Netflix Data Analysis

The Netflix dataset consists of data collected by Netflix on the movies and TV shows that they offer. This data includes description, year of release, and other metadata about the content. Analysis of this dataset can be used to understand viewer preferences, analysis of the Netflix dataset might include:

- What type of shows that Netflix is focusing on?
- What type of content that Netflix focus more?
- The trend of the shows on Netflix by years
- Top 5 directors and actors that appeared most on Netflix?
- Most produced genres on Netflix
- Sentiment content analysis on Netflix shows over the years

Installing and importing necessary packages for analysis

In [ ]:
```
pip install textblob
```

In [ ]:
```
pip install plotly==5.11.0
```

In [11]:
```
import numpy as np # linear algebra
import pandas as pd # for data preparation
import matplotlib.pyplot as plt # for data visualization
%matplotlib inline
from textblob import TextBlob # for sentiment analysis
```

In [12]:
```
pd.options.mode.chained_assignment = None
```

Inputing the CSV file and initial checking of the data

In [14]:
```
df = pd.read_csv('netflix_titles.csv')
```

In [15]:
```
df.shape
```

Out[15]:
```
(8807, 12)
```

In [16]:
```
df.dtypes
```

Out[16]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [19]:
```
df["rating"].unique()
```

Out[19]:
```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', '74 min', '84 min', '66 min', 'NR', nan,
       'TV-Y7-FV', 'UR'], dtype=object)
```

In [20]:
```
rating_list = ['74 min', '84 min', '66 min']
```

['74 min', '84 min', '66 min'] should not be in the rating, move them to their respective column

In [21]:
```
wrong_rating_list = df[df.rating.isin(rating_list)]
```

In [22]:
```
wrong_rating_list
```

Out[22]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min	NaN	Movies	Louis C.K. muses on religion, eternal love, gi...
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min	NaN	Movies	Emmy-winning comedy writer Louis C.K. brings h...
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	66 min	NaN	Movies	The comic puts his trademark hilarious/thought...

In [23]:
```
df.loc[df['show_id'] == 's5542', 'duration'] = '74 min'
df.loc[df['show_id'] == 's5795', 'duration'] = '84 min'
df.loc[df['show_id'] == 's5814', 'duration'] = '66 min'
```

In [24]:
```
wrong_rating_list = df[df.rating.isin(rating_list)]
```

In [25]:
```
## Louis C.K. show rating are TV-MA after researching
df.loc[df['director'] == 'Louis C.K.', 'rating'] = 'TV-MA'
```

In [26]:
```
df[df['director']=='Louis C.K.']
```

Out[26]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	TV-MA	74 min	Movies	Louis C.K. muses on religion, eternal love, gi...
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	TV-MA	84 min	Movies	Emmy-winning comedy writer Louis C.K. brings h...
5813	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	August 15, 2016	2015	TV-MA	66 min	Movies	The comic puts his trademark hilarious/thought...

Creating the rating distribution table for the pie chart

In [28]:
```
r_distribution = df['rating'].value_counts().rename_axis('rating').reset_index(name='counts')
r_distribution
```

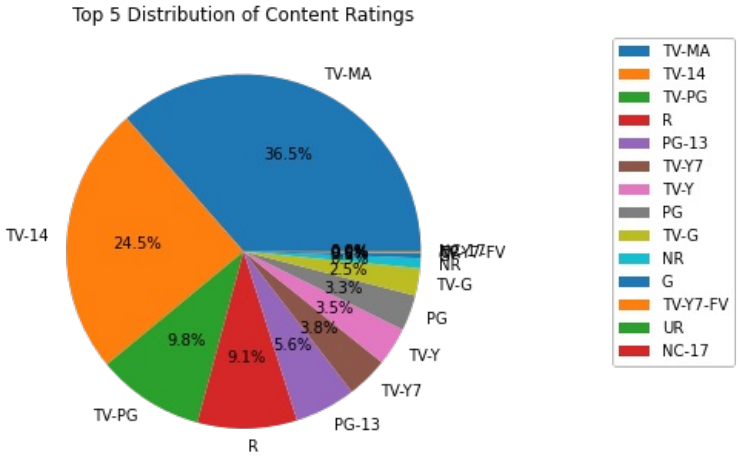
Out[28]:

	rating	counts
0	TV-MA	3210
1	TV-14	2160
2	TV-PG	863
3	R	799
4	PG-13	490
5	TV-Y7	334
6	TV-Y	307
7	PG	287

8	TV-G	220
9	NR	80
10	G	41
11	TV-Y7-FV	6
12	UR	3
13	NC-17	3

In [29]:
```
## Setting labels and values
p_labels = r_distribution.rating
p_values = r_distribution.counts
```

In [30]:
```
## Visualizing the pie chart
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
ax.pie(p_values, labels = p_labels, autopct='%1.1f%%')
plt.title('Top 5 Distribution of Content Ratings')
plt.legend(bbox_to_anchor=(1.25,1))
plt.show()
```



The data in the chart suggests that the vast majority of Netflix's offereings are intended for adult viewers as they are rated "TV-MA"

Content Trend by Year

In [32]:
```
# Extracting the movie and tv show, grouping them together by the year and prepping for the chart
df1 = df[['type','release_year']]
df1 = df1.rename(columns={"release_year": "Year of Release"})
df2 = df1.groupby(['Year of Release', 'type']).size().reset_index(name="Content Counts")
df2 = df2[df2['Year of Release'] >= 2010]
```

In [33]:
```
df2
```

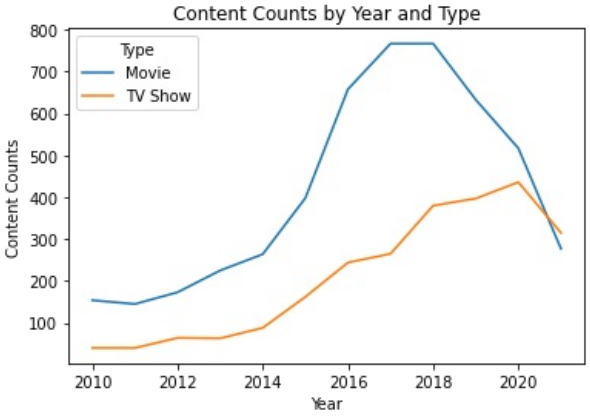
Out[33]:

	Year of Release	type	Content Counts
95	2010	Movie	154
96	2010	TV Show	40
97	2011	Movie	145
98	2011	TV Show	40
99	2012	Movie	173
100	2012	TV Show	64
101	2013	Movie	225
102	2013	TV Show	63
103	2014	Movie	264
104	2014	TV Show	88
105	2015	Movie	398
106	2015	TV Show	162

107	2016	Movie	658
108	2016	TV Show	244
109	2017	Movie	767
110	2017	TV Show	265
111	2018	Movie	767
112	2018	TV Show	380
113	2019	Movie	633
114	2019	TV Show	397
115	2020	Movie	517
116	2020	TV Show	436
117	2021	Movie	277
118	2021	TV Show	315

In [34]:

```
#reshaping the dataframe
df3 = df2.pivot(index='Year of Release', columns='type', values='Content Counts')

# Create the plot
df3.plot()

# Customize the plot
plt.xlabel('Year')
plt.ylabel('Content Counts')
plt.title('Content Counts by Year and Type')
plt.legend(title='Type')
plt.show()
```



The line graph shows that Netflix's production is focused more on movies. However, there has been a decline in production decline around 2019, which can be due to the pandemic. Around 2021, movie production seems to decline more than TV shows for the first time since 2010.

Top 5 directors and actors that appeared most on Netflix

In [36]:

```
df.head(1)
```

Out[36]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...

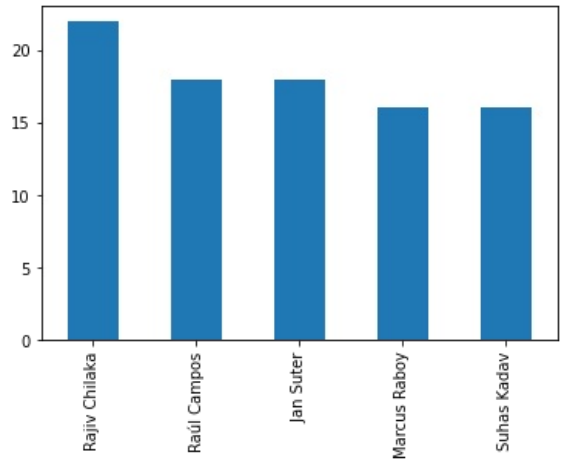
In [ ]:

Top 5 directors

In [37]:

```
# drop rows with NaN values in the director column
clean_data = df.dropna(subset=['director'])

# spliting the name
clean_data.loc[:, 'director_split'] = clean_data['director'].apply(lambda x: x.split(','))
```

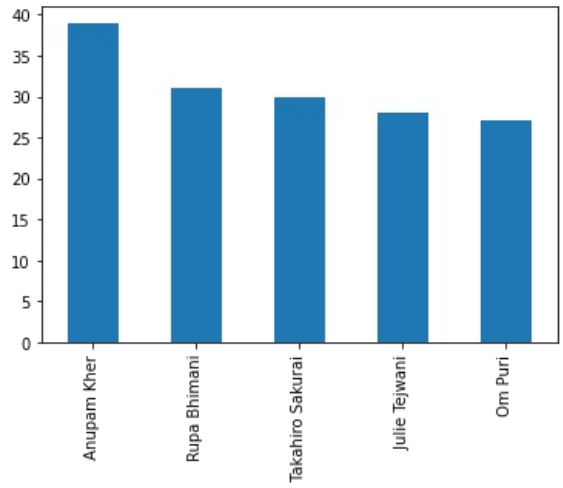
```
# use the explode function to create a new row for each split name
clean_data = clean_data.explode('director_split')

# count the frequency of each name and sort in descending order
name_counts = clean_data['director_split'].value_counts().sort_values(ascending=False)

# get the top 5 names
top_names = name_counts.head(5)

# plot the top names using a bar chart
top_names.plot(kind='bar')
plt.show()
```



Top 5 actors

In [38]:

```
# drop rows with NaN values in the cast column
clean_data_cast = df.dropna(subset=['cast'])

# split the names
clean_data_cast['cast_split'] = clean_data_cast['cast'].apply(lambda x: x.split(','))

# use the explode function to create a new row for each split name
clean_data_cast = clean_data_cast.explode('cast_split')

# count the frequency of each name and sort in descending order
name_counts = clean_data_cast['cast_split'].value_counts().sort_values(ascending=False)

# get the top 5 names
top_names = name_counts.head(5)

# plot the top names using a bar chart
top_names.plot(kind='bar')
plt.show()
```

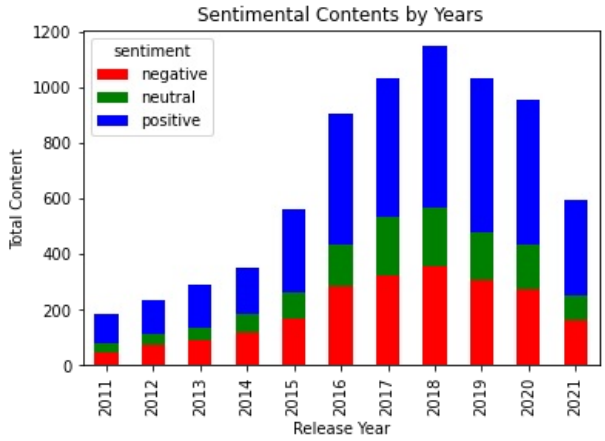


Sentimental Contents Analysis

This analysis is done to find out what type of content Netflix is producing over the years.

In [39]:

```
# select only the rows with a year greater than 2010
df = df[df['release_year'] > 2010]

# create an empty list to store the sentiments
sentiments = []

# loop through the description column and analyze the sentiment of each description
for description in df['description']:
    sentiment = TextBlob(description).sentiment.polarity
    # assign a label based on the polarity range
    if sentiment > 0:
        label = 'positive'
    elif sentiment < 0:
        label = 'negative'
    else:
        label = 'neutral'
    sentiments.append(label)

# add the sentiments to the DataFrame
df['sentiment'] = sentiments

# group the data by year and compute the count of each sentiment for each year
yearly_sentiment = df.groupby('release_year')['sentiment'].value_counts()

# pivot the data from the sentiment column into columns
yearly_sentiment = yearly_sentiment.unstack()

# plot the yearly sentiment
plot = yearly_sentiment.plot(kind='bar', stacked=True, color=['red', 'green', 'blue'], legend=True)

plt.title('Sentimental Contents by Years')
plt.xlabel('Release Year')
plt.ylabel('Total Content')

plt.show()
```



From the stacked bar chart, we can see that Netflix focuses more on overall positive content on their platform.

Top genres on Netflix

In [75]:

```
# Filter the dataframe to only include rows where the year is >= 2010
df_filtered = df[df['release_year'] >= 2010]

# Split the genres column into separate rows
df_splitted = df_filtered.assign(listed_in=df_filtered.listed_in.str.split(',')).explode('listed_in')

# Drop rows with null values in the genres column
df_cleaned = df_splitted.dropna(subset=['listed_in'])

# Group the data by year
df_grouped = df_cleaned.groupby('release_year')

# Count the frequency of each genre within each year group
genre_counts = df_grouped['listed_in'].value_counts().unstack()
```

In [76]:

```
genre_counts
```

Out[76]:

listed_in	Anime Features	Children & Family Movies	Classic & Cult TV	Comedies	Crime TV Shows	Cult Movies	Documentaries	Docuseries	Dramas	Faith & Spirituality	...	Romantic TV Shows	Sci-Fi & Fantasy
release_year													

2011	6.0	NaN	NaN	42.0	NaN	NaN	NaN	3.0	84.0	3.0	...	NaN	NaN
2012	6.0	NaN	NaN	32.0	3.0	6.0	NaN	NaN	66.0	3.0	...	NaN	NaN
2013	9.0	NaN	NaN	42.0	NaN	3.0	3.0	2.0	66.0	5.0	...	NaN	NaN
2014	6.0	NaN	NaN	42.0	NaN	NaN	3.0	NaN	99.0	13.0	...	NaN	4.0
2015	3.0	NaN	3.0	47.0	3.0	NaN	3.0	NaN	140.0	5.0	...	NaN	2.0
2016	6.0	8.0	NaN	81.0	3.0	3.0	8.0	NaN	219.0	8.0	...	NaN	NaN
2017	6.0	7.0	NaN	70.0	3.0	3.0	10.0	2.0	233.0	24.0	...	NaN	NaN
2018	12.0	21.0	NaN	94.0	NaN	NaN	3.0	4.0	239.0	32.0	...	NaN	11.0
2019	6.0	8.0	NaN	74.0	3.0	NaN	3.0	NaN	210.0	19.0	...	NaN	2.0
2020	3.0	9.0	3.0	94.0	NaN	NaN	2.0	NaN	148.0	14.0	...	NaN	NaN
2021	6.0	3.0	NaN	26.0	3.0	NaN	NaN	NaN	59.0	6.0	...	3.0	NaN

11 rows × 69 columns

In [80]:
```
# replacing the nan values with 0 to make calulation on the numbers.
genre_counts = genre_counts.fillna(0)
```

In [81]:
```
genre_counts
```

Out[81]:

listed_in	Anime Features	Children & Family Movies	Classic & Cult TV	Comedies	Crime TV Shows	Cult Movies	Documentaries	Docuseries	Dramas	Faith & Spirituality	...	Romantic TV Shows	Sci-Fi & Fantasy
release_year													
2011	6.0	0.0	0.0	42.0	0.0	0.0	0.0	3.0	84.0	3.0	...	0.0	0.0
2012	6.0	0.0	0.0	32.0	3.0	6.0	0.0	0.0	66.0	3.0	...	0.0	0.0
2013	9.0	0.0	0.0	42.0	0.0	3.0	3.0	2.0	66.0	5.0	...	0.0	0.0
2014	6.0	0.0	0.0	42.0	0.0	0.0	3.0	0.0	99.0	13.0	...	0.0	4.0
2015	3.0	0.0	3.0	47.0	3.0	0.0	3.0	0.0	140.0	5.0	...	0.0	2.0
2016	6.0	8.0	0.0	81.0	3.0	3.0	8.0	0.0	219.0	8.0	...	0.0	0.0
2017	6.0	7.0	0.0	70.0	3.0	3.0	10.0	2.0	233.0	24.0	...	0.0	0.0
2018	12.0	21.0	0.0	94.0	0.0	0.0	3.0	4.0	239.0	32.0	...	0.0	11.0
2019	6.0	8.0	0.0	74.0	3.0	0.0	3.0	0.0	210.0	19.0	...	0.0	2.0
2020	3.0	9.0	3.0	94.0	0.0	0.0	2.0	0.0	148.0	14.0	...	0.0	0.0
2021	6.0	3.0	0.0	26.0	3.0	0.0	0.0	0.0	59.0	6.0	...	3.0	0.0

11 rows × 69 columns

In [98]:
```
top_genres = genre_counts.apply(lambda x: x.nlargest(3).index, axis=1)
```

In [99]:
```
print(top_genres)
```

```
release_year
2011    Index([' International Movies', 'Comedies', ' ...
2012    Index([' International Movies', 'Comedies', 'D...
2013    Index([' International Movies', 'Dramas', 'Com...
2014    Index([' International Movies', 'Dramas', 'Com...
2015    Index([' International Movies', 'Dramas', 'Com...
2016    Index([' International Movies', 'Dramas', 'Com...
2017    Index([' International Movies', 'Dramas', 'Com...
2018    Index([' International Movies', 'Dramas', ' In...
2019    Index([' International Movies', 'Dramas', 'Com...
2020    Index([' International Movies', 'Dramas', 'Com...
2021    Index([' International Movies', 'Dramas', 'Com...
dtype: object
```

From the list we can see that International movies is shown as the top genre This is expected because from our previous chart both our top director and cast are categorized from international movies. Hence, international movies is shown here as the top genre.

In [104…

```
genre_counts = genre_counts.fillna(0)

def get_top_genres(row):
  # Select the top 4 genres
  top_genres = row.nlargest(4).index
  # Remove the top genre
  top_genres = top_genres[1:]
  # Remove the 'genre_' prefix from each genre name
  top_genres = top_genres.str.replace('listed_in_', '')
  return top_genres

# Get the names of the top 3 genres for each year
top_genres = genre_counts.apply(get_top_genres, axis=1)

print(top_genres)
```

```
release_year
2011    Index(['Comedies', ' Dramas', 'Dramas'], dtype...
2012    Index(['Comedies', 'Dramas', ' Independent Mov...
2013    Index(['Dramas', 'Comedies', ' Independent Mov...
2014    Index(['Dramas', 'Comedies', ' Dramas'], dtype...
2015    Index(['Dramas', 'Comedies', ' Independent Mov...
2016    Index(['Dramas', 'Comedies', ' Independent Mov...
2017    Index(['Dramas', 'Comedies', ' Independent Mov...
2018    Index(['Dramas', ' Independent Movies', 'Comed...
2019    Index(['Dramas', 'Comedies', ' Dramas'], dtype...
2020    Index(['Dramas', 'Comedies', ' Romantic Movies...
2021    Index(['Dramas', 'Comedies', ' Dramas'], dtype...
dtype: object
```

To get the accurate presentation, I have removed the internatial movies from our results. From there, we can see that Dramas and Comedies are Netflix's top genres throughout the years.

Conclusion

We could get further analysis by getting data on the rating of each movie from IMDB/ rotten tomatoes and find out what type of genre gets the most popular rating by the viewers. We could also build a movie prediction model based on that dataset. Furthermore, if we could also gather the budget data on each title, it could be used to plot the trend in spending on movies and TV shows over the years.