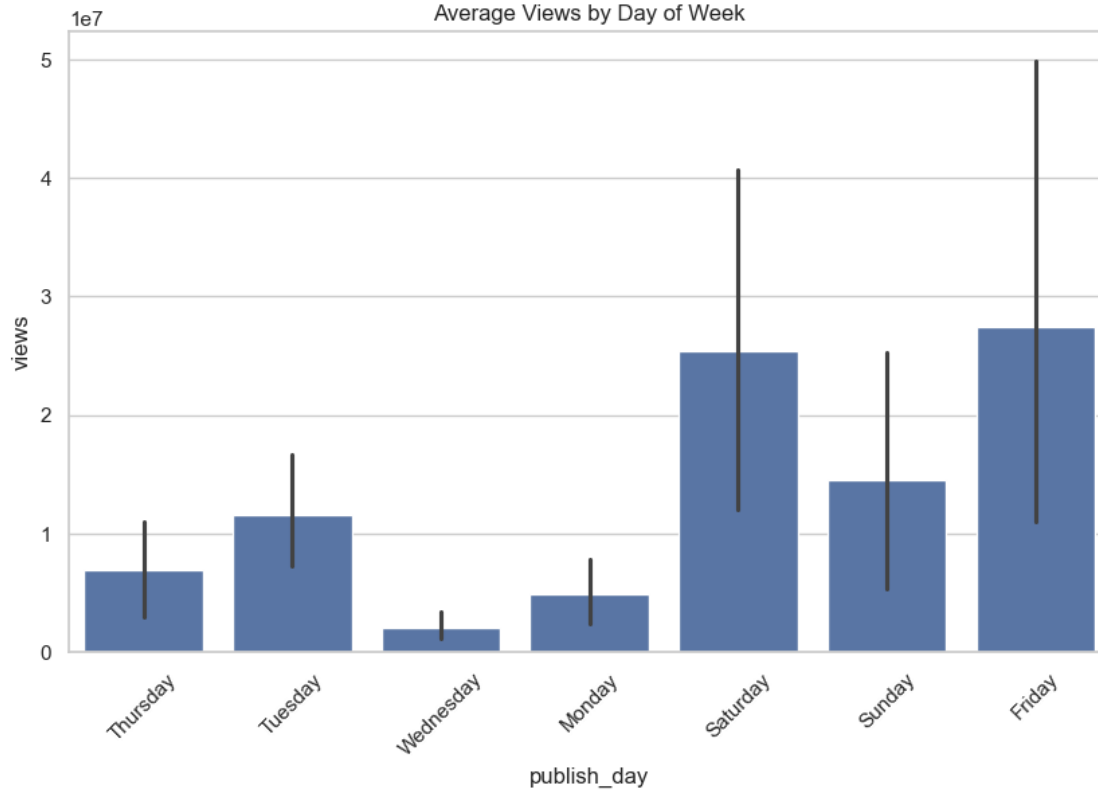
100%|

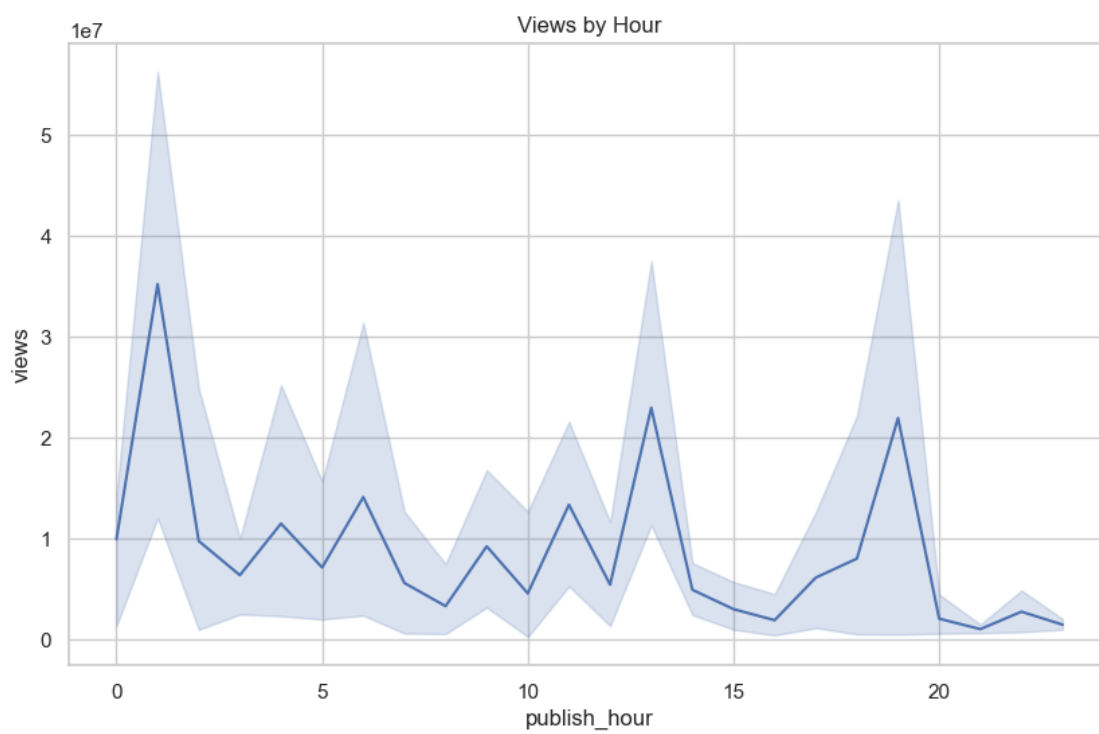
| 7/7 [00:02<00:00, 2.82it/s]

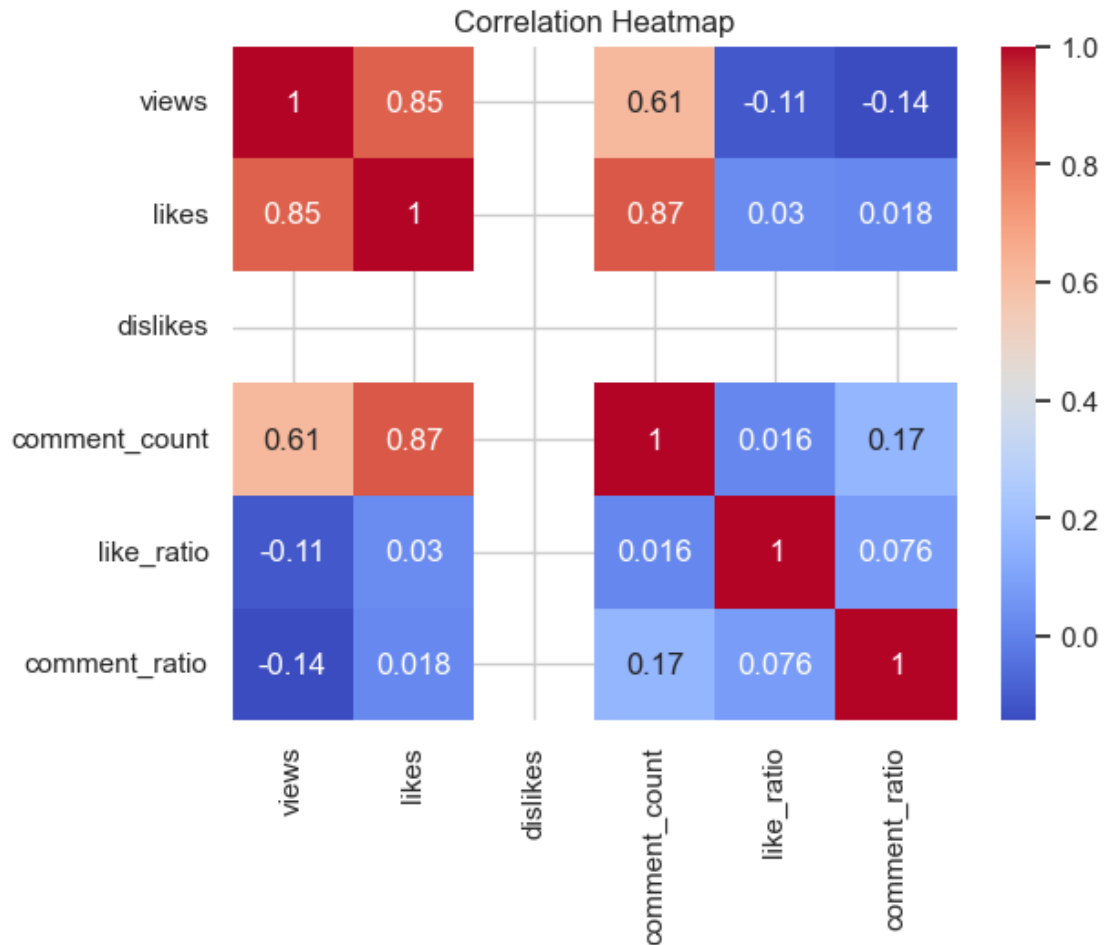
Top 10 Trending Videos by Views:

	title	channel_title	\
149	Good Foods vs Bad Foods! Learn Teeth-Friendl...	BoomZy Doo	
252	Grand Theft Auto VI Trailer 2	Rockstar Games	
178	Grand Theft Auto VI Trailer 2	Rockstar Games	
1	Grand Theft Auto VI Trailer 2	Rockstar Games	
51	Grand Theft Auto VI Trailer 2	Rockstar Games	
301	Grand Theft Auto VI Trailer 2	Rockstar Games	
200	Grand Theft Auto VI Trailer 2	Rockstar Games	
104	wow	dednahype	
134	CRAZIEST IMPOSSIBLE SHOT YET @AlwaysHockey	Dude Perfect	
299	CRAZIEST IMPOSSIBLE SHOT YET @AlwaysHockey	Dude Perfect	

	country	views	likes	comment_count
149	PK	95910880	1255994	0
252	IT	91668200	5388406	453935
178	RU	91668200	5388406	453935
1	US	91668200	5388406	453935
51	IN	91668200	5388406	453935
301	DE	91668200	5388406	453935
200	FR	91668200	5388406	453935
104	PK	77700532	923818	941
134	PK	75139313	3425631	2690
299	IT	75139313	3425631	2690







Saved 350 trending videos to 'youtube\_trending\_live.csv'

```
[1]: # YouTube Trending Video Analyzer - Interactive & Visualization Enhanced
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob
from googleapiclient.discovery import build
from tqdm import tqdm

sns.set(style='whitegrid')
%matplotlib inline

# ----- 1. Set up API -----
API_KEY = 'AIzaSyC1Yhq66YmNp-v4VCgI96gJf_-dR1NgCgY'
youtube = build('youtube', 'v3', developerKey=API_KEY)
```

```

# ----- 2. Country Selection -----
print("Select Country (or type 'ALL' for global analysis):")
print("Options: US, IN, PK, RU, FR, IT, DE")
user_input = input("Enter country code or 'ALL': ").strip().upper()

countries = ['US', 'IN', 'PK', 'RU', 'FR', 'IT', 'DE'] if user_input == 'ALL'
else [user_input]

# ----- 3. Fetch Trending Videos -----
def fetch_trending(country_code, max_results=50):
    request = youtube.videos().list(
        part='snippet,statistics',
        chart='mostPopular',
        regionCode=country_code,
        maxResults=max_results
    )
    response = request.execute()
    videos = []
    for item in response.get('items', []):
        video = {
            'video_id': item['id'],
            'title': item['snippet']['title'],
            'channel_title': item['snippet']['channelTitle'],
            'category_id': item['snippet']['categoryId'],
            'publish_time': item['snippet']['publishedAt'],
            'views': int(item['statistics'].get('viewCount', 0)),
            'likes': int(item['statistics'].get('likeCount', 0)),
            'dislikes': int(item['statistics'].get('dislikeCount', 0)),
            'comment_count': int(item['statistics'].get('commentCount', 0)),
            'country': country_code
        }
        videos.append(video)
    return videos

# ----- 4. Collect and Combine -----
all_videos = []
for country in tqdm(countries):
    all_videos.extend(fetch_trending(country, max_results=50))

df = pd.DataFrame(all_videos)

# ----- 5. Feature Engineering -----
df['publish_time'] = pd.to_datetime(df['publish_time'])
df['publish_day'] = df['publish_time'].dt.day_name()
df['publish_hour'] = df['publish_time'].dt.hour
df['like_ratio'] = df['likes'] / (df['likes'] + df['dislikes'] + 1)
df['comment_ratio'] = df['comment_count'] / (df['views'] + 1)

```

```

df['title_sentiment'] = df['title'].apply(lambda x: TextBlob(x).sentiment.
    ↪polarity)

# ----- 6. Load Category Mapping -----
cat_map = youtube.videoCategories().list(part='snippet', regionCode='US').
    ↪execute()
cat_lookup = {item['id']: item['snippet']['title'] for item in cat_map['items']}
df['category_name'] = df['category_id'].map(cat_lookup)

# ----- 7. Display & Export -----
df_sorted = df.sort_values(by='views', ascending=False).head(200) # show top
    ↪200 videos

print("\n Top 10 Trending Channels:")
top_channels = df_sorted['channel_title'].value_counts().head(10)
print(top_channels)

# ----- 8. Visualizations -----
# Pie chart: Views by Category
cat_views = df_sorted.groupby('category_name')['views'].sum().
    ↪sort_values(ascending=False)
cat_views.plot.pie(autopct='%1.1f%%', figsize=(8, 8), title='Share of Views by
    ↪Category')
plt.ylabel('')
plt.show()

# Bar chart: Top Channels
top_channels.plot(kind='barh', color='skyblue', title='Top Trending Channels')
plt.xlabel("Number of Trending Videos")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()

# ----- 9. Export for Tableau -----
df_sorted.to_csv('youtube_trending_analysis_ready.csv', index=False)
print(f" Saved {len(df_sorted)} records to 'youtube_trending_analysis_ready.
    ↪csv'")

```

Select Country (or type 'ALL' for global analysis):

Options: US, IN, PK, RU, FR, IT, DE

Enter country code or 'ALL': ALL

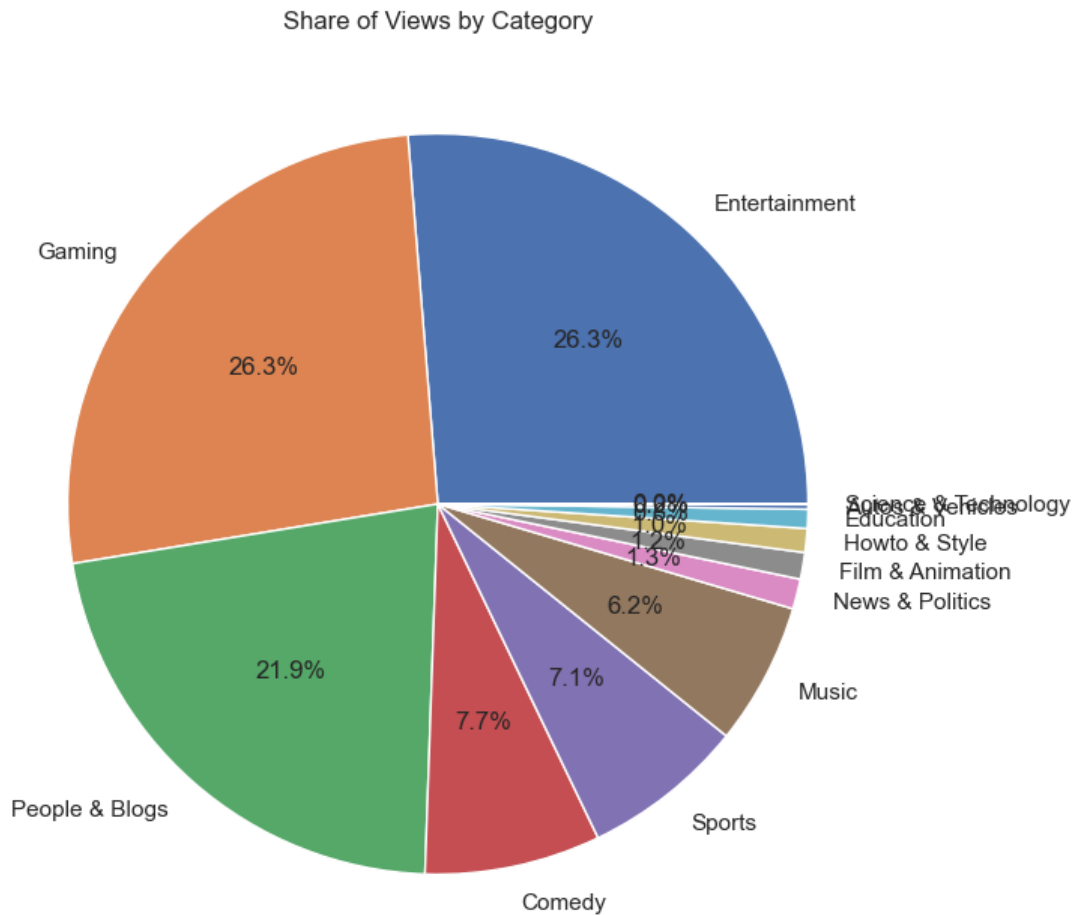
100%|

| 7/7 [00:02<00:00, 3.16it/s]

Top 10 Trending Channels:  
channel\_title

Rockstar Games	6
CBS Sports Golazo	4
Vijay Television	3
ISSEI /	3
i_roblox_queen	3
Netflix	3
Vogue	3
Jahaann	2
Live Speedy	2
Cheri's World	2

Name: count, dtype: int64



C:\Users\tvino\AppData\Local\Temp\ipykernel\_8032\4249839230.py:87: UserWarning: Glyph 12356 (\N{HIRAGANA LETTER I}) missing from font(s) Arial.

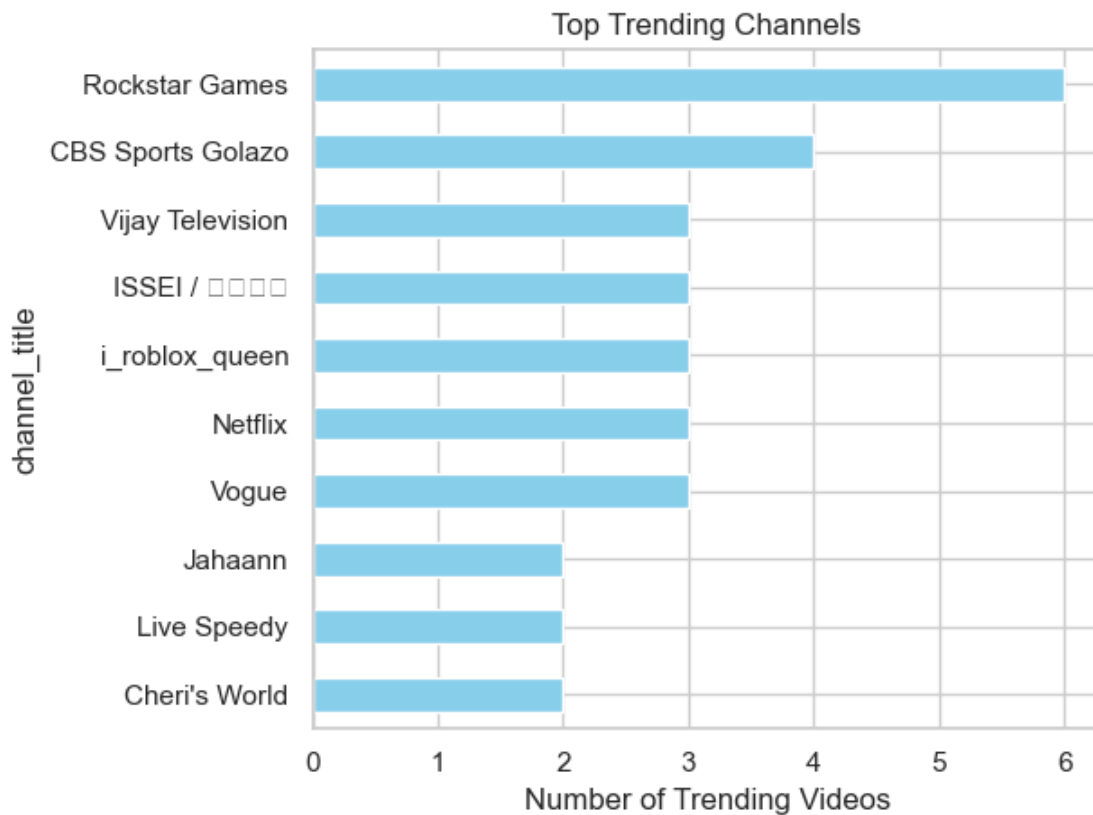
plt.tight\_layout()

C:\Users\tvino\AppData\Local\Temp\ipykernel\_8032\4249839230.py:87: UserWarning: Glyph 12387 (\N{HIRAGANA LETTER SMALL TU}) missing from font(s) Arial.

```

plt.tight_layout()
C:\Users\tvino\AppData\Local\Temp\ipykernel_8032\4249839230.py:87: UserWarning:
Glyph 12379 (\N{HIRAGANA LETTER SE}) missing from font(s) Arial.
plt.tight_layout()
C:\Users\tvino\Downloads\anaconda\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 12356 (\N{HIRAGANA
LETTER I}) missing from font(s) Arial.
fig.canvas.print_figure(bytes_io, **kw)
C:\Users\tvino\Downloads\anaconda\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 12387 (\N{HIRAGANA
LETTER SMALL TU}) missing from font(s) Arial.
fig.canvas.print_figure(bytes_io, **kw)
C:\Users\tvino\Downloads\anaconda\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 12379 (\N{HIRAGANA
LETTER SE}) missing from font(s) Arial.
fig.canvas.print_figure(bytes_io, **kw)

```



Saved 200 records to 'youtube\_trending\_analysis\_ready.csv'

[ ]: