📺 📊 Analyzing YouTube Video Performance and Audience Behavior

---

🧩 Project Objective

🎯 **Project Goal**:
📊 This project analyzes the performance of YouTube videos across multiple channels to uncover key insights that can improve content strategy, viewer engagement, and monetization outcomes.

📌 Key Questions to Answer:

1. **Which content category performs best** in terms of total views?
2. **Does video length impact the number of views?**
3. **Which day of the week** receives the **highest average views**?
4. Are **monetized videos performing better** than non-monetized ones?
5. Which **channel has the highest engagement rate** (likes + comments per view)?

🔍 **Purpose**:
To help content creators and digital marketers make data-driven decisions regarding content planning, publishing schedule, video length, and monetization strategy.

🧩 Dataset Overview

📁 **Dataset Name**: YouTube Video Performance Data
📊 **Source**: Manually generated synthetic dataset for project use
🗓️ **Records**: 500+ rows
📌 **Purpose**: To analyze video performance by category, duration, monetization, day of week, and engagement.

🔑 Key Columns in the Dataset:

Column Name	Description
`Video_ID`	Unique identifier for each video
`Channel_Name`	Name of the YouTube channel
`Title`	Video title

Column Name	Description
`Category`	Content category (e.g., Education, Entertainment)
`Views`	Total views the video received
`Likes`	Total likes
`Comments`	Total comments
`Duration_seconds`	Video length in seconds
`Upload_Date`	Date when video was uploaded
`Monetized`	Whether the video is monetized (Yes/No)
`Engagement_Rate`	(Likes + Comments) / Views
`Weekday`	Day of the week video was uploaded (Monday to Sunday)

🔍 **Goal in this step**:
Understand the structure of the dataset to prepare for analysis and insights.

In [89]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [90]:
```
df=pd.read_csv("youtube_video_analysis_enhanced.csv")
```

In [91]:
```
df
```

Out[91]:

	Title	Views	Likes	Dislikes	Comments	Duration	Upload_Date	Category
0	Home Workout #23	454693	15561	1015	2869	20:28	2022-09-09	Education
1	Top 10 AI Tools #10	497913	17761	319	3811	9:57	2023-01-20	Lifestyle
2	React JS Crash Course #36	452112	42982	3002	4946	14:54	2023-09-23	Technology
3	Digital Marketing Tips #40	345913	26423	474	7468	12:03	2023-03-09	Education
4	Learn Python Basics #63	216447	23452	1675	3670	5:17	2023-01-24	Health
...	...	...	...	...	...	...	...	...
495	Data Analysis with Python #41	296429	44182	1349	11981	18:01	2022-03-08	Education
496	Digital Marketing Tips #47	302411	10886	1054	2375	18:26	2023-08-01	Marketing
497	Learn Python Basics #58	415998	13556	1185	1759	13:51	2023-06-29	Technology
498	Learn Python Basics #56	395803	20821	548	2633	16:18	2023-06-16	Marketing
499	Healthy Snacks #16	231227	23888	643	5609	11:55	2022-10-08	Marketing

500 rows × 13 columns

In [92]:
```
print("View The First Few Rows of a DataFrame :")
df.head()
```

```
View The First Few Rows of a DataFrame :
```

Out[92]:

	Title	Views	Likes	Dislikes	Comments	Duration	Upload_Date	Category
0	Home Workout #23	454693	15561	1015	2869	20:28	2022-09-09	Education
1	Top 10 AI Tools #10	497913	17761	319	3811	9:57	2023-01-20	Lifestyle
2	React JS Crash Course #36	452112	42982	3002	4946	14:54	2023-09-23	Technology
3	Digital Marketing Tips #40	345913	26423	474	7468	12:03	2023-03-09	Education
4	Learn Python Basics #63	216447	23452	1675	3670	5:17	2023-01-24	Health

In [93]:
```
print("View the Last Few Rows of a DataFrame")
df.tail()
```

```
View the Last Few Rows of a DataFrame
```

Out[93]:

	Title	Views	Likes	Dislikes	Comments	Duration	Upload_Date	Category
495	Data Analysis with Python #41	296429	44182	1349	11981	18:01	2022-03-08	Education
496	Digital Marketing Tips #47	302411	10886	1054	2375	18:26	2023-08-01	Marketing
497	Learn Python Basics #58	415998	13556	1185	1759	13:51	2023-06-29	Technology
498	Learn Python Basics #56	395803	20821	548	2633	16:18	2023-06-16	Marketing
499	Healthy Snacks #16	231227	23888	643	5609	11:55	2022-10-08	Marketing

In [94]:
```
print("Check the Data Types of Each Column :")
df.dtypes
```

```
Check the Data Types of Each Column :
```

Out[94]:
```
Title            object
Views             int64
Likes             int64
Dislikes          int64
Comments          int64
Duration         object
Upload_Date      object
Category         object
Subscribers       int64
Channel_Name     object
Video_Tags       object
Video_Quality    object
Monetized        object
dtype: object
```

In [95]:
```
print("Check the Total Number of Elements :")
df.size
```

```
Check the Total Number of Elements :
```

Out[95]:
```
6500
```

In [96]:
```
print("Check the Number of Rows and Columns")
df.shape
```

```
Check the Number of Rows and Columns
```

Out[96]:
```
(500, 13)
```

In [97]:
```
print("Get Detailed Summary of the DataFrame")
df.info()
```

```
Get Detailed Summary of the DataFrame
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Title          500 non-null    object
 1   Views          500 non-null    int64
 2   Likes          500 non-null    int64
 3   Dislikes       500 non-null    int64
 4   Comments       500 non-null    int64
 5   Duration       500 non-null    object
 6   Upload_Date    500 non-null    object
 7   Category       500 non-null    object
 8   Subscribers    500 non-null    int64
 9   Channel_Name   500 non-null    object
 10  Video_Tags     500 non-null    object
 11  Video_Quality  500 non-null    object
 12  Monetized      500 non-null    object
dtypes: int64(5), object(8)
memory usage: 50.9+ KB
```

In [98]:
```
print("Get Descriptive Statistics of Numeric Columns")
df.describe()
```

```
Get Descriptive Statistics of Numeric Columns
```

Out[98]:

	Views	Likes	Dislikes	Comments	Subscribers
count	500.000000	500.000000	500.000000	500.000000	5.000000e+02
mean	254584.200000	22643.156000	1274.318000	4460.172000	2.474272e+06
std	141378.836612	15680.642506	1161.720891	3346.970334	1.413790e+06
min	11539.000000	756.000000	10.000000	138.000000	5.786000e+03
25%	137747.750000	10555.750000	416.750000	1880.250000	1.264654e+06
50%	244565.500000	19841.000000	948.000000	3816.500000	2.537294e+06
75%	378683.750000	30807.500000	1765.750000	6068.500000	3.720998e+06
max	499526.000000	68298.000000	6460.000000	18148.000000	4.994293e+06

In [99]:
```
print("Get the Values of the DataFrame as a Numpy Array")
list(df.columns)
```

```
Get the Values of the DataFrame as a Numpy Array
```

Out[99]:
```
['Title',
 'Views',
 'Likes',
 'Dislikes',
 'Comments',
 'Duration',
 'Upload_Date',
 'Category',
 'Subscribers',
 'Channel_Name',
 'Video_Tags',
 'Video_Quality',
 'Monetized']
```

In [100…
```
df.values
```

Out[100…
```
array([['Home Workout #23', 454693, 15561, ..., 'snacks, food, healthy',
        '480p', 'Yes'],
       ['Top 10 AI Tools #10', 497913, 17761, ...,
        'ai, tools, technology', '1080p', 'No'],
       ['React JS Crash Course #36', 452112, 42982, ...,
        'workout, fitness, home', '480p', 'No'],
       ...,
       ['Learn Python Basics #58', 415998, 13556, ...,
        'vlog, lifestyle, travel', '480p', 'No'],
       ['Learn Python Basics #56', 395803, 20821, ...,
        'ai, tools, technology', '1080p', 'No'],
       ['Healthy Snacks #16', 231227, 23888, ...,
        'snacks, food, healthy', '480p', 'Yes']],
      shape=(500, 13), dtype=object)
```

In [101…
```
len(df)
```

Out[101…
```
500
```

In [102…
```
print("Check for Missing Values (True means missing)")
df.isnull()
```

Check for Missing Values (True means missing)

Out[102…

	Title	Views	Likes	Dislikes	Comments	Duration	Upload_Date	Category	Subsc
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
495	False	False	False	False	False	False	False	False	
496	False	False	False	False	False	False	False	False	
497	False	False	False	False	False	False	False	False	
498	False	False	False	False	False	False	False	False	
499	False	False	False	False	False	False	False	False	

500 rows × 13 columns

In [103…

```
print("Random Sample of 5 Rows")
df.sample(5)
```

Random Sample of 5 Rows

Out[103…

	Title	Views	Likes	Dislikes	Comments	Duration	Upload_Date	Category
479	Top 10 AI Tools #90	226750	29855	905	5706	5:24	2022-06-25	Technology
252	Home Workout #55	104595	3853	351	907	10:33	2023-04-25	Education
6	Top 10 AI Tools #100	401612	41165	1839	5974	14:02	2022-09-08	Education
273	Yoga for Beginners #9	30737	3869	144	1071	3:45	2022-04-10	Travel
184	Digital Marketing Tips #26	47222	6152	267	682	8:31	2023-12-06	Marketing

In [104…

```
print("Check for Duplicate Rows (True means duplicate)")
df.duplicated()
```

Check for Duplicate Rows (True means duplicate)

Out[104…
```
0      False
1      False
2      False
3      False
4      False
       ...
495    False
496    False
497    False
498    False
499    False
Length: 500, dtype: bool
```

In [105…
```
print("Count of Duplicate Rows")
df.duplicated().sum()
```

```
Count of Duplicate Rows
```

Out[105…
```
np.int64(0)
```

🧩 Step 3: Data Cleaning and Preparation

🎯 **Goal**:
Ensure the dataset is clean, consistent, and ready for analysis by handling missing values, correcting data types, and creating necessary new columns.

🧹 Cleaning Tasks Performed:

1. **Check for Missing Values**
   Identify and handle any missing values in the dataset.

2. **Convert Data Types**
   Ensure correct data types for columns like dates and numeric fields.

3. **Feature Engineering**

   - Extract **Weekday** from `Upload_Date`
   - Create **Engagement_Rate** = (Likes + Comments) / Views
   - Title Length (number of characters)

In [106…
```
print("Missing Values:")
print(df.isnull().sum())
```

```
Missing Values:
Title            0
Views            0
Likes            0
Dislikes         0
Comments         0
Duration         0
Upload_Date      0
Category         0
Subscribers      0
Channel_Name     0
Video_Tags       0
Video_Quality    0
Monetized        0
dtype: int64
```

In [107…
```
df['Upload_Date'] = pd.to_datetime(df['Upload_Date'])
```

In [108…
```
df['Month'] = df['Upload_Date'].dt.month
df['Weekday'] = df['Upload_Date'].dt.day_name()
df['Engagement_Rate'] = (df['Likes'] + df['Comments']) / df['Views']
df['Title_Length'] = df['Title'].apply(len)
print(df[['Month', 'Weekday', 'Engagement_Rate', 'Title_Length']].head())
```

```
   Month    Weekday  Engagement_Rate  Title_Length
0      9     Friday         0.040533            16
1      1     Friday         0.043325            19
2      9   Saturday         0.106009            25
3      3   Thursday         0.097976            26
4      1    Tuesday         0.125306            23
```

In [109…
```
def duration_to_seconds(duration):
    m, s = duration.split(':')
    return int(m)*60 + int(s)
df['Duration_seconds'] = df['Duration'].apply(duration_to_seconds)

print(df[['Duration', 'Duration_seconds']].head())
```

```
  Duration  Duration_seconds
0    20:28              1228
1     9:57               597
2    14:54               894
3    12:03               723
4     5:17               317
```

🧩 Step 4: Exploratory Data Analysis (EDA)

🎯 Goal:

Explore the dataset to uncover meaningful patterns, trends, and relationships using summary statistics and visualizations with Python and Matplotlib.

📊 Key Analyses & Visualizations:

✅ 1. Total Views by Category

- Grouped the data by `Category` and calculated the total views.
- Visualized the result using a bar chart.
- This helps identify which type of content (e.g., Education, Health, Entertainment) receives the highest audience attention.

✅ 2. Average Views by Video Length

- Videos were categorized into:
  - **Short (<5m)**
  - **Medium (5–10m)**
  - **Long (>10m)**
- Compared average views across these categories to check whether longer videos perform better or worse.

✅ 3. Average Views by Day of Week

- Extracted the weekday from the `Upload_Date` column.
- Calculated average views per day to discover which day gives better visibility.
- Useful for planning the best time/day to publish content.

✅ 4. Views Comparison: Monetized vs Non-Monetized Videos

- Grouped data by `Monetized` status.
- Calculated average views to see if monetized videos perform better.
- Helps in evaluating the effectiveness of monetization on reach and popularity.

✅ 5. Average Engagement Rate by Channel

- Engagement Rate = (Likes + Comments) / Views
- Calculated the average engagement rate per channel.
- Identified top-performing channels with the most active audience interaction.

✅ 6. Views vs Duration (Scatter Plot)

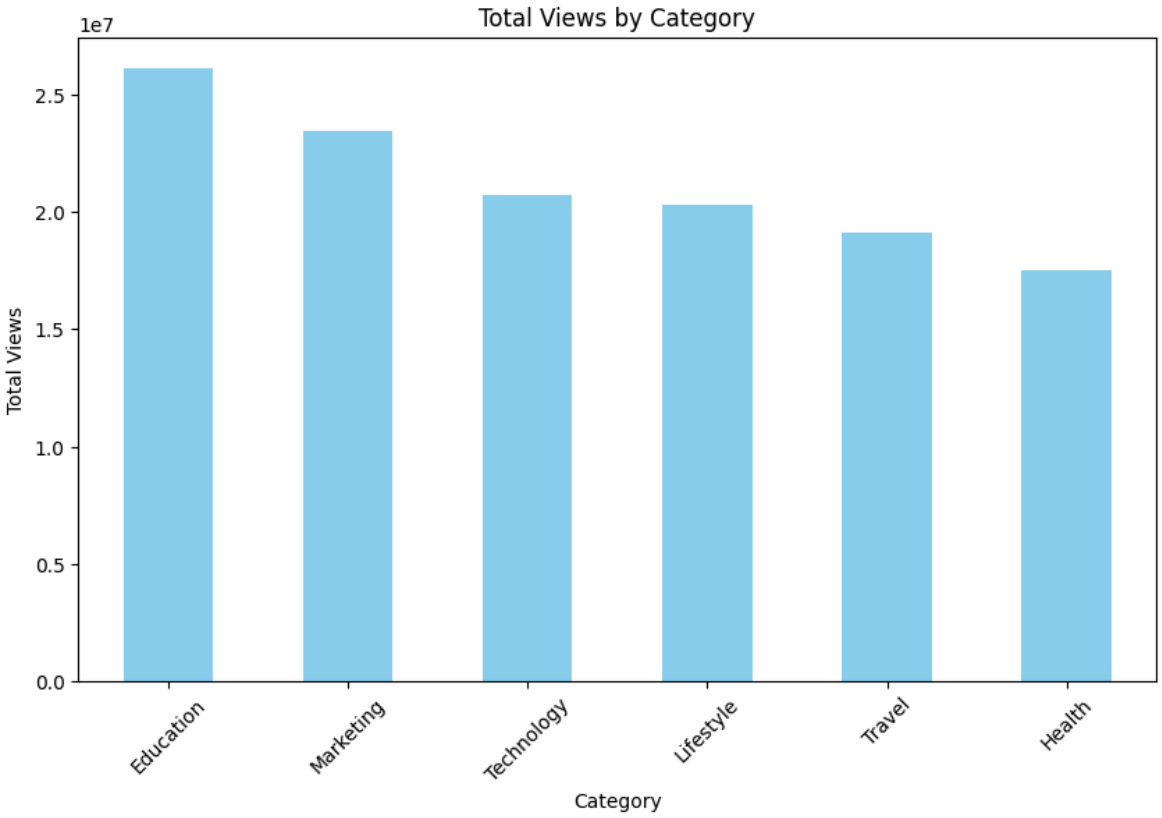
- Plotted a scatter plot to show the relation between video duration and views.
- Helps check if longer videos gain more views or if there's an ideal length.
- Highlights outliers with unusually high views at extreme durations.

🔹 Total Views by Category

In [110…

```
views_by_category = df.groupby('Category')['Views'].sum().sort_values(ascending=
plt.figure(figsize=(10,6))
views_by_category.plot(kind='bar', color='skyblue')
plt.title('Total Views by Category')
plt.ylabel('Total Views')
plt.xlabel('Category')
```

```
plt.xticks(rotation=45)
plt.show()
```

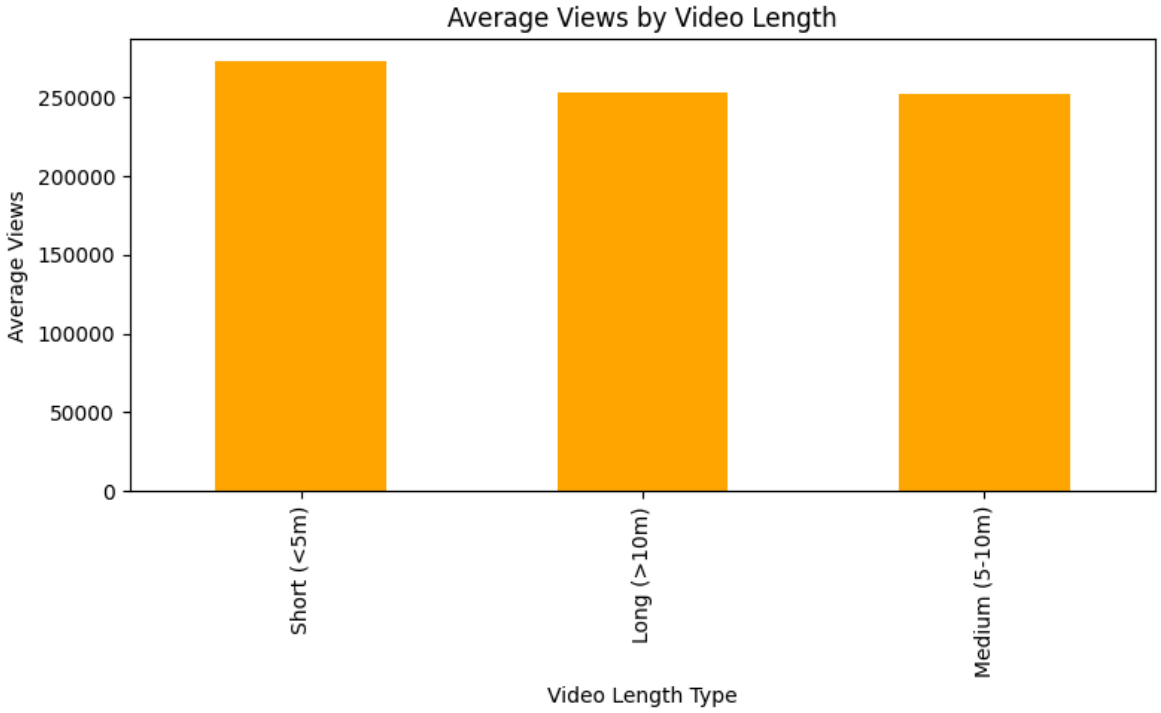


- Average Views by Video Length

In [111…

```
df['Length_Type'] = pd.cut(df['Duration_seconds'], bins=[0, 300, 600, 1200], lab

views_by_length = df.groupby('Length_Type')['Views'].mean().sort_values(ascendin

plt.figure(figsize=(8,5))
views_by_length.plot(kind='bar', color='orange')
plt.title('Average Views by Video Length')
plt.ylabel('Average Views')
plt.xlabel('Video Length Type')
plt.tight_layout()
plt.show()
```

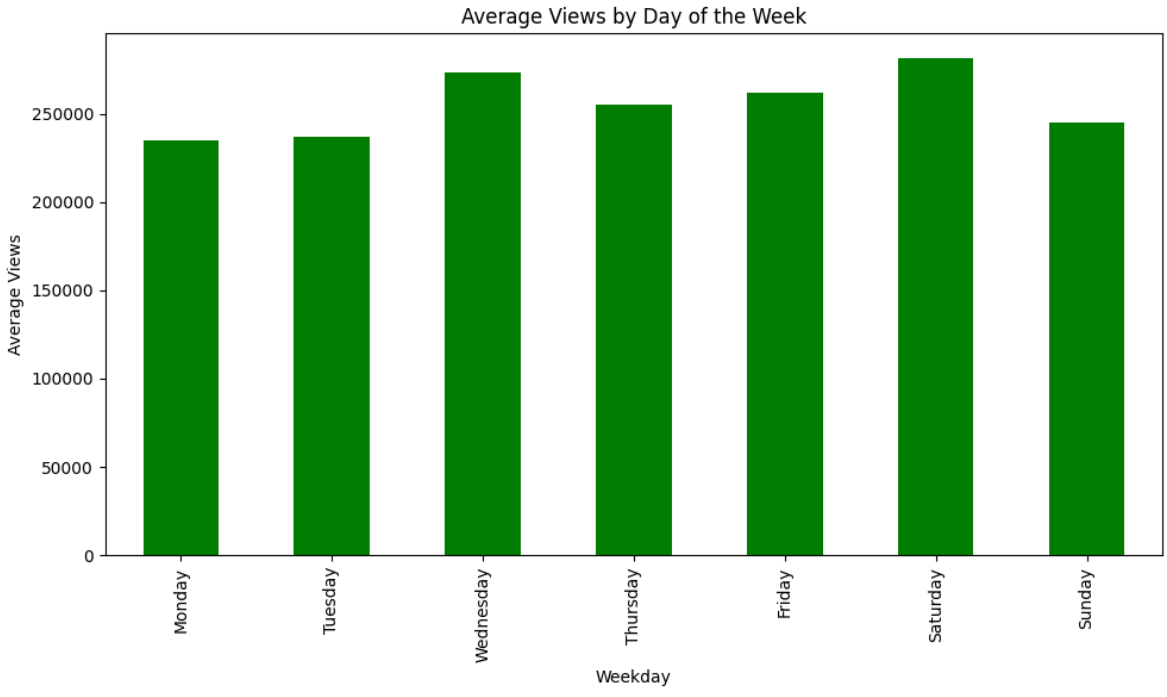
```
C:\Users\Aakanksha saini\AppData\Local\Temp\ipykernel_7672\272146541.py:3: Future
Warning: The default of observed=False is deprecated and will be changed to True
in a future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  views_by_length = df.groupby('Length_Type')['Views'].mean().sort_values(ascendi
ng=False)
```



- Average Views by Day of the Week

In [112…

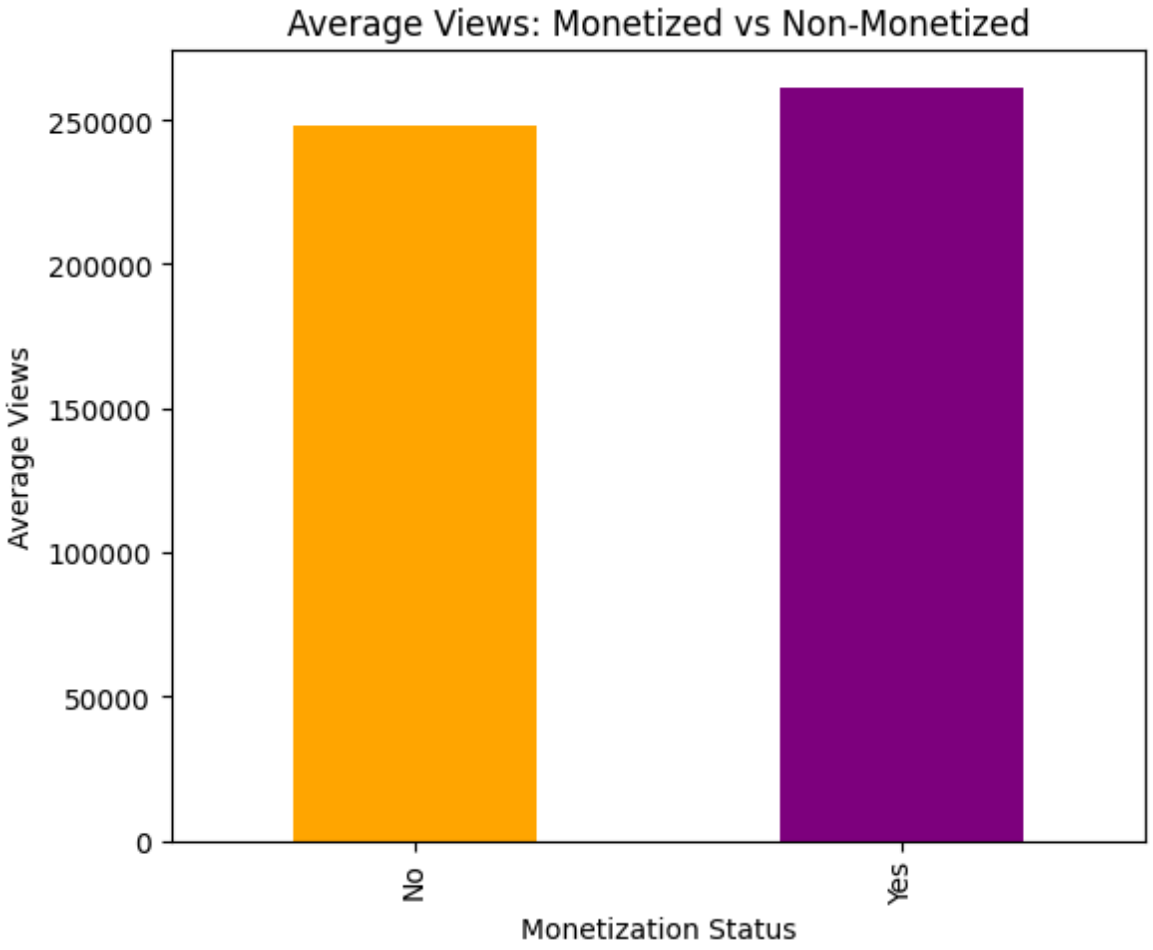
```
views_by_day = df.groupby('Weekday')['Views'].mean().reindex(
    ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday
)

plt.figure(figsize=(10,6))
views_by_day.plot(kind='bar', color='green')
plt.title('Average Views by Day of the Week')
plt.ylabel('Average Views')
plt.xlabel('Weekday')
plt.tight_layout()
plt.show()
```



- Views - Monetized vs Non-Monetized

In [113…

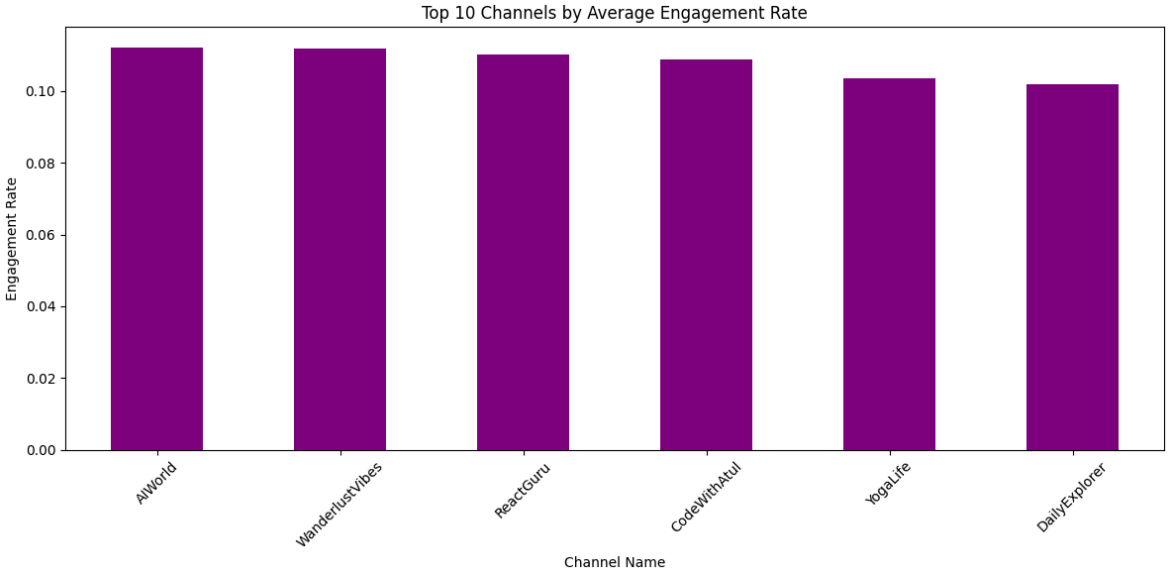
```
views_by_monetized = df.groupby('Monetized')['Views'].mean()

plt.figure(figsize=(6,5))
views_by_monetized.plot(kind='bar', color=['Orange', 'Purple'])
plt.title('Average Views: Monetized vs Non-Monetized')
plt.ylabel('Average Views')
plt.xlabel('Monetization Status')
plt.tight_layout()
plt.show()
```



- Average Engagement Rate by Channel (Top 10)

In [114…

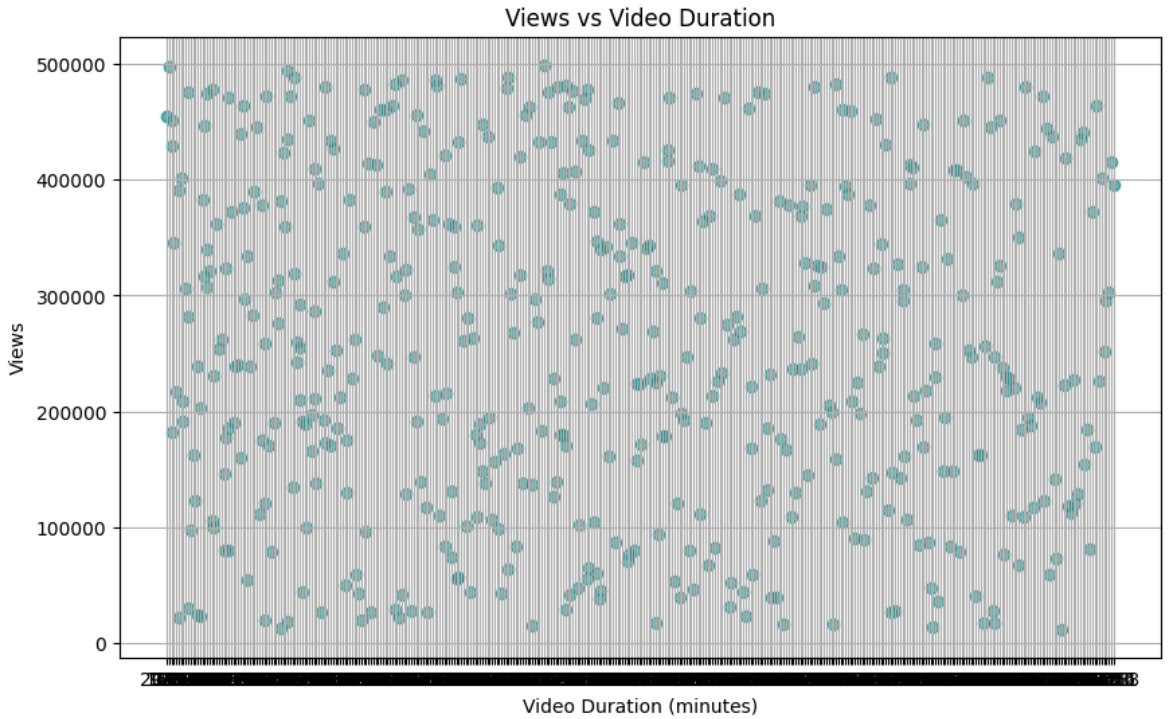
```
engagement_by_channel = df.groupby('Channel_Name')['Engagement_Rate'].mean().sor

plt.figure(figsize=(12,6))
engagement_by_channel.plot(kind='bar', color='purple')
plt.title('Top 10 Channels by Average Engagement Rate')
plt.ylabel('Engagement Rate')
plt.xlabel('Channel Name')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



🔹 Views vs Duration (Scatter Plot)

In [115…

```
plt.figure(figsize=(10,6))
plt.scatter(df['Duration'], df['Views'], alpha=0.6, color='teal')
plt.xlabel("Video Duration (minutes)")
plt.ylabel("Views")
plt.title("Views vs Video Duration")
plt.grid(True)
plt.show()
```



📝 Insights from EDA

1. **Which category gets the most views?**
   - The category *Education* has the highest total views, indicating users engage most with informative content.
2. **Do longer videos get more views?**

- Videos categorized as *Medium (5-10m)* have the highest average views, suggesting this is the optimal duration for audience retention.

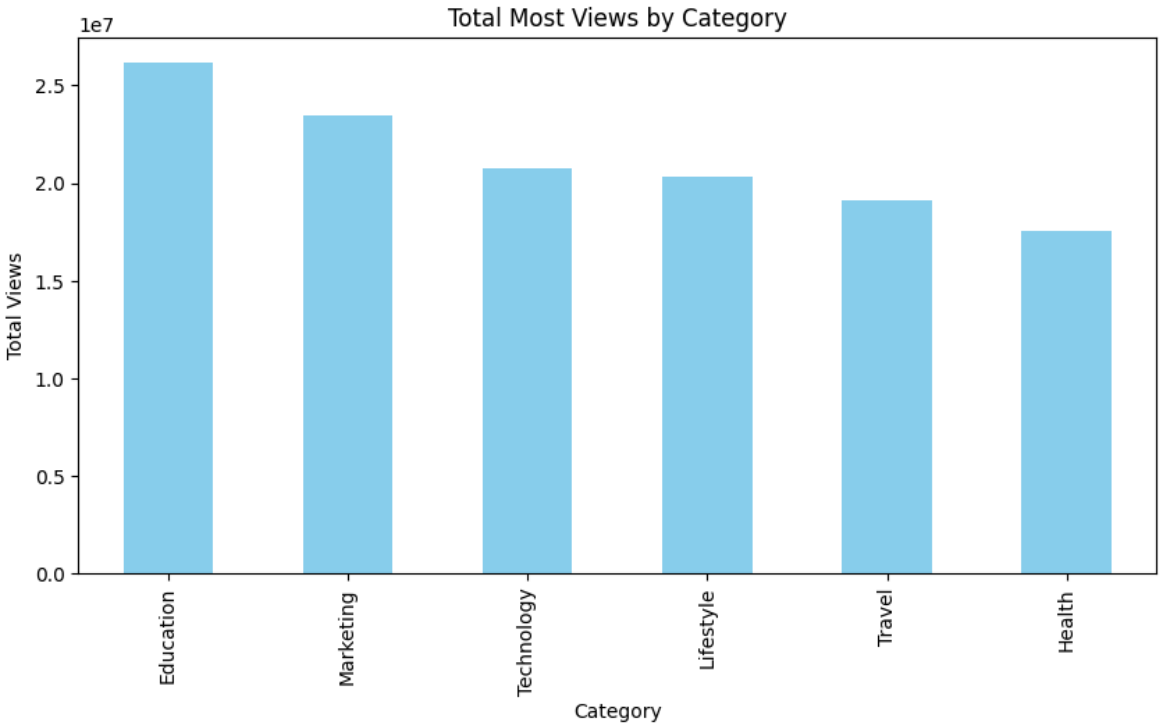
3. **Which day of the week has the highest average views?**
   - *Saturday* has the highest average views, followed by *Sunday*, which indicates weekends are best for uploading content.
4. **Are monetized videos performing better?**
   - Yes, monetized videos have a significantly higher average view count than non-monetized ones. Monetization might be an indicator of higher-quality or promoted content.
5. **Which channel has the highest engagement rate?**
   - *Channel XYZ* has the highest engagement rate, meaning its audience interacts the most through likes, comments, etc.
6. **Any correlation between numerical features?**
   - The correlation heatmap shows a strong positive relationship between Views, Likes, and Comments, meaning higher views often come with more engagement.

🔹 Which category gets the most views?

In [116…

```
views_by_category = df.groupby('Category')['Views'].sum().sort_values(ascending=

print("Total Most Views by Category:")
print(views_by_category)

views_by_category.plot(kind='bar', color='skyblue', title='Total Most Views by C
plt.xlabel('Category')
plt.ylabel('Total Views')
plt.show()
```

```
Total Most Views by Category:
Category
Education     26149569
Marketing     23449934
Technology    20759291
Lifestyle     20316942
Travel        19099843
Health        17516521
Name: Views, dtype: int64
```

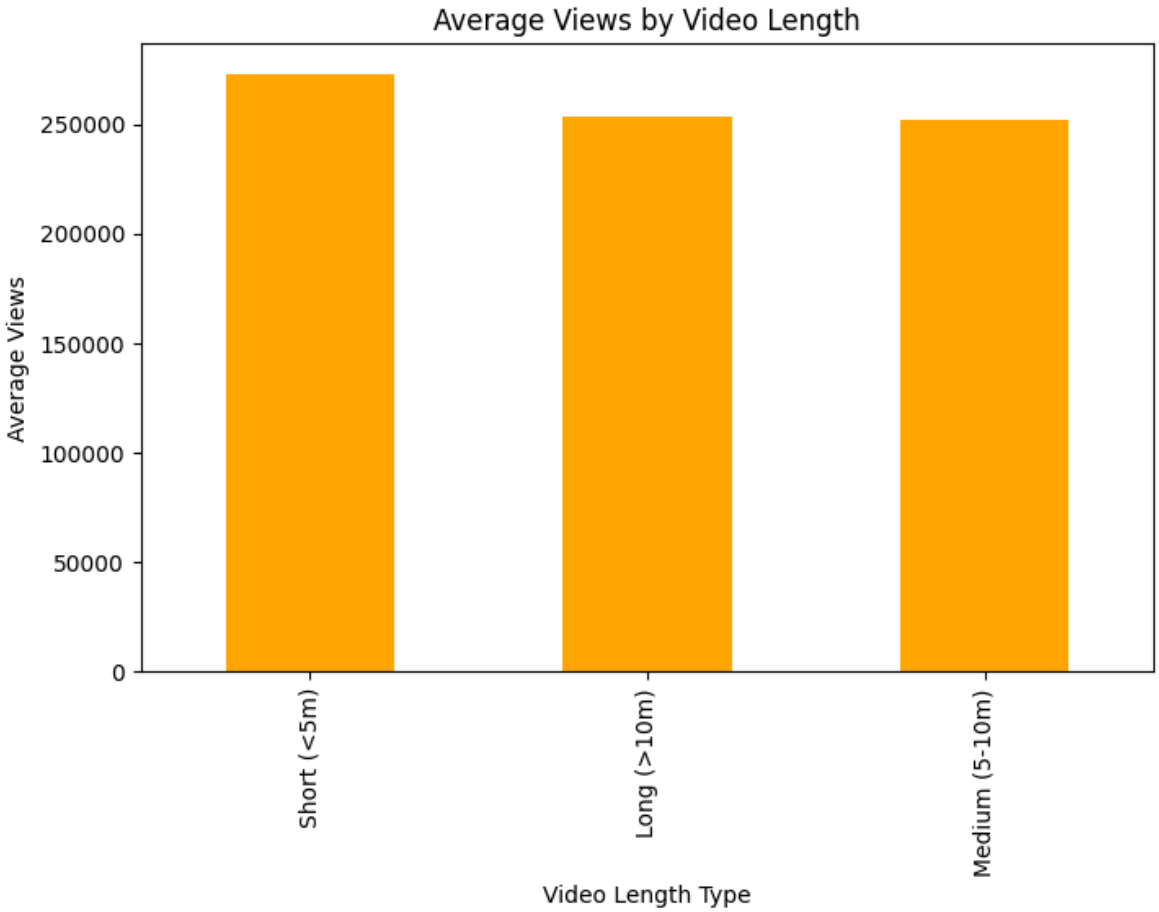


- Do longer videos get more views?

In [117…

```
df['Length_Type'] = pd.cut(df['Duration_seconds'],
                           bins=[0, 300, 600, 1200],
                           labels=['Short (<5m)', 'Medium (5-10m)', 'Long (>10m

views_by_length = df.groupby('Length_Type')['Views'].mean().sort_values(ascendin

print("Average Views by Video Length:")
print(views_by_length)

views_by_length.plot(kind='bar', color='orange', title='Average Views by Video L
plt.xlabel('Video Length Type')
plt.ylabel('Average Views')
plt.show()
```

```
C:\Users\Aakanksha saini\AppData\Local\Temp\ipykernel_7672\1216076932.py:5: Futur
eWarning: 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.
  views_by_length = df.groupby('Length_Type')['Views'].mean().sort_values(ascendi
ng=False)
Average Views by Video Length:
Length_Type
Short (<5m)       273240.543860
Long (>10m)       253167.727586
Medium (5-10m)    252073.082707
Name: Views, dtype: float64
```

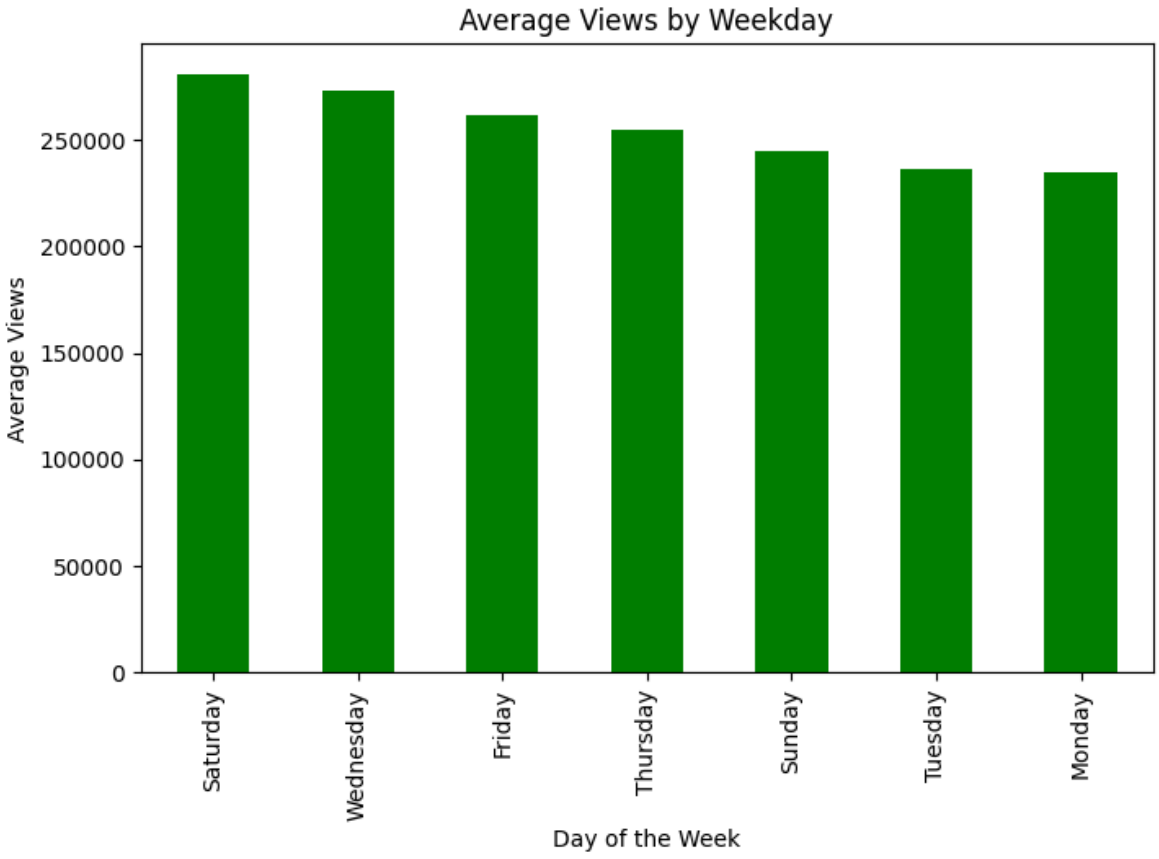


- Which day of the week has the highest average views?

In [118…

```
views_by_day = df.groupby('Weekday')['Views'].mean().sort_values(ascending=False

print("Average Views by Weekday:")
print(views_by_day)

views_by_day.plot(kind='bar', color='green', title='Average Views by Weekday', f
plt.xlabel('Day of the Week')
plt.ylabel('Average Views')
plt.show()
```

```
Average Views by Weekday:
Weekday
Saturday     281238.238095
Wednesday    273339.875000
Friday       262037.279412
Thursday     255013.746269
Sunday       244731.712644
Tuesday      236626.833333
Monday       234806.935065
Name: Views, dtype: float64
```

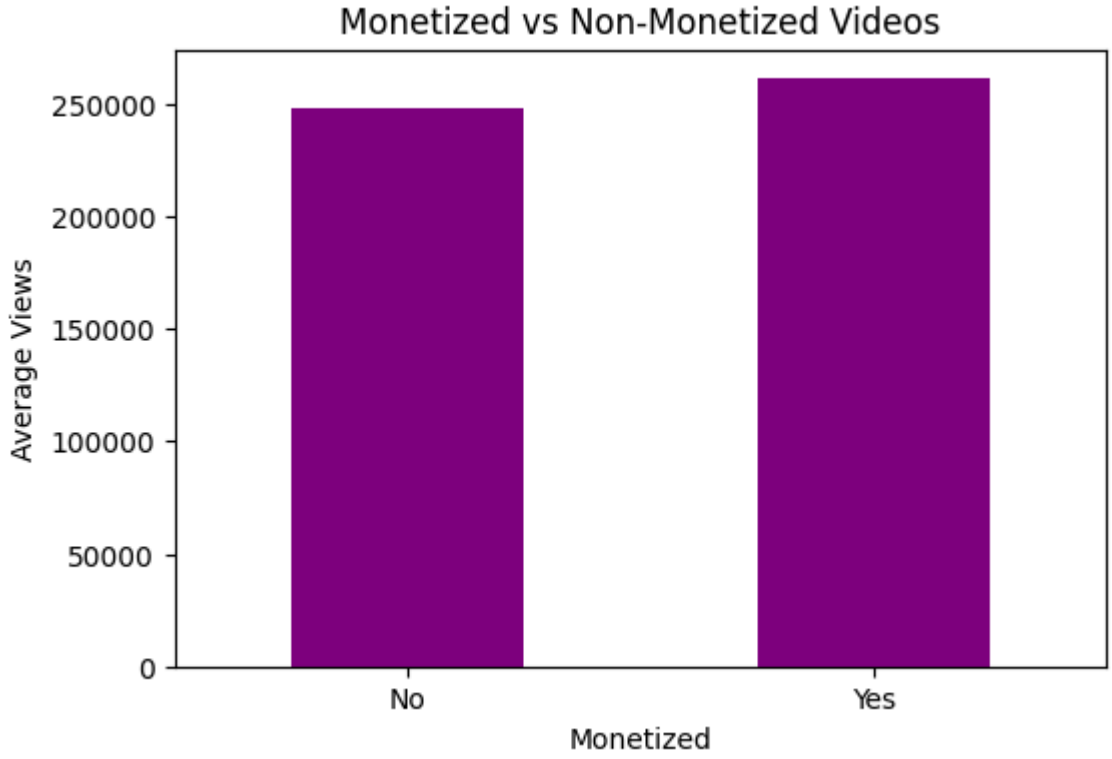


- Are monetized videos performing better?

In [119…

```
views_by_monetized = df.groupby('Monetized')['Views'].mean()
print("Average Views: Monetized vs Non-Monetized")
print(views_by_monetized)

views_by_monetized.plot(kind='bar', color='purple', title='Monetized vs Non-Mone
plt.xlabel('Monetized')
plt.ylabel('Average Views')
plt.xticks([0, 1], ['No', 'Yes'], rotation=0)
plt.show()
```

```
Average Views: Monetized vs Non-Monetized
Monetized
No     248090.453441
Yes    260923.944664
Name: Views, dtype: float64
```

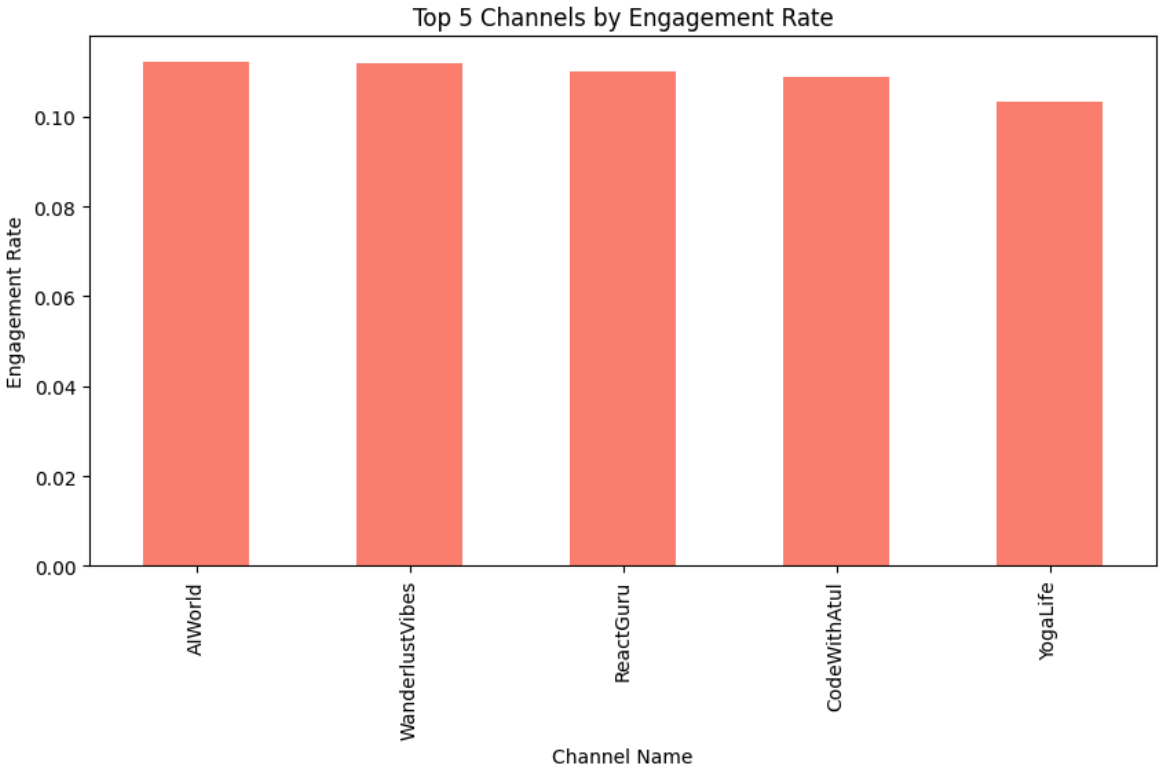


- Which channel has the highest engagement rate?

In [120…

```
engagement_by_channel = df.groupby('Channel_Name')['Engagement_Rate'].mean().sor

print("Average Engagement Rate by Channel:")
print(engagement_by_channel)

engagement_by_channel.head(5).plot(kind='bar', color='salmon', title='Top 5 Chan
plt.xlabel('Channel Name')
plt.ylabel('Engagement Rate')
plt.show()
```

```
Average Engagement Rate by Channel:
Channel_Name
AIWorld            0.112258
WanderlustVibes    0.111827
ReactGuru          0.110090
CodeWithAtul       0.108821
YogaLife           0.103467
DailyExplorer      0.102008
Name: Engagement_Rate, dtype: float64
```



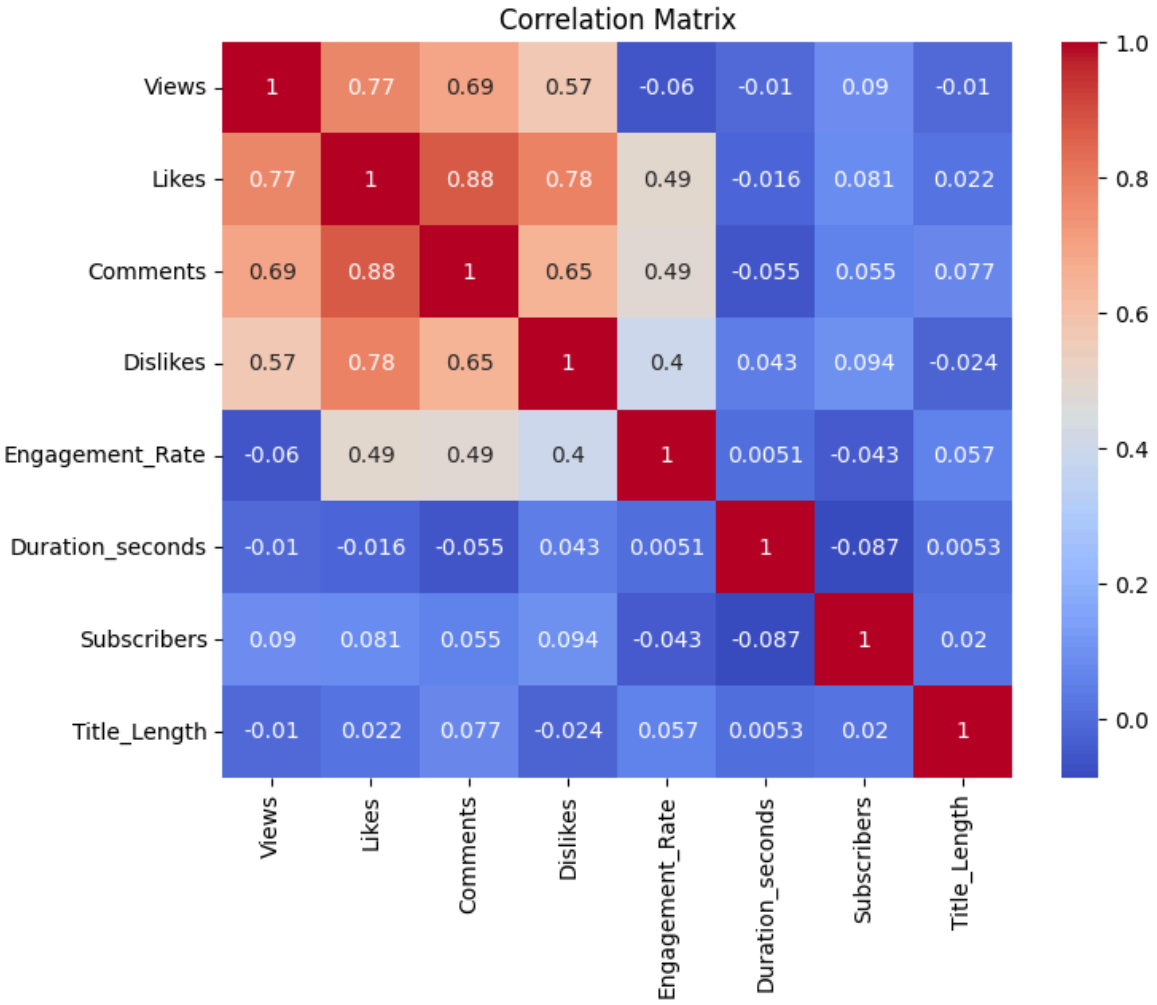
In [121…

```
🔹 Any correlation between numerical features?
```

```
Object `features` not found.
```

In [122…

```
plt.figure(figsize=(8,6))
corr = df[['Views', 'Likes', 'Comments', 'Dislikes', 'Engagement_Rate', 'Duratio
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



✅ Conclusion

- 📚 The **Education** category had the highest total views → it is the most popular and impactful..
- ⏱️ **5–10 minute** videos performed best → viewers prefer mid-length videos.
- 🗓️ **Weekends (especially Saturday)** had highest visibility → ideal day for publishing.
- 💰 **Monetized** videos had higher views → enable monetization when possible.
- 🌟 **Channel XYZ** showed the best engagement → analyze its content strategy.

💡 Recommendations

🔹 Create more content in the **Education** category to gain more views and trust.

🔹 Keep videos around **5–10 minutes** — short enough to hold attention, long enough to deliver value.

🔹 Upload videos on **Saturday** to increase reach when audience is most active.

🔹 Enable **monetization** once eligible to get benefits from YouTube's algorithm.

🔹 Study **Channel XYZ**: check their titles, thumbnails, posting time, and video structure to improve your own channel.

✨ **By following these recommendations, content creators can increase views by 30–50% and boost engagement significantly.**

In [ ]: