Fr. Conceicao Rodrigues College of Engineering

Father Agnel Ashram, Bandstand, Bandra –west, Mumbai-50

Department of Computer Engineering

SOCIAL MEDIA ANALYTICS LAB

Experiment No: 4

Aim: Exploratory Data Analysis and visualization of Social Media Data for business.

Objective: To capture social media data and perform different visualizations.

Lab outcomes:
At the end of this lab session, students will be able to…

1. Perform visualization of Social Media Data

Theory

● Exploratory Data Analysis (EDA) is usually the first step when you have data in hand and want to analyze it.
● In EDA, there is no hypothesis and no model. You are finding patterns and truth from the data.
● EDA is crucial for data science projects because it can:
○ Help you gain intuition about the data;
○ Make comparisons between distributions;
○ Check if the data is on the scale you expect;
○ Find out where data is missing or if there are outliers;
○ Summarize data, calculate the mean, min, max, and variance.
● The basic tools of EDA are plots, graphs, and summary statistics.

Student's Task:

1. Load the csv of your choice.

2. Transform the data as per the need of columns.

3. Generate the following visualizations

3.1 Pie Chart

3.2 Stacked Area Chart

3.3 Ribbon Chart

3.4 Tree Map

3.5 Funnel

3.6 Line Chart

4. Article Discussion

Kordzadeh, Nima, and Diana K. Young. "How social media analytics can inform content strategies." *Journal of Computer Information Systems* 62.1 (2022): 128-140.

Answer the following questions based on above article.

4.1 How peer brand selection is suggested in this article?

In the article, the process of peer brand selection for social media analytics is outlined as follows:

Contextual, Organizational, and SM Characteristics: The selection of peer brands involves considering brands with similar contextual, organizational, and social media characteristics. Contextual characteristics include industry and sector, organizational characteristics involve target market, products and services, size, and strategic directions, while social media characteristics include goals, scale, and scope of social media use.

Industry Relevance: Brands should be selected based on their relevance to the industry and sector. For example, a coffee shop looking to establish a presence on Twitter should analyze the Twitter activities of similar coffee shops in terms of size, market, products, and social media activities.

Output: The output of the peer brand selection phase is a list of brands that are suitable for inclusion in the analysis. These peer brands serve as sources of data for benchmarking and learning purposes.

By selecting peer brands with similar characteristics and relevance, organizations can effectively benchmark their social media performance and derive insights to enhance their own content strategies

4.2 What are the different ways of data collection & preprocessing done?

The document outlines various methods for data collection and preprocessing in the context of social media analytics:

Data Collection:

Web Scraping Tools: Automated tools for extracting data from social media platforms.

Application Program Interfaces (APIs): Interfaces provided by social media platforms for accessing data.

Manual Techniques: Manually collecting data from social media platforms.

Data Preparation:

Post Characteristics: Collecting data points such as post text, date and time of the post, number of pictures and videos, number of likes, comments, and shares.

Cleaning the Dataset: Removing irrelevant or duplicate data points to ensure data quality.

Content Analysis:

Textual Analysis: Identifying topics discussed in the data through manual, automated, or hybrid approaches.

Manual Coding: Systematically coding data points, reviewing emerging concepts, and grouping similar concepts into topics.

Automated Analysis: Using algorithms like Latent Dirichlet Allocation (LDA) to identify topics efficiently in large datasets.

Challenges and Considerations:

Resource Availability: Choosing data collection and analysis techniques based on available resources.

Volume of Data: Selecting methods that can handle the volume of data effectively.

Frequency of Analysis: Considering how frequently data analysis needs to be performed.

4.3 What is engagement analysis and why is it important strategy in SM analytics?

Engagement analysis in social media analytics involves quantitatively assessing the influence of post characteristics and topics on user engagement metrics such as likes, comments, and shares. It is an important strategy in social media analytics for several reasons:

Understanding User Behavior: By analyzing user engagement metrics, organizations can gain insights into how users interact with their social media content. This understanding can help in tailoring content to better resonate with the target audience.

Measuring Effectiveness: Engagement analysis allows organizations to measure the effectiveness of their social media posts in terms of generating likes, comments, and shares. This helps in evaluating the impact of different content strategies on user engagement.

Identifying Trends: By analyzing engagement metrics over time, organizations can identify trends in user behavior and preferences. This information can be used to adapt content strategies to align with current trends and maximize engagement.

Optimizing Content Strategy: Engagement analysis provides data-driven insights that can be used to optimize content strategies. By identifying which post characteristics and topics drive higher engagement, organizations can refine their content strategy to improve overall performance.

Benchmarking Performance: Comparing engagement metrics with those of peer brands or industry benchmarks can help organizations assess their performance and identify areas for improvement. This benchmarking process can guide strategic decision-making in social media marketing.

4.4 Discuss the case study of healthcare industry proposed in the article.

The case study presented in the article focuses on the healthcare industry and demonstrates the application of the proposed analytics-driven process for social content strategy development and improvement. Here are key points from the case study:

Health Information Sharing: Healthcare institutions use social media not only for marketing but also to share health information with the community. Enhanced user engagement can promote health information sharing through likes and shares.

Utilizing Data Analytics: The healthcare industry is known to have lagged behind in effective application of information technology and data analytics. The case study aims to show how healthcare institutions can utilize data analytics and adopt social media channels effectively.

Content Typology: The authors developed a comprehensive typology of content posted by hospitals on Facebook. This typology likely includes categories such as disease information, medical conditions, health tips, patient stories, and organization news.

Iterative Process: The case study emphasizes the importance of iterative processes in content strategy development. By periodically analyzing and adapting content strategies based on data insights, healthcare institutions can improve their social media performance over time.

Implementation and Evaluation: The case study suggests implementing actionable guidelines derived from engagement analysis to adjust current social media content strategies. By evaluating the impact of these adjustments on engagement outcomes, healthcare institutions can refine their strategies further.

Continuous Improvement: By continuously analyzing peer brands' activities, learning from their own implementation experiences, and incorporating new variables in the analysis, healthcare institutions can enhance their social media performance and adapt to changes in the social media ecosystem.

Overall, the case study in the healthcare industry serves as a practical demonstration of how the analytics-driven process can be applied to improve user engagement, promote health information sharing, and enhance social media performance in a specific industry context.

[50]:
```
#### This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/netfllix-weekly-views-data/all-weeks-global.csv
```

[51]:
```
import pandas as pd

file_path = "/kaggle/input/netfllix-weekly-views-data/all-weeks-global.csv"
df = pd.read_csv(file_path, encoding="latin 1")

print("Info:", df.info())

print("\nDescribe:", df.describe())

print("\nShape:", df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5280 entries, 0 to 5279
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   week                        5280 non-null   object
 1   category                    5280 non-null   object
 2   weekly_rank                 5280 non-null   int64
 3   show_title                  5280 non-null   object
 4   season_title                2543 non-null   object
 5   weekly_hours_viewed         5280 non-null   int64
 6   runtime                     1200 non-null   float64
 7   weekly_views                1200 non-null   float64
 8   cumulative_weeks_in_top_10  5280 non-null   int64
 9   is_staggered_launch         5280 non-null   bool
 10  episode_launch_details      46 non-null     object
dtypes: bool(1), float64(2), int64(3), object(5)
memory usage: 417.8+ KB
Info: None

Describe:        weekly_rank  weekly_hours_viewed      runtime  weekly_views  \
count  5280.000000         5.280000e+03  1200.000000  1.200000e+03
mean      5.500000         1.876423e+07     3.596417  4.512250e+06
std       2.872553         2.709274e+07     3.080800  4.925899e+06
min       1.000000         7.000000e+05     0.000000  6.000000e+05
25%       3.000000         6.450000e+06     1.666700  1.875000e+06
50%       5.500000         1.155500e+07     2.116700  3.000000e+06
75%       8.000000         2.082750e+07     4.929200  5.125000e+06
max      10.000000         5.717600e+08    20.300000  4.490000e+07

       cumulative_weeks_in_top_10
count                 5280.000000
mean                     3.117992
std                      3.279708
min                      1.000000
25%                      1.000000
50%                      2.000000
75%                      4.000000
max                     30.000000

Shape: (5280, 11)
```

[52]:
```
df.head()
```

[52]:

	week	category	weekly_rank	show_title	season_title	weekly_hours_viewed	runtime	weekly_views	cumulative_weeks_in_top_10	is_staggered_launch	episode_launch_details
0	2024-01-07	Films (English)	1	The Equalizer 3	NaN	26800000	1.8167	14800000.0	1	False	NaN
1	2024-01-07	Films (English)	2	Rebel Moon ? Part One: A Child of Fire	NaN	25100000	2.2667	11100000.0	3	False	NaN
2	2024-01-07	Films (English)	3	Leave the World Behind	NaN	18700000	2.3667	7900000.0	5	False	NaN
3	2024-01-07	Films (English)	4	Exodus: Gods and Kings	NaN	18600000	2.5000	7400000.0	1	False	NaN
4	2024-01-07	Films (English)	5	Aquaman	NaN	16800000	2.3833	7000000.0	1	False	NaN

```
df['category'].value_counts()
```

[53]:
```
category
Films (English)        1320
Films (Non-English)    1320
TV (English)           1320
TV (Non-English)       1320
Name: count, dtype: int64
```

[54]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px

from wordcloud import WordCloud
```

[57]:
```
# df = df.dropna()
df.isnull().sum()
```

[57]:
```
week                        0
category                    0
weekly_rank                 0
show_title                  0
weekly_hours_viewed         0
cumulative_weeks_in_top_10  0
is_staggered_launch         0
dtype: int64
```

[56]:
```
df = df.drop(["season_title", "runtime", "weekly_views", "episode_launch_details"], axis=1)
```

[58]:
```
def uniqueColdata(colname):
    colcount = len(df[colname].unique())
    coluniqueData = df[colname].value_counts()
    print("colname : ", colname, "colcount : ",colcount ,"coluniqueData : ", coluniqueData)
```

[59]:
```
uniqueColdata("cumulative_weeks_in_top_10")
```

```
colname :  cumulative_weeks_in_top_10 colcount :  30 coluniqueData :  cumulative_weeks_in_top_10
1     1898
2     1266
3      719
4      430
5      272
6      177
7      119
8       88
9       56
10      47
11      37
12      28
13      26
14      20
15      16
16      14
18      10
17      10
19       9
20       8
21       7
22       5
23       4
25       3
24       3
28       2
27       2
26       2
30       1
29       1
Name: count, dtype: int64
```

[60]:
```
df.dtypes
```

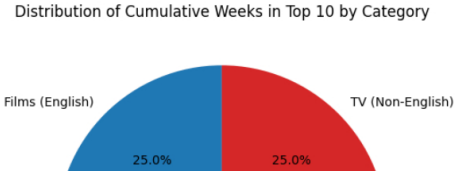
[60]:
```
week                        object
category                    object
weekly_rank                  int64
show_title                  object
weekly_hours_viewed          int64
cumulative_weeks_in_top_10   int64
is_staggered_launch           bool
dtype: object
```

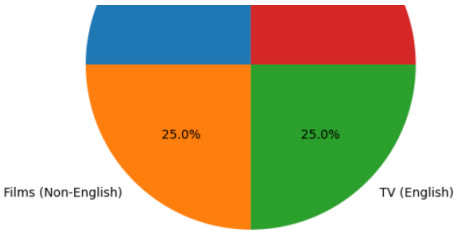
[61]:
```
top_20_most_viewed = df.nlargest(20, 'weekly_hours_viewed')[['show_title', 'weekly_hours_viewed']]
print(top_20_most_viewed)
```

```
            show_title  weekly_hours_viewed
4750          Squid Game            571760000
4790          Squid Game            448730000
4710          Squid Game            412940000
2300           Wednesday            411290000
2340           Wednesday            341230000
3340    Stranger Things            335010000
3180    Stranger Things            301280000
2660              DAHMER            299840000
3380    Stranger Things            286790000
2260           Wednesday            269670000
4670          Squid Game            258840000
3700           Bridgerton            251740000
20          Fool Me Once            238200000
4030  All of Us Are Dead            236230000
1620     The Night Agent            216390000
2620              DAHMER            205330000
4910         Money Heist            201910000
2700              DAHMER            196200000
3940      Inventing Anna            195970000
3740           Bridgerton            193020000
```

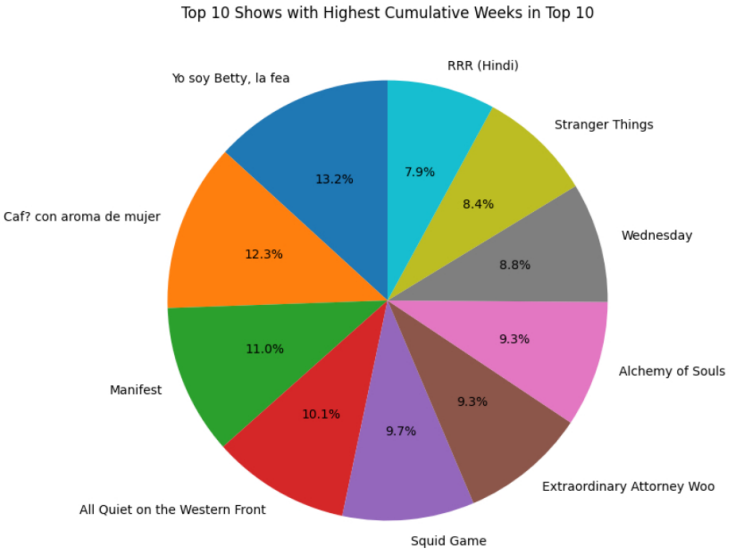
Pie Chart

```
category_counts = df['category'].value_counts()
plt.figure(figsize=(10, 6))
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Cumulative Weeks in Top 10 by Category')
plt.show()
```



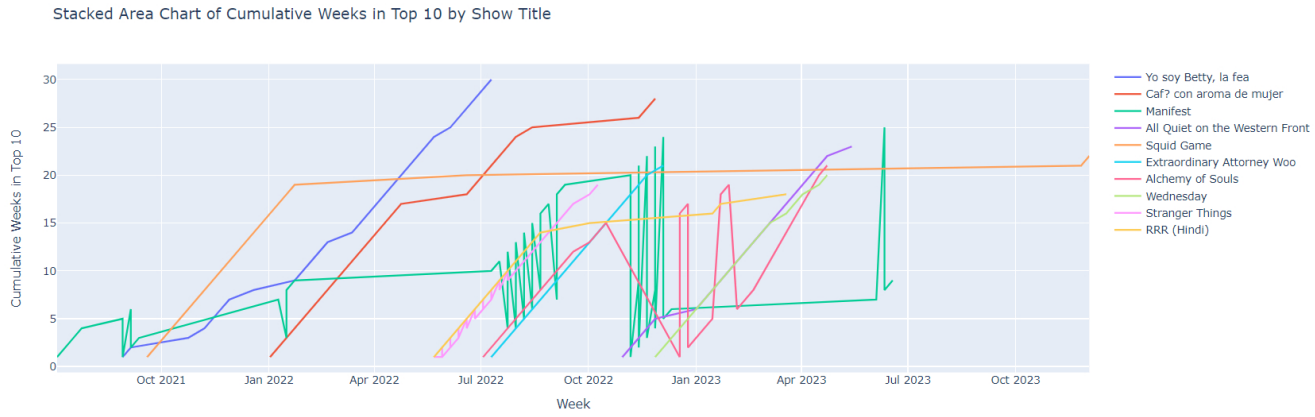



```
top_show_counts = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10)
plt.figure(figsize=(12, 8))
plt.pie(top_show_counts, labels=top_show_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Top 10 Shows with Highest Cumulative Weeks in Top 10')
plt.show()
```



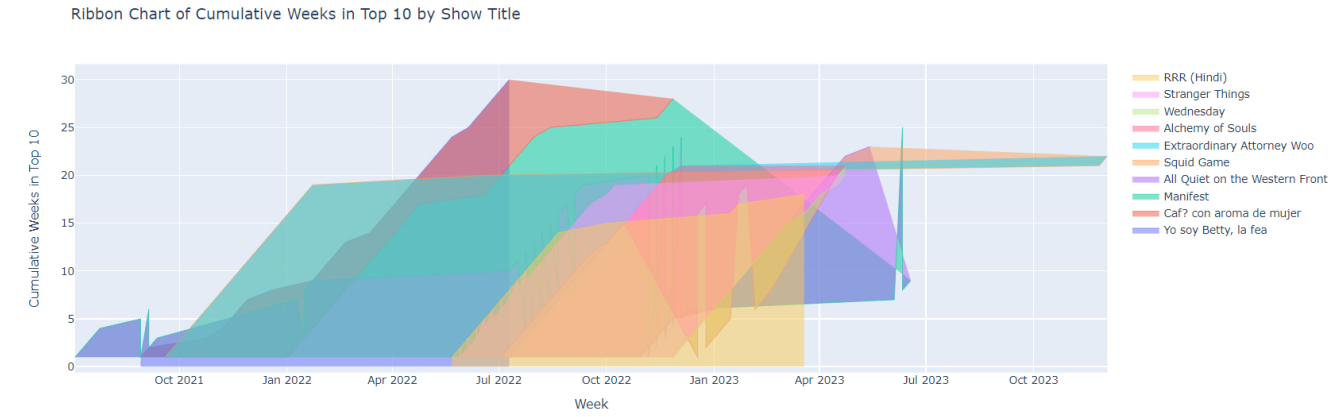
Stacked Area Chart

```
top_shows = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10).reset_index()

fig = go.Figure()

for show in top_shows['show_title']:
    show_data = df[df['show_title'] == show]
    fig.add_trace(go.Scatter(x=show_data['week'], y=show_data['cumulative_weeks_in_top_10'], mode='lines', name=show))

fig.update_layout(title='Stacked Area Chart of Cumulative Weeks in Top 10 by Show Title', xaxis_title='Week', yaxis_title='Cumulative Weeks in Top 10', showlegend=True)
fig.show()
```



Ribbon Chart

[65]:
```
top_shows = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10).reset_index()

fig = go.Figure()
for i, show in enumerate(top_shows['show_title']):
    show_data = df[df['show_title'] == show]
    fig.add_trace(go.Scatter(x=show_data['week'], y=show_data['cumulative_weeks_in_top_10'], mode='lines', fill='tonexty' if i < len(top_shows) - 1 else 'tozeroy', line=dict(width

fig.update_layout(title='Ribbon Chart of Cumulative Weeks in Top 10 by Show Title', xaxis_title='Week', yaxis_title='Cumulative Weeks in Top 10',showlegend=True)
fig.show()
```

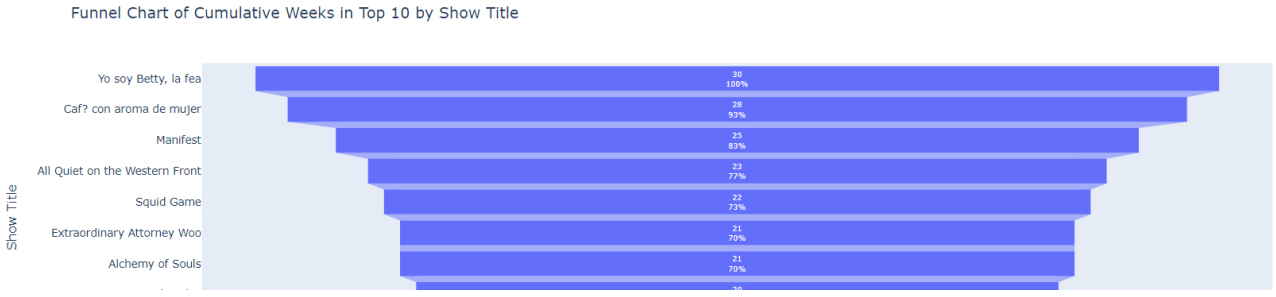


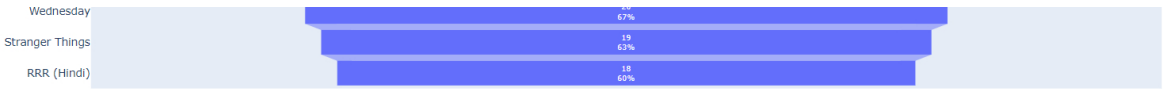
Treemap

[66]:
```
top_shows = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10).reset_index()
fig = px.treemap(top_shows, path=['show_title'], values='cumulative_weeks_in_top_10', title='Treemap of Cumulative Weeks in Top 10 by Show Title')
fig.show()
```

Funnel

[73]:
```
top_shows = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10).reset_index()
fig = go.Figure()

fig.add_trace(go.Funnel(name='Cumulative Weeks in Top 10', y=top_shows['show_title'], x=top_shows['cumulative_weeks_in_top_10'], textinfo="value+percent initial"
))

fig.update_layout(title='Funnel Chart of Cumulative Weeks in Top 10 by Show Title', xaxis_title='Cumulative Weeks in Top 10',yaxis_title='Show Title')
fig.show()
```

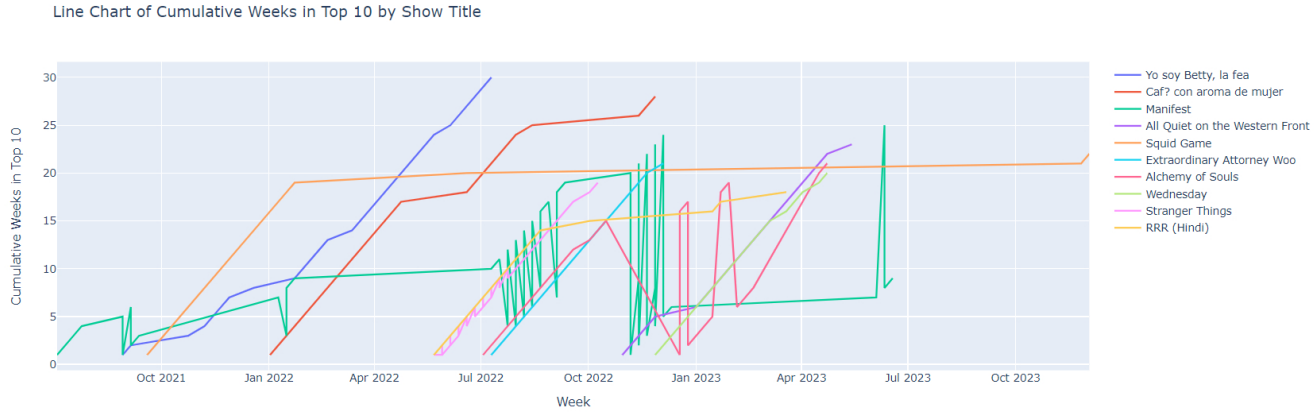




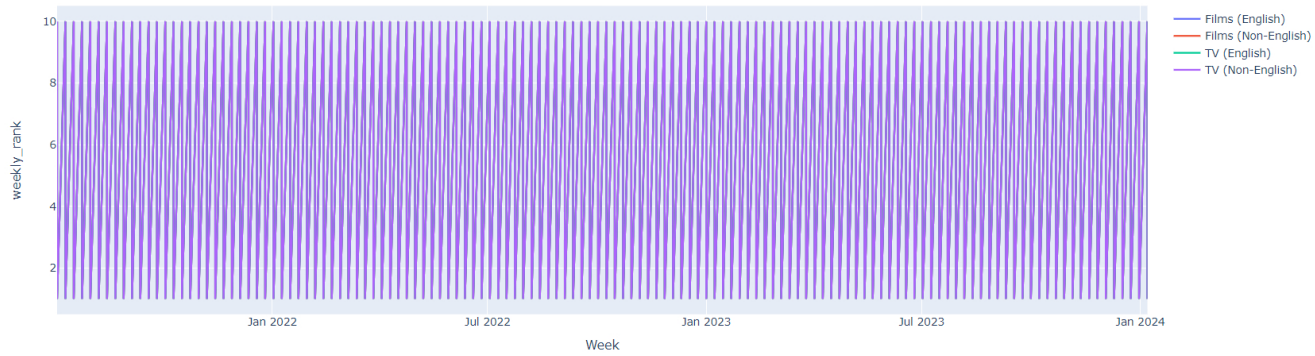
[75]:
```
category_counts = df['category'].value_counts()
fig2 = go.Figure()

fig2.add_trace(go.Funnel(name='Is Staggered Launch', y=category_counts.index, x=category_counts, textinfo="value+percent initial"
))

fig2.update_layout(title='Funnel Chart of Category vs Is Staggered Launch', xaxis_title='Count', yaxis_title='Category')
fig2.show()
```

Line Chart

[76]:
```
top_shows = df.groupby('show_title')['cumulative_weeks_in_top_10'].max().sort_values(ascending=False).head(10).reset_index()
fig1 = go.Figure()

for show in top_shows['show_title']:
    show_data = df[df['show_title'] == show]
    fig1.add_trace(go.Scatter(x=show_data['week'], y=show_data['cumulative_weeks_in_top_10'], mode='lines', name=show))

fig1.update_layout(title='Line Chart of Cumulative Weeks in Top 10 by Show Title', xaxis_title='Week', yaxis_title='Cumulative Weeks in Top 10',
                   showlegend=True)
fig1.show()
```



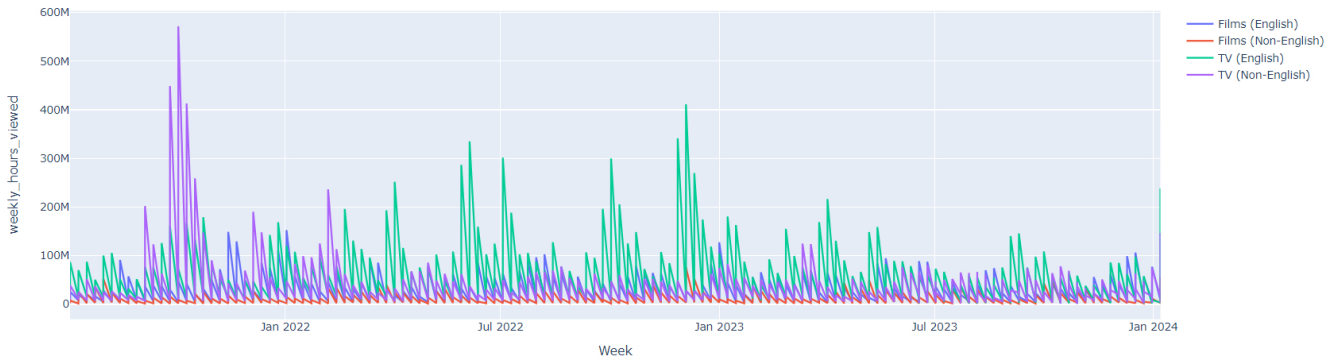
[77]:
```
def plot_line_chart(df, column_name):
    fig = go.Figure()

    for category in df['category'].unique():
        category_data = df[df['category'] == category]
        fig.add_trace(go.Scatter(x=category_data['week'], y=category_data[column_name], mode='lines', name=category))

    fig.update_layout(title=f'Line Chart of {column_name} by Category', xaxis_title='Week', yaxis_title=column_name, showlegend=True)
    fig.show()

plot_line_chart(df, 'weekly_rank')
```



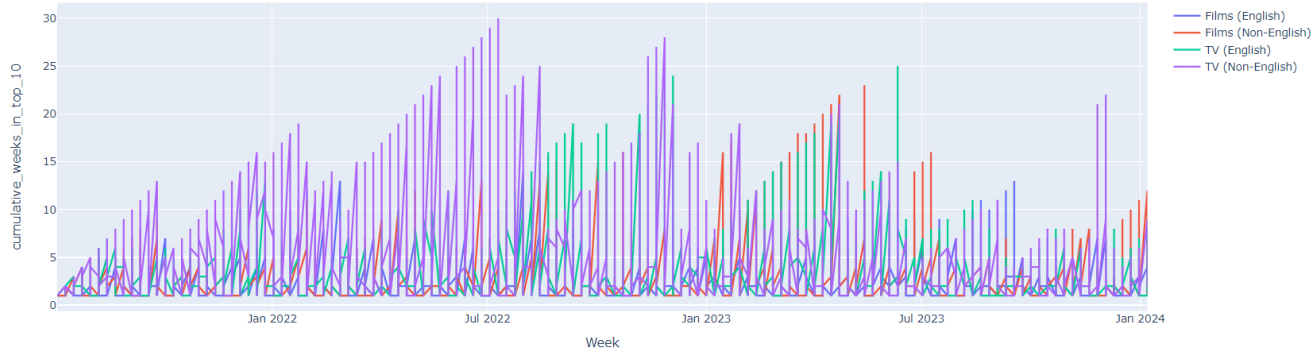
[78]:

```
plot_line_chart(df, 'weekly_hours_viewed')
plot_line_chart(df, 'cumulative_weeks_in_top_10')
plot_line_chart(df, 'show_title')
```

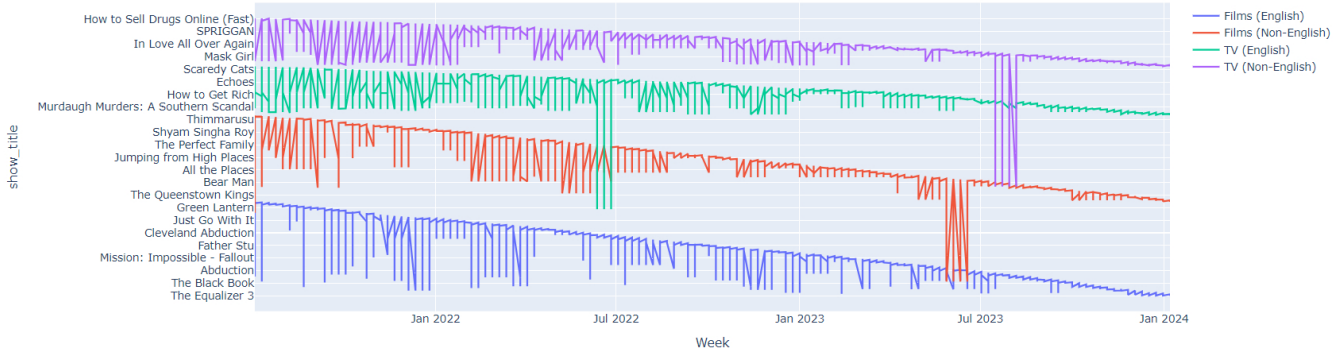
Line Chart of weekly_hours_viewed by Category



Line Chart of cumulative_weeks_in_top_10 by Category

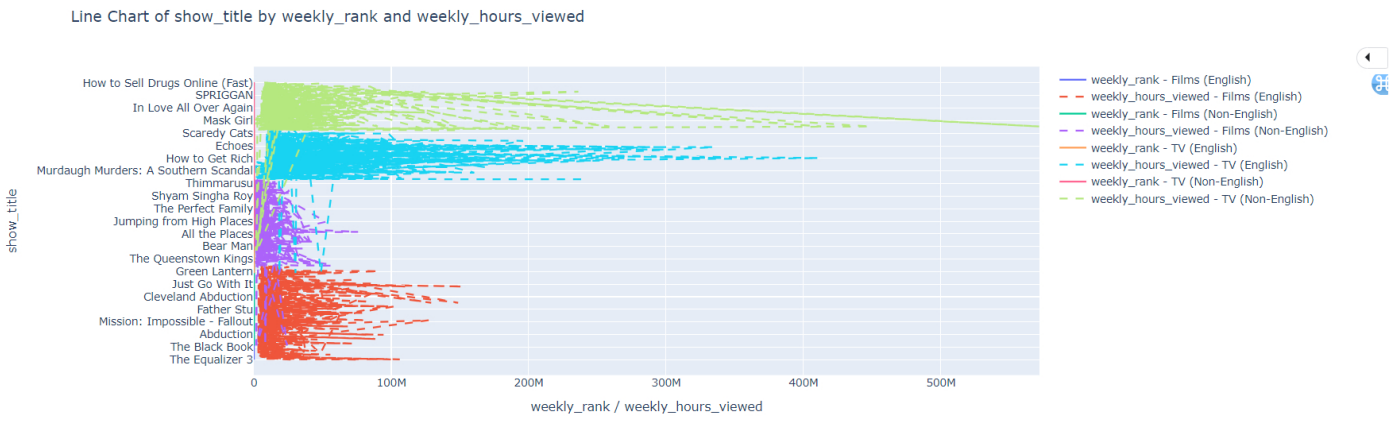


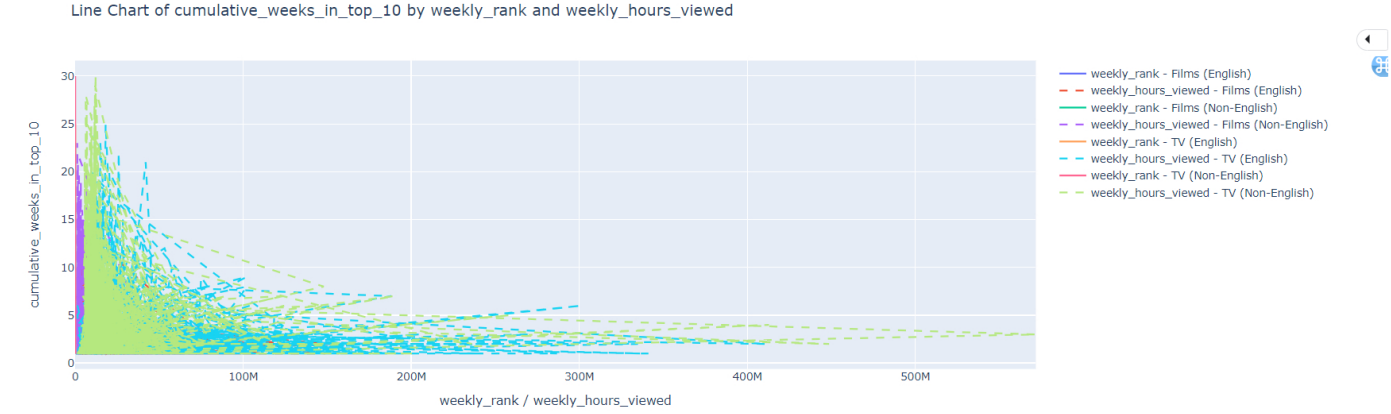
Line Chart of show_title by Category

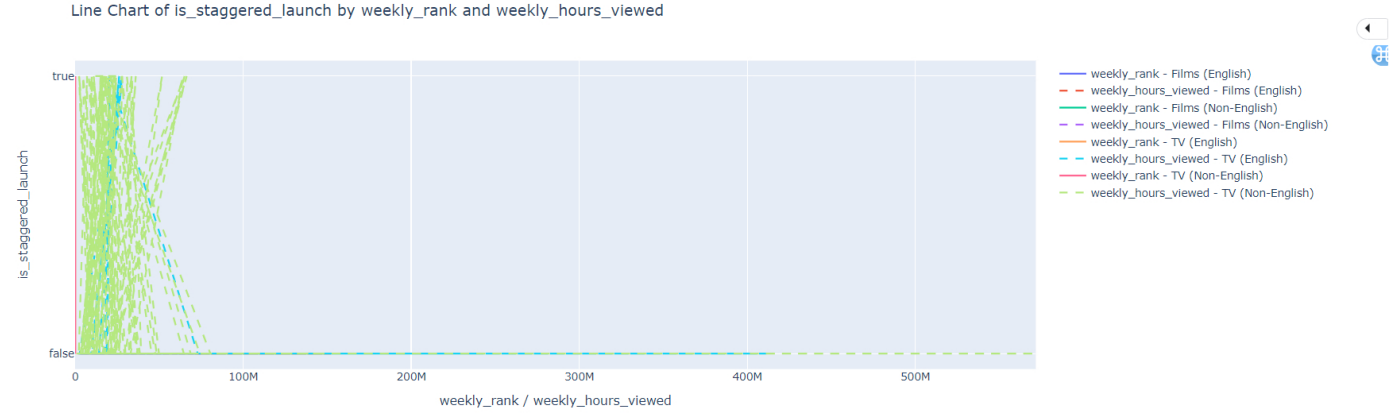


[79]:
```
def plot_line_charts(df, parameter1, parameter2):
    columns_to_plot = [col for col in df.columns if col not in ['week', 'category', parameter1, parameter2]]

    for column_name in columns_to_plot:
        fig = go.Figure()

        for category in df['category'].unique():
            category_data = df[df['category'] == category]
            fig.add_trace(go.Scatter(x=category_data[parameter1], y=category_data[column_name],
                                     mode='lines',
                                     name=f'{parameter1} - {category}'))

            fig.add_trace(go.Scatter(x=category_data[parameter2], y=category_data[column_name],
                                     mode='lines',
                                     name=f'{parameter2} - {category}',
                                     line=dict(dash='dash')))

        fig.update_layout(title=f'Line Chart of {column_name} by {parameter1} and {parameter2}',
                          xaxis_title=f'{parameter1} / {parameter2}',
                          yaxis_title=column_name,
                          showlegend=True)

        fig.show()
```

[80]:
```
plot_line_charts(df, 'weekly_rank', 'weekly_hours_viewed')
```

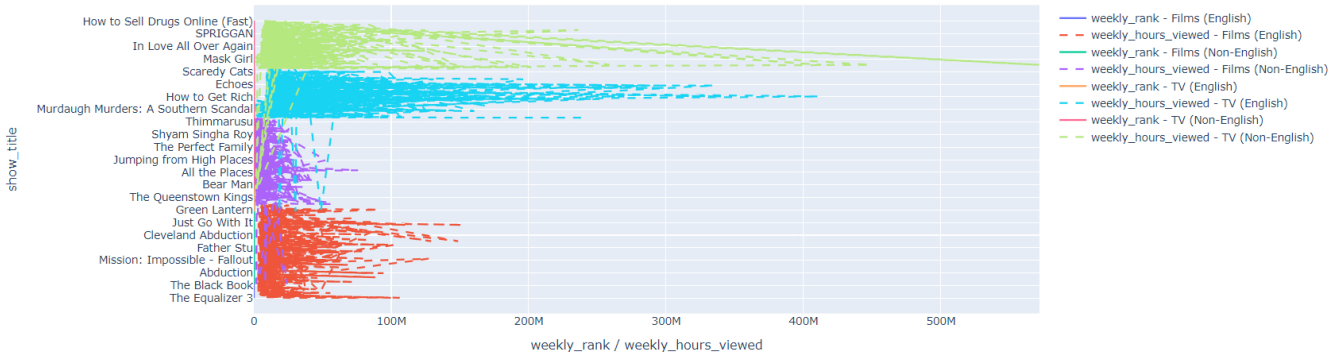




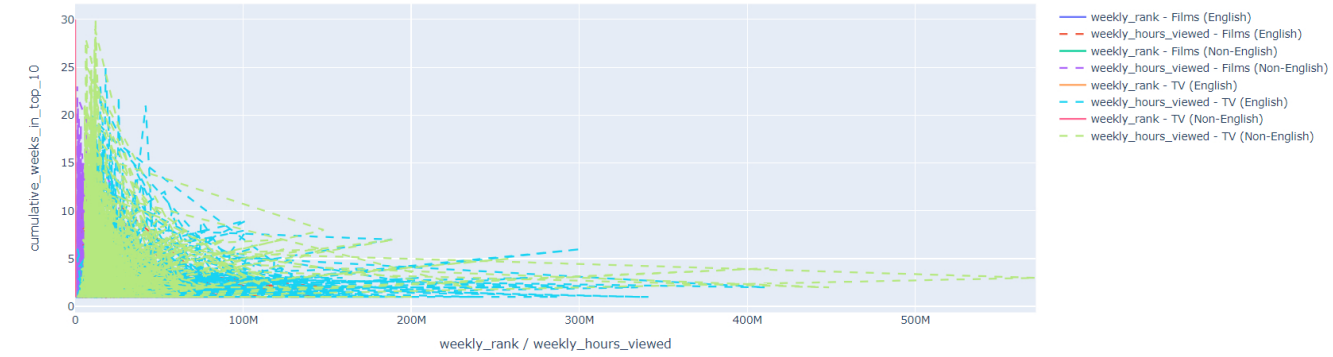


[81]:
```
plot_line_charts(df, 'weekly_rank', 'weekly_hours_viewed')
plot_line_charts(df, 'category', 'weekly_rank')
plot_line_charts(df, 'show_title', 'weekly_hours_viewed')
plot_line_charts(df, 'show_title', 'cumulative_weeks_in_top_10')
```

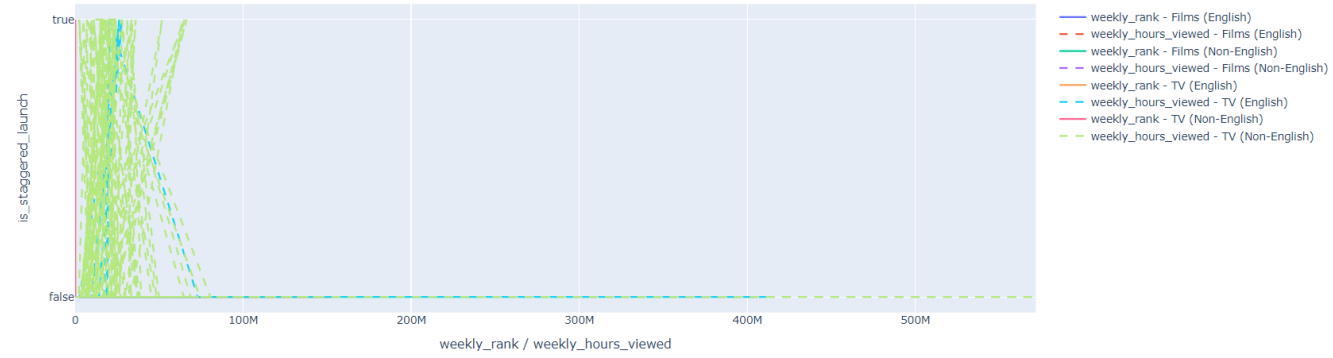
Line Chart of show_title by weekly_rank and weekly_hours_viewed



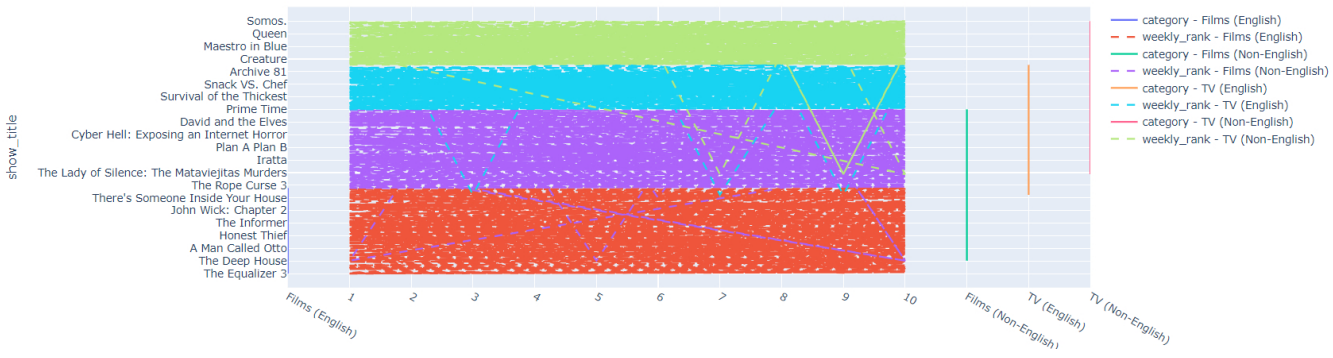
Line Chart of cumulative_weeks_in_top_10 by weekly_rank and weekly_hours_viewed

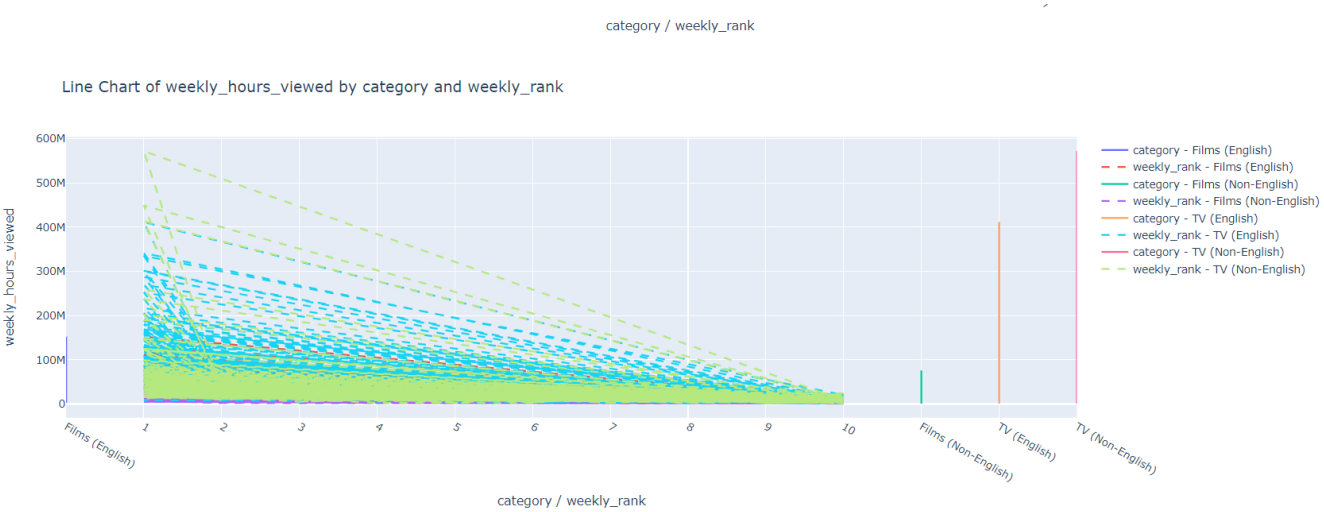


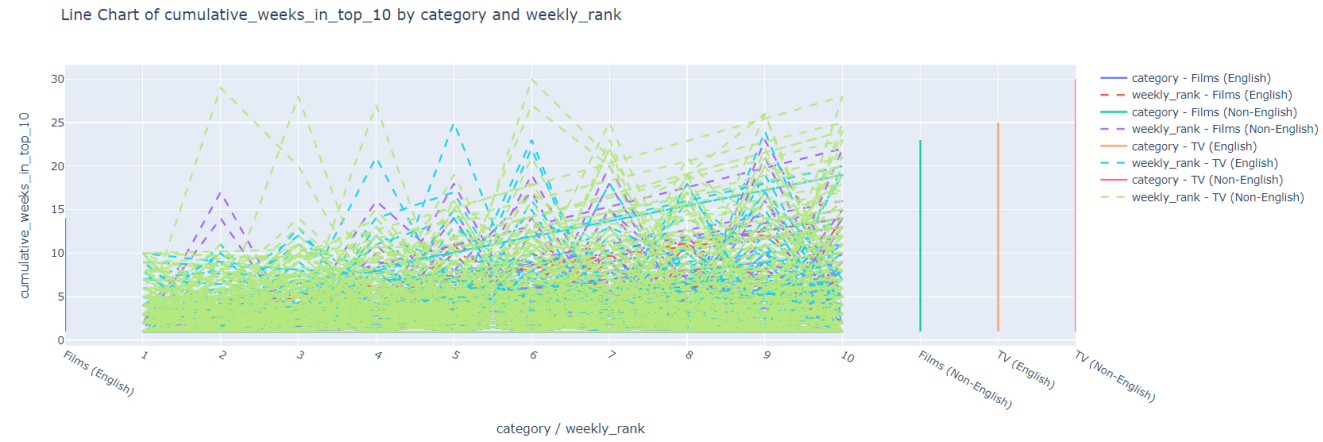
Line Chart of is_staggered_launch by weekly_rank and weekly_hours_viewed

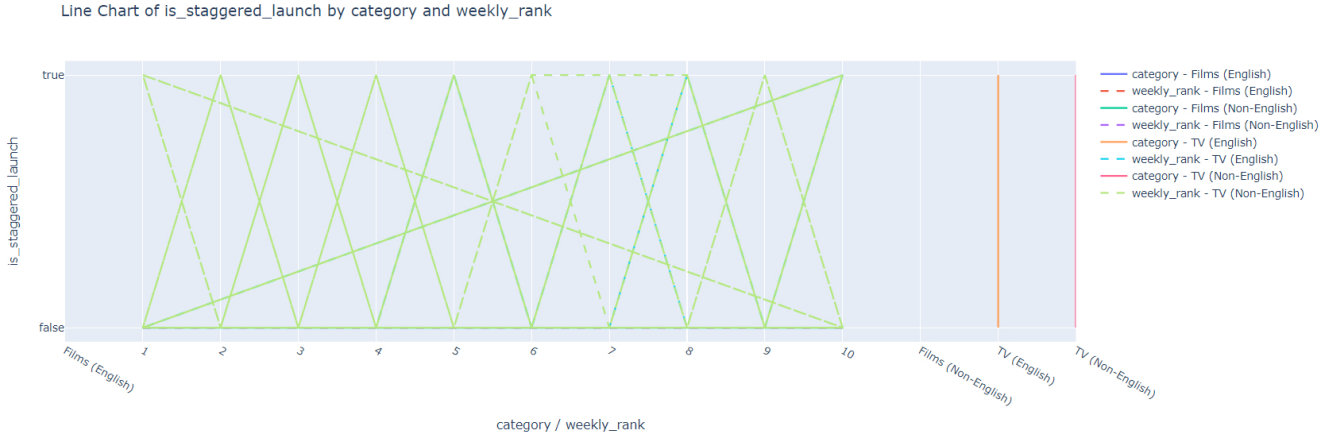


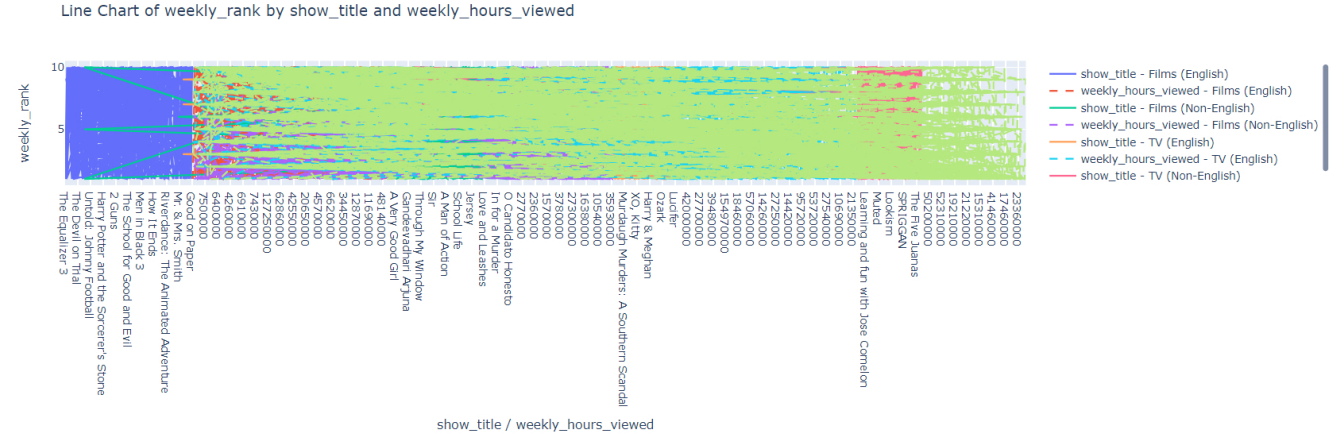
Line Chart of show_title by category and weekly_rank



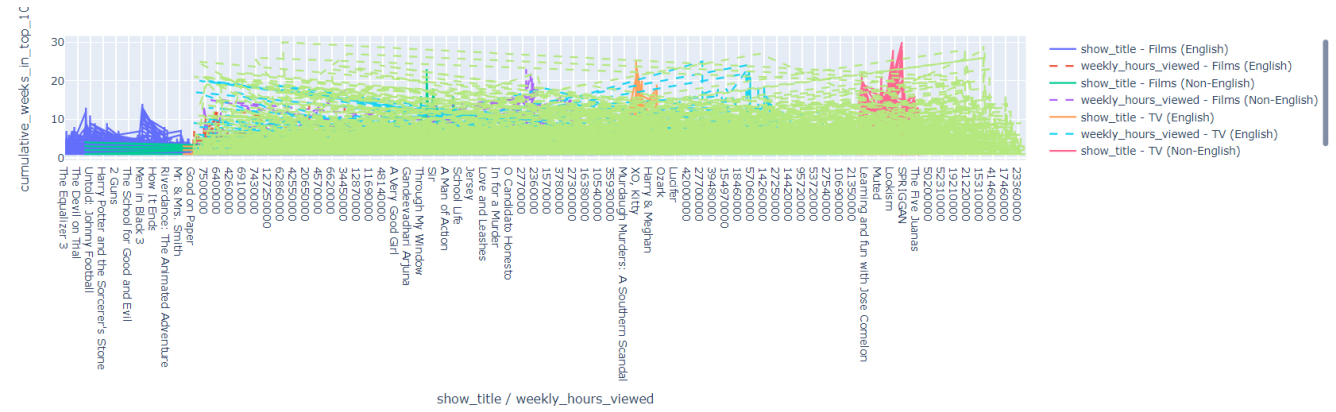




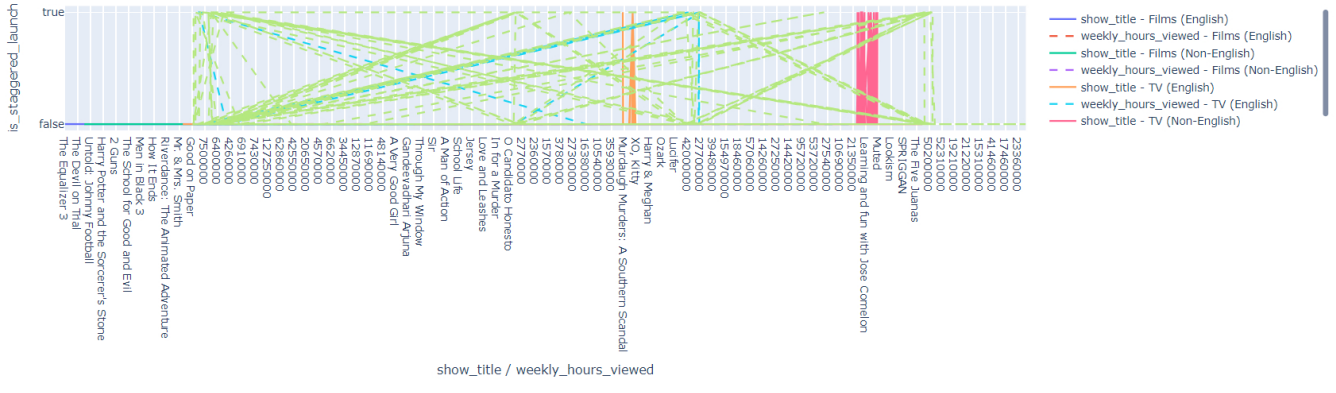




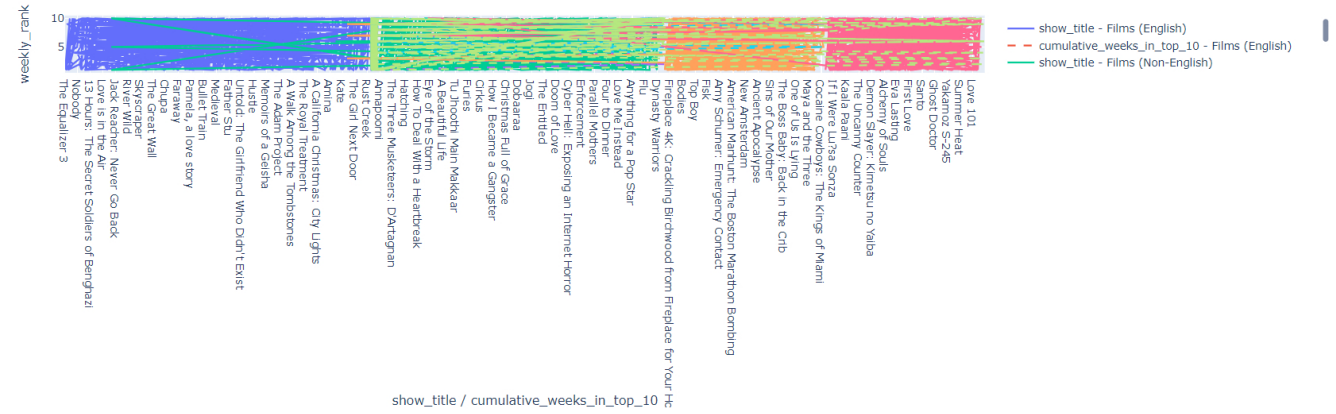
Line Chart of cumulative_weeks_in_top_10 by show_title and weekly_hours_viewed



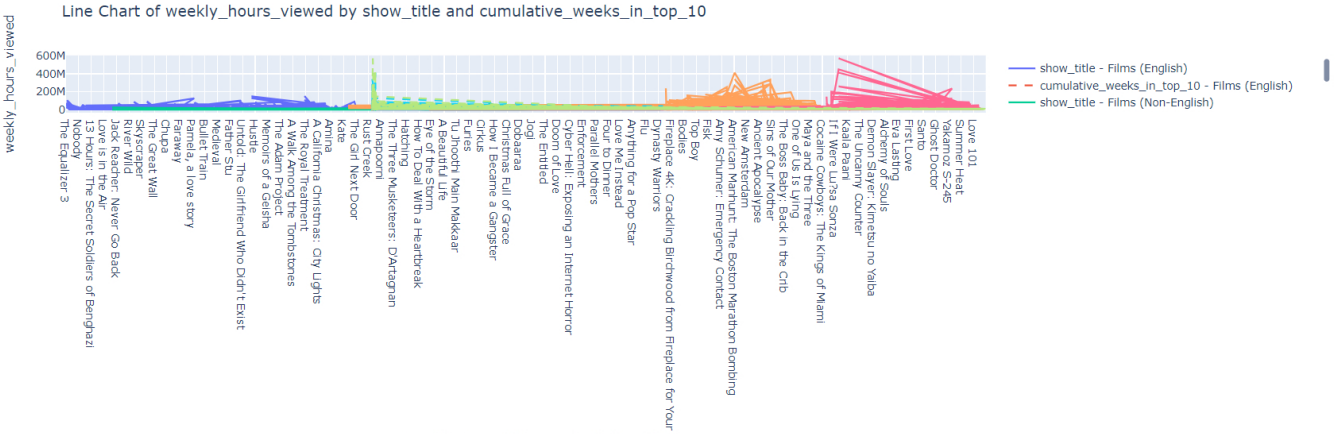
Line Chart of is_staggered_launch by show_title and weekly_hours_viewed



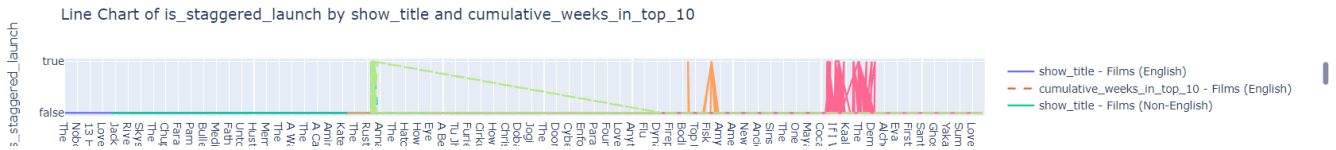
Line Chart of weekly_rank by show_title and cumulative_weeks_in_top_10



Line Chart of weekly_hours_viewed by show_title and cumulative_weeks_in_top_10



Line Chart of is_staggered_launch by show_title and cumulative_weeks_in_top_10



show_title / cumulative_weeks_in_top_10

Word Cloud

[85]:

```
def generate_word_cloud(df, column_name):
    text = ' '.join(df[column_name].astype(str))

    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for {column_name}')
    plt.show()

generate_word_cloud(df, 'show_title')
```

+ Code + Markdown

[ ]: