In [66]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
```

In [3]:
```
df=pd.read_csv("netflix1.csv")
df
```

Out[3]:

	show_id	type	title	director	country	date_added	release_yea
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993
...	...	...	...	...	...	...	...
8785	s8797	TV Show	Yunus Emre	Not Given	Turkey	1/17/2017	2016
8786	s8798	TV Show	Zak Storm	Not Given	United States	9/13/2018	2016
8787	s8801	TV Show	Zindagi Gulzar Hai	Not Given	Pakistan	12/15/2016	2012
8788	s8784	TV Show	Yoko	Not Given	Pakistan	6/23/2018	2016
8789	s8786	TV Show	YOM	Not Given	Pakistan	06-07-2018	2016

8790 rows × 10 columns

In [4]:
```
df.shape
```

Out[4]: (8790, 10)

In [5]:
```
df.head(5 )
```

Out[5]:

	show_id	type	title	director	country	date_added	release_year	ra
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	P
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	T
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	T
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	T
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	T

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

	release_year
count	8790.000000
mean	2014.183163
std	8.825466
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64
```

In [9]:
```
df.dropna(axis=1)
```

Out[9]:

	show_id	type	title	director	country	date_added	release_yea
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993
...	...	...	...	...	...	...	...
8785	s8797	TV Show	Yunus Emre	Not Given	Turkey	1/17/2017	2016
8786	s8798	TV Show	Zak Storm	Not Given	United States	9/13/2018	2016
8787	s8801	TV Show	Zindagi Gulzar Hai	Not Given	Pakistan	12/15/2016	2012
8788	s8784	TV Show	Yoko	Not Given	Pakistan	6/23/2018	2016
8789	s8786	TV Show	YOM	Not Given	Pakistan	06-07-2018	2016

8790 rows × 10 columns

In [10]:
```
df.dropna(axis=0,inplace=True)
```

In [11]:
```
df.duplicated().sum()
```

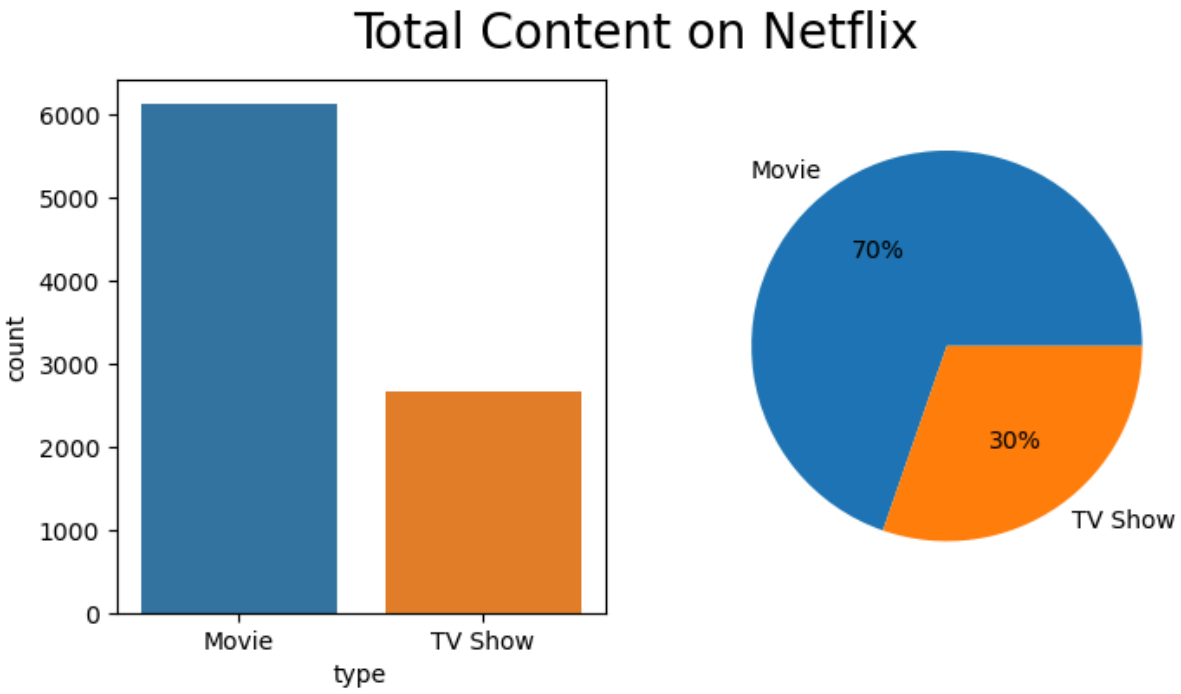
Out[11]:
```
0
```

In [68]:
```
df['type'].value_counts()
```

Out[68]:
```
Movie      6126
TV Show    2664
Name: type, dtype: int64
```

In [75]:
```
freq=df['type'].value_counts()
fig, axes=plt.subplots(1,2, figsize=(8, 4))
sns.countplot(df, x=df['type'], ax=axes[0])
plt.pie(freq, labels=['Movie', 'TV Show'], autopct='%.0f%%')
plt.suptitle('Total Content on Netflix', fontsize=20)
```

Out[75]:
```
Text(0.5, 0.98, 'Total Content on Netflix')
```



In [15]:
```
df.drop_duplicates(inplace=True)
```

In [20]:
```
df.dropna(subset=['director', 'listed_in', 'country'],
inplace=True)
```

In [77]:
```
df['rating'].value_counts()
```

Out[77]:
```
TV-MA       3205
TV-14       2157
TV-PG        861
R            799
PG-13        490
TV-Y7        333
TV-Y         306
PG           287
TV-G         220
NR            79
G             41
TV-Y7-FV       6
NC-17          3
UR             3
Name: rating, dtype: int64
```

In [22]:
```
df
```

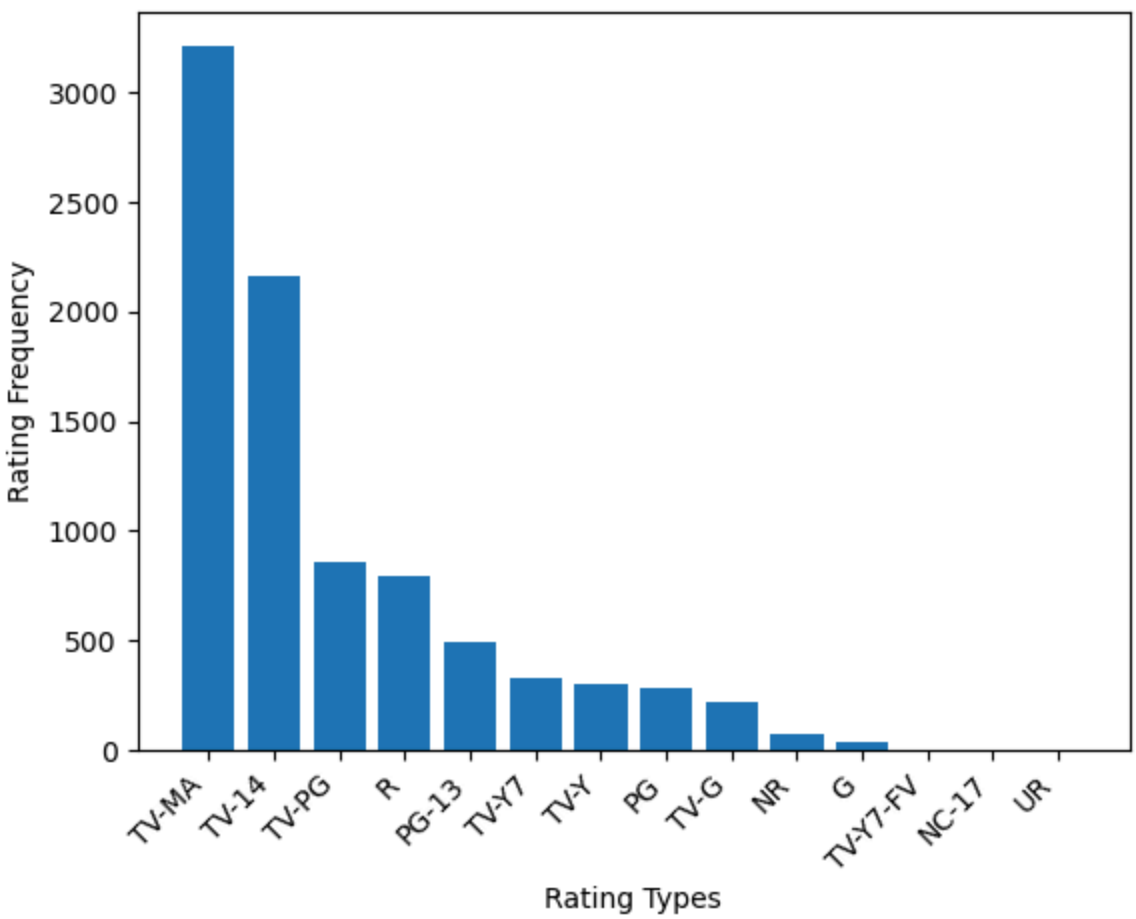
Out[22]:

	show_id	type	title	director	country	date_added	release_yea
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993
...	...	...	...	...	...	...	...
8785	s8797	TV Show	Yunus Emre	Not Given	Turkey	1/17/2017	2016
8786	s8798	TV Show	Zak Storm	Not Given	United States	9/13/2018	2016
8787	s8801	TV Show	Zindagi Gulzar Hai	Not Given	Pakistan	12/15/2016	2012
8788	s8784	TV Show	Yoko	Not Given	Pakistan	6/23/2018	2016
8789	s8786	TV Show	YOM	Not Given	Pakistan	06-07-2018	2016

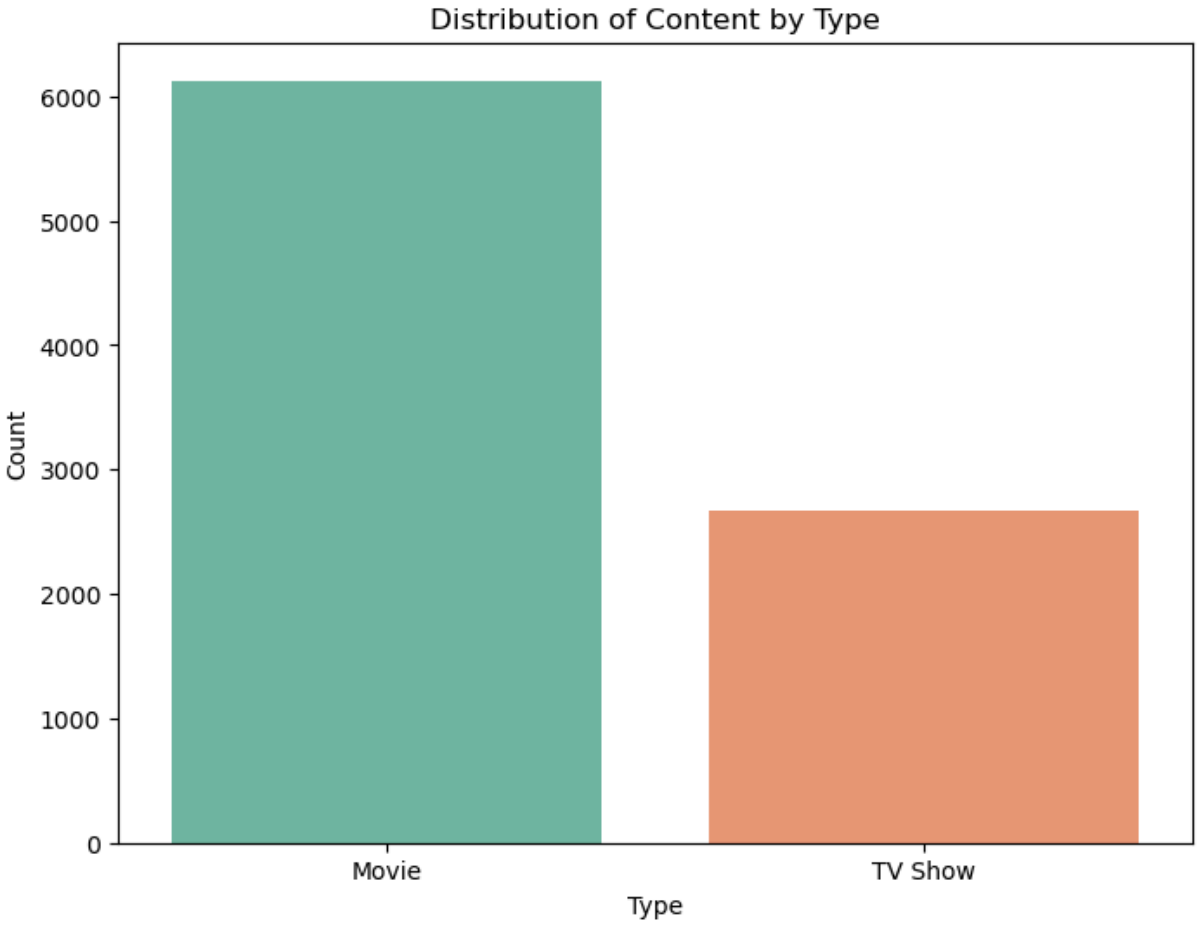
8790 rows × 10 columns

In [90]:

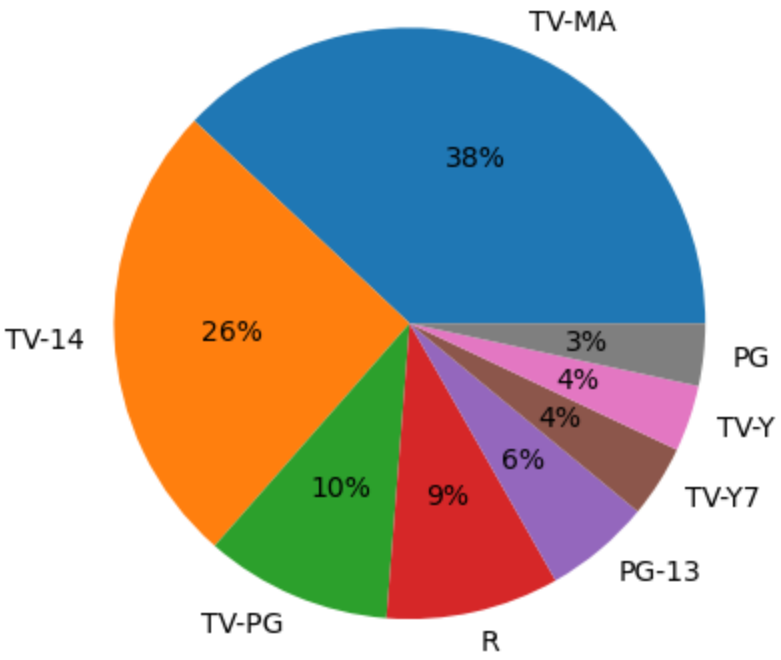
```
ratings = df['rating'].value_counts().reset_index()
ratings.columns = ['rating', 'count']  # Rename columns for clarity
ratings = ratings.sort_values(by='count', ascending=False)

plt.bar(ratings['rating'], ratings['count'])
plt.xticks(rotation=45, ha='right')
plt.xlabel("Rating Types")
plt.ylabel("Rating Frequency")
plt.suptitle('Rating on Netflix', fontsize=20)
plt.show()
```



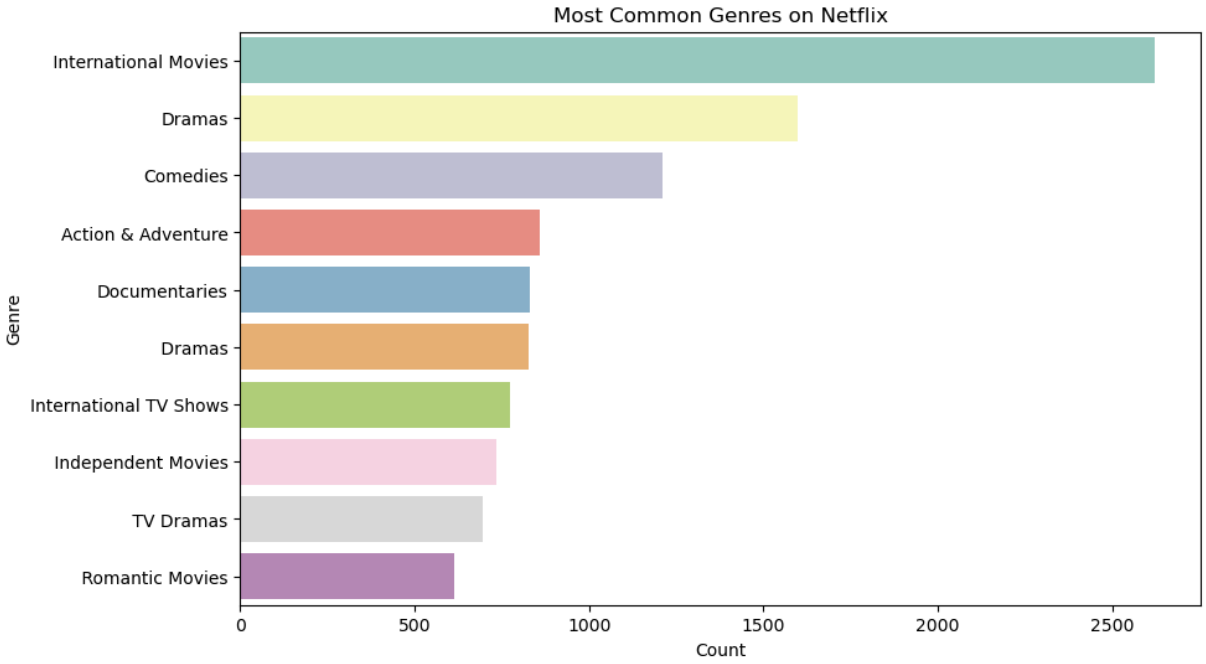
In [55]:
```
# Count the number of Movies and TV Shows
type_counts = df['type'].value_counts()
# Plot the distribution
plt.figure(figsize=(8, 6))
sns.barplot(x=type_counts.index, y=type_counts.values,
palette='Set2')
plt.title('Distribution of Content by Type')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



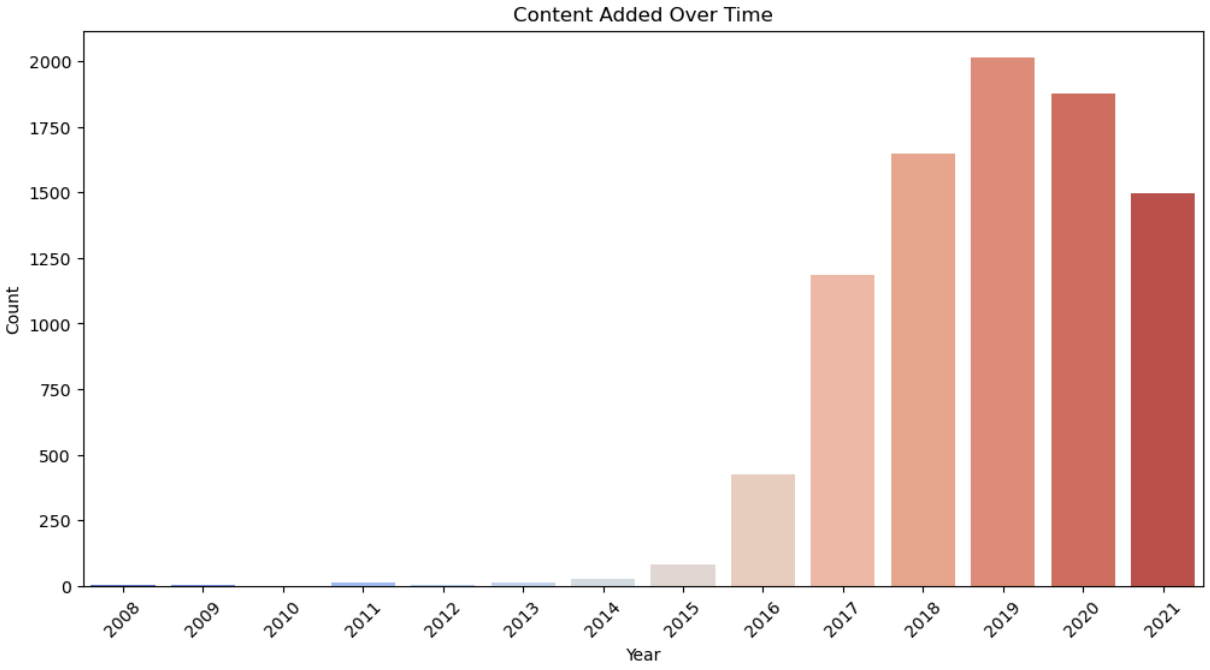
In [92]:
```
# Assuming df is your DataFrame
ratings = df['rating'].value_counts().reset_index()
ratings.columns = ['rating', 'count']  # Rename columns for clarity
ratings = ratings.sort_values(by='count', ascending=False)

# Plot pie chart for top 8 ratings
plt.pie(ratings['count'][:8], labels=ratings['rating'][:8], autopct='%.0f%%'
plt.suptitle('Rating on Netflix', fontsize=20)
plt.show()
```



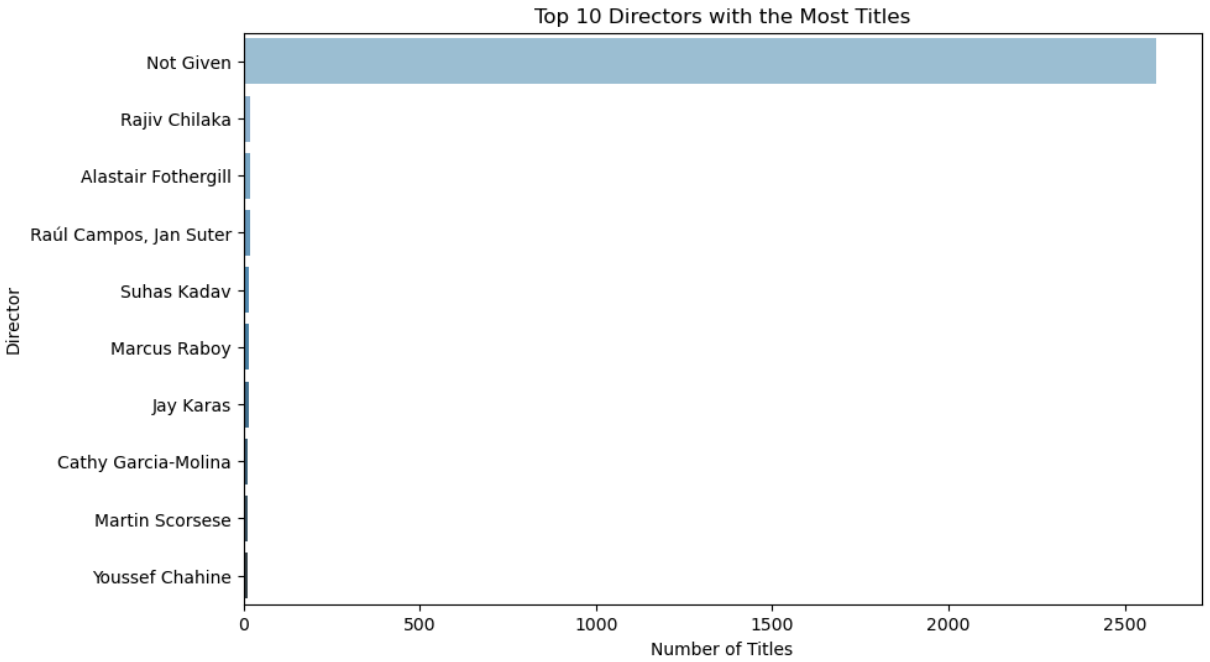
In [59]:
```
# Split the 'listed_in' column and count genres
df['genres'] = df['listed_in'].apply(lambda x: x.split(','))
all_genres = sum(df['genres'], [])
genre_counts = pd.Series(all_genres).value_counts().head(10)
# Plot the most common genres
plt.figure(figsize=(10, 6))
sns.barplot(x=genre_counts.values, y=genre_counts.index,
palette='Set3')
plt.title('Most Common Genres on Netflix')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```



In [94]:
```
# Ensure 'date_added' is in datetime format
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Extract year and month
df['year_added'] = df['date_added'].dt.year
df['month_added'] = df['date_added'].dt.month

# Plot content added over the years
plt.figure(figsize=(12, 6))
sns.countplot(x='year_added', data=df, palette='coolwarm')
plt.title('Content Added Over Time')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



In [63]:
```
# Count titles by director
top_directors = d['director'].value_counts().head(10)
# Plot top directors
plt.figure(figsize=(10, 6))
sns.barplot(x=top_directors.values, y=top_directors.index,
palette='Blues_d')
plt.title('Top 10 Directors with the Most Titles')
plt.xlabel('Number of Titles')
plt.ylabel('Director')
plt.show()
```



In [98]:
```
pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeabl
eNote: you may need to restart the kernel to use updated packages.
  WARNING: The script wordcloud_cli.exe is installed in 'C:\Users\akansha ra
wat\AppData\Roaming\Python\Python311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
warning, use --no-warn-script-location.
Collecting wordcloud
  Obtaining dependency information for wordcloud from https://files.pythonho
sted.org/packages/f5/b0/247159f61c5d5d6647171bef84430b7efad4db504f0229674024
f3a4f7f2/wordcloud-1.9.3-cp311-cp311-win_amd64.whl.metadata
  Downloading wordcloud-1.9.3-cp311-cp311-win_amd64.whl.metadata (3.5 kB)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib
\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-p
ackages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\si
te-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib->wordcloud) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anacon
da3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Downloading wordcloud-1.9.3-cp311-cp311-win_amd64.whl (300 kB)
   ---------------------------------------- 0.0/300.2 kB ? eta -:--:--
   ------------------------------------ --- 276.5/300.2 kB 5.7 MB/s eta 0:0
0:01
   ---------------------------------------- 300.2/300.2 kB 4.6 MB/s eta 0:0
0:00
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.3
```

In [104…

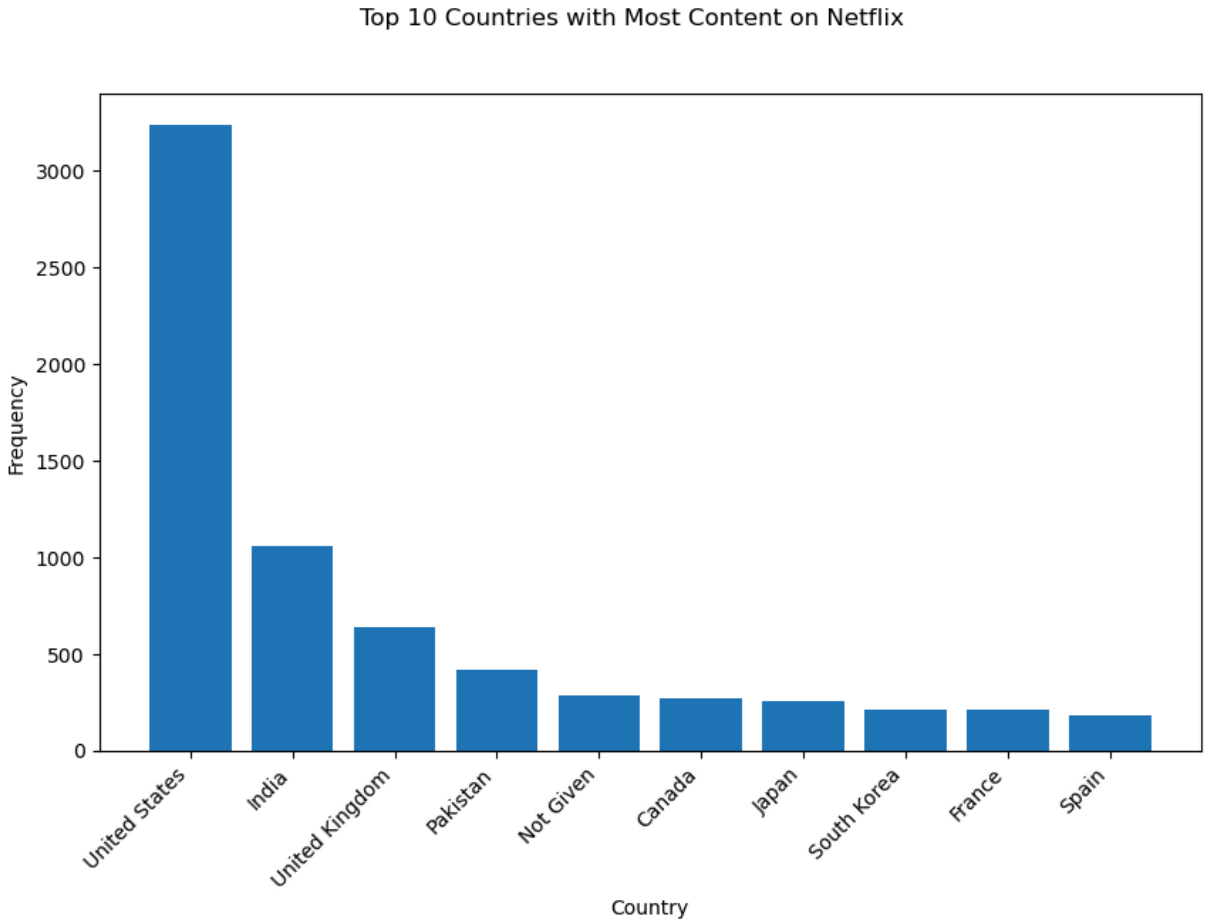
```python
from wordcloud import WordCloud

# Generate word cloud for movie titles
movie_titles = df[df['type'] == 'Movie']['title']
wordcloud = WordCloud(width=800, height=400, background_color='black').gener

# Plot word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[104], line 1
----> 1 from wordcloud import WordCloud
      3 # Generate word cloud for movie titles
      4 movie_titles = df[df['type'] == 'Movie']['title']

ModuleNotFoundError: No module named 'wordcloud'
```

In [103…
```
df['country'].value_counts()
```

Out[103…
```
United States     3240
India             1057
United Kingdom     638
Pakistan           421
Not Given          287
                  ...
Iran                 1
West Germany         1
Greece               1
Zimbabwe             1
Soviet Union         1
Name: country, Length: 86, dtype: int64
```

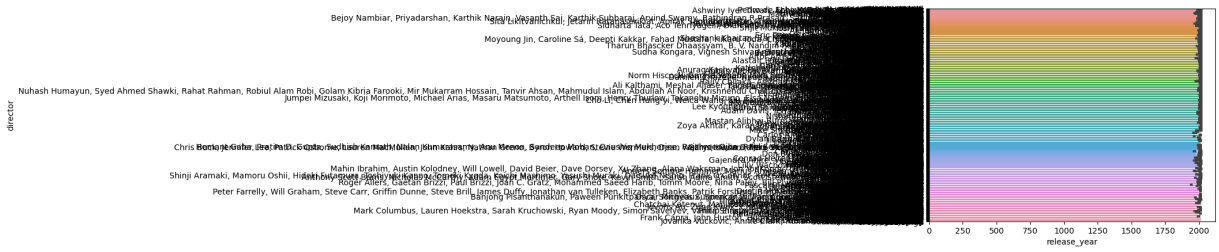
In [99]:
```
# Assuming 'df' is your DataFrame
top_ten_countries = df['country'].value_counts().reset_index()
top_ten_countries.columns = ['country', 'count']  # Rename columns for clari
top_ten_countries = top_ten_countries.sort_values(by='count', ascending=Fals

# Plot top 10 countries with most content
plt.figure(figsize=(10, 6))
plt.bar(top_ten_countries['country'], top_ten_countries['count'])
plt.xticks(rotation=45, ha='right')
plt.xlabel("Country")
plt.ylabel("Frequency")
plt.suptitle("Top 10 Countries with Most Content on Netflix")
plt.show()
```



barplot

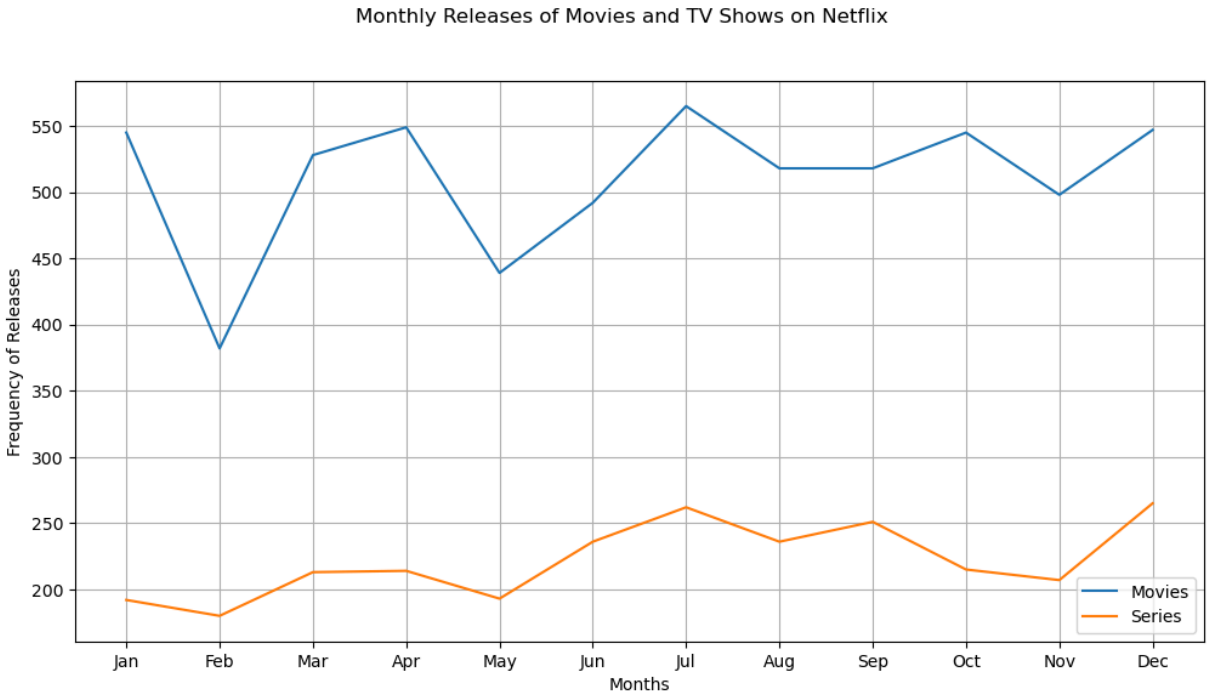
In [40]:
```
sns.barplot(y=df.director,x=df.release_year)
```

Out[40]: <Axes: xlabel='release_year', ylabel='director'>



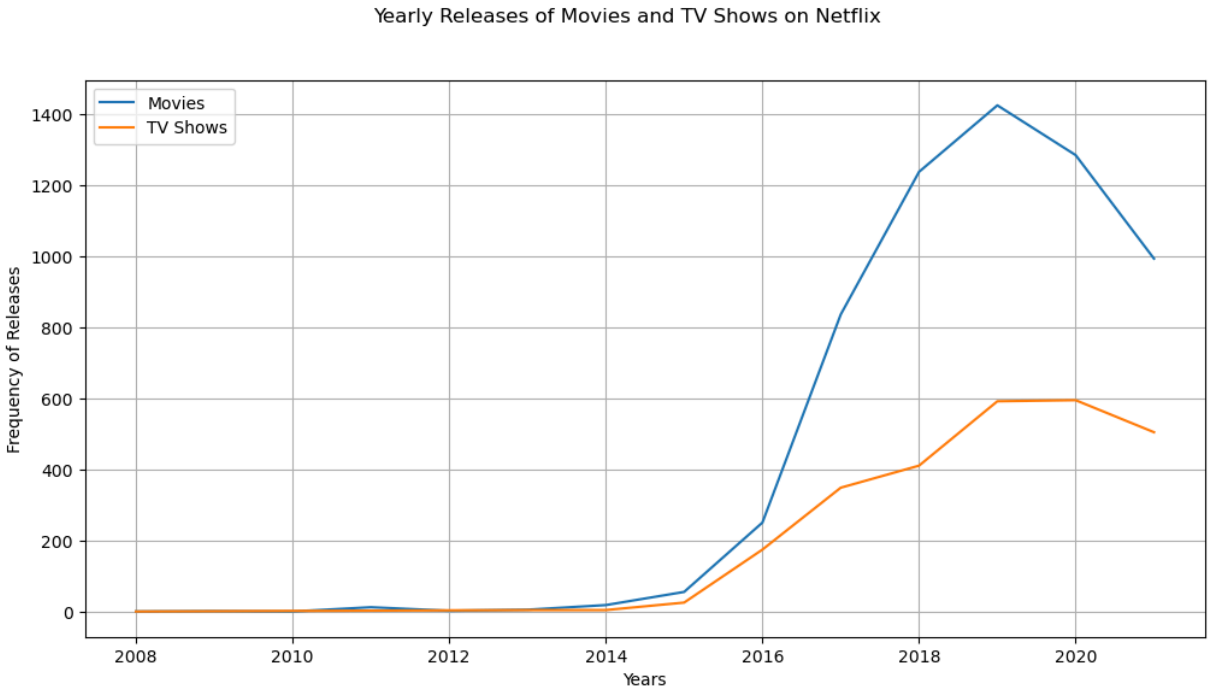
In [105…
```
# Assuming 'df' is your DataFrame and 'date_added' is already in datetime fo
df['year'] = df['date_added'].dt.year
df['month'] = df['date_added'].dt.month
df['day'] = df['date_added'].dt.day

# Calculate monthly releases for movies and TV shows
monthly_movie_release = df[df['type'] == 'Movie']['month'].value_counts().so
monthly_series_release = df[df['type'] == 'TV Show']['month'].value_counts()

# Plotting the monthly releases
plt.figure(figsize=(12, 6))
```

```
plt.plot(monthly_movie_release.index, monthly_movie_release.values, label='M
plt.plot(monthly_series_release.index, monthly_series_release.values, label=
plt.xlabel("Months")
plt.ylabel("Frequency of Releases")
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', '
plt.legend()
plt.grid(True)
plt.suptitle("Monthly Releases of Movies and TV Shows on Netflix")
plt.show()
```



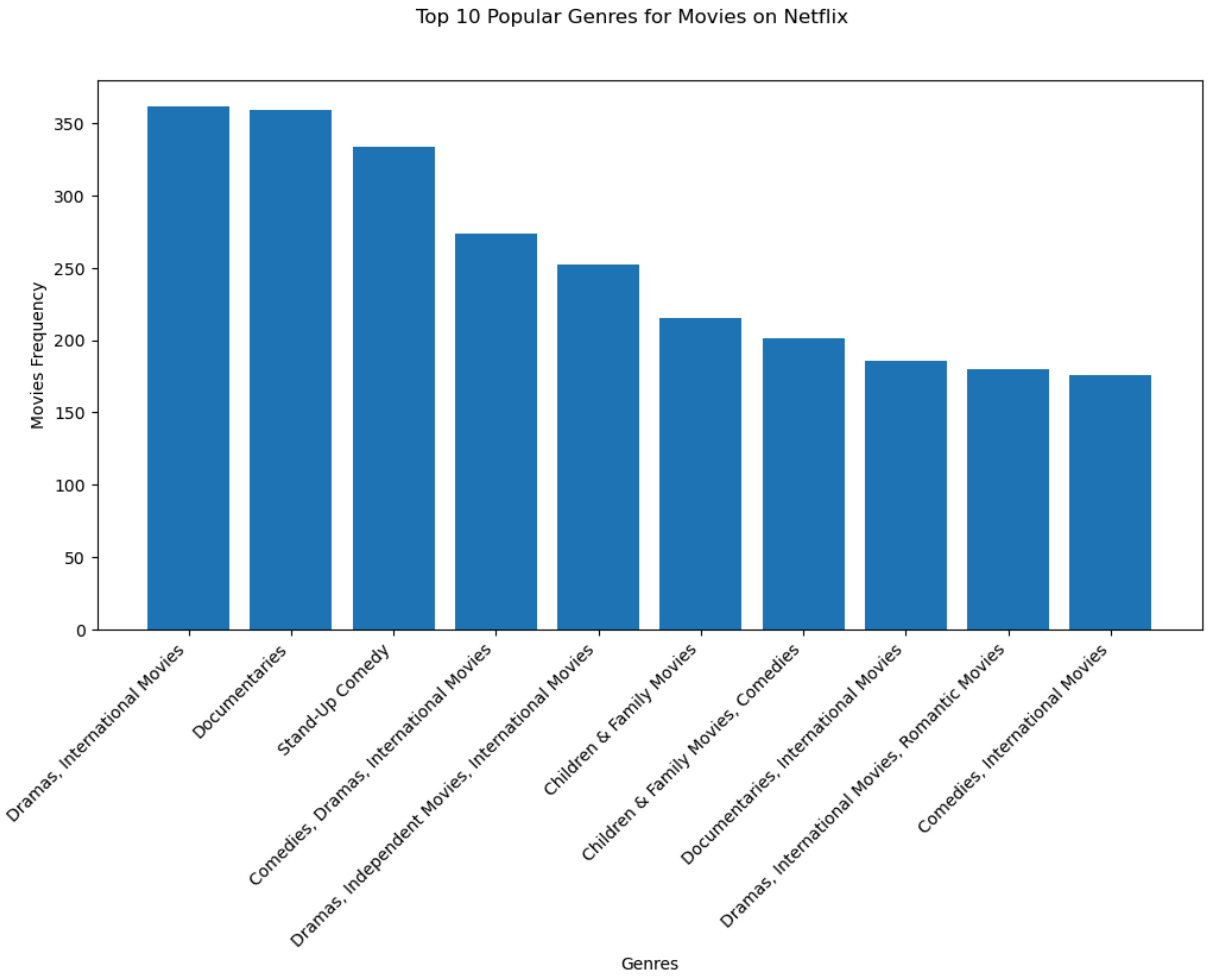
In [106…

```
# Calculate yearly releases for movies and TV shows
yearly_movie_releases = df[df['type'] == 'Movie']['year'].value_counts().sor
yearly_series_releases = df[df['type'] == 'TV Show']['year'].value_counts().

# Plotting the yearly releases
plt.figure(figsize=(12, 6))
plt.plot(yearly_movie_releases.index, yearly_movie_releases.values, label='M
plt.plot(yearly_series_releases.index, yearly_series_releases.values, label=
plt.xlabel("Years")
plt.ylabel("Frequency of Releases")
plt.grid(True)
plt.suptitle("Yearly Releases of Movies and TV Shows on Netflix")
plt.legend()
plt.show()
```



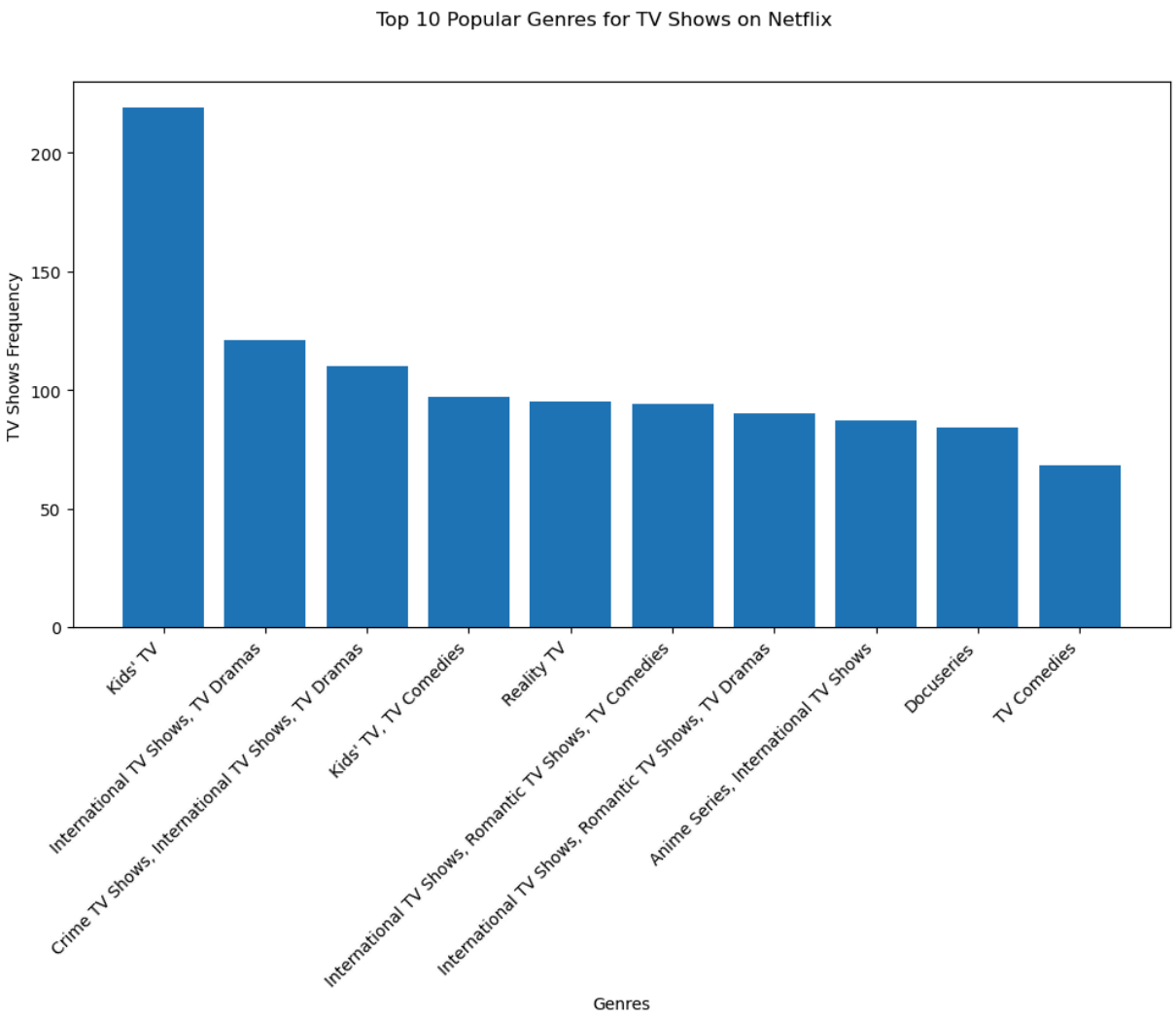
In [107…

```
# Calculate top 10 popular genres for movies and TV shows
popular_movie_genre = df[df['type'] == 'Movie'].groupby("listed_in").size().
popular_series_genre = df[df['type'] == 'TV Show'].groupby("listed_in").size

# Plot top 10 popular movie genres
plt.figure(figsize=(12, 6))
plt.bar(popular_movie_genre.index, popular_movie_genre.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("Movies Frequency")
plt.suptitle("Top 10 Popular Genres for Movies on Netflix")
plt.show()
```



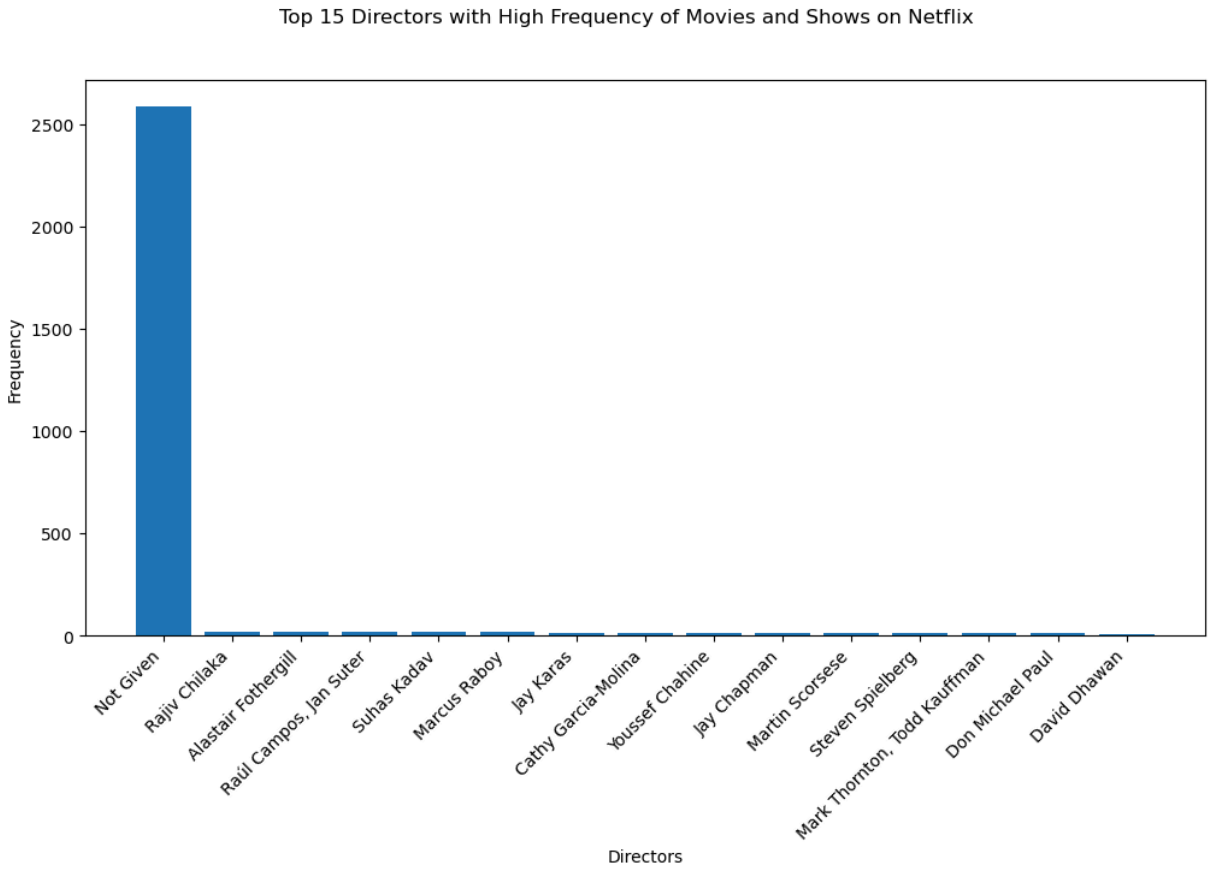
In [108…

```
# Plot top 10 popular TV show genres
plt.figure(figsize=(12, 6))
plt.bar(popular_series_genre.index, popular_series_genre.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Genres")
plt.ylabel("TV Shows Frequency")
plt.suptitle("Top 10 Popular Genres for TV Shows on Netflix")
plt.show()
```



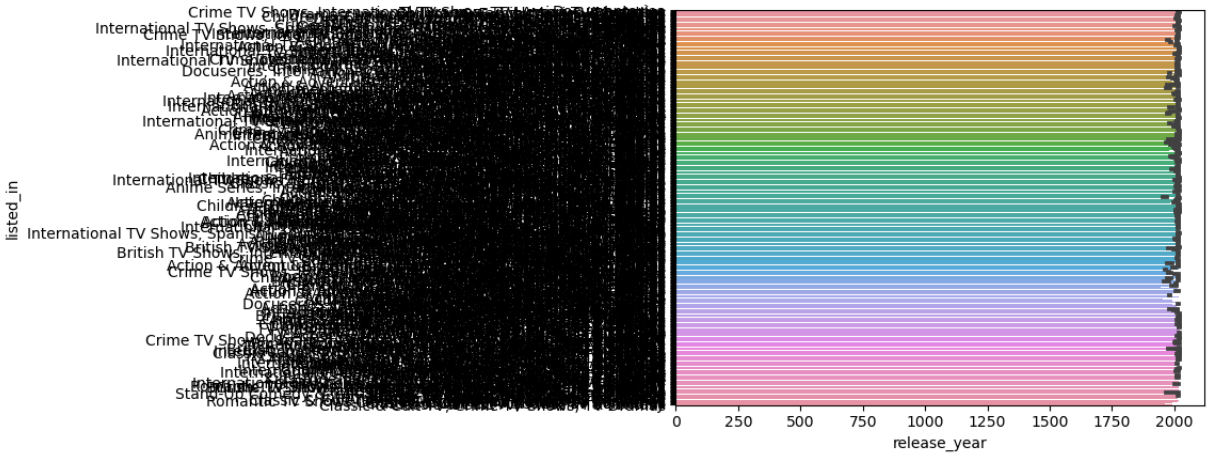
In [109…

```
# Calculate top 15 directors with the highest frequency of movies and shows
directors = df['director'].value_counts().reset_index()
directors.columns = ['director', 'count']  # Rename columns for clarity
directors = directors.sort_values(by='count', ascending=False).head(15)

# Plot top 15 directors
plt.figure(figsize=(12, 6))
plt.bar(directors['director'], directors['count'])
plt.xticks(rotation=45, ha='right')
plt.xlabel("Directors")
plt.ylabel("Frequency")
plt.suptitle("Top 15 Directors with High Frequency of Movies and Shows on Ne
plt.show()
```



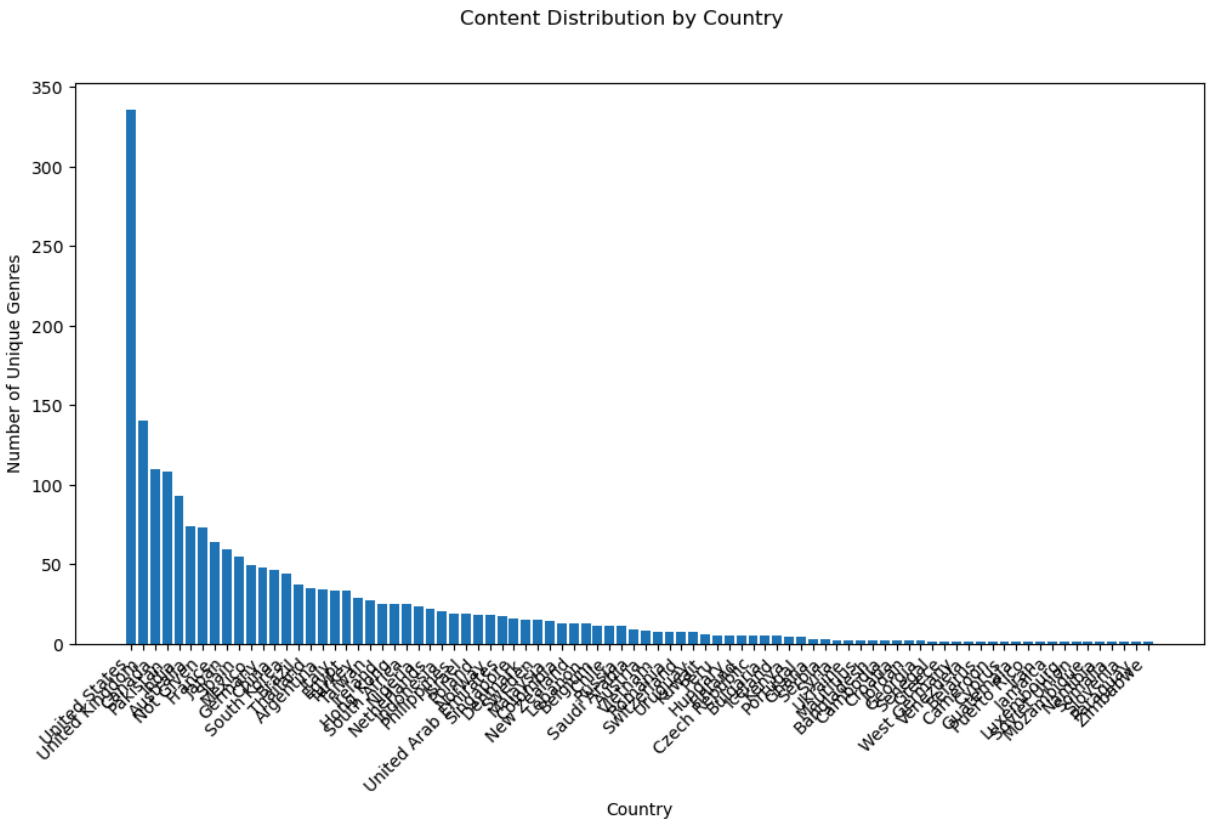
In [54]:
```
sns.barplot(y="listed_in",x="release_year",data=df)
```

Out[54]: <Axes: xlabel='release_year', ylabel='listed_in'>



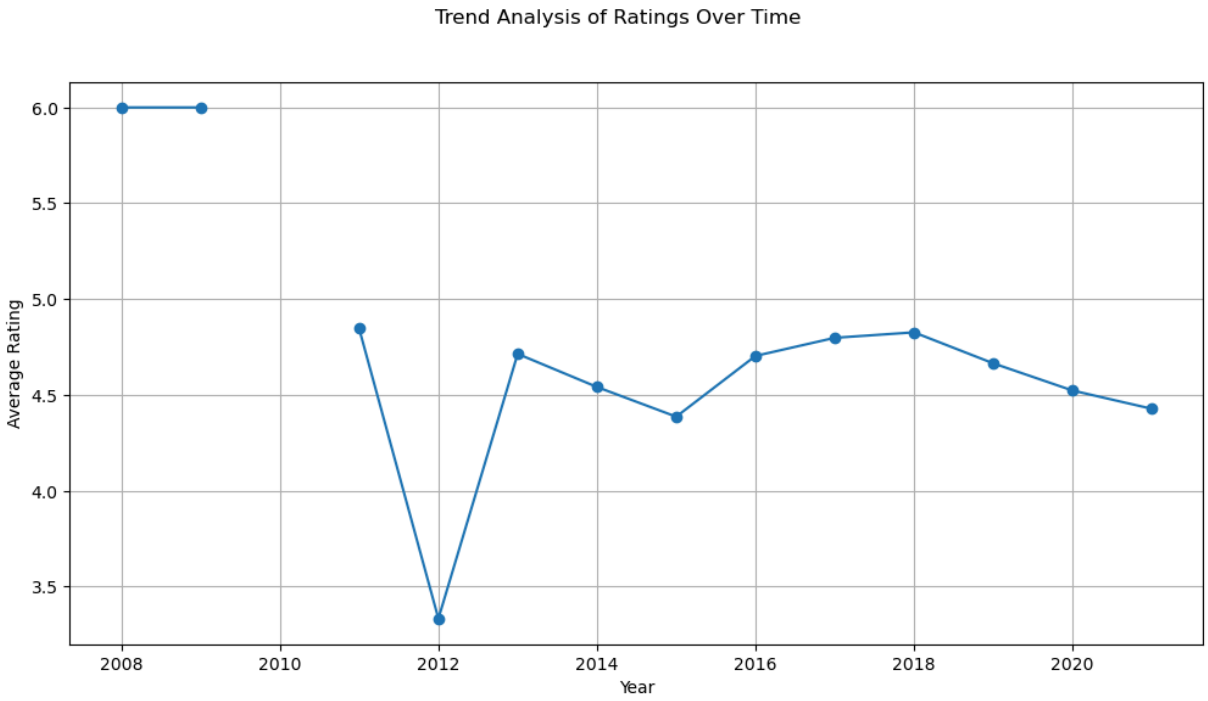
In [110…
```
# Count content by country and sort by the number of unique genres
country_genre_counts = df.groupby('country')['listed_in'].nunique().reset_ir
country_genre_counts.columns = ['country', 'unique_genres']
country_genre_counts = country_genre_counts.sort_values(by='unique_genres',

plt.figure(figsize=(12, 6))
plt.bar(country_genre_counts['country'], country_genre_counts['unique_genres
plt.xticks(rotation=45, ha='right')
plt.xlabel("Country")
plt.ylabel("Number of Unique Genres")
```

```
plt.suptitle("Content Distribution by Country")
plt.show()
```



In [111…

```
# Convert 'rating' to categorical and assign numerical values
rating_mapping = {'G': 1, 'PG': 2, 'PG-13': 3, 'R': 4, 'NC-17': 5, 'TV-Y': 1
df['rating_numeric'] = df['rating'].map(rating_mapping)

# Average rating by year
average_rating_per_year = df.groupby('year')['rating_numeric'].mean().reset_

plt.figure(figsize=(12, 6))
plt.plot(average_rating_per_year['year'], average_rating_per_year['rating_nu
plt.xlabel("Year")
plt.ylabel("Average Rating")
plt.grid(True)
plt.suptitle("Trend Analysis of Ratings Over Time")
plt.show()
```

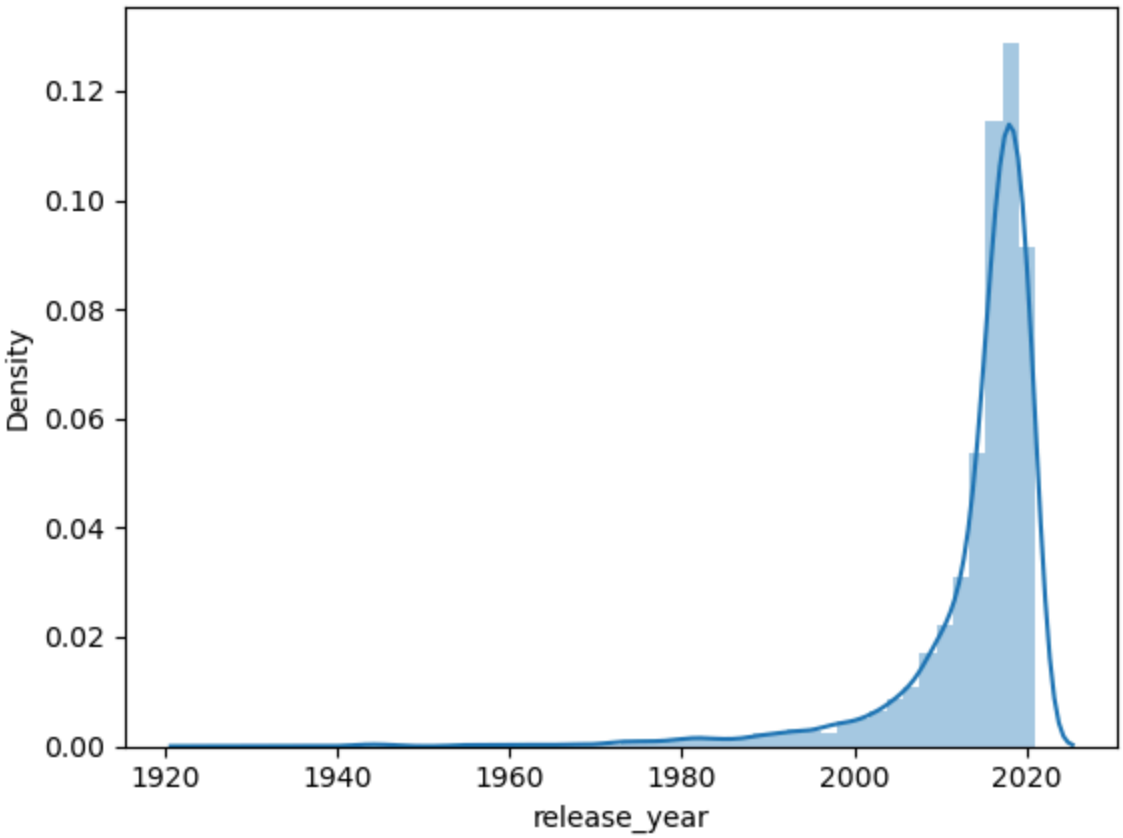


distribution plot

In [36]:
```
sns.distplot(df["release_year"], hist=True
             )
```

```
C:\Users\akansha rawat\AppData\Local\Temp\ipykernel_12048\2244818705.py:1: U
serWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["release_year"], hist=True
```

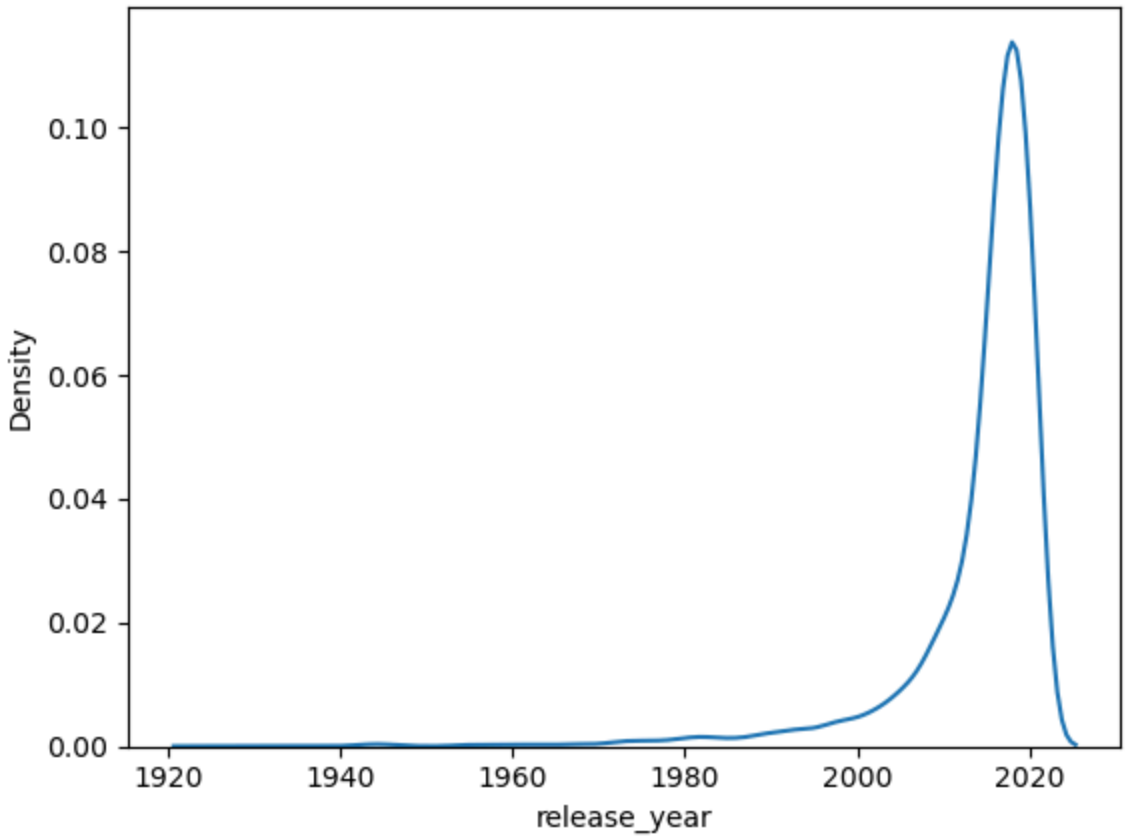
Out[36]:
```
<Axes: xlabel='release_year', ylabel='Density'>
```



kde plot

In [47]:
```
sns.kdeplot(df["release_year"])
```

Out[47]:
```
<Axes: xlabel='release_year', ylabel='Density'>
```

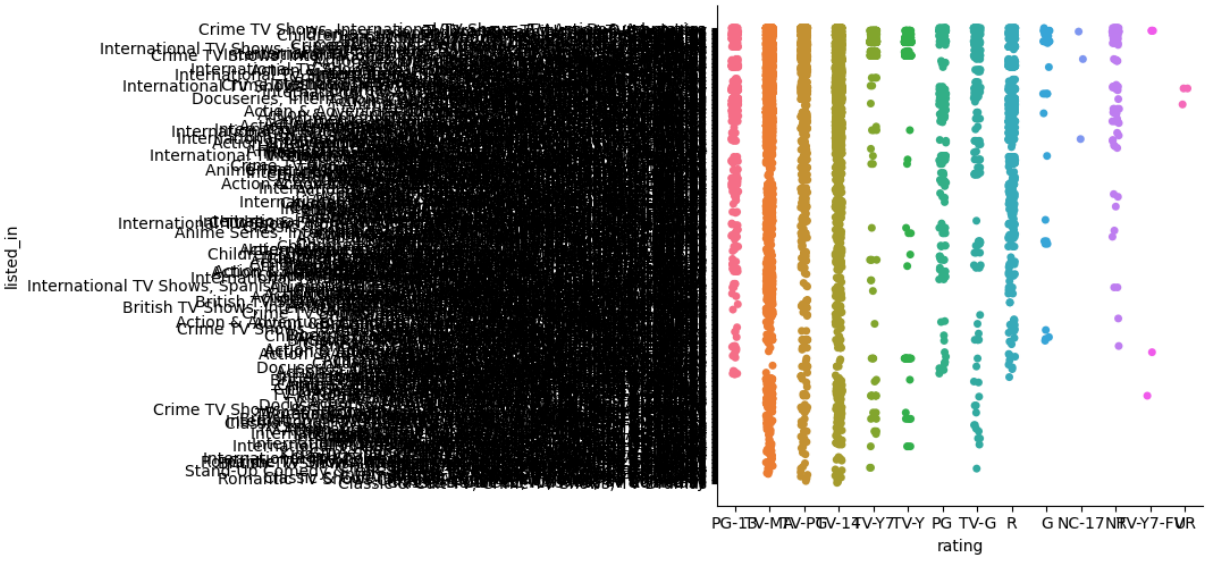


categoriacl plot

In [49]:
```
sns.catplot(x="rating",y="listed_in",data=df, hue="rating")
```

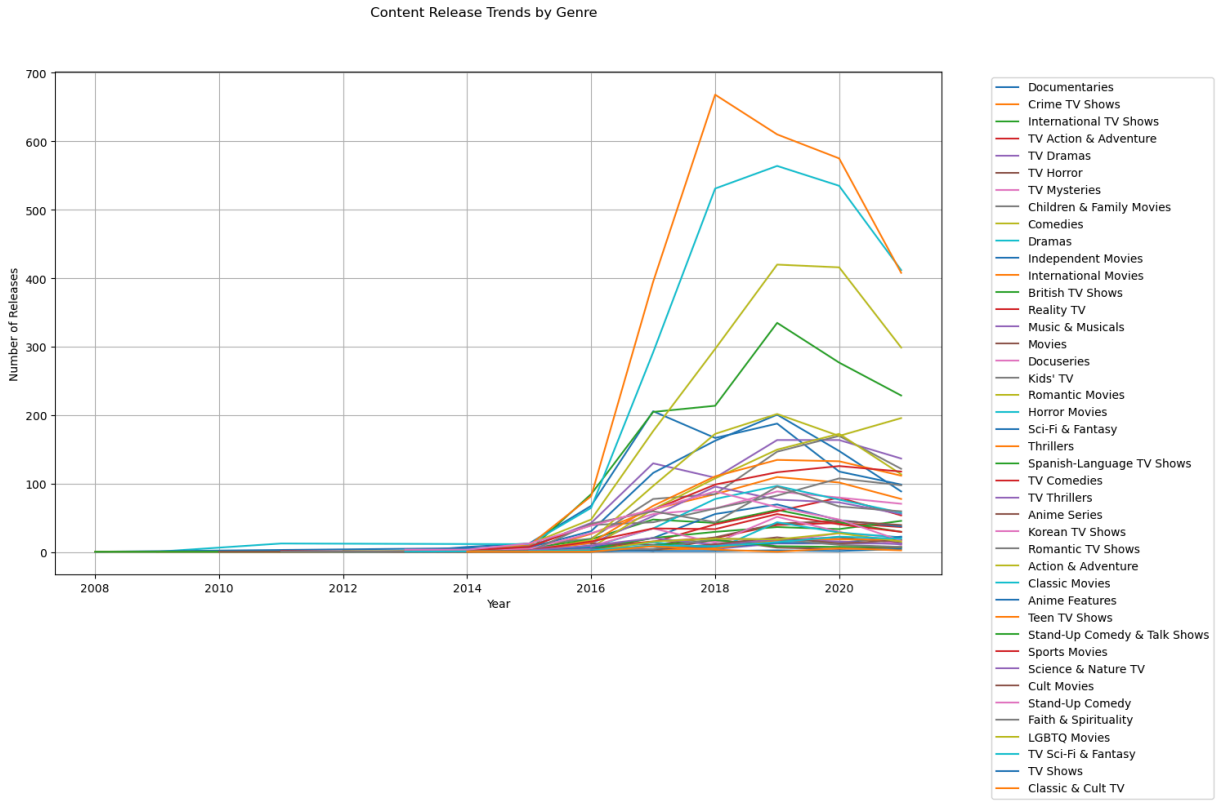
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarn
ing: Tight layout not applied. The left and right margins cannot be made lar
ge enough to accommodate all axes decorations.
  self._figure.tight_layout(*args, **kwargs)
```

Out[49]: <seaborn.axisgrid.FacetGrid at 0x2aa7d2ab650>



In [113…

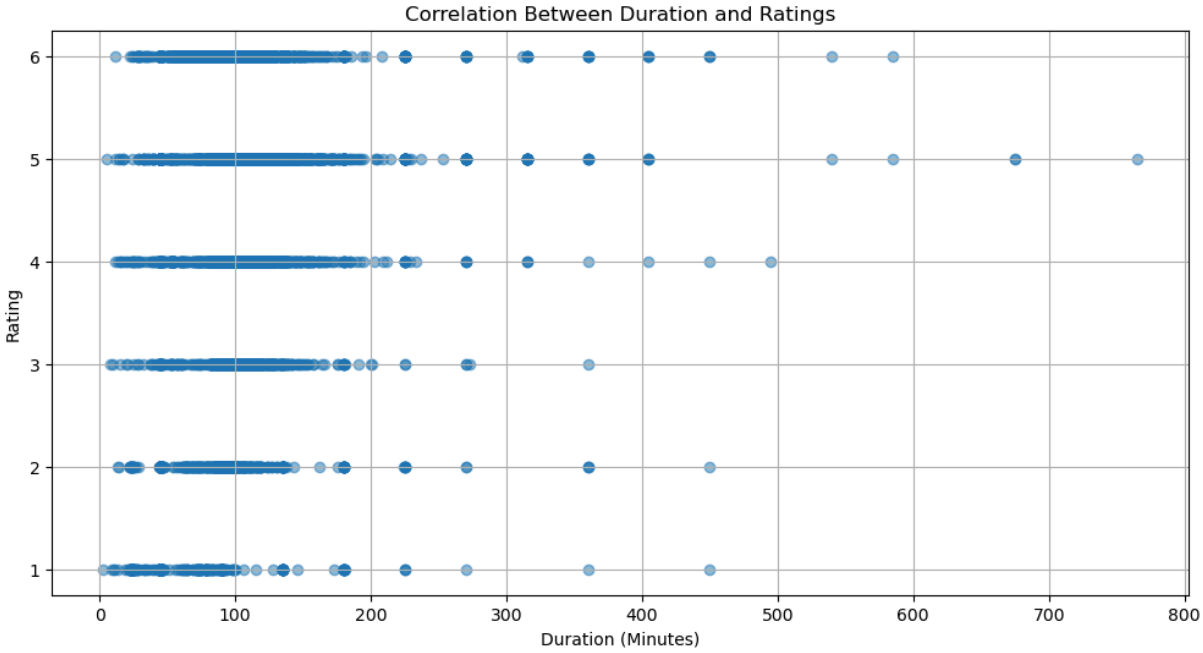
```
# Split 'listed_in' into separate genres
df['genres'] = df['listed_in'].str.split(', ').apply(lambda x: [genre.strip(

# Create a DataFrame for each genre with release counts by year
genre_release_trends = pd.DataFrame()
for genre in df['genres'].explode().unique():
    genre_df = df[df['genres'].apply(lambda x: genre in x)]
    genre_release = genre_df.groupby('year').size().reset_index(name='count'
    genre_release['genre'] = genre
    genre_release_trends = pd.concat([genre_release_trends, genre_release])

plt.figure(figsize=(14, 8))
for genre in genre_release_trends['genre'].unique():
    genre_data = genre_release_trends[genre_release_trends['genre'] == genre
    plt.plot(genre_data['year'], genre_data['count'], label=genre)

plt.xlabel("Year")
plt.ylabel("Number of Releases")
plt.legend(loc='best', bbox_to_anchor=(1.05, 1))
plt.grid(True)
plt.suptitle("Content Release Trends by Genre")
plt.show()
```



In [112…

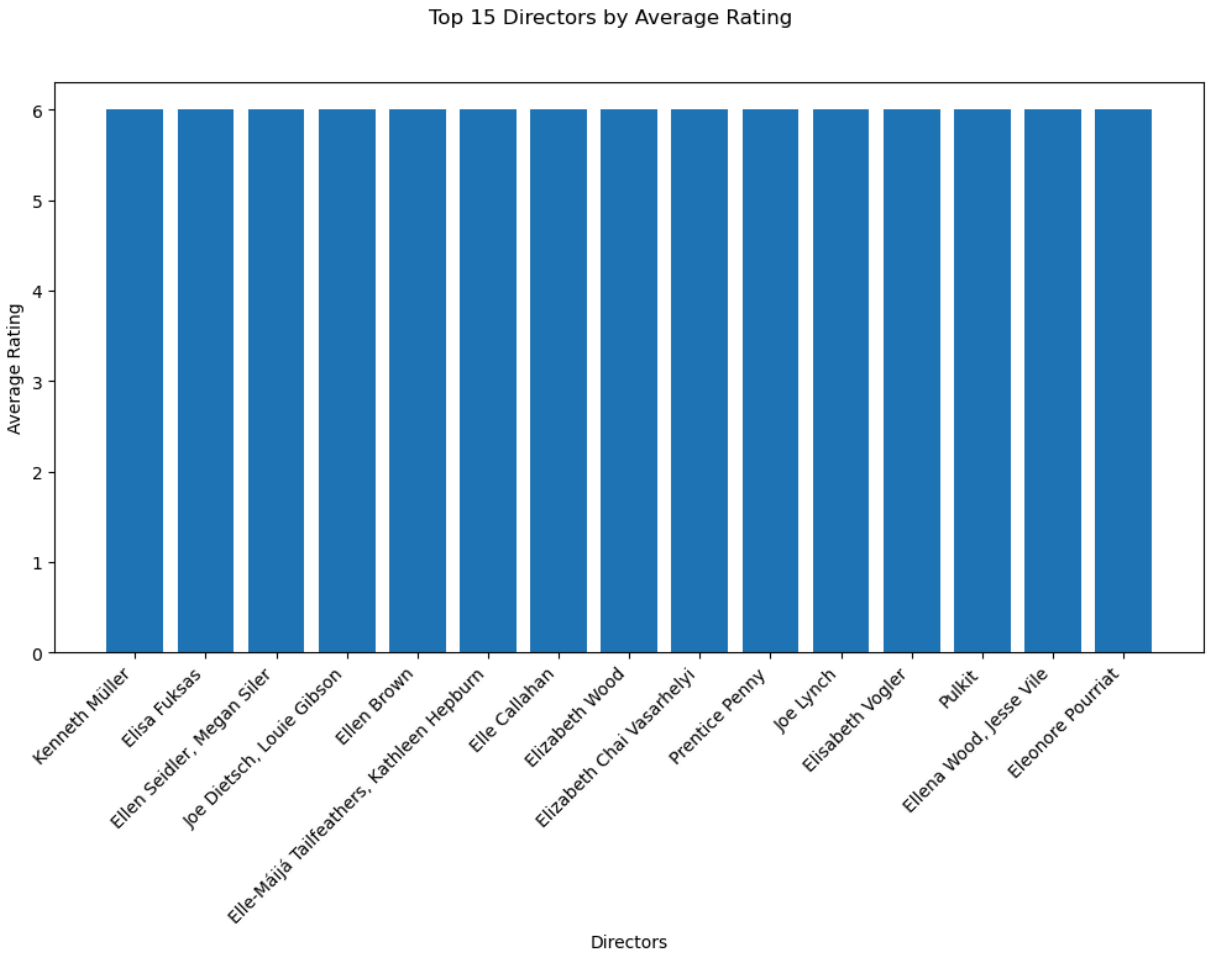
```
# Convert duration to minutes (assuming the format is "X min" or "X Seasons"
def parse_duration(duration):
    if 'min' in duration:
        return int(duration.split()[0])
    elif 'Season' in duration:
        return int(duration.split()[0]) * 45  # Approximate value
    return None
```

```
df['duration_minutes'] = df['duration'].apply(parse_duration)

# Drop rows with NaN duration
df = df.dropna(subset=['duration_minutes'])

# Plot duration vs rating
plt.figure(figsize=(12, 6))
plt.scatter(df['duration_minutes'], df['rating_numeric'], alpha=0.5)
plt.xlabel("Duration (Minutes)")
plt.ylabel("Rating")
plt.title("Correlation Between Duration and Ratings")
plt.grid(True)
plt.show()
```



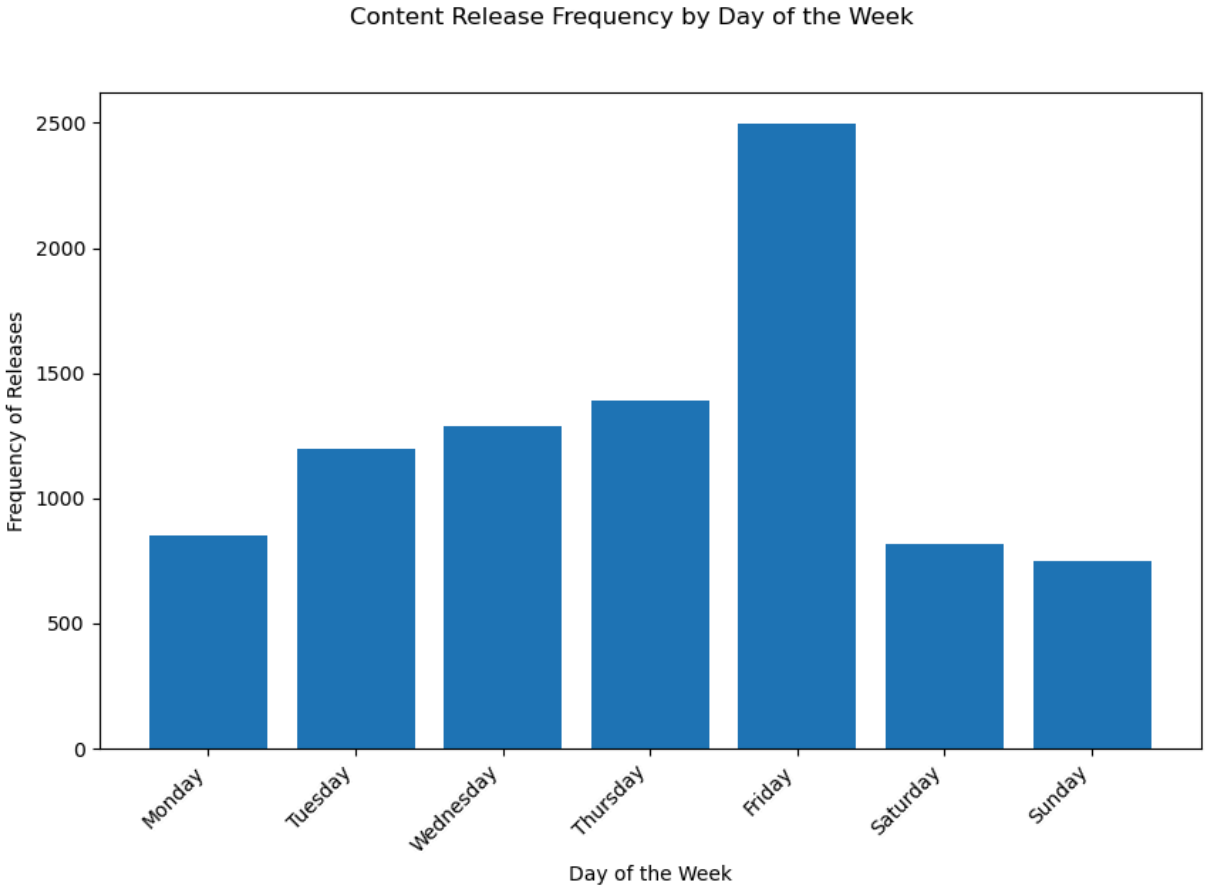
In [114…

```
# Calculate average rating for each director
director_rating = df.groupby('director')['rating_numeric'].mean().reset_inde
director_rating = director_rating.sort_values(by='rating_numeric', ascending

# Plot average rating for top directors
plt.figure(figsize=(12, 6))
plt.bar(director_rating['director'], director_rating['rating_numeric'])
plt.xticks(rotation=45, ha='right')
plt.xlabel("Directors")
plt.ylabel("Average Rating")
plt.suptitle("Top 15 Directors by Average Rating")
plt.show()
```



In [115…

```
# Extract day of the week from 'date_added'
df['day_of_week'] = df['date_added'].dt.day_name()

# Count content by day of the week
day_of_week_counts = df['day_of_week'].value_counts().reindex(['Monday', 'Tu

# Plot content release frequency by day of the week
plt.figure(figsize=(10, 6))
plt.bar(day_of_week_counts.index, day_of_week_counts.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Day of the Week")
plt.ylabel("Frequency of Releases")
plt.suptitle("Content Release Frequency by Day of the Week")
plt.show()
```



In [ ]: