Streaming Content Recommendation System

- NIDHI MISHRA 20MIA1007
- CHETAN SRIVASTAVA 20MIA1122
- UTKARSH ARYAN 20MIA1155

Importing the Datasets from GitHub

```
!wget https://github.com/utkarsh-aryan/Datasets-for-projects/raw/main/disney_plus_t
!wget https://github.com/utkarsh-aryan/Datasets-for-projects/raw/main/amazon_prime_
!wget https://github.com/utkarsh-aryan/Datasets-for-projects/raw/main/netflix_title
!wget https://github.com/utkarsh-aryan/Datasets-for-projects/raw/main/hulu_titles.c
```

```
--2023-04-07 07:21:07--  https://github.com/utkarsh-aryan/Datasets-for-pro
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/utkarsh-aryan/Datasets-for-pro
--2023-04-07 07:21:07--  https://raw.githubusercontent.com/utkarsh-aryan/D
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
HTTP request sent, awaiting response... 200 OK
Length: 383571 (375K) [text/plain]
Saving to: ‘disney_plus_titles.csv.1’

disney_plus_titles. 100%[===================>] 374.58K  --.-KB/s    in 0.0

2023-04-07 07:21:08 (10.8 MB/s) - ‘disney_plus_titles.csv.1’ saved [383571

--2023-04-07 07:21:08--  https://github.com/utkarsh-aryan/Datasets-for-pro
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/utkarsh-aryan/Datasets-for-pro
--2023-04-07 07:21:08--  https://raw.githubusercontent.com/utkarsh-aryan/D
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
HTTP request sent, awaiting response... 200 OK
Length: 3972416 (3.8M) [text/plain]
Saving to: ‘amazon_prime_titles.csv.1’

amazon_prime_titles 100%[===================>]   3.79M  --.-KB/s    in 0.0

2023-04-07 07:21:09 (55.7 MB/s) - ‘amazon_prime_titles.csv.1’ saved [39724

--2023-04-07 07:21:09--  https://github.com/utkarsh-aryan/Datasets-for-pro
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/utkarsh-aryan/Datasets-for-pro
--2023-04-07 07:21:10--  https://raw.githubusercontent.com/utkarsh-aryan/D
```

```
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
HTTP request sent, awaiting response... 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: ‘netflix_titles.csv.1’

netflix_titles.csv. 100%[===================>]   3.24M  --.-KB/s    in 0.0

2023-04-07 07:21:10 (48.7 MB/s) - ‘netflix_titles.csv.1’ saved [3399671/33

--2023-04-07 07:21:10--  https://github.com/utkarsh-aryan/Datasets-for-pro
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/utkarsh-aryan/Datasets-for-pro
--2023-04-07 07:21:10--  https://raw.githubusercontent.com/utkarsh-aryan/D
```

DATASET

For this project we will use 4 datasets containing of listings of all the movies and tv shows available on Netflix, Hulu, Disney Plus and Amazon Prime, along with details such as - cast, directors, ratings, release year, duration, etc.

Dataset preprocessing and cleaning

```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.io as pio
from plotly.offline import iplot
from plotly.subplots import make_subplots
from wordcloud import WordCloud, STOPWORDS
import random
import re
```

In the following codes

df1- *Amazon Prime Dataset*

df2- *Hulu Dataset*

df3 - *Disney Plus Dataset*

df4 - *Netflix Dataset*

```
df1 = pd.read_csv("amazon_prime_titles.csv", delimiter=",", encoding="latin-1", par
```

```
df1 = pd.read_csv("amazon_prime_titles.csv", delimiter=",", encoding="latin-1", pa
df2 = pd.read_csv("hulu_titles.csv", delimiter=",", encoding="latin-1", parse_dates
df3 = pd.read_csv("disney_plus_titles.csv", delimiter=",", encoding="latin-1", pars
df4 = pd.read_csv("netflix_titles.csv", delimiter=",", encoding="latin-1", parse_da
```

```
print("The size and shape of dataset 1")
print(df1.size)
print(df1.shape)
```

```
The size and shape of dataset 1
106348
(9668, 11)
```

```
print("The size and shape of dataset 2")
print(df2.size)
print(df2.shape)
```

```
The size and shape of dataset 2
33803
(3073, 11)
```

```
print("The size and shape of dataset 3")
print(df3.size)
print(df3.shape)
```

```
The size and shape of dataset 3
15950
(1450, 11)
```

```
print("The size and shape of dataset 4")
print(df4.size)
print(df4.shape)
```

```
The size and shape of dataset 4
96877
(8807, 11)
```

```
df1.dtypes
```

```
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

```
df2.dtypes
```

```
type                    object
title                   object
director                object
cast                   float64
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

```
df3.dtypes
```

```
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

```
df4.dtypes
```

```
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

DATA CLEANING

We will go through all 4 datasets to clean them.

```
df1["date_added"] = df1["date_added"].dt.year
df1["date_added"].unique()
```

```
array([2021.,   nan])
```

```
df1["date_added"].fillna(0, inplace=True)
df1["date_added"] = df1["date_added"].astype(int)
df1.head()
```

show_id	type	title	director	cast	country	date_added	rele
s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	2021	
s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	2021	
s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	2021	
		Pink:	Sonia	Interviews with: Pink, Adele,	United		

```
df1.loc[df1["date_added"]==0, ]
```

	type	title	director	cast	country	date_added	re

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9668 entries, s1 to s9668
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   type          9668 non-null   object
 1   title         9668 non-null   object
 2   director      7586 non-null   object
 3   cast          8435 non-null   object
 4   country       672 non-null    object
 5   date_added    9668 non-null   int64
 6   release_year  9668 non-null   int64
 7   rating        9331 non-null   object
 8   duration      9668 non-null   object
 9   listed_in     9668 non-null   object
 10  description   9668 non-null   object
dtypes: int64(2), object(9)
memory usage: 906.4+ KB
```

s21	Show	Zubaan Mahal	NaN	NaN	NaN		0

```
df1.duplicated().sum()
```

```
0
```

s9664	Movie	Pride Of The	Joseph H.	Leo Gorcey, Bobby	NaN		0

```
df1.fillna("No Data", inplace=True)
df1.isnull().sum()
```

```
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

FOR DATASET 2(HULU), we will also convert float64 to string

```
df2['cast'] = df2['cast'].astype(str)
```

```
df2["date_added"] = df2["date_added"].dt.year
df2["date_added"].unique()
```

```
array([2021., 2020., 2019., 2018., 2017., 2016., 2015., 2014., 2013.,
       2012., 2011., 2010., 2009., 2008., 2006.,   nan])
```

```
df2["date_added"].fillna(0, inplace=True)
df2["date_added"] = df2["date_added"].astype(int)
df2.head()
```

show_id	type	title	director	cast	country	date_added	release_yea
s1	Movie	Ricky Velez: Here's Everything	NaN	nan	NaN	2021	202
s2	Movie	Silent Night	NaN	nan	NaN	2021	202
s3	Movie	The Marksman	NaN	nan	NaN	2021	202

```
df2.loc[df2["date_added"]==0, ]
```

s3048	TV Show	Black Butler: Book of Circus	NaN	nan	Japan	0
s3049	TV Show	Blade Dance of the Elementalers	NaN	nan	Japan	0
s3050	TV Show	Boys Before Flowers	NaN	nan	South Korea	0
s3051	TV Show	Buffy the Vampire Slayer	NaN	nan	United States	0
s3052	TV Show	Doctora Juguetes	NaN	nan	NaN	0
s3053	TV Show	Firefly	NaN	nan	United States	0
s3054	TV Show	Frasier	NaN	nan	United States	0
s3055	TV Show	Hey Arnold!	NaN	nan	United States	0
s3056	TV Show	Horrible Histories (UK)	NaN	nan	United Kingdom	0
s3057	TV Show	Kimi Ni Todoke: From Me to You	NaN	nan	Japan	0
s3058	TV Show	Medium	NaN	nan	United States	0

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3073 entries, s1 to s3073
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   type          3073 non-null   object
 1   title         3073 non-null   object
 2   director      3 non-null      object
 3   cast          3073 non-null   object
 4   country       1620 non-null   object
 5   date_added    3073 non-null   int64 
 6   release_year  3073 non-null   int64 
 7   rating        2553 non-null   object
```

```
 8   duration      2594 non-null   object
 9   listed_in     3073 non-null   object
 10  description   3069 non-null   object
dtypes: int64(2), object(9)
memory usage: 288.1+ KB
```

TV

```
df2.duplicated().sum()
```

```
0
```

Show

```
df2.fillna("No Data", inplace=True)
df2.isnull().sum()
```

```
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

For Dataset 3 (DISNEY)

```
df3["date_added"] = df3["date_added"].dt.year
df3["date_added"].unique()
```

```
array([2021., 2020., 2019.,   nan])
```

```
df3["date_added"].fillna(0, inplace=True)
df3["date_added"] = df3["date_added"].astype(int)
df3.head()
```

	type	title	director	cast	country	date_added

```
df3.loc[df3["date_added"]==0, ]
```

show_id	type	title	director	cast	country	date_added	relea
s1440	TV Show	Disney Kirby Buckets	NaN	Jacob Bertrand, Mekai Curtis, Cade Sutton, Oli...	United States	0	
s1441	TV Show	Disney Mech-X4	NaN	Nathaniel Potvin, Raymond Cham, Kamran Lucas, ...	Canada	0	
s3	Movie	Mammoth	Karen Disher	Lequizamo, Denis	States		2021

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1450 entries, s1 to s1450
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   type          1450 non-null   object
 1   title         1450 non-null   object
 2   director      977 non-null    object
 3   cast          1260 non-null   object
 4   country       1231 non-null   object
 5   date_added    1450 non-null   int64 
 6   release_year  1450 non-null   int64 
 7   rating        1447 non-null   object
 8   duration      1450 non-null   object
 9   listed_in     1450 non-null   object
 10  description   1450 non-null   object
dtypes: int64(2), object(9)
memory usage: 135.9+ KB
```

```
df3.duplicated().sum()
```

```
0
```

```
df3.fillna("No Data", inplace=True)
df3.isnull().sum()
```

```
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
```

```
description     0
dtype: int64
```

FOR DATASET 4 (NETFLIX)

```
df4["date_added"] = df4["date_added"].dt.year
df4["date_added"].unique()
```

```
array([2021., 2020., 2019., 2018., 2017., 2016., 2015., 2014., 2013.,
       2012., 2011., 2009., 2008.,   nan, 2010.])
```

```
df4["date_added"].fillna(0, inplace=True)
df4["date_added"] = df4["date_added"].astype(int)
df4.head()
```

show_id	type	title	director	cast	country	date_added	releas
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021	
s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021	
s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021	
		Jailbirds					

```
df4.loc[df4["date_added"]==0, ]
```

show_id	type	title	director	cast	country	date_added	rele
s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe, Jon Hamm, Adam Godley, Chris...	United Kingdom	0	
s6175	TV Show	Anthony Bourdain: Parts Unknown	NaN	Anthony Bourdain	United States	0	

```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8807 entries, s1 to s8807
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   type          8807 non-null   object
 1   title         8807 non-null   object
 2   director      6173 non-null   object
 3   cast          7982 non-null   object
 4   country       7976 non-null   object
 5   date_added    8807 non-null   int64 
 6   release_year  8807 non-null   int64 
 7   rating        8803 non-null   object
 8   duration      8804 non-null   object
 9   listed_in     8807 non-null   object
 10  description   8807 non-null   object
dtypes: int64(2), object(9)
memory usage: 825.7+ KB
```

```
df4.duplicated().sum()
```

```
0
```

```
df4.fillna("No Data", inplace=True)
df4.isnull().sum()
```

```
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

ALL THE DATASETS ARE CLEAN

```
df1.head()
```

show_id	type	title	director	cast	country	date_added	rele
s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	2021	
s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	2021	
s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	2021	
		Pink:	Sonia	Interviews with: Pink. Adele.	United		

```
df2.head()
```

show_id	type	title	director	cast	country	date_added	release_yea
s1	Movie	Ricky Velez: Here's Everything	No Data	nan	No Data	2021	202
s2	Movie	Silent Night	No Data	nan	No Data	2021	202
s3	Movie	The Marksman	No Data	nan	No Data	2021	202

```
df3.head()
```

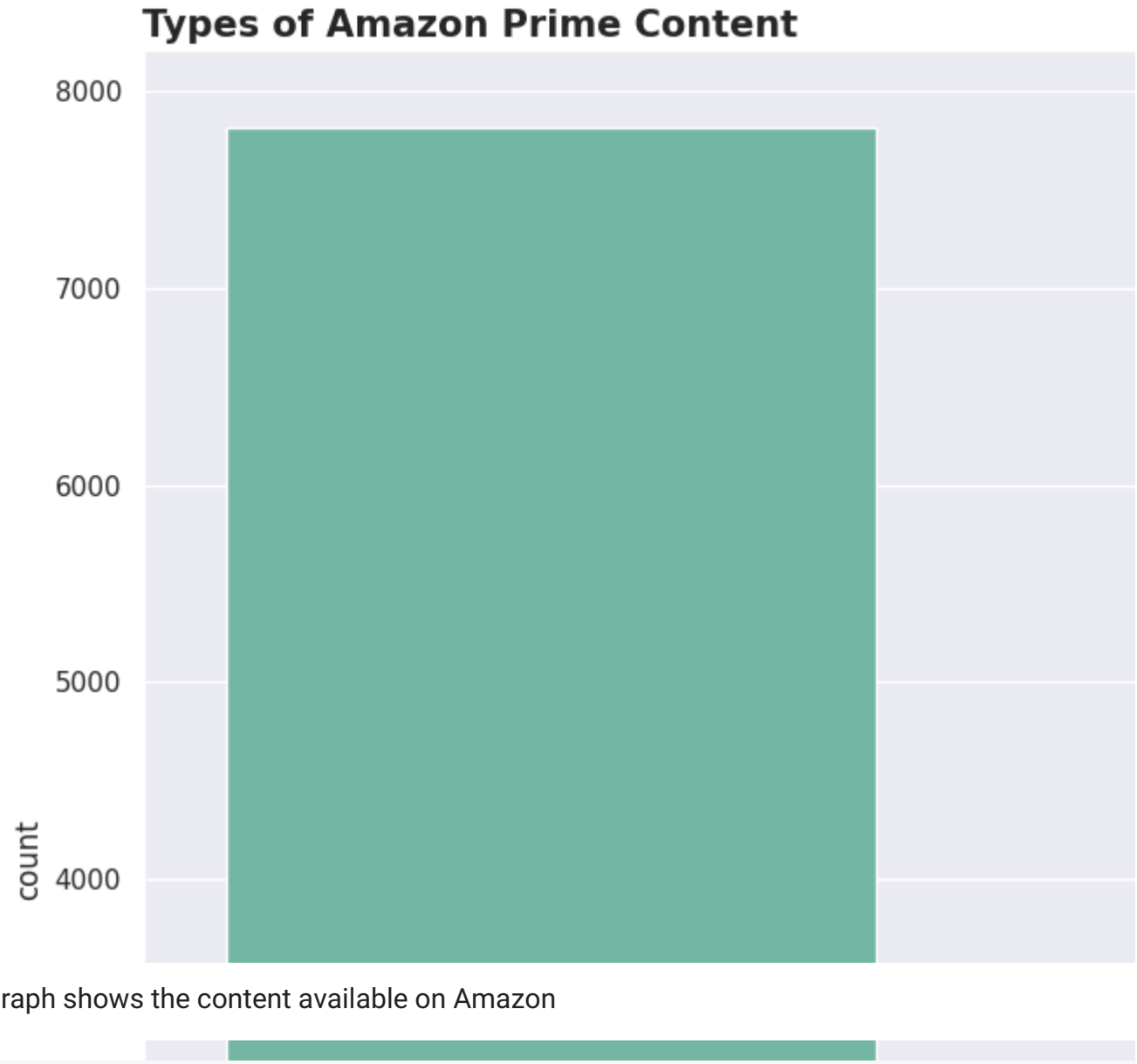
```
df4.head()
```

show_id	type	title	director	cast	country	date_added	releas
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	No Data	United States	2021	
s2	TV Show	Blood & Water	No Data	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021	
s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	No Data	2021	
		Jailbirds					

Visualization of DF1 (AMAZON Prime)

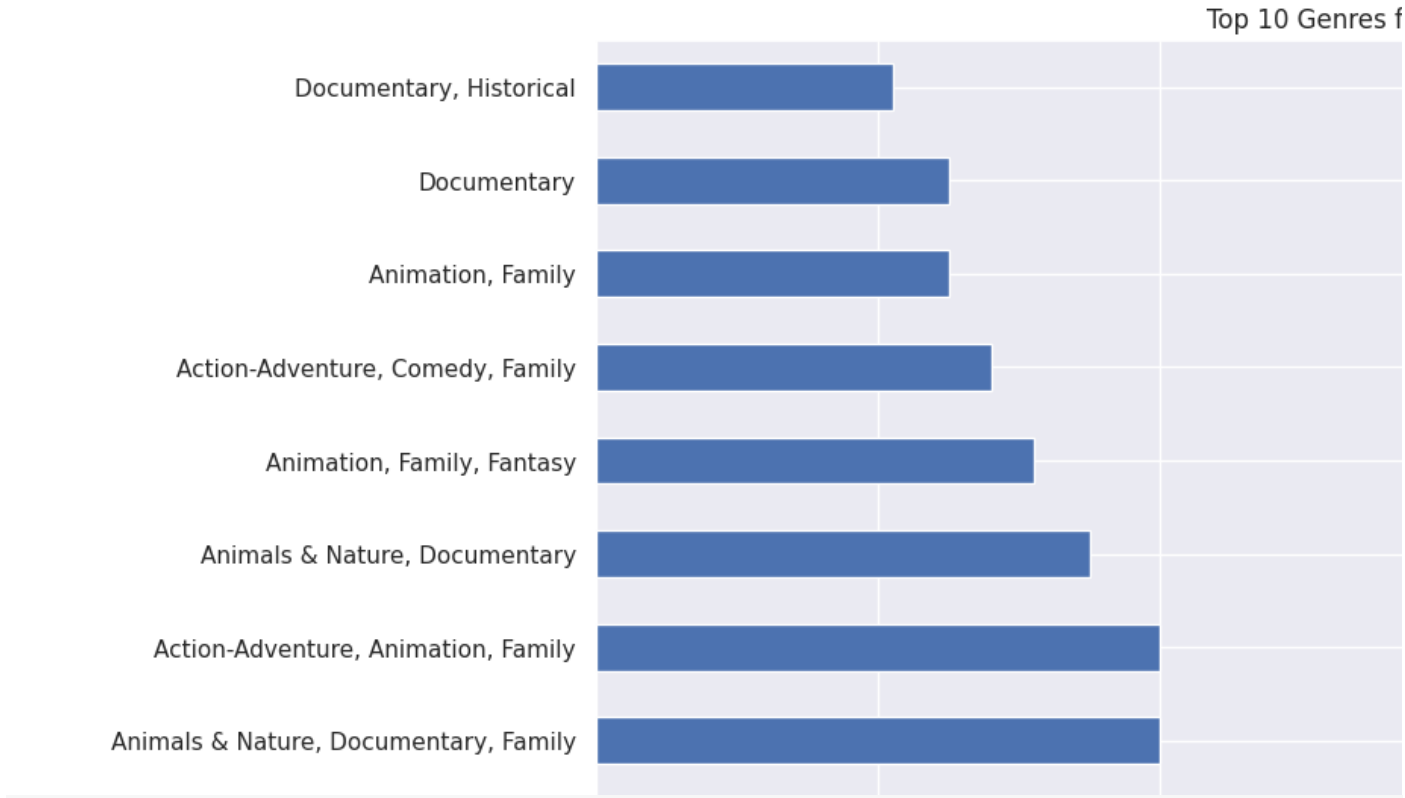
```
#Vertical·BarChart
sns.set(style="darkgrid")
ax = sns.countplot(x="type", data=df1, palette="Set2")
ax.set_title(f'Types of Amazon Prime Content', fontsize=15, fontweight='bold', posi
```

```
Text(0.2, 1.0, 'Types of Amazon Prime Content')
```



The graph shows the content available on Amazon

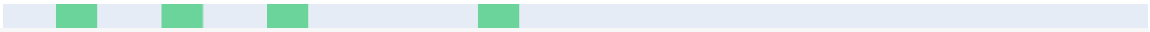
```
#Horizontal BarChart
plt.figure(figsize = (15,8))
plt.title('Top 10 Genres for Movies in Amazon Prime')
df3[df3["type"]=="Movie"]["listed_in"].value_counts()[:10].plot(kind='barh')
plt.show()
```




```
#Vertical BarChart
d1 = df1[df1["type"] == "TV Show"]
d2 = df1[df1["type"] == "Movie"]

col = "release_year"

vc1 = d1[col].value_counts().reset_index()
vc1 = vc1.rename(columns = {col : "count", "index" : col})
vc1['percent'] = vc1['count'].apply(lambda x : 100*x/sum(vc1['count']))
vc1 = vc1.sort_values(col)

vc2 = d2[col].value_counts().reset_index()
vc2 = vc2.rename(columns = {col : "count", "index" : col})
vc2['percent'] = vc2['count'].apply(lambda x : 100*x/sum(vc2['count']))
vc2 = vc2.sort_values(col)

trace1 = go.Bar(x=vc1[col], y=vc1["count"], name="TV Shows", marker=dict(color="#a6
trace2 = go.Bar(x=vc2[col], y=vc2["count"], name="Movies", marker=dict(color="#6ad4
data = [trace1, trace2]
layout = go.Layout(title="Content added over the years", legend=dict(x=0.1, y=1.1,
fig = go.Figure(data, layout=layout)
fig.show()
```


```
#Vertical BarChart
col = "rating"

vc1 = d1[col].value_counts().reset_index()
vc1 = vc1.rename(columns = {col : "count", "index" : col})
vc1['percent'] = vc1['count'].apply(lambda x : 100*x/sum(vc1['count']))
vc1 = vc1.sort_values(col)

vc2 = d2[col].value_counts().reset_index()
vc2 = vc2.rename(columns = {col : "count", "index" : col})
vc2['percent'] = vc2['count'].apply(lambda x : 100*x/sum(vc2['count']))
vc2 = vc2.sort_values(col)

trace1 = go.Bar(x=vc1[col], y=vc1["count"], name="TV Shows", marker=dict(color="#a6
trace2 = go.Bar(x=vc2[col], y=vc2["count"], name="Movies", marker=dict(color="#6ad4
data = [trace1, trace2]
layout = go.Layout(title="Content added over the years", legend=dict(x=0.1, y=1.1,
fig = go.Figure(data, layout=layout)
fig.show()
```

Visualization of DF2 (Hulu)

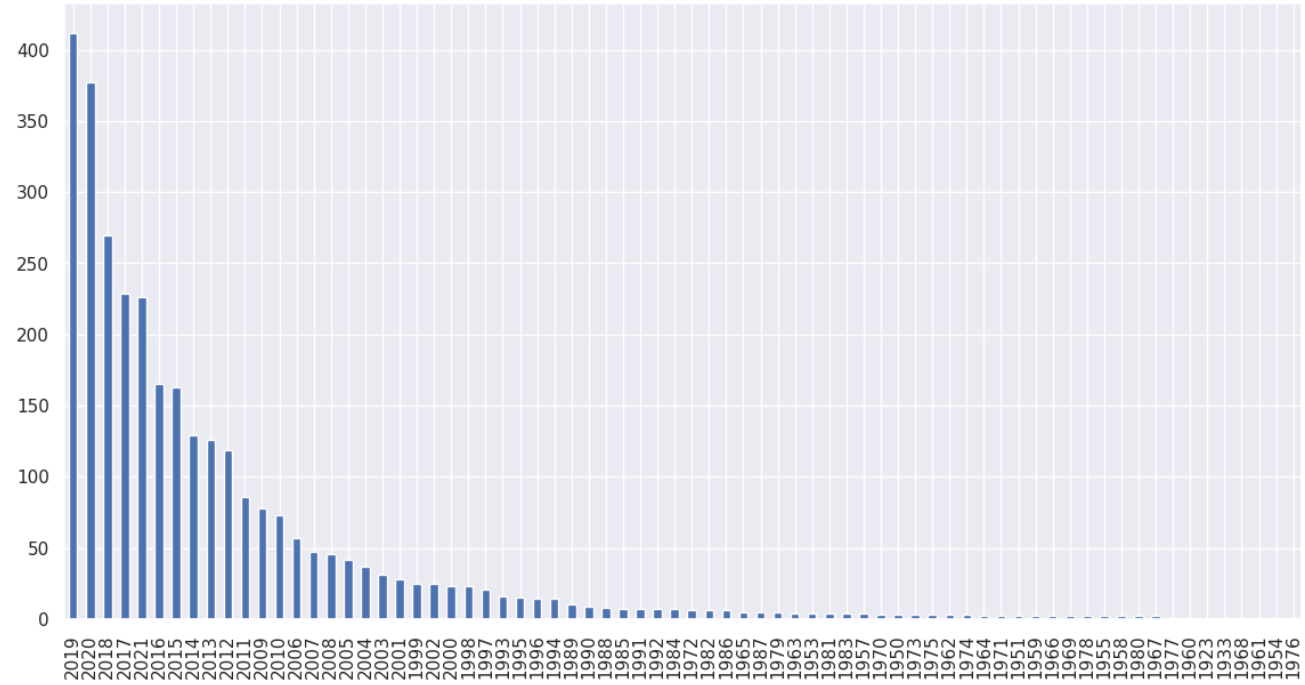


```
#Pie Chart
plt.figure(figsize=(14, 7))
labels=['TV Show', 'Movie']
plt.pie(df2['type'].value_counts().sort_values(),labels=labels,explode=[0.1,0.1],au
plt.title('Type of Hulu Content')
plt.axis('equal')
plt.show()
```

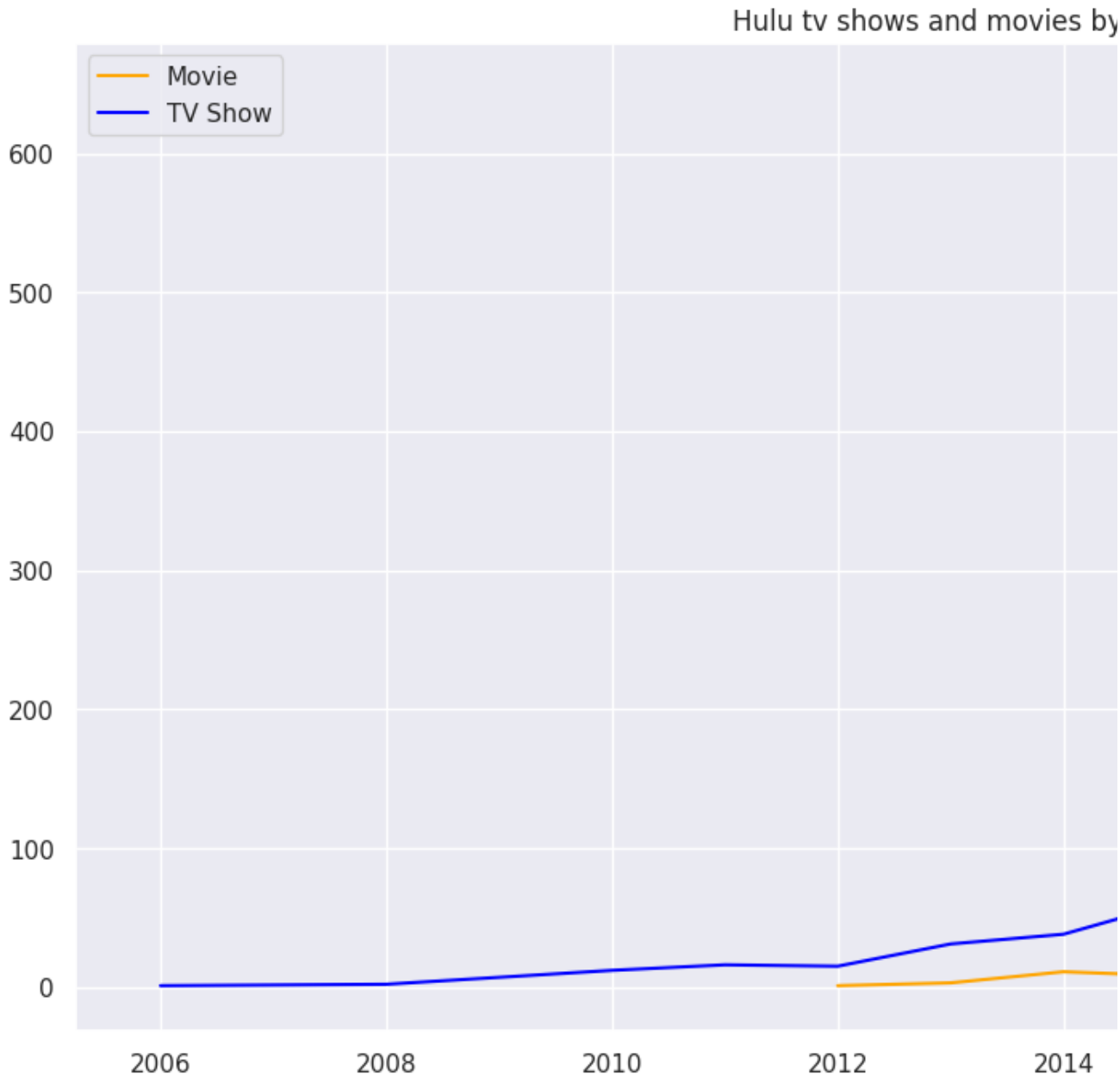
Type of Hulu Content

```
#Vertical BarChart
plt.figure(figsize=(14, 7))
df2['release_year'].value_counts().plot(kind='bar')
```

```
<Axes: >
```

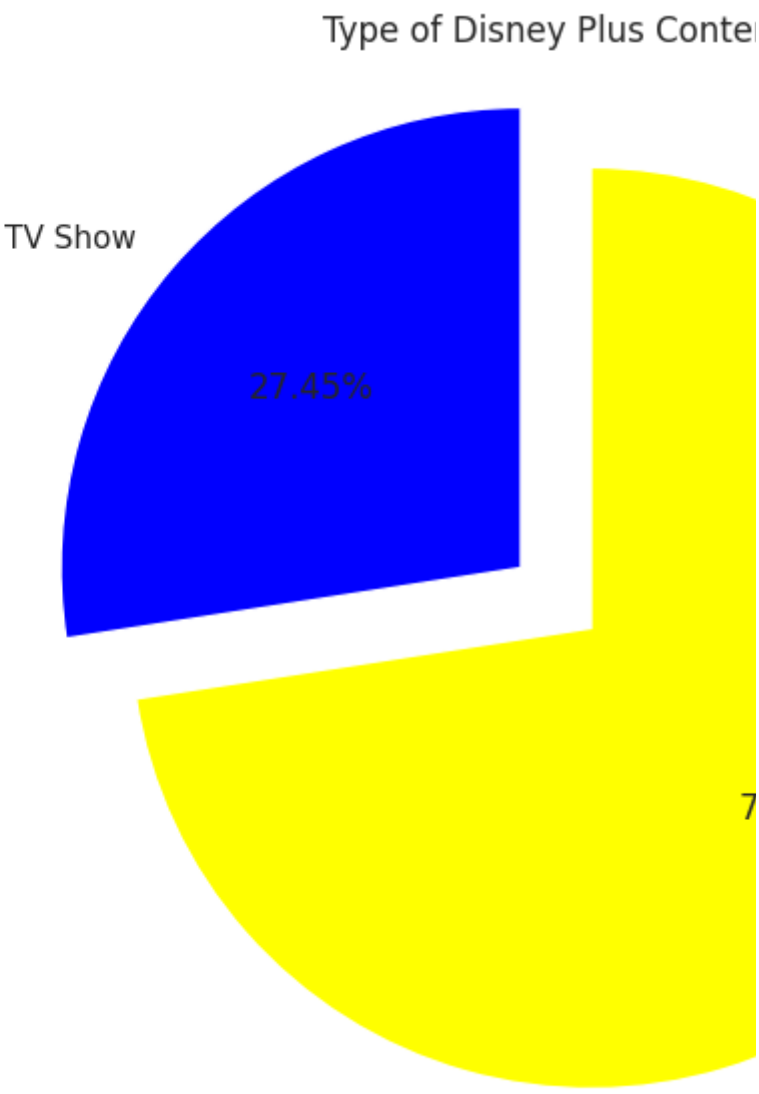


```
# Line Graph
plt.figure(figsize=(15,8))
movie = df2[df2['type'] == 'Movie' ]
tv = df2[df2['type'] == 'TV Show']
movie = movie[movie['date_added']>2000]
tv = tv[tv['date_added']>2000]
added_counts= movie['date_added'].value_counts()
added_tv_counts= tv['date_added'].value_counts()
sns.lineplot(x=added_counts.index,y=added_counts.values, color="orange", label='Mov
sns.lineplot(x=added_tv_counts.index,y=added_tv_counts.values, color="blue", label=
plt.title('Hulu tv shows and movies by year added')
plt.legend()
plt.show()
```



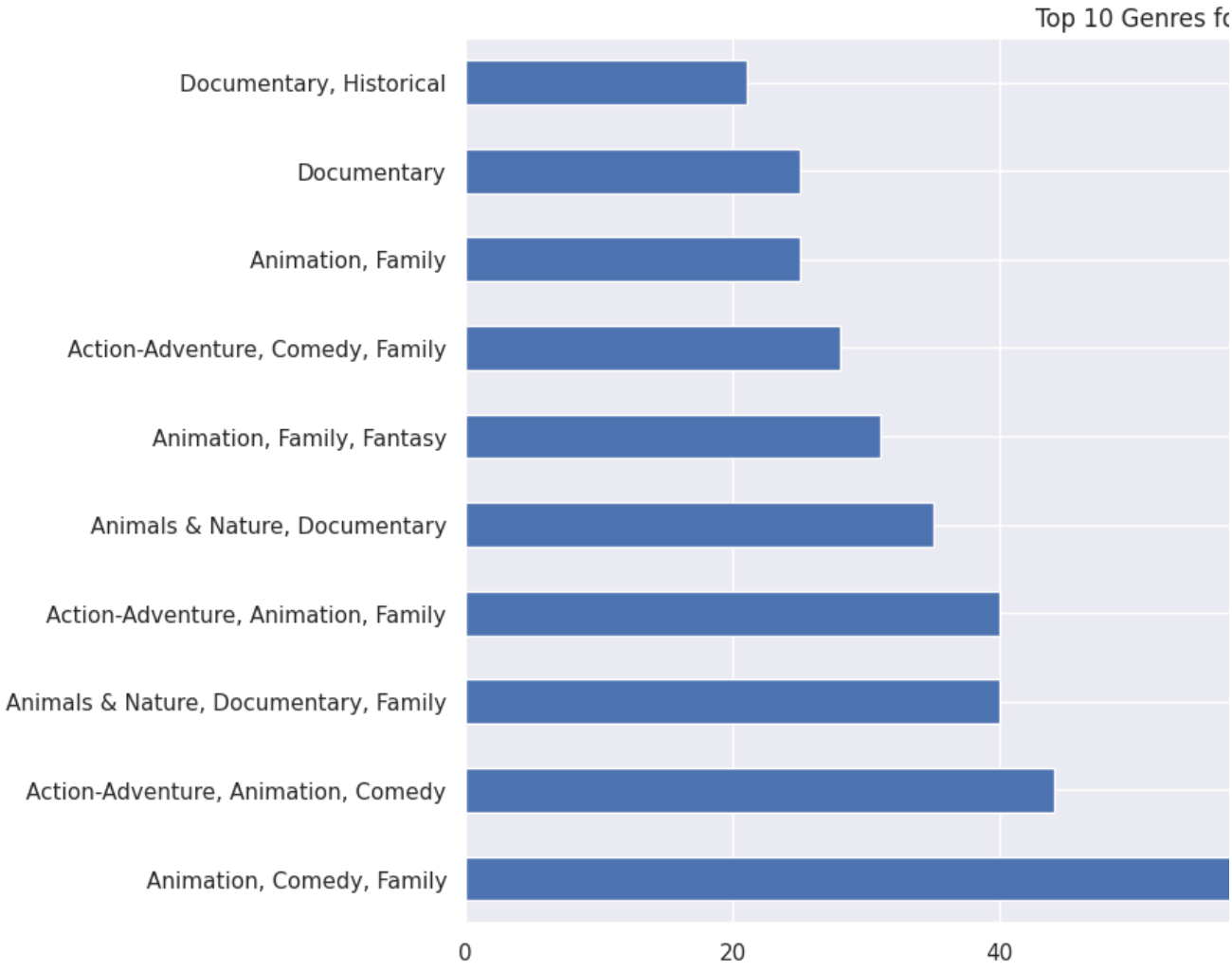
Visualization of DF3 (Disney Plus)

```
#Pie Chart
plt.figure(figsize=(14, 7))
labels=['TV Show', 'Movie']
plt.pie(df3['type'].value_counts().sort_values(),labels=labels,explode=[0.1,0.1],au
plt.title('Type of Disney Plus Content')
plt.axis('equal')
plt.show()
```

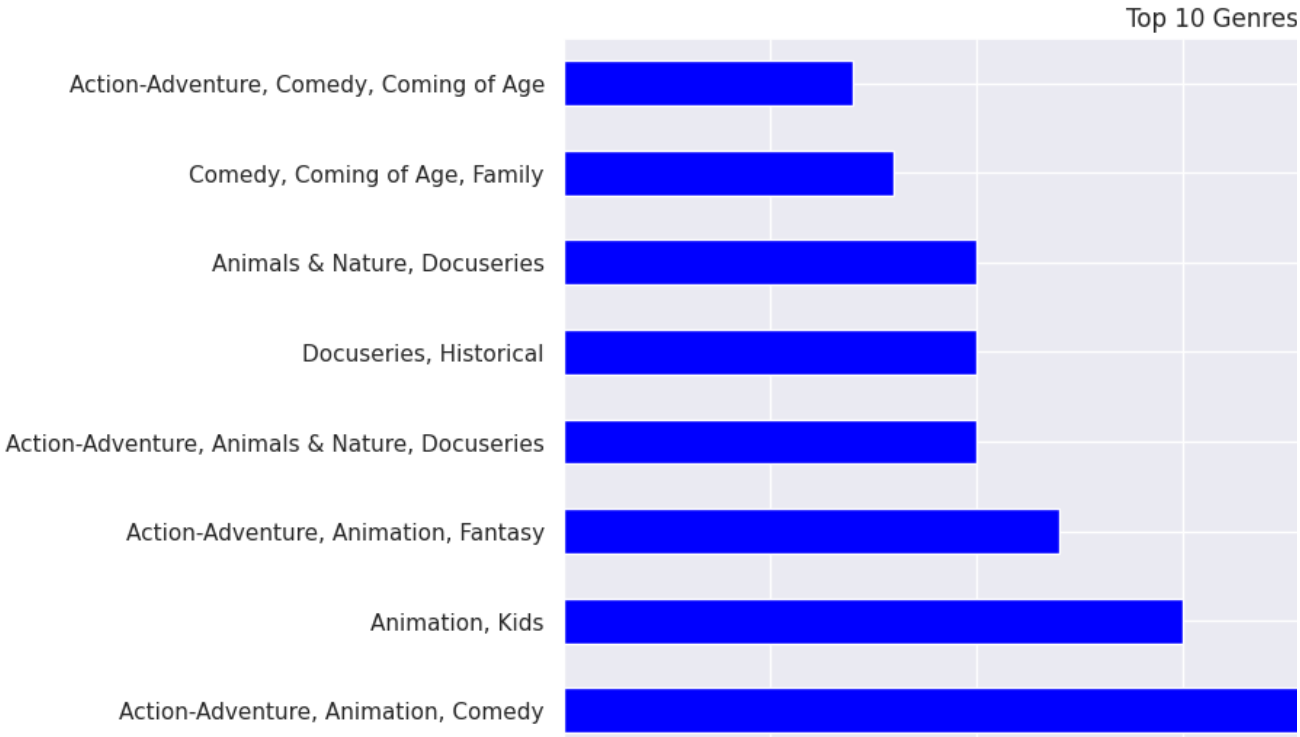


From the above chart it is clearly observed that the on Disney plus platform 72.68% of the content is movies and 27.45% is TV shows.

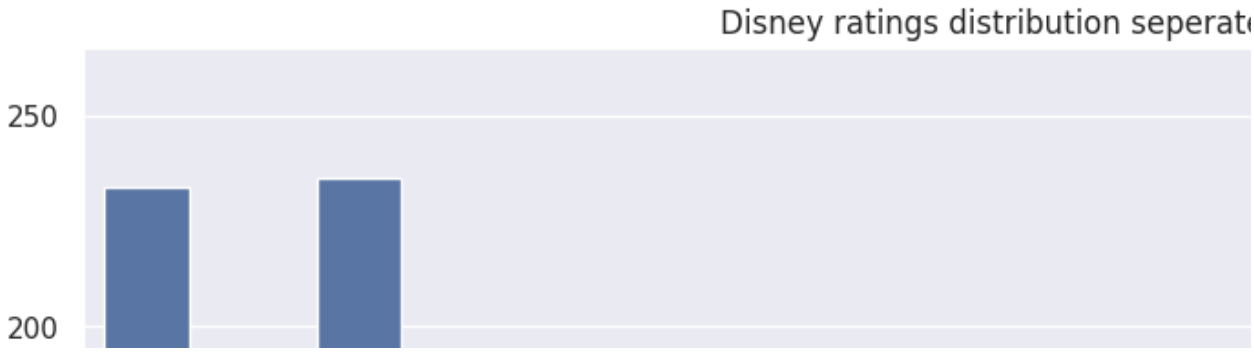
```
# Horizontal Bar Graph
plt.figure(figsize = (15,8))
plt.title('Top 10 Genres for Movies in Disney Platform')
df3[df3["type"]=="Movie"]["listed_in"].value_counts()[:10].plot(kind='barh')
plt.show()
```




```
# Horizontal Bar Graph
plt.figure(figsize = (15,8))
plt.title('Top 10 Genres for TV SHOW in Disney Platform')
df3[df3["type"]=="TV Show"]["listed_in"].value_counts()[:10].plot(kind='barh',color
plt.show()
```




```
# Vertical Bar Graph
plt.figure(figsize = (15,8))
plt.title('Disney ratings distribution seperated by type of release')
sns.countplot(x='rating', data=df3, hue='type')
plt.show()
```



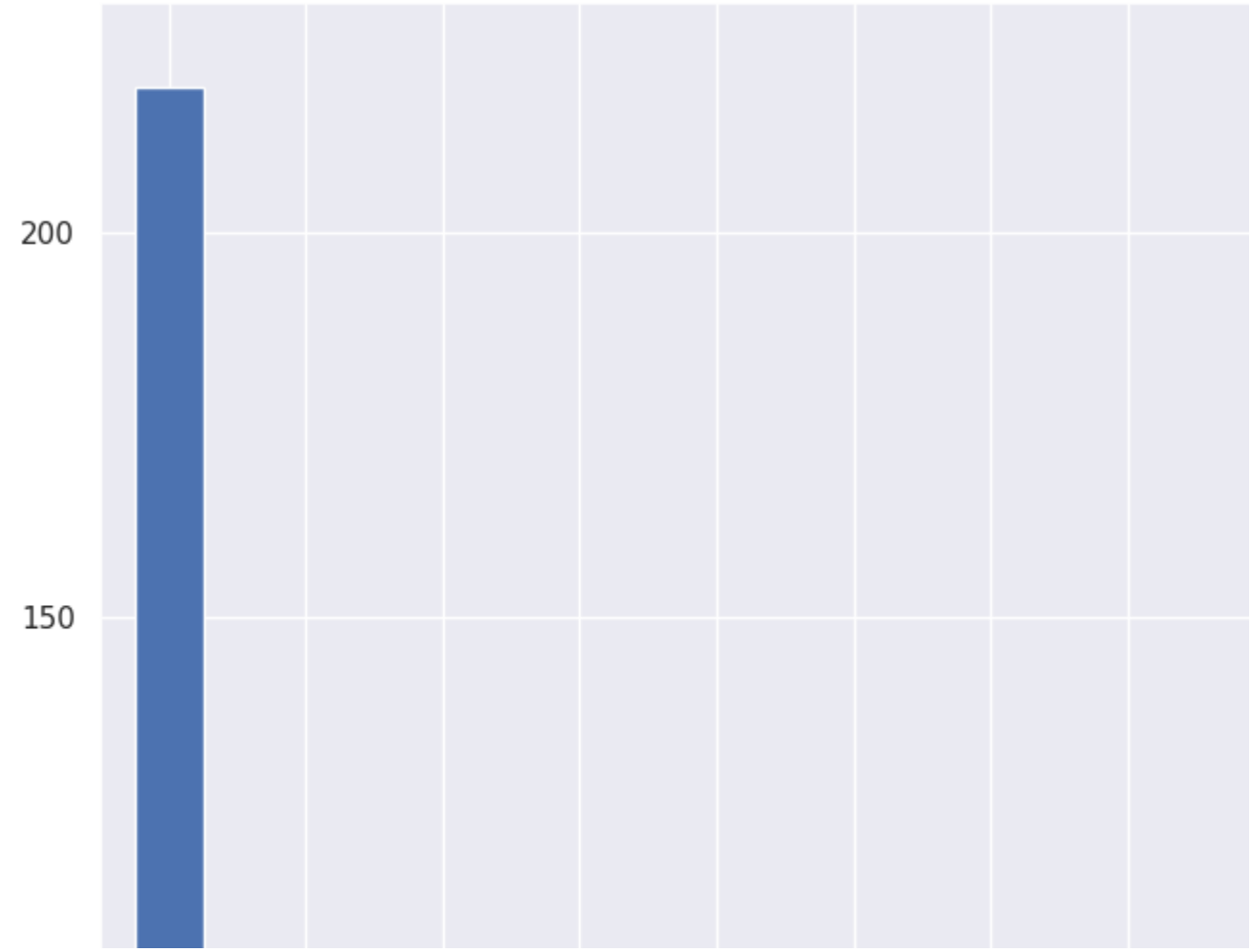
The rating's meaning are:

'TV-PG': 'Older Kids', TV-MA: 'Adults', TV-Y7-FV: 'Older Kids', TV-Y7: 'Older Kids', TV-14: 'Teens', R: 'Adults', TV-Y: 'Kids', NR: 'Adults', PG-13: 'Teens', TV-G: 'Kids', PG: 'Older Kids', G: 'Kids', UR: 'Adults', NC-17: 'Adults'

In 2019 nearly 800 Movies and TV Shows were added while in 2020 and 2021 it decreased to half the number of 2019.

```
# Vertical Bar Graph
df_tv_show = df3[df3['type']=='TV Show']
df_tv_show['duration'].value_counts().plot(kind='bar')
```

```
<Axes: >
```

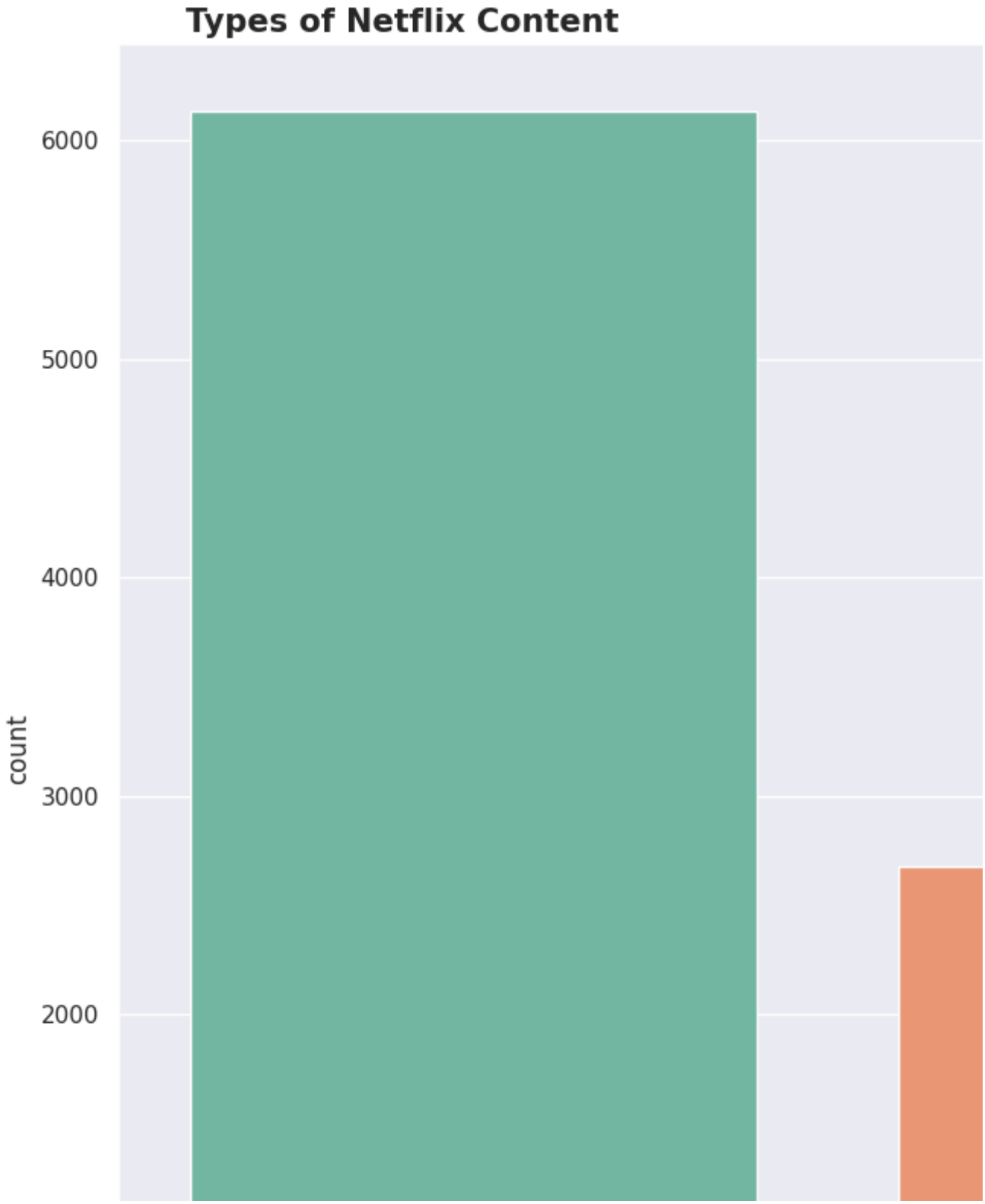


From the above grph it is clearly stated that the maximum number of TV shows contains only 1 season i.e. more than 200 in number and it decerases as we maximise the number of season.

Visualization of DF4 (Netflix)

```
# Vertical Bar Graph
sns.set(style="darkgrid")
ax = sns.countplot(x="type", data=df4, palette="Set2")
ax.set_title(f'Types of Netflix Content', fontsize=15, fontweight='bold', position=
```

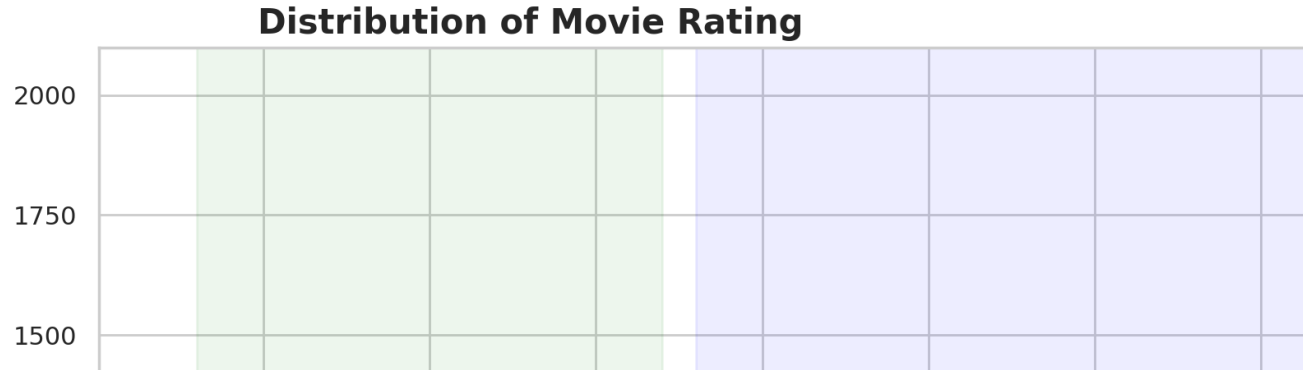
```
Text(0.2, 1.0, 'Types of Netflix Content')
```



This illustrates that netflix content include twice the number of movies when compared with the number of TV Shows.

```
# Vertical Bar Graph
sns.set_style('whitegrid') # plot with grid

movie = df4[df4['type'] == 'Movie']
rating_order =  ['G', 'TV-Y', 'TV-G', 'PG', 'TV-Y7', 'TV-PG', 'PG-13', 'TV-14', 'R'
movie_rating = movie['rating'].value_counts()[rating_order]
```

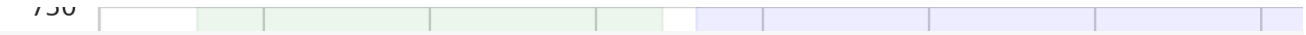
```
def rating_barplot(data, title, height, h_lim=None):
    fig, ax = plt.subplots(1,1, figsize=(14, 7), dpi=200)
    if h_lim :
        ax.set_ylim(0, h_lim)

    ax.bar(data.index, data,  color="#e0e0e0", width=0.52, edgecolor='black')

    color =  ['green',  'blue',  'orange',  'red']
    span_range = [[0, 2], [3,  6], [7, 8], [9, 11]]

    for idx, sub_title in enumerate(['Little Kids', 'Older Kids', 'Teens', 'Mature'
        ax.annotate(sub_title,
                    xy=(sum(span_range[idx])/2 ,height),
                    xytext=(0,0), textcoords='offset points',
                    va="center", ha="center",
                    color="w", fontsize=16, fontweight='bold',
                    bbox=dict(boxstyle='round4', pad=0.4, color=color[idx], alpha=0
        ax.axvspan(span_range[idx][0]-0.4,span_range[idx][1]+0.4,  color=color[idx]
    ax.set_title(f'Distribution of {title} Rating', fontsize=15, fontweight='bold',
    fig.tight_layout()
    plt.show()



rating_barplot(movie_rating, 'Movie', 1200, 2100)
```



The rating's meaning are:

'TV-PG': 'Older Kids', TV-MA: 'Adults', TV-Y7-FV: 'Older Kids', TV-Y7: 'Older Kids', TV-14: 'Teens', R: 'Adults', TV-Y: 'Kids', NR: 'Adults', PG-13: 'Teens', TV-G: 'Kids', PG: 'Older Kids', G: 'Kids', UR: 'Adults', NC-17: 'Adults'

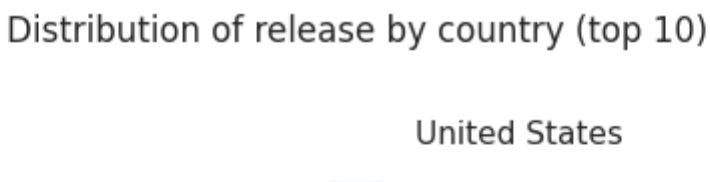
Furthur this graph shows that most content is targeted at mature and older teen audience



```
# Vertical Bar Graph
plt.figure(figsize = (15,8))
plt.title('Netflix country distribution seperated by type of release')
sns.countplot(x='country', data=df4, hue='type', order=df4.country.value_counts().i
plt.show()
```

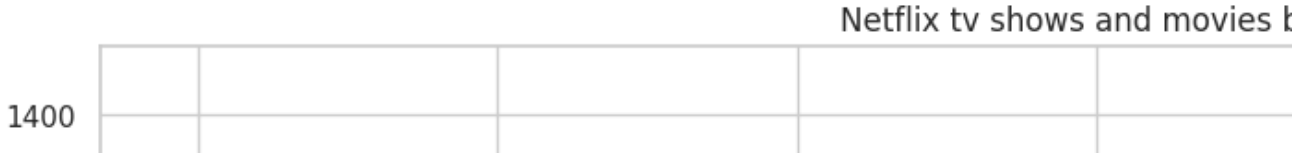
This double bar graph shows the amount of content on Netflix Library by their country of origin

```
#Pie Graph
country_counts = df4['country'].value_counts(sort=True)
country_df = pd.DataFrame(country_counts)
country_df = country_df.reset_index()
country_df.columns = ['country', 'counts']
country_pie_df = country_df.head(10)
plt.figure(figsize = (15,8))
colors = sns.color_palette('pastel')[0:10]
plt.title('Distribution of release by country (top 10)')
plt.pie(country_pie_df['counts'], labels = country_pie_df['country'], colors = colo
plt.show()
```



This graph shows the amount of releases per country till now

```
#Line Graph
plt.figure(figsize=(15,8))
movie = df4[df4['type'] == 'Movie' ]
tv = df4[df4['type'] == 'TV Show']
movie = movie[movie['date_added']>2000]
tv = tv[tv['date_added']>2000]
added_counts= movie['date_added'].value_counts()
added_tv_counts= tv['date_added'].value_counts()
sns.lineplot(x=added_counts.index,y=added_counts.values, color="orange", label='Mov
sns.lineplot(x=added_tv_counts.index,y=added_tv_counts.values, color="blue", label=
plt.title('Netflix tv shows and movies by year added')
plt.legend()
plt.show()
```



This graph shows that the number of content quickly ramped up after 2014 and has slowed down recently.

Summary of all 4 streaming sites

```
#Line Graph
plt.figure(figsize=(15,8))
dd1 = df1[df1['type'] == 'Movie' ]
dd2 = df2[df2['type'] == 'Movie' ]
dd3 = df3[df3['type'] == 'Movie' ]
dd4 = df4[df4['type'] == 'Movie' ]
added_counts1= dd1['release_year'].value_counts()
added_counts2= dd2['release_year'].value_counts()
added_counts3= dd3['release_year'].value_counts()
added_counts4= dd4['release_year'].value_counts()

sns.lineplot(x=added_counts1.index,y=added_counts1.values, color="orange", label='A
sns.lineplot(x=added_counts2.index,y=added_counts2.values, color="blue", label='Hul
sns.lineplot(x=added_counts3.index,y=added_counts3.values, color="green", label='Di
sns.lineplot(x=added_counts4.index,y=added_counts4.values, color="black", label='Ne

plt.title('Movies by release year in major Streaming sites')
plt.legend()
plt.show()
```

This Graph shows how Amazon Prime has more legacy programs and quickly catching up to its rivals in acquiring new ccontent

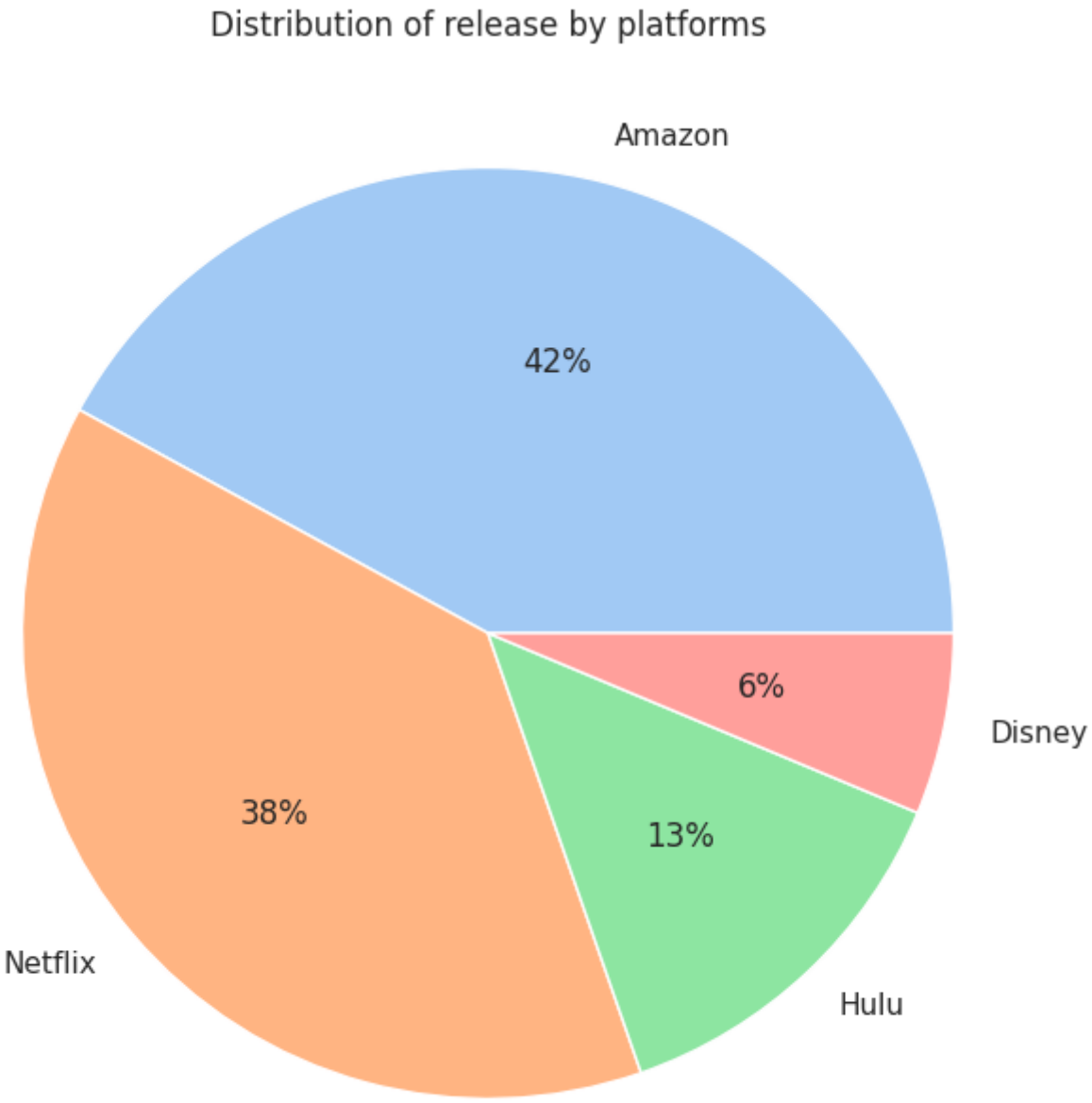
```
#Line Graph
plt.figure(figsize=(15,8))
dt1 = df1[df1['type'] == 'TV Show' ]
dt2 = df2[df2['type'] == 'TV Show' ]
dt3 = df3[df3['type'] == 'TV Show' ]
dt4 = df4[df4['type'] == 'TV Show' ]
dt1 = dt1[dt1['release_year']>1980]
dt2 = dt2[dt2['release_year']>1980]
dt3 = dt3[dt3['release_year']>1980]
dt4 = dt4[dt4['release_year']>1980]
added_countst1= dt1['release_year'].value_counts()
added_countst2= dt2['release_year'].value_counts()
added_countst3= dt3['release_year'].value_counts()
added_countst4= dt4['release_year'].value_counts()

sns.lineplot(x=added_countst1.index,y=added_countst1.values, color="orange", label=
sns.lineplot(x=added_countst2.index,y=added_countst2.values, color="blue", label='H
sns.lineplot(x=added_countst3.index,y=added_countst3.values, color="green", label='
sns.lineplot(x=added_countst4.index,y=added_countst4.values, color="black", label='

plt.title('TV SHOWS by release year in major Streaming sites')
plt.legend()
plt.show()
```

This shows that when it comes to TV Shows, Netflix is in the lead. While its closest rivals are trying to catch it.


```
df1['platform'] = 'Amazon'
df2['platform'] = 'Hulu'
df3['platform'] = 'Disney'
df4['platform'] = 'Netflix'
```

```
frames = [df1, df2, df3, df4]
result = pd.concat(frames, keys=["Amazon", "Hulu", "Disney", "Netflix"])
```

```
#Pie Chart
platform_counts = result['platform'].value_counts(sort=True)
platform_df = pd.DataFrame(platform_counts)
platform_df = platform_df.reset_index()
platform_df.columns = ['platform', 'counts']
platform_pie_df = platform_df
plt.figure(figsize = (15,8))
colors = sns.color_palette('pastel')[0:10]
plt.title('Distribution of release by platforms')
plt.pie(platform_pie_df['counts'], labels = platform_pie_df['platform'], colors = c
plt.show()
```



This Pie Chart shows the total content available by platform

```
#Word Cloud
text = ' '.join(result['listed_in'])
plt.rcParams['figure.figsize'] = (12,12)
wordcloud = WordCloud(background_color = 'black',colormap='vlag', width = 1200,  he
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

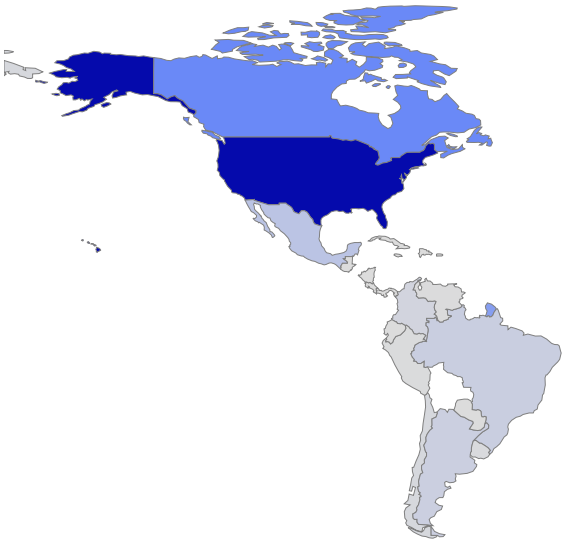
This word cloud shows which word are more commonly used in the content listings.

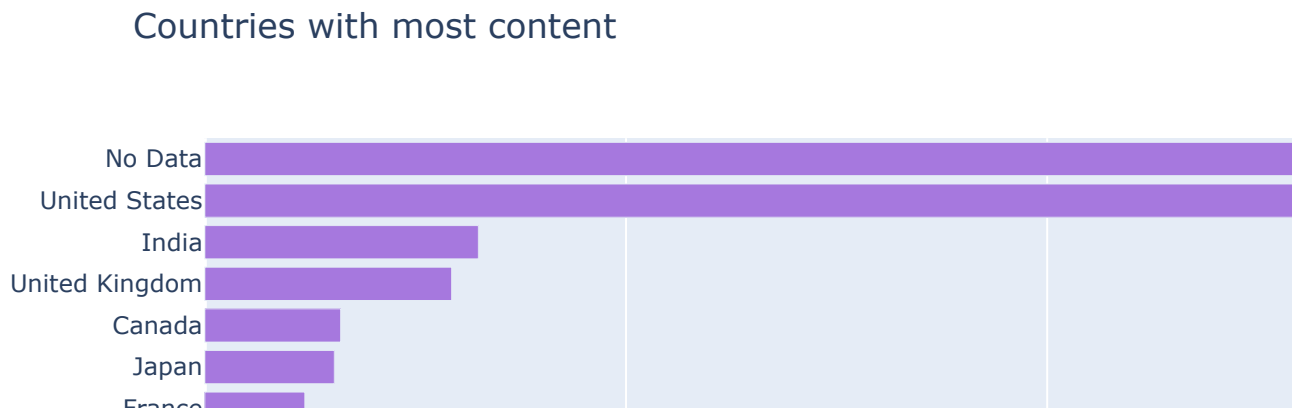
Interactive Map showing Content by Country of Origin

```
country_codes = {'afghanistan': 'AFG', 'albania': 'ALB', 'algeria': 'DZA', 'america
'antigua and barbuda': 'ATG','argentina': 'ARG','armenia': 'ARM','aruba': 'ABW','au
 'bahrain': 'BHR','bangladesh': 'BGD','barbados': 'BRB','belarus': 'BLR','belgium':
 'bolivia': 'BOL','bosnia and herzegovina': 'BIH','botswana': 'BWA','brazil': 'BRA'
 'burkina faso': 'BFA','burma': 'MMR','burundi': 'BDI','cabo verde': 'CPV','cambodi
 'central african republic': 'CAF','chad': 'TCD','chile': 'CHL','china': 'CHN','col
 'Congo republic': 'COG','cook islands': 'COK','costa rica': 'CRI',"cote d'ivoire":
```

```
 'czech republic': 'CZE','denmark': 'DNK','djibouti': 'DJI','dominica': 'DMA','domi
 'el salvador': 'SLV','equatorial guinea': 'GNQ','eritrea': 'ERI','estonia': 'EST',
 'fiji': 'FJI','finland': 'FIN','france': 'FRA','french polynesia': 'PYF','gabon':
 'ghana': 'GHA','gibraltar': 'GIB','greece': 'GRC','greenland': 'GRL','grenada': 'G
 'guinea-bissau': 'GNB','guinea': 'GIN','guyana': 'GUY','haiti': 'HTI','honduras':
 'indonesia': 'IDN','iran': 'IRN','iraq': 'IRQ','ireland': 'IRL','isle of man': 'IM
 'jordan': 'JOR','kazakhstan': 'KAZ','kenya': 'KEN','kiribati': 'KIR','north korea'
 'kyrgyzstan': 'KGZ','laos': 'LAO','latvia': 'LVA','lebanon': 'LBN','lesotho': 'LSO
 'luxembourg': 'LUX','macau': 'MAC','macedonia': 'MKD','madagascar': 'MDG','malawi'
'marshall islands': 'MHL','mauritania': 'MRT','mauritius': 'MUS','mexico': 'MEX','m
 'montenegro': 'MNE','morocco': 'MAR','mozambique': 'MOZ','namibia': 'NAM','nepal':
 'nicaragua': 'NIC','nigeria': 'NGA','niger': 'NER','niue': 'NIU','northern mariana
 'panama': 'PAN','papua new guinea': 'PNG','paraguay': 'PRY','peru': 'PER','philipp
 'qatar': 'QAT','romania': 'ROU','russia': 'RUS','rwanda': 'RWA','saint kitts and n
 'saint pierre and miquelon': 'SPM','saint vincent and the grenadines': 'VCT','samo
 'saudi arabia': 'SAU','senegal': 'SEN','serbia': 'SRB','seychelles': 'SYC','sierra
 'slovenia': 'SVN','solomon islands': 'SLB','somalia': 'SOM','south africa': 'ZAF',
 'suriname': 'SUR','swaziland': 'SWZ','sweden': 'SWE','switzerland': 'CHE','syria':
 'thailand': 'THA','timor-leste': 'TLS','togo': 'TGO','tonga': 'TON','trinidad and
 'tuvalu': 'TUV','uganda': 'UGA','ukraine': 'UKR','united arab emirates': 'ARE','un
 'uzbekistan': 'UZB','vanuatu': 'VUT','venezuela': 'VEN','vietnam': 'VNM','virgin i
 'zimbabwe': 'ZWE'}

## countries
from collections import Counter
colorscale = ["#f7fbff", "#ebf3fb", "#deebf7", "#d2e3f3", "#c6dbef", "#b3d2e9", "#9
    "#85bcdb", "#6baed6", "#57a0ce", "#4292c6", "#3082be", "#2171b5", "#1361a9",
    "#08519c", "#0b4083", "#08306b"
]

def geoplot(ddf):
    country_with_code, country = {}, {}
    shows_countries = ", ".join(result['country'].dropna()).split(", ")
    for c,v in dict(Counter(shows_countries)).items():
        code = ""
        if c.lower() in country_codes:
            code = country_codes[c.lower()]
        country_with_code[code] = v
        country[c] = v

    data = [dict(
            type = 'choropleth',
            locations = list(country_with_code.keys()),
            z = list(country_with_code.values()),
            colorscale = [[0,"rgb(5, 10, 172)"],[0.65,"rgb(40, 60, 190)"],[0.75,"rg
                        [0.80,"rgb(90, 120, 245)"],[0.9,"rgb(106, 137, 247)"],[1,"r
            autocolorscale = False,
            reversescale = True,
            marker = dict(
                line = dict (
                    color = 'gray',
                    width = 0.5
                ) ),
            colorbar = dict(
                autotick = False,
```

```
                title = ''),
          ) ]

    layout = dict(
        title = '',
        geo = dict(
            showframe = False,
            showcoastlines = False,
            projection = dict(
                type = 'Mercator'
            )
        )
    )

    fig = dict( data=data, layout=layout )
    iplot( fig, validate=False, filename='d3-world-map' )
    return country

country_vals = geoplot(result)
tabs = Counter(country_vals).most_common(25)

labels = [_[0] for _ in tabs][::-1]
values = [_[1] for _ in tabs][::-1]
trace1 = go.Bar(y=labels, x=values, orientation="h", name="", marker=dict(color="#a

data = [trace1]
layout = go.Layout(title="Countries with most content", height=700, legend=dict(x=0
fig = go.Figure(data, layout=layout)
fig.show()
```





This interactive Plotly Map and graph shows the countries most represented by the streaming Content sites.

MODEL

```
#Final Dataset used in Model
result.head()
```

	show_id	type	title	director	cast	country	date_added	relea
Amazon	**s1**	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	2021	
	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin	India	2021	

```
#Define Function to remove non relevant symbols and covert it to lower case
def preprocessing(desc):
    desc = desc.lower()
    desc = re.sub('[-=+,#/\?:^$.@*\"※~&%·!』\\‘|\(\)\[\]\<\>`\'…》]', ' ', desc)
    desc = " ".join(desc.split())

    return desc
```

```
#Applying Preprocessing to create new description
result["new_description"] = result["description"].apply(lambda x: preprocessing(x))
print(result.shape)
result.head()
```

```
(22998, 13)
```

	show_id	type	title	director	cast	country	date_added	relea
Amazon	**s1**	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	2021	
	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	2021	
	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	2021	

```
from gensim.models.fasttext import FastText as FT_gensim
#Corpus is Generated from the new description
corpus = result["new_description"].tolist()
#Corpus is used to generate sentence list
sentences = [re.split(' ', str(sentence)) for sentence in corpus]
print(corpus[0])
print(sentences[0])
```

```
a small fishing village must procure a local doctor to secure a lucrative busi
['a', 'small', 'fishing', 'village', 'must', 'procure', 'a', 'local', 'doctor'
```

```
embedding_size = 30
#Defining the FastText Model for Vectorization
FT_model = FT_gensim(vector_size=embedding_size, min_count=2, min_n=2, max_n=5, sg=
                         sample=0.001, window=5, alpha=0.025, min_alpha=0.0001)
#Building Vocabulary from the sentence list
FT_model.build_vocab(sentences)

print('corpus_count: ', FT_model.corpus_count)
print('corpus_total_words: ', FT_model.corpus_total_words)
#Trainig the FastText Model on sentence list
FT_model.train(sentences,
    epochs=FT_model.epochs,
    total_examples=FT_model.corpus_count, total_words=FT_model.corpus_total_words)

print(FT_model)
```

```
corpus_count:  22998
corpus_total_words:  796584
FastText<vocab=22977, vector_size=30, alpha=0.025>
```

```
FT_vector = []
#Applying the trained FT model on the corpus to generate Vectors
for item in corpus:
    FT_vector.append(FT_model.wv[str(item)])
FT_vector = np.asarray(FT_vector)
```

```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
#Generating KMeans clusters from the vectors generated
kmeanModel = KMeans(n_clusters=50, random_state=42).fit(FT_vector)
cluster_id = kmeanModel.predict(FT_vector)
#Adding the cluster idd of each data point in the dataset
result["cluster_id"] = cluster_id
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureW

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the va
```

```
#Dataset after Pre-Processing and Clustering
result.head()
```

		type	title	director	cast	country	date_added	relea
	show_id							
Amazon	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	2021	
	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	2021	
	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert LaSardo, R...	United States	2021	
	s4	Movie	Pink: Staying True	Sonia Anderson	Interviews with: Pink, Adele, BeyoncÃ©, Britne...	United States	2021	

```
#The Recommendation Sytemm using Clustering
#It takes 2 inputs, the movie name and the exploration flag
#If exp is 0, the sytem searches within the c;uster of searched title
#If exp is 1, it will search on nearby cluster
#This helps in keeping the recommendations fresh and promoting exploration to new c
def recommendation_system(title_name, exp):

    top_k = 5
    title_row = result[result["title"] == title_name].copy()
    if (exp == 1):
        search_df = result[result["cluster_id"].isin(title_row["cluster_id"]-1)].co
    else:
        search_df = result[result["cluster_id"].isin(title_row["cluster_id"])].copy
    search_df = search_df.drop(search_df[search_df["title"] == title_name].index)

    search_df["Similarity"] = search_df.apply(lambda x: FT_model.wv.similarity(titl
    search_df.sort_values(by=["Similarity"], ascending=False, inplace=True)

    return search_df[["title", "Similarity"]].head(top_k)
```

```
recommendation_system("Ernest Saves Christmas", 0)
```

		title	Similarity
show_id			
Netflix	s1027	Barbie & Chelsea: The Lost Birthday	[0.98405844]
Amazon	s4109	The Yummy Gummy Search For Santa	[0.98195267]
	s9378	Noddy Saves Christmas	[0.980627]
Hulu	s1248	Elf	[0.9798547]
Netflix	s5017	Trailer Park Boys Live at the North Pole	[0.9780766]

```
recommendation_system("National Parks Adventure", 0)
```

		title	Similarity
show_id			
Netflix	s4052	2,215	[0.9926673]
	s1917	Rize	[0.992051]
	s1124	Magical Andes	[0.9911406]
Hulu	s490	Summer of Soul	[0.99111295]
	s723	The Crime of the Century	[0.990374]

```
recommendation_system("Secrets of Deception", 1)
```

		title	Similarity
show_id			
Amazon	s7049	Ascension	[0.9817956]
	s816	TAN	[0.97967124]
Hulu	s1733	Cashback	[0.9794499]
Amazon	s3779	Santa Jaws	[0.97803766]
Disney	s263	Night at the Museum	[0.9772865]

End of Code