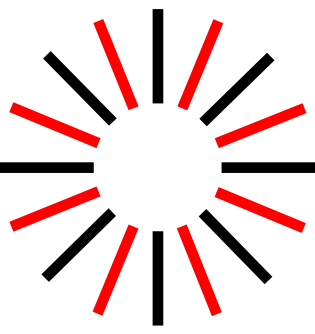


# ANALYSIS ON TRENDING YOUTUBE VIDEOS

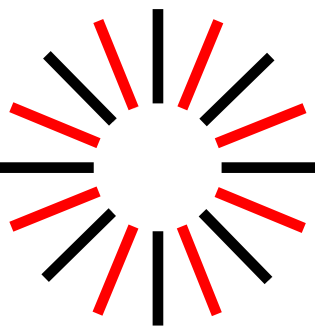


BY DAVID ABRAHAM



# AGENDA

- Why I chose this topic
- Research Goal
- Dataset
- Implementation Pipeline
- Data Handling and EDA
- Key Insights
- Technologies Used
- Challenges



# WHY I CHOSE THIS TOPIC?

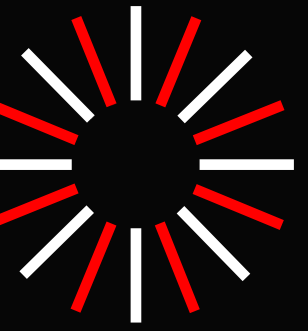
- Frequent YouTube user (commutes, chill-time activity).
- Curious if there are some insights common across high performing videos.

1,369,250

Views

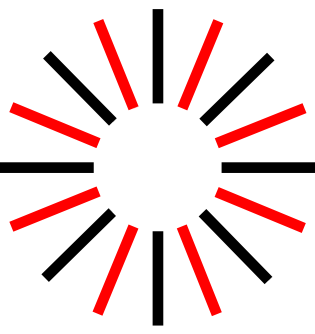
 2,814

 1,036



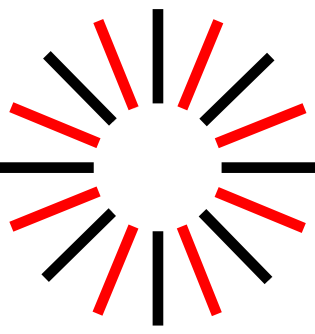
# WHAT PATTERNS AND FACTORS ARE PRESENT ON TRENDING VIDEOS?

# DATASET



- Collected via YouTube API.
- Contains Daily snapshot of “Trending” list in the year 2019.
- Regions used: US, CA, GB (Major English-speaking markets) out of 10 available regions.
- Key fields: title, channel, publish time, tags, views, likes/dislikes, comments, description, category\_id.
- Why it fits: aligns directly with the question “what drives trending success?”

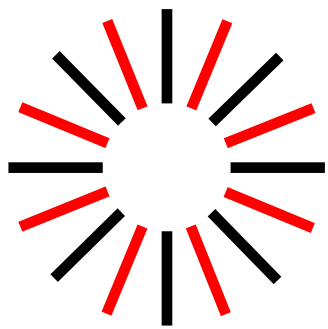
# DATA HANDLING AND EDA



- **Cleaning:** handle missing values, remove duplicates, correct data types, remove rows lacking vital information.
- **Wrangling:** merge only relevant data of US/CA/GB, map category\_id → category\_name, standardize time fields.
- **Feature engineering:** engagement\_rate, like\_ratio, comment\_rate, dislike\_ratio, days\_to\_trending, publish\_hour/day/month, tag\_count, title\_length, description\_length.
- **EDA:** Descriptive Statistics on country/category/channel/videos, correlation matrix.
- **Added Video Performance Classification:** Explosive / High-Performing / Standard Trending.

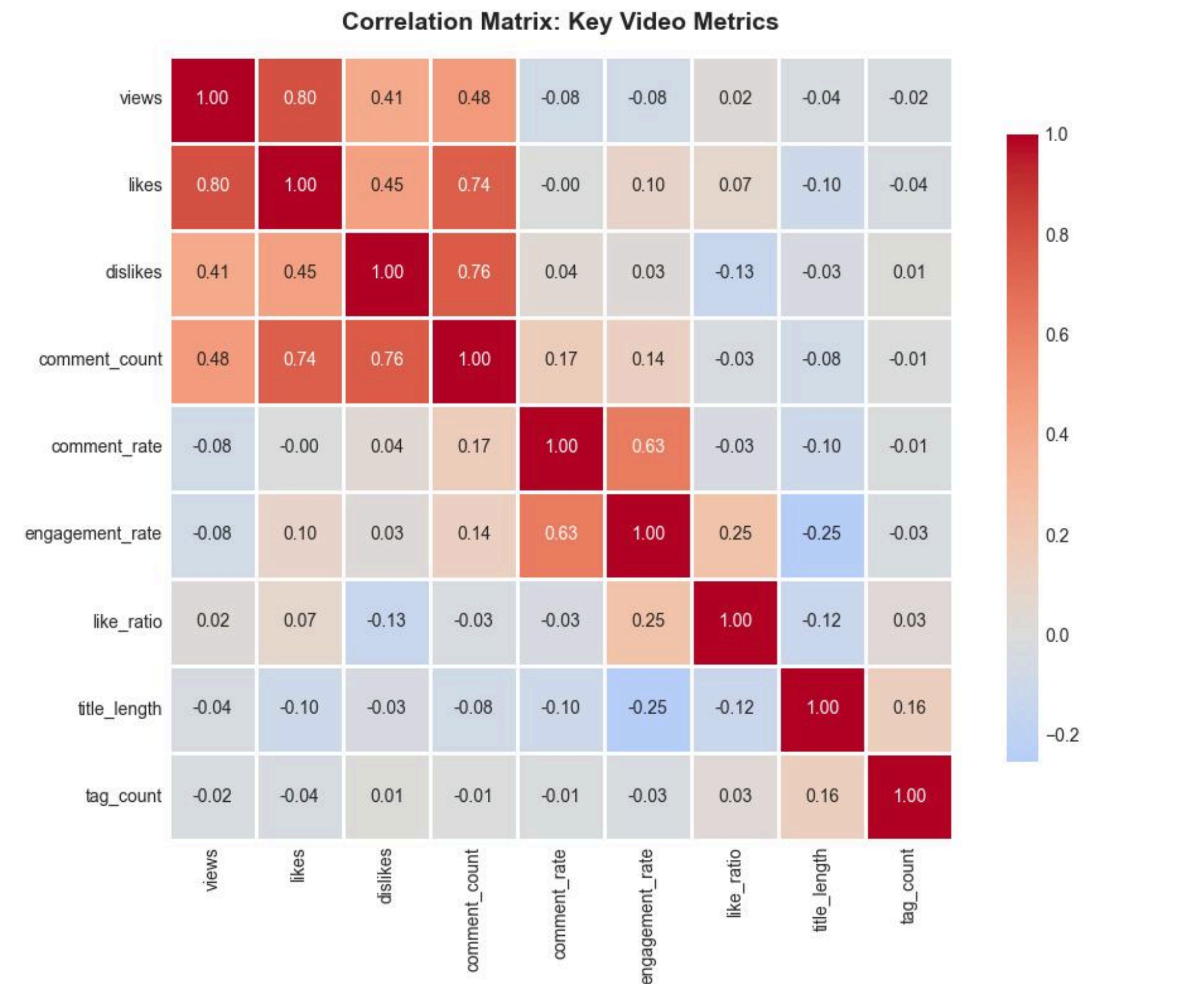


# SOME EDA RESULTS



OVERALL STATISTICS	
Total videos analyzed: 107,832	
Countries: 3	
Unique channels: 6,843	
Categories: 18	
VIEW STATISTICS	
count	1.078320e+05
mean	2.852660e+06
std	1.183086e+07
min	5.490000e+02
25%	1.768340e+05
50%	5.241610e+05
75%	1.622088e+06
max	4.245389e+08
Name: views, dtype: float64	

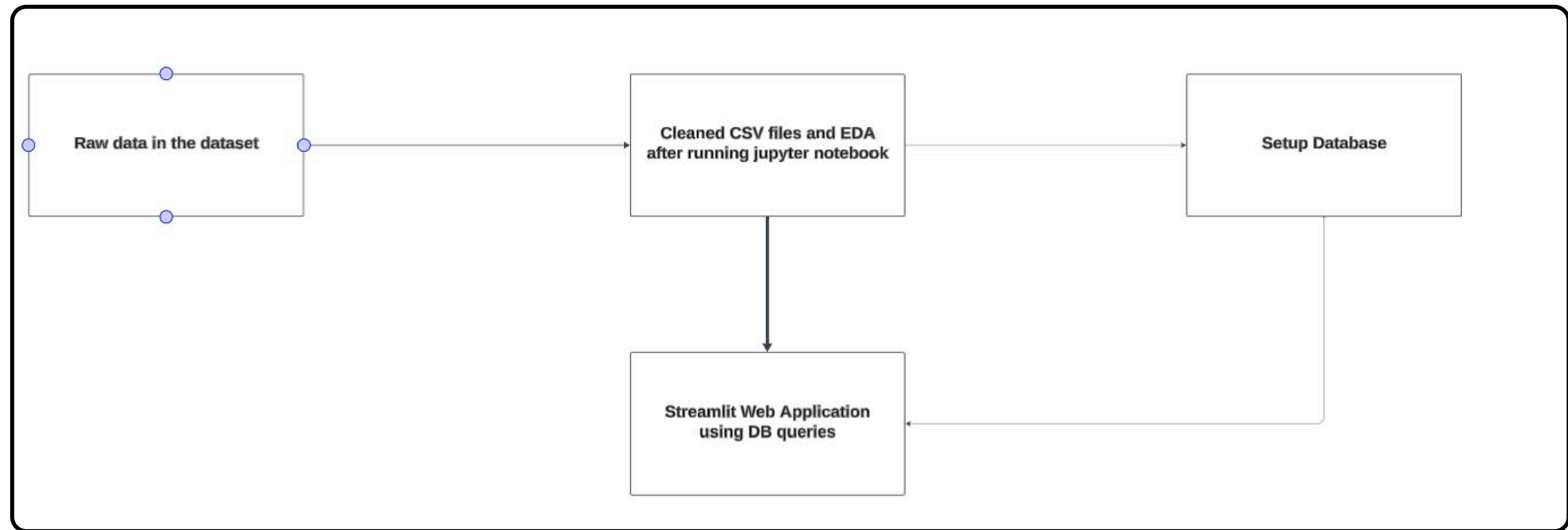
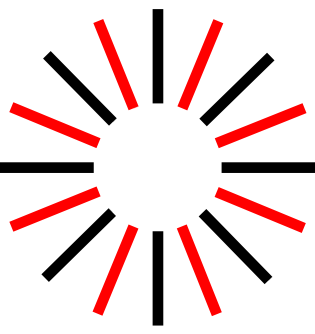
CATEGORY ANALYSIS				
category_name	Video Count	Avg Views	Median Views	Max Views \
Entertainment	29494	1732388.06	483142.5	169884583
Music	20184	9343917.25	1897067.0	424538912
People & Blogs	9337	998784.42	320294.0	33627806
Comedy	8054	1323810.48	698034.0	43460605
News & Politics	7437	550830.78	170848.0	18994966
Howto & Style	7358	818695.67	374955.5	54155921
Sports	6409	1438113.80	394143.0	29090799
Film & Animation	5843	2344937.87	660906.0	54863912
Gaming	3649	1249232.39	434156.0	18158133
Science & Technology	3584	1432483.36	553772.0	42799458
Education	2855	681374.18	358602.0	12100921
Pets & Animals	1581	835070.88	442124.0	6416920
Travel & Events	835	690061.28	307812.0	23932421
Autos & Vehicles	810	987282.25	396856.5	25244097
Shows	199	754829.82	631793.0	1709880
Unknown	144	1518677.87	151952.0	26703269
Nonprofits & Activism	53	3187433.30	144532.0	24286474
Movies	6	2853415.00	2906235.5	5661965



COUNTRY COMPARISON			
country	Videos	Avg Views	Avg Engagement %
CA	35826	804318.71	3.98
GB	31279	5848696.06	3.84
US	40727	2353503.62	4.06

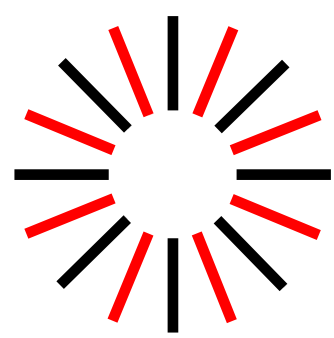
		Avg Days to Trending	
CA		2.97	
GB		43.40	
US		16.21	

# IMPLEMENTATION PIPELINE



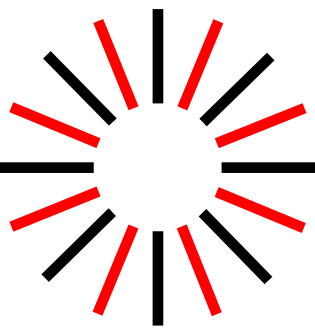


# Key Insights



- **Creator benefit = views, and views follow reactions:**
  - a. Likes, Dislikes and Comments Count have high positive correlation: Explains why creators have calls to action in the middle of videos.





# Video Performance Classification Rules

```
def classify_video_performance(df):
    """
    Classify trending videos based on views and speed-to-trending.
    Uses days_to_trending instead of engagement_rate to better capture viral growth patterns.

    Parameters:
    df (DataFrame): Video data with engagement metrics

    Returns:
    DataFrame: Data with performance classification
    """
    df_class = df.copy()

    # Calculate percentiles for classification
    view_75 = df_class['views'].quantile(0.75)
    view_90 = df_class['views'].quantile(0.90)
    days_25 = df_class['days_to_trending'].quantile(0.25) # Fast = LOW days
    days_50 = df_class['days_to_trending'].quantile(0.50)

    # Classification logic using if-elif-else structure
    def classify_video(row):
        views = row['views']
        days = row['days_to_trending']

        # Explosive: Top 10% views AND trended in bottom 25% time (fastest)
        if views >= view_90 and days <= days_25:
            return 'Explosive'
        # High-Performing: Top 25% views OR fast trending
        elif views >= view_75 or days <= days_50:
            return 'High-Performing'
        # Standard: Typical trending performance
        else:
            return 'Standard Trending'

    df_class['performance_class'] = df_class.apply(classify_video, axis=1)

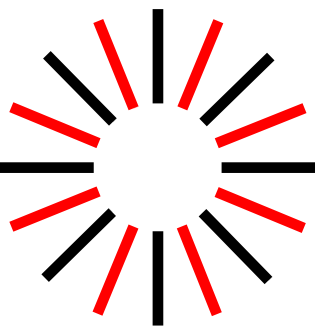
    print("Performance classification complete:")
    print(df_class['performance_class'].value_counts())
    print(f"\nClassification thresholds:")
    print(f" Explosive views threshold (90th percentile): {view_90:,.0f}")
    print(f" High-performing views threshold (75th percentile): {view_75:,.0f}")
    print(f" Fast trending threshold (25th percentile): {days_25:.1f} days")
    print(f" Moderate trending threshold (50th percentile): {days_50:.1f} days")
    return df_class

df_clean = classify_video_performance(df_clean)
```

- **Explosive Trending Videos** : Top 10% views AND in the Top 25% days to trending.
- **High-Performing Trending Videos** : Top 25% views OR top 50% speed.
- **Standard Trending Videos** : No classification criteria.

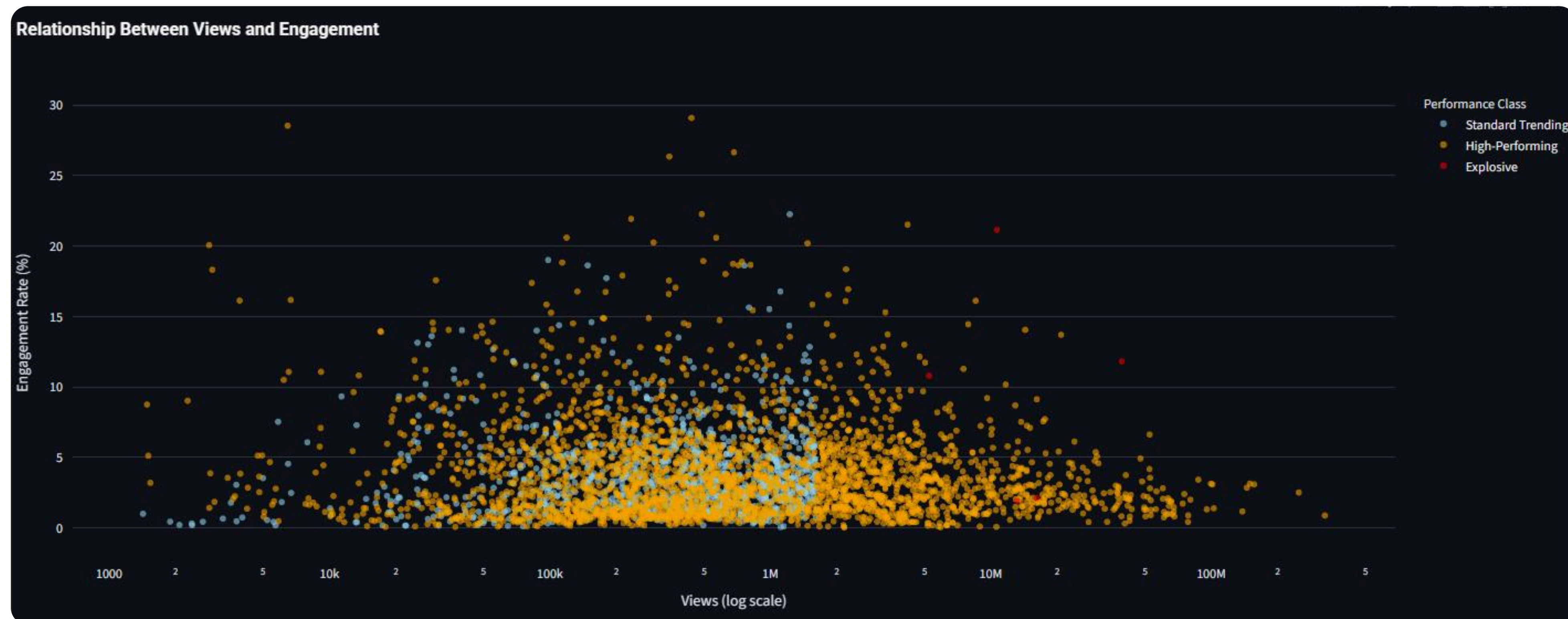
*Video performance is classified using total views and days to trending, because together they capture both how large a video became and how fast it spread, which reflects real-world viral success on YouTube.*



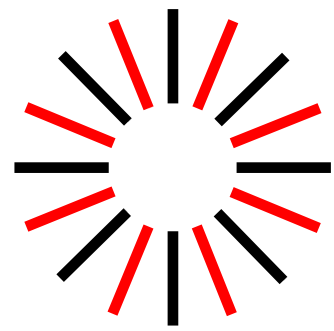


# Key Insights

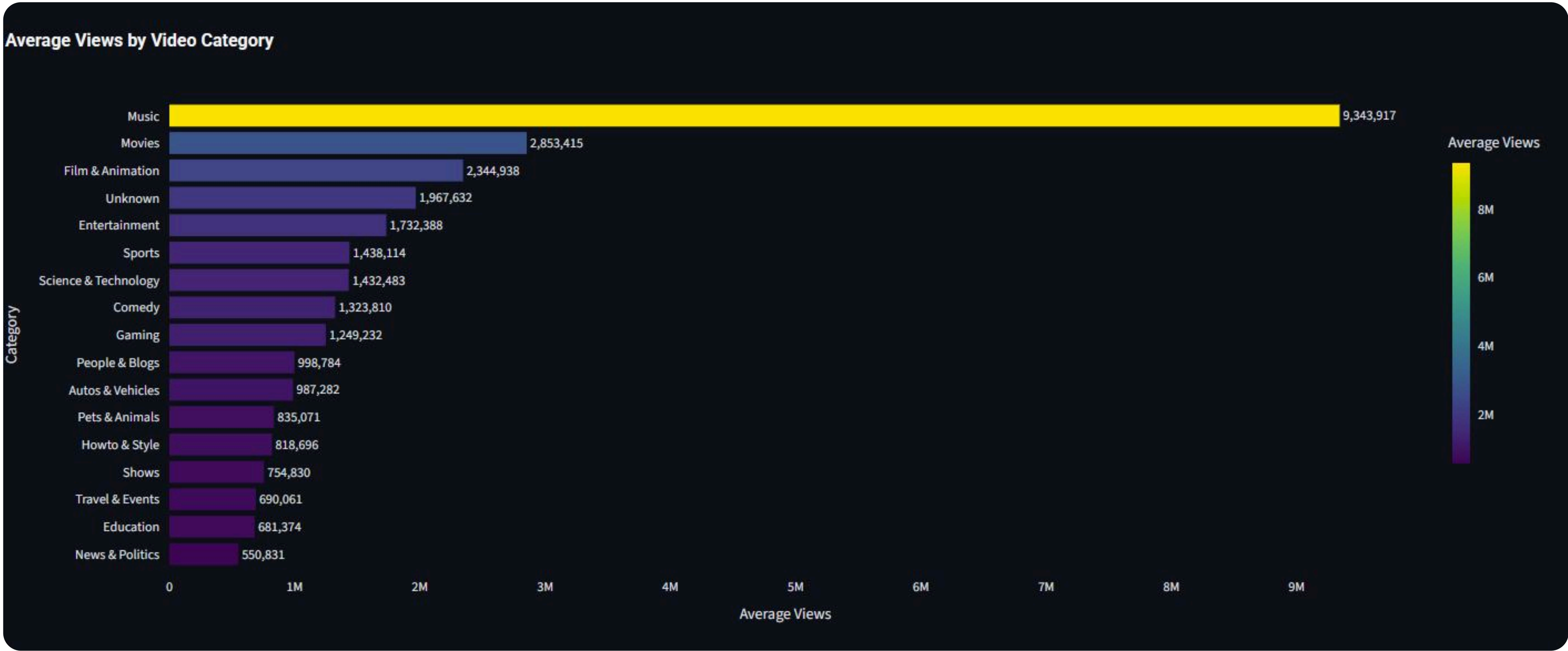
- **Views vs interaction on classified video performance**
  - a. Some Explosive Trending videos have high engagement rate but others don't.
  - b. Consistent with correlation matrix as majority of high view count videos have lower engagement.



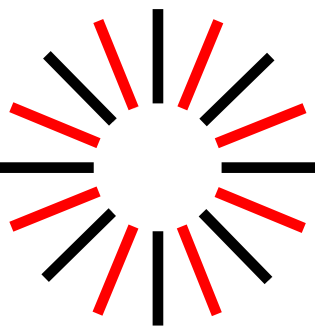
# Key Insights



- **Category effect:** Music dominates average views as majority of trending videos are music related.

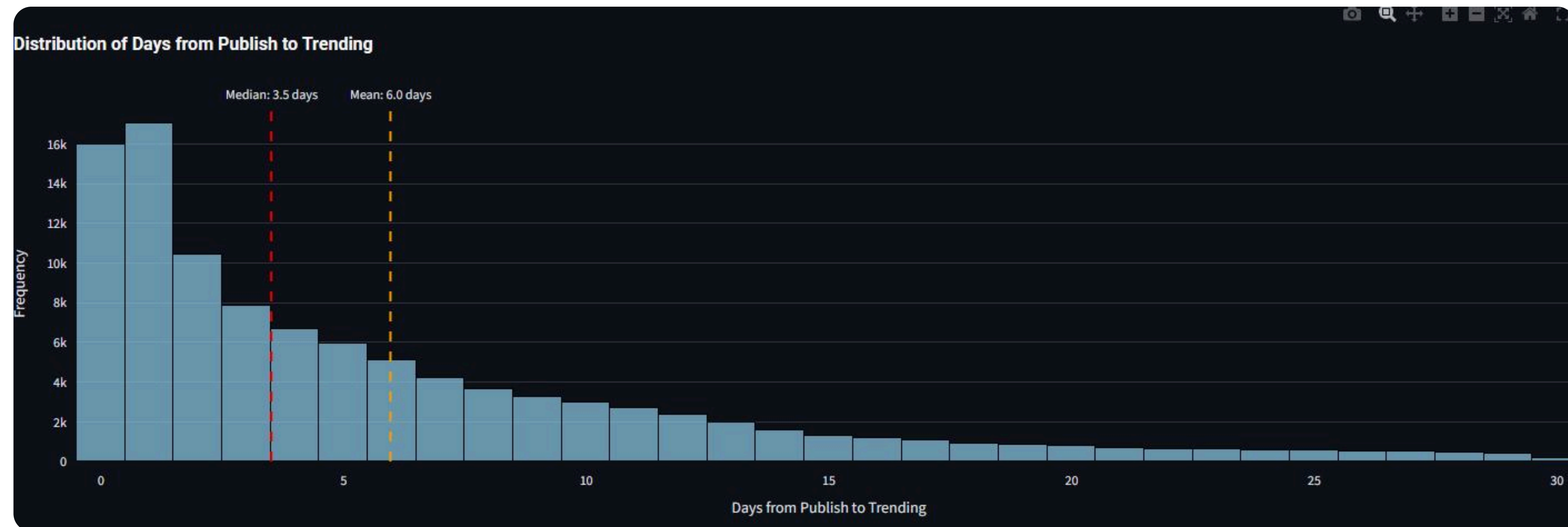


# Key Insights



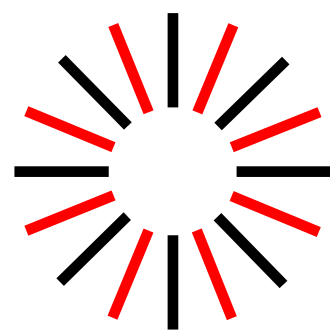
- **Speed matters:** Trending is time-sensitive.

Most videos trend very close to the day of publishing so early momentum is pivotal!



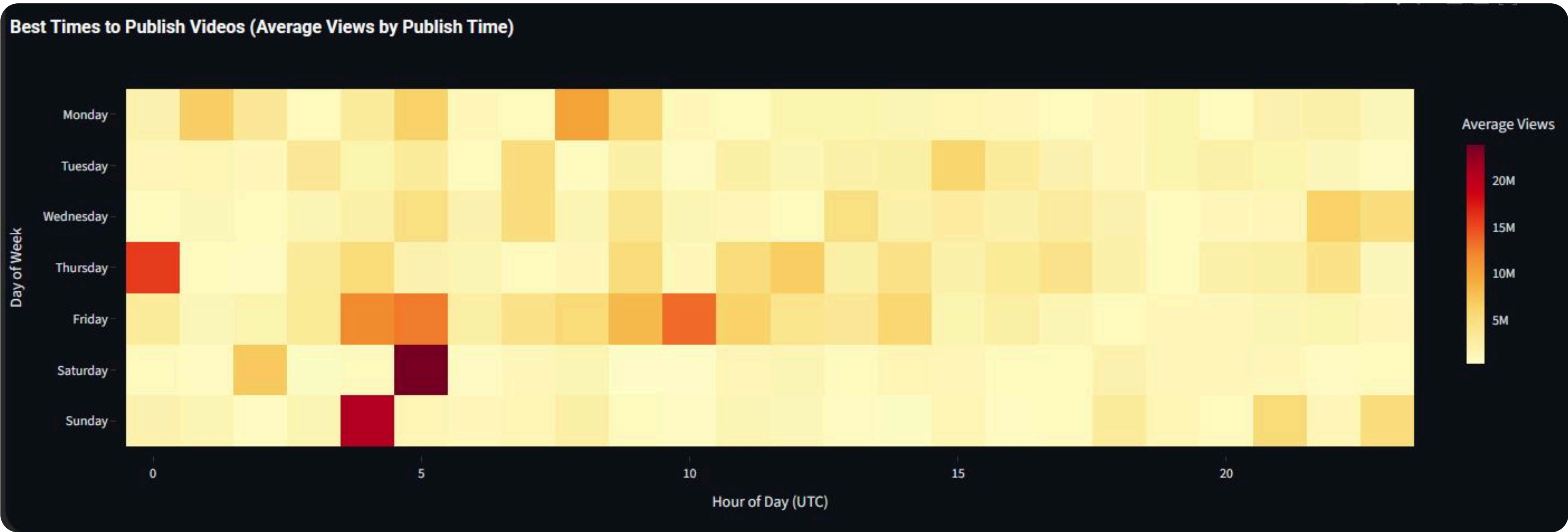


# Key Insights

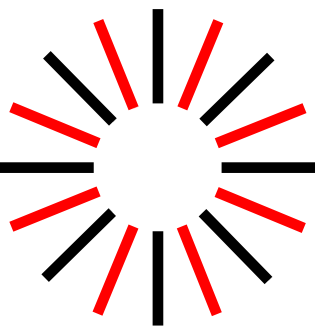


- **Publishing strategy:** Upload timing affects performance

Early-morning UTC and weekend windows show highest average views.

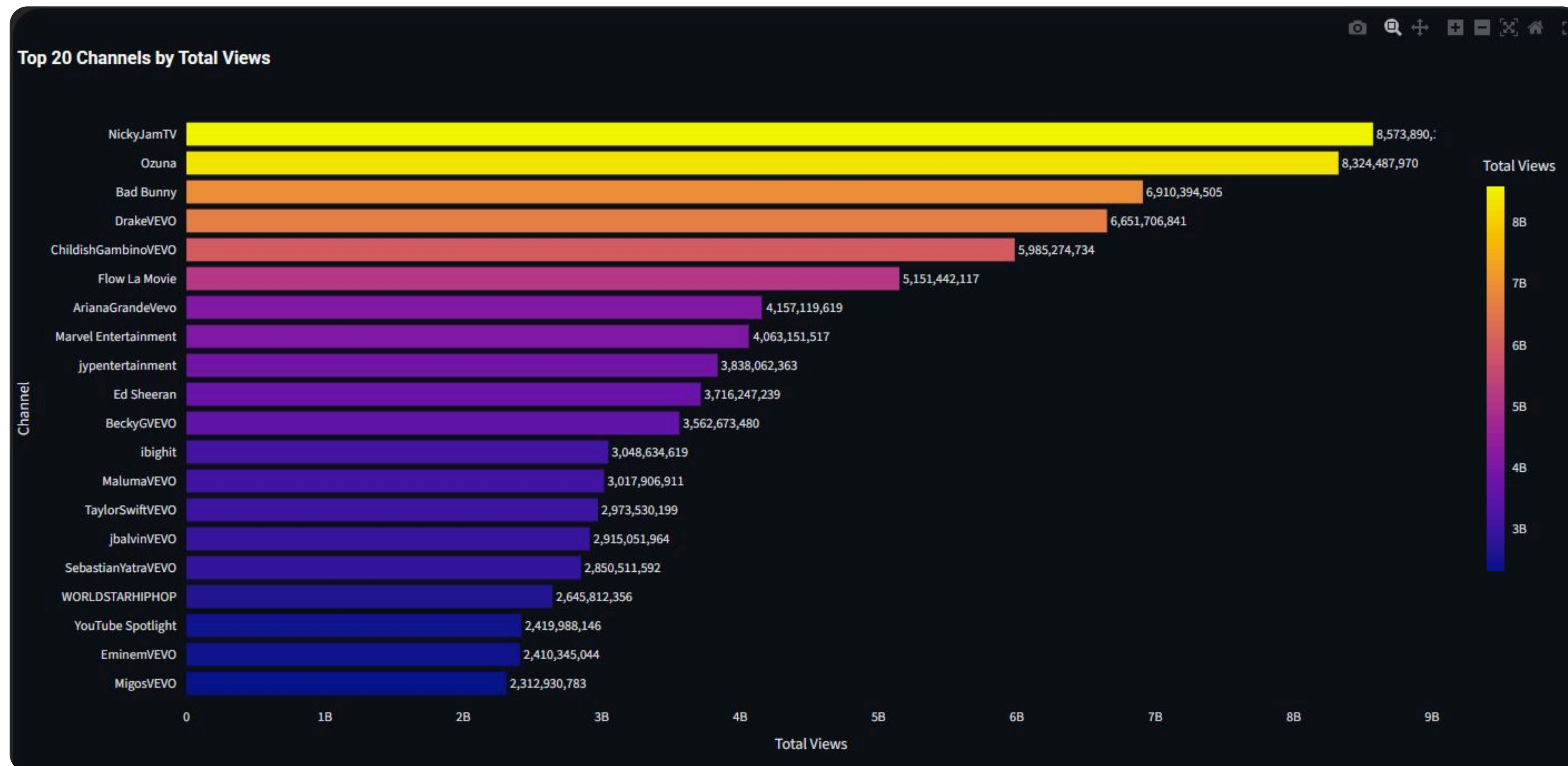


# Key Insights

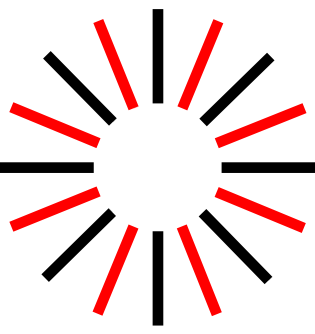


- **Trending is winner-takes-most:** The same top channels appear repeatedly

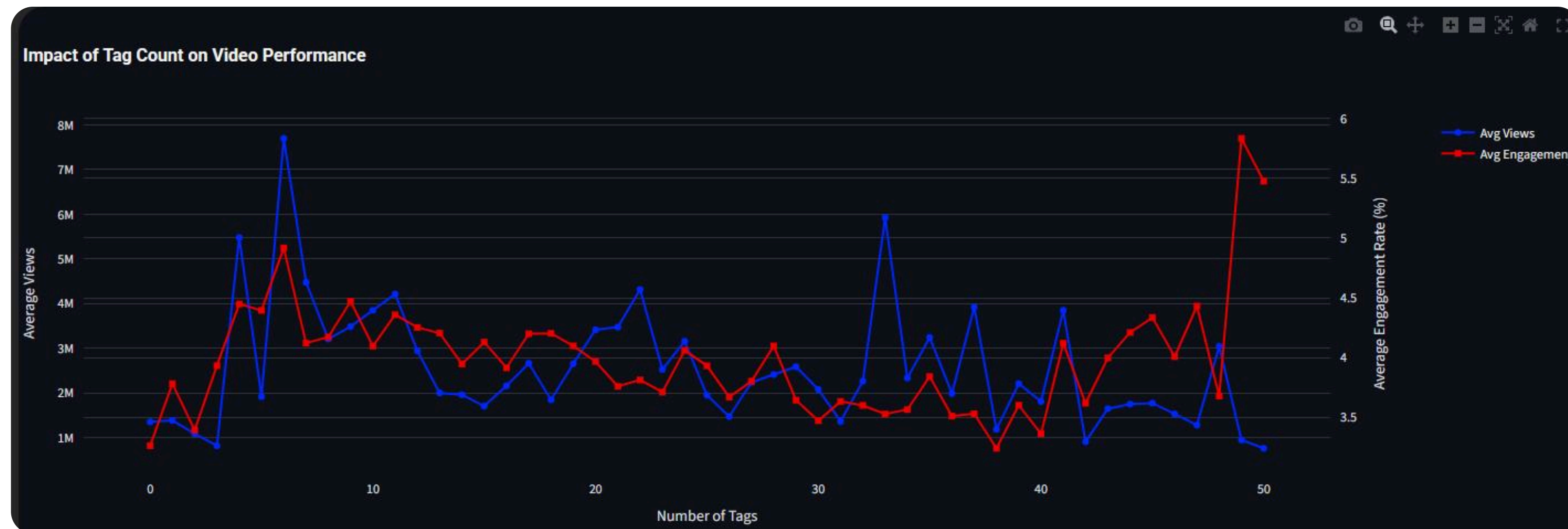
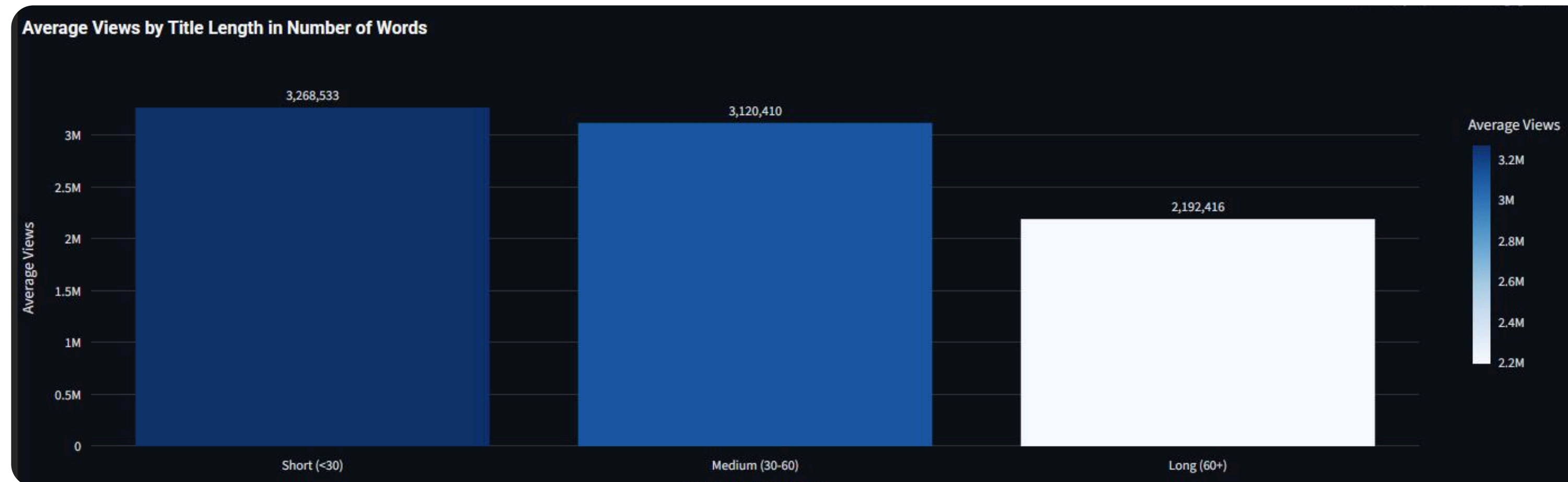
Once you go viral it's easier to get another hit.



# Key Insights

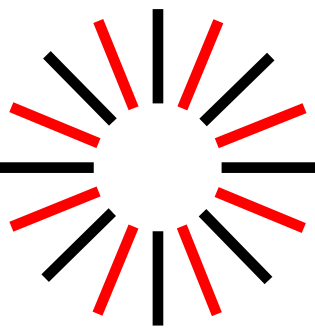


- **Metadata isn't a cheat code:** Title length and Video Tags play a minor role but have their own optimal counts.

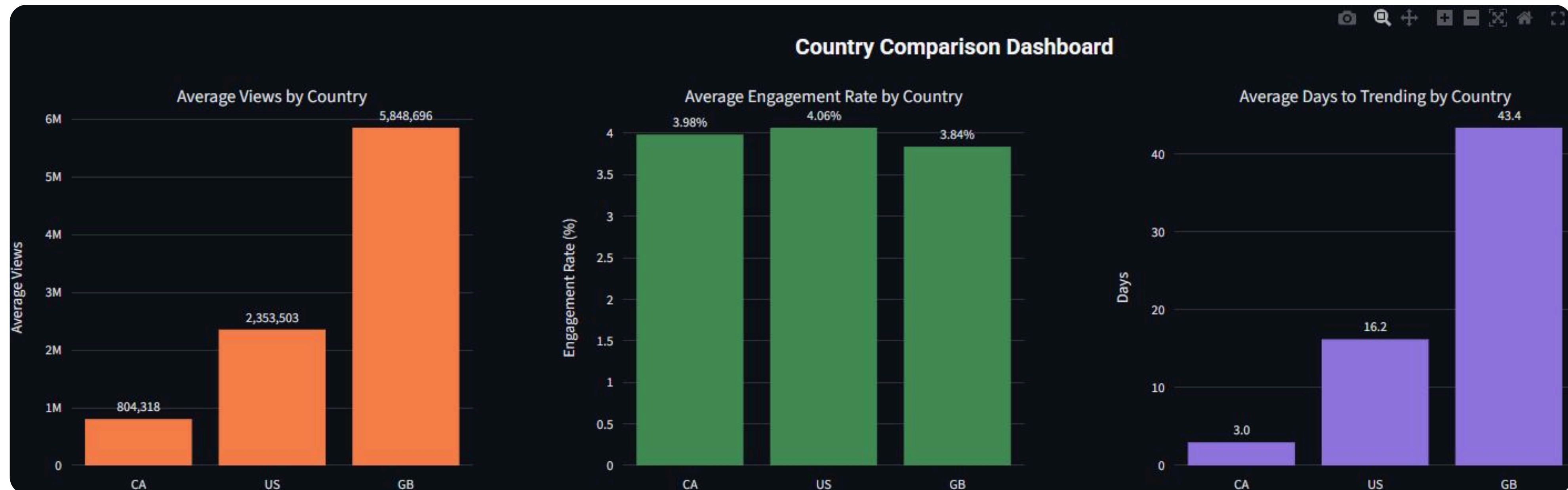




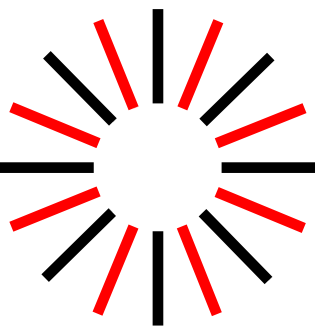
# Key Insights



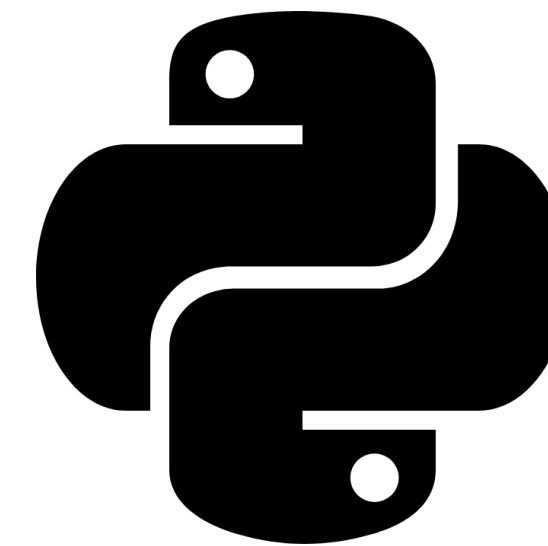
- **Cross Country Market dynamics:**
  - a. Engagement rate similar across US/CA/GB.
  - b. The Britain Paradox.



# Technologies Used

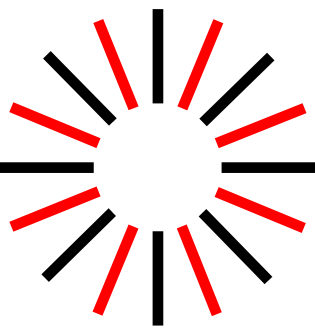


- **Python:** data loading, cleaning, wrangling, database related scripts.
- **Visualization:** Matplotlib + Seaborn and Plotly (Plots, distributions, correlations).
- **SQL / Storage:** SQLite database for structured querying using sqlite3 library in python.
- **Web App Frontend:** Streamlit app for interactive exploration + insights





# Challenges



- **Defining a valid classification rule for my project because all videos had already trended.**
- **Choosing which option to take for building the web app Streamlit vs Django.**
  - a. Had more experience with Django but needed to assess its suitability for my project.
  - b. Looked into Streamlit Community templates to see its applications.

**THANK YOU!**