Netflix Data

About Netflix

- Netflix is a global subscription-based streaming service.
- It offers movies, TV shows, documentaries, and Netflix Originals.
- Launched: 1997 as a DVD rental service; transitioned to streaming in 2007.
- Available in 190+ countries, supporting multiple languages.
- Known for personalized recommendations using advanced algorithms.

What is EDA (Exploratory Data Analysis)?

- **Definition**: EDA is the process of analyzing datasets to summarize their main characteristics, often using visualizations and statistical techniques.

Purpose

- To understand the structure of the data.
- To detect patterns, trends, and relationships.
- To identify missing values, outliers, or errors.
- To form hypotheses and guide further analysis.

Steps Involved:

- **Loading the Data:** Importing datasets into the analysis environment.
- **Data Cleaning:** Handling missing values, duplicates, and inconsistencies.
- **Descriptive Statistics:** Summarizing numerical and categorical data.
- **Visualizations:** Creating charts like histograms, scatter plots, and heatmaps.
- **Tools for EDA:**

Python libraries like Pandas, Matplotlib, Seaborn, Plotly. Jupyter Notebook is often used for performing and documenting EDA interactively.

In [4]:
```
# Before start the EDA me must import library which is used in my analysis
```

- **NumPy** is a Python library for efficient numerical computation, offering support for multi-dimensional arrays and mathematical functions.

In [6]:
```
# import numpy
import numpy as np
```

- **Pandas** is a Python library for data manipulation and analysis, providing data structures like DataFrames to handle and analyze structured data efficiently.

In [8]:
```
# import pandas
import pandas as pd
```

- **Seaborn** is a Python visualization library based on Matplotlib that provides a high-level interface for creating informative and attractive statistical graphics.

In [10]:
```
# import seaborn
import seaborn as sns
```

- **Matplotlib** is a Python plotting library used to create static, interactive, and animated visualizations in a variety of formats like line plots, bar charts, histograms, and more.

In [12]:
```
# import matplotlib
```

```
import matplotlib.pyplot as plt
```

In [13]:
```
# import p
import plotly.express as px
```

In [14]:
```
# In some case some warning occures so import warning
import warnings
warnings.filterwarnings("ignore") # It can ignore warning
```

loading the dataset is the first step in any EDA process

In [16]:
```
# we load data in any variable , i am using df.
```

In [17]:
```
df=pd.read_csv("C:/Users/acer/Downloads/gta 5/netflix_titles.csv")
```

In [18]:
```
df
```

Out[18]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descri
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	A father the e hi fil
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	cro path pa Cape
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To p his f fi pov drug
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	F flirta and ta a
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a coa ce kno tra
...	...	...	...	...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	A po carto a rep ar
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	While alon sp to your
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Comedies, Horror Movies	Look surviv world ov
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Children & Family Movies, Comedies	Dra from c f superh
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals	A sc bu boy w his wa

8807 rows × 12 columns

Data Exploration

In [20]:

```
#Use the head() method to view the first few rows

df.head()   ## Default is first 5 rows
```

Out[20]:

	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...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

In [21]:
```
# Use df.tail() to display the last 5 rows of the dataset.

df.tail()  # Default is the last 5 rows
```

Out[21]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descripti
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2007	R	158 min	Cult Movies, Dramas, Thrillers	A politi cartoon a cri repor and a
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7	2 Seasons	Kids' TV, Korean TV Shows, TV Comedies	While livi alone i spoo town young
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R	88 min	Comedies, Horror Movies	Looking survive i world tak over z
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG	88 min	Children & Family Movies, Comedies	Dragg from civili life form superher
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14	111 min	Dramas, International Movies, Music & Musicals	A scrap but p boy wor his way i a t

In [22]:
```
#Use the columns attribute to get a list of column names.
df.columns
```

Out[22]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [23]:
```
# Use the info() method to get an overview of the data types.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

In [24]:
```
# Use the shape attribute to get the number of rows and columns.
df.shape
```

Out[24]: (8807, 12)

Data Cleaning

Check for missing values and handle them appropriately.

In [28]:
```
# check nan values
df.isnull()    # Returns a Boolean DataFrame where True indicates missing values
```

Out[28]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	False	False	False	False	True	False	False	False	False	False	False	False
1	False	False	False	True	False	False	False	False	False	False	False	False
2	False	False	False	False	False	True	False	False	False	False	False	False
3	False	False	False	True	True	True	False	False	False	False	False	False
4	False	False	False	True	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
8802	False	False	False	False	False	False	False	False	False	False	False	False
8803	False	False	False	True	True	True	False	False	False	False	False	False
8804	False	False	False	False	False	False	False	False	False	False	False	False
8805	False	False	False	False	False	False	False	False	False	False	False	False
8806	False	False	False	False	False	False	False	False	False	False	False	False

8807 rows × 12 columns

In [29]:
```
# check nan values in numercial show all in sum by coloumns
df.isnull().sum()
```

Out[29]:
```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

In [30]:
```
# check nan values in  percentage

df.isnull().sum()/len(df)*100
```

Out[30]:
```
show_id          0.000000
type             0.000000
title            0.000000
director        29.908028
cast             9.367549
country          9.435676
date_added       0.113546
release_year     0.000000
rating           0.045418
duration         0.034064
listed_in        0.000000
description      0.000000
dtype: float64
```

In [31]:
```
# There are 6 coloumn in which are of nan values
```

- director ~ **29%**
- cast ~ **9.36%**
- country ~ **9.43**
- date_added,rating,duration all are less than ~ **1**

fillna() function is used to fill missing values (NaN) in a DataFrame with a specified value or method, ensuring complete data for analysis.

In [34]:
```
# fill director by unkonw toh help of fillna
df["director"]=df["director"].fillna("unknow")
```

In [35]:
```
# Fill NaN values in the entire DataFrame with 'Unknown'
df['cast'].fillna('Unknown', inplace=True)
```

In [37]:
```
# there are 9% nan values my approcah is to fill nan value with most occuring values by using of mode
```

In [38]:
```
df["country"]=df["country"].fillna("unknow")
```

In [40]:
```
# drop rows to handle missing values
df=df.dropna(subset=['rating', 'date_added','duration'])
```

In [41]:
```
# We convert date into different columns because to find out trends over time
```

In [117…
```
# Convert 'date_added' to datetime
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Extract temporal features
df['year_added'] = df['date_added'].dt.year
df['month_added'] = df['date_added'].dt.month
df['day_of_week'] = df['date_added'].dt.day_name()
```

In [261…
```
df.drop(columns="date_added",inplace=True)
```

In [45]:
```
# # Convert 'date_added' to datetime and  Extract temporal features it create some nan values
df.isnull().sum()/len(df)*100
```

Out[45]:
```
show_id         0.000000
type            0.000000
title           0.000000
director        0.000000
cast            0.000000
country         0.000000
date_added      1.001138
release_year    0.000000
rating          0.000000
duration        0.000000
listed_in       0.000000
description     0.000000
year_added      1.001138
month_added     1.001138
day_of_week     1.001138
dtype: float64
```

In [46]:
```
# ffill (Forward Fill) method, missing values ko previous available value se fill karta hai
df[["year_added","month_added","day_of_week"]]=df[["year_added","month_added","day_of_week"]].fillna(df[["year_
```

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 15 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   cast          8790 non-null   object        
 5   country       8790 non-null   object        
 6   date_added    8702 non-null   datetime64[ns]
 7   release_year  8790 non-null   int64         
 8   rating        8790 non-null   object        
 9   duration      8790 non-null   object        
 10  listed_in     8790 non-null   object        
 11  description   8790 non-null   object        
 12  year_added    8790 non-null   object        
 13  month_added   8790 non-null   object        
 14  day_of_week   8790 non-null   object        
dtypes: datetime64[ns](1), int64(1), object(13)
memory usage: 1.1+ MB
```

In [49]:
```
df["duration"].fillna(df["duration"].mode()[0],inplace=True)
```

In [50]:
```
df.isnull().sum()/len(df)*100
```

Out[50]:
```
show_id         0.000000
type            0.000000
title           0.000000
director        0.000000
cast            0.000000
country         0.000000
date_added      1.001138
release_year    0.000000
rating          0.000000
duration        0.000000
listed_in       0.000000
description     0.000000
year_added      0.000000
month_added     0.000000
day_of_week     0.000000
dtype: float64
```

In [51]:
```
# now i am changing coloumn name for my better understanding
df.rename(columns={"listed_in":"genres"},inplace=True)
```

In [52]:
```
# drop desdescription becasue its not Help full for my analysis
df.drop(columns="description",inplace=True)
```

In [53]:
```
movie_data = df[df['type'] == 'Movie']
tv_show_data = df[df['type'] == 'TV Show']
```

In [54]:
```
# Movies_data have duration all are in mintuees
# tv_show_data have duration all in season
```

In [55]:
```
tv_show_data["duration"].unique()
```

Out[55]:
```
array(['2 Seasons', '1 Season', '9 Seasons', '4 Seasons', '5 Seasons',
       '3 Seasons', '6 Seasons', '7 Seasons', '10 Seasons', '8 Seasons',
       '17 Seasons', '13 Seasons', '15 Seasons', '12 Seasons',
       '11 Seasons'], dtype=object)
```

In [113…
```
#Movies have durations like '90 min', extract the numeric part
movie_data['duration_minutes'] = movie_data['duration'].str.extract('(\d+)').astype(float)

## TV Shows have durations like '1 Season' or '2 Seasons', extract the numeric part
tv_show_data['duration_seasons'] = tv_show_data['duration'].str.extract('(\d+)').astype(float)
```

In [115…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 15 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   cast          8790 non-null   object
 5   country       8790 non-null   object
 6   date_added    8702 non-null   datetime64[ns]
 7   release_year  8790 non-null   int32
 8   rating        8790 non-null   object
 9   duration      8790 non-null   object
 10  genres        8790 non-null   object
 11  description   8790 non-null   object
 12  year_added    8790 non-null   object
 13  month_added   8790 non-null   object
 14  day_of_week   8790 non-null   object
dtypes: datetime64[ns](1), int32(1), object(13)
memory usage: 1.0+ MB
```

Code Breakdown: of above we can use this on both data movies and tv_shows

movie_data['duration']

This refers to the column in your DataFrame that contains the duration of movies in a string format (e.g., "90 min", "120 min", etc.).

str.extract('(\d+)')

- his method uses a regular expression to extract the numerical digits (the number of minutes) from the string in each row.
- '(\d+)' is the regular expression:
- \d+ means one or more digits (e.g., 90 from "90 min").
- Parentheses () capture the matched value, which in this case is the number of minutes.

astype(float)

- After extracting the number, astype(float) converts the extracted string (the digits) into a float type.
- This makes the values in the duration_minutes column numeric, so they can be used for further calculations or analysis.

In [59]:
```
tv_show_data.isnull().sum()
```

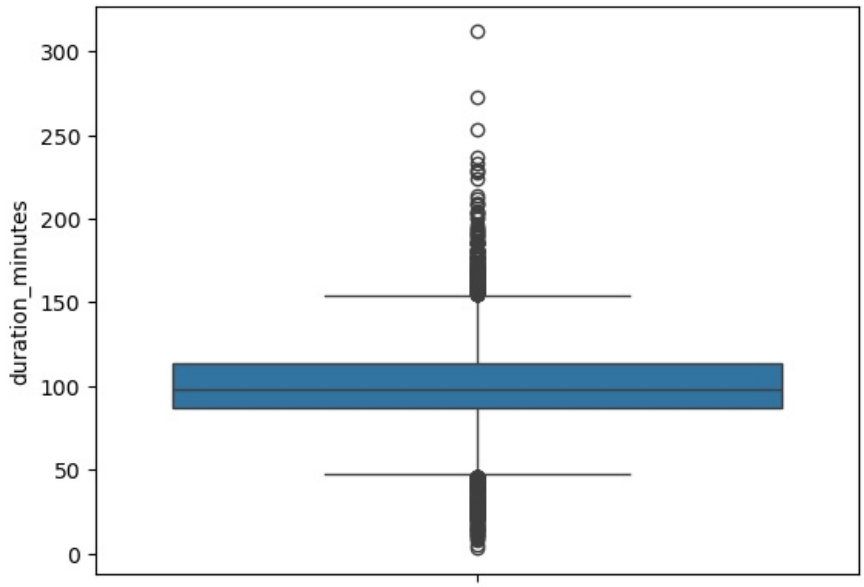
Out[59]:
```
show_id             0
type                0
title               0
director            0
cast                0
country             0
date_added         88
release_year        0
rating              0
duration            0
genres              0
description         0
year_added          0
month_added         0
day_of_week         0
duration_seasons    0
dtype: int64
```

In [60]:
```
# boxplot show the outliers
```

Detecting outliers in duration minutes and shows

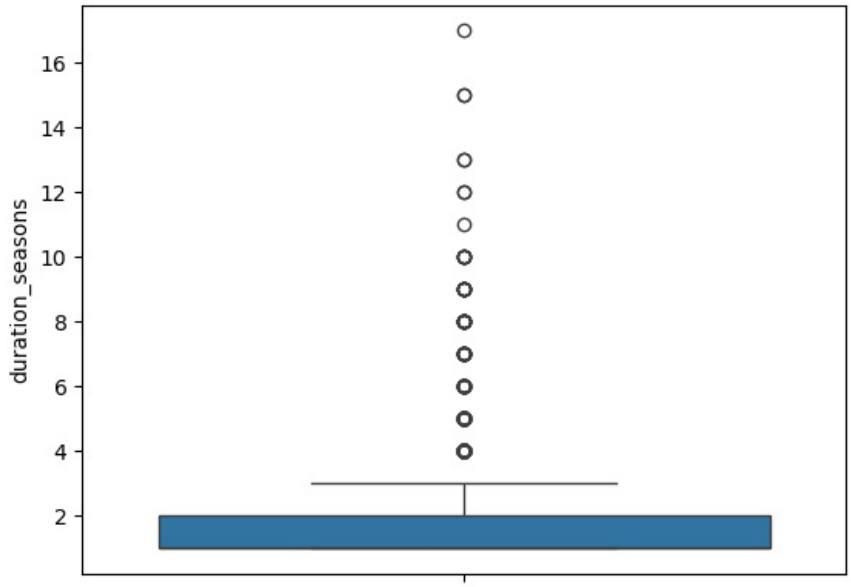
In [62]:
```
sns.boxplot(movie_data["duration_minutes"])
```

Out[62]:
```
<Axes: ylabel='duration_minutes'>
```



In [63]:
```
sns.boxplot(tv_show_data["duration_seasons"])
```

Out[63]: <Axes: ylabel='duration_seasons'>

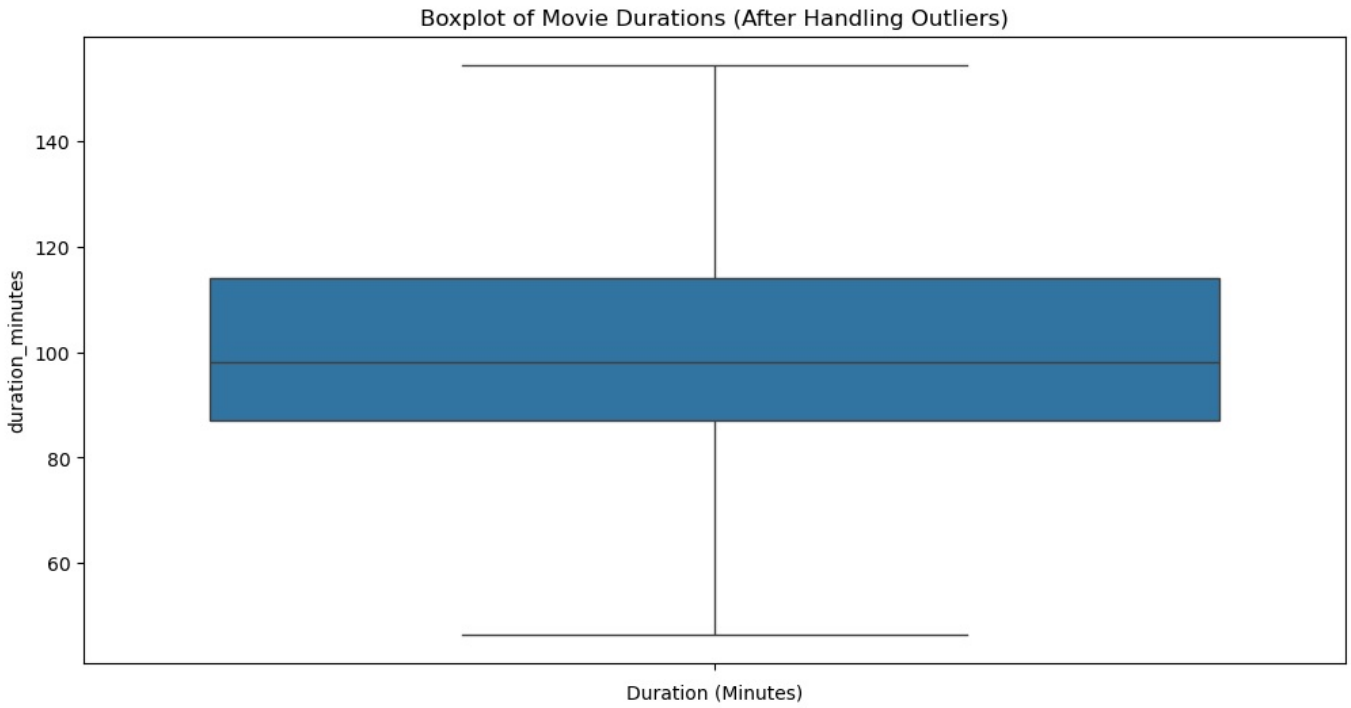


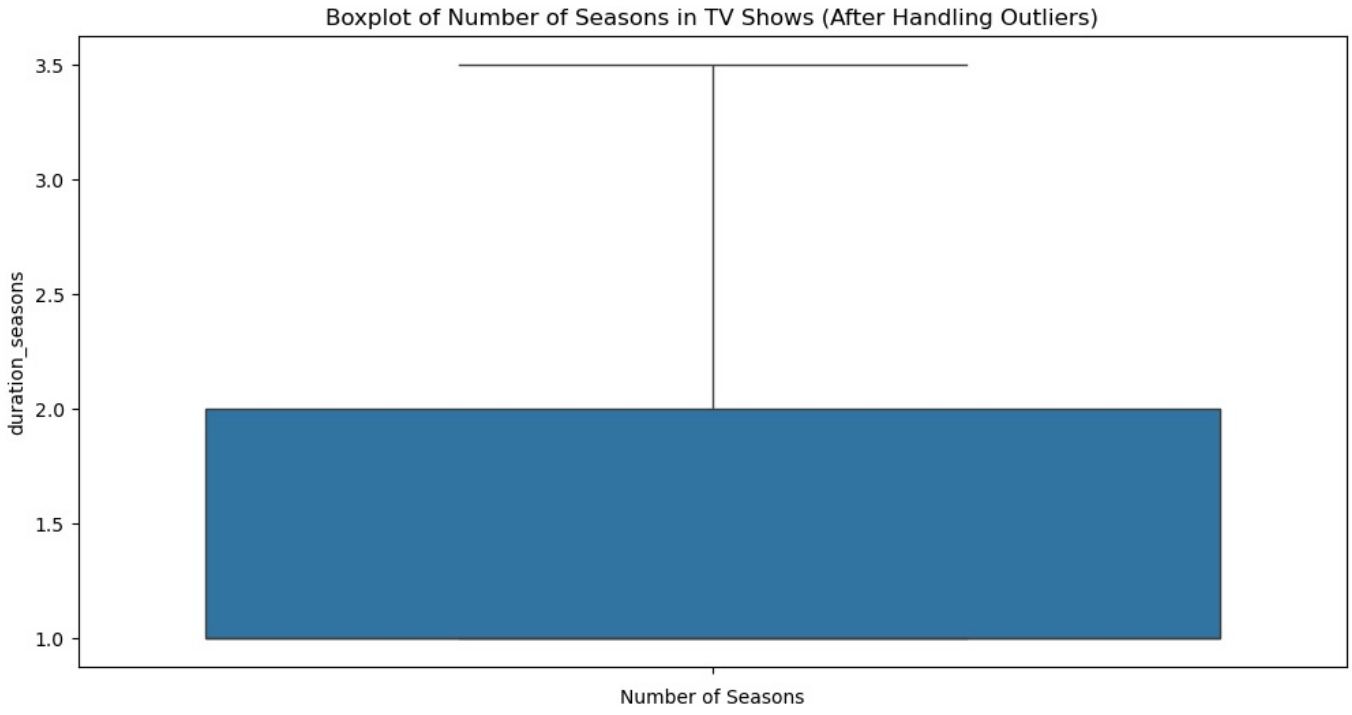
In [64]:
```
# movie_data and # tv_shows_data have outlier so it best to handle outliers otherwise it create problems while
```

In [65]:
```
# Function to detect and handle outliers using IQR
def handle_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)  # First quartile (25th percentile)
    Q3 = data[column].quantile(0.75)  # Third quartile (75th percentile)
    IQR = Q3 - Q1                     # Interquartile Range

    # Define lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

     # Handle outliers by capping them to the lower and upper bounds
    data[column] = np.where(data[column] < lower_bound, lower_bound, data[column])
    data[column] = np.where(data[column] > upper_bound, upper_bound, data[column])

    return data

# Detect and handle outliers in Movie durations
movie_data = handle_outliers_iqr(movie_data, 'duration_minutes')

# Detect and handle outliers in TV Show seasons
tv_show_data = handle_outliers_iqr(tv_show_data, 'duration_seasons')
```

In [66]:
```
# Visualization after handling outliers
plt.figure(figsize=(12, 6))
sns.boxplot(movie_data['duration_minutes'])
```

```
plt.title('Boxplot of Movie Durations (After Handling Outliers)')
plt.xlabel('Duration (Minutes)')
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(tv_show_data['duration_seasons'])
plt.title('Boxplot of Number of Seasons in TV Shows (After Handling Outliers)')
plt.xlabel('Number of Seasons')
plt.show()
```

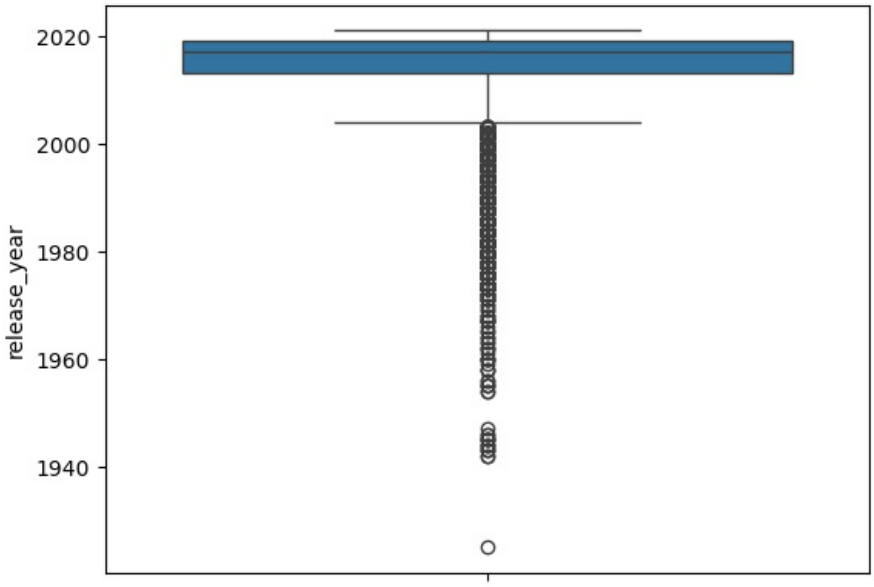




Detecting outliers in release_year

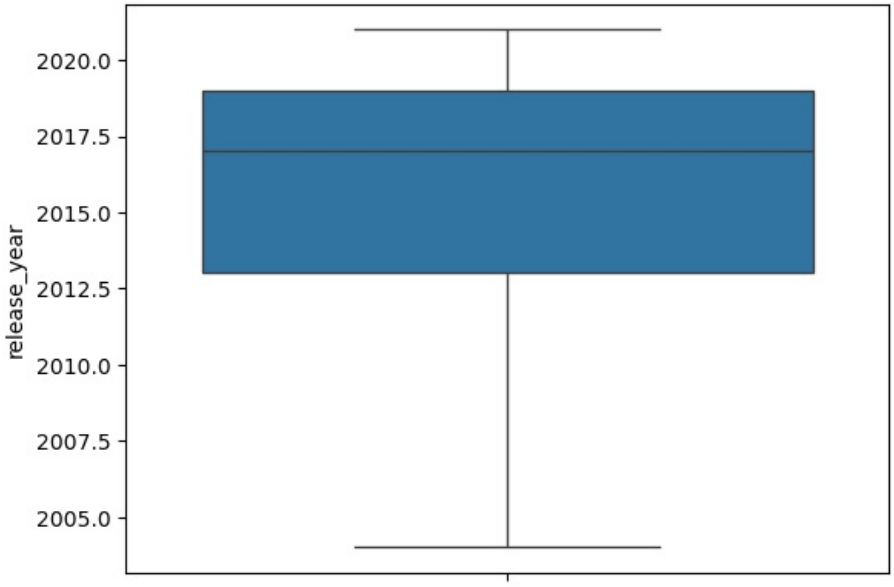
In [68]:
```
sns.boxplot(df["release_year"])
```

Out[68]:
```
<Axes: ylabel='release_year'>
```



In [69]:
```
df=handle_outliers_iqr(df,"release_year")
```

In [70]:
```
sns.boxplot(df["release_year"])
plt.show()
```



In [71]:
```
# Due to oulier it my release year in float converting int
```

In [72]:
```
df["release_year"]=df["release_year"].astype(int)
```

Descriptive Statistics:

Compute basic descriptive statistics such as mean, median, mode, range, and standard deviation for relevant variables.

In [270…
```
df.describe()
```

Out[270…

	release_year	year_added	month_added	listed_in
count	8790.000000	8702.000000	8702.000000	0.0
mean	2015.305802	2018.889681	6.654217	NaN
std	4.951764	1.567252	3.430974	NaN
min	2004.000000	2008.000000	1.000000	NaN
25%	2013.000000	2018.000000	4.000000	NaN
50%	2017.000000	2019.000000	7.000000	NaN
75%	2019.000000	2020.000000	10.000000	NaN
max	2021.000000	2021.000000	12.000000	NaN

In [ ]:
```
### I have completed the data cleaning process, and now I am ready to proceed with the data visualization phase
```

- Create visualizations to represent the distribution of content over different genres.

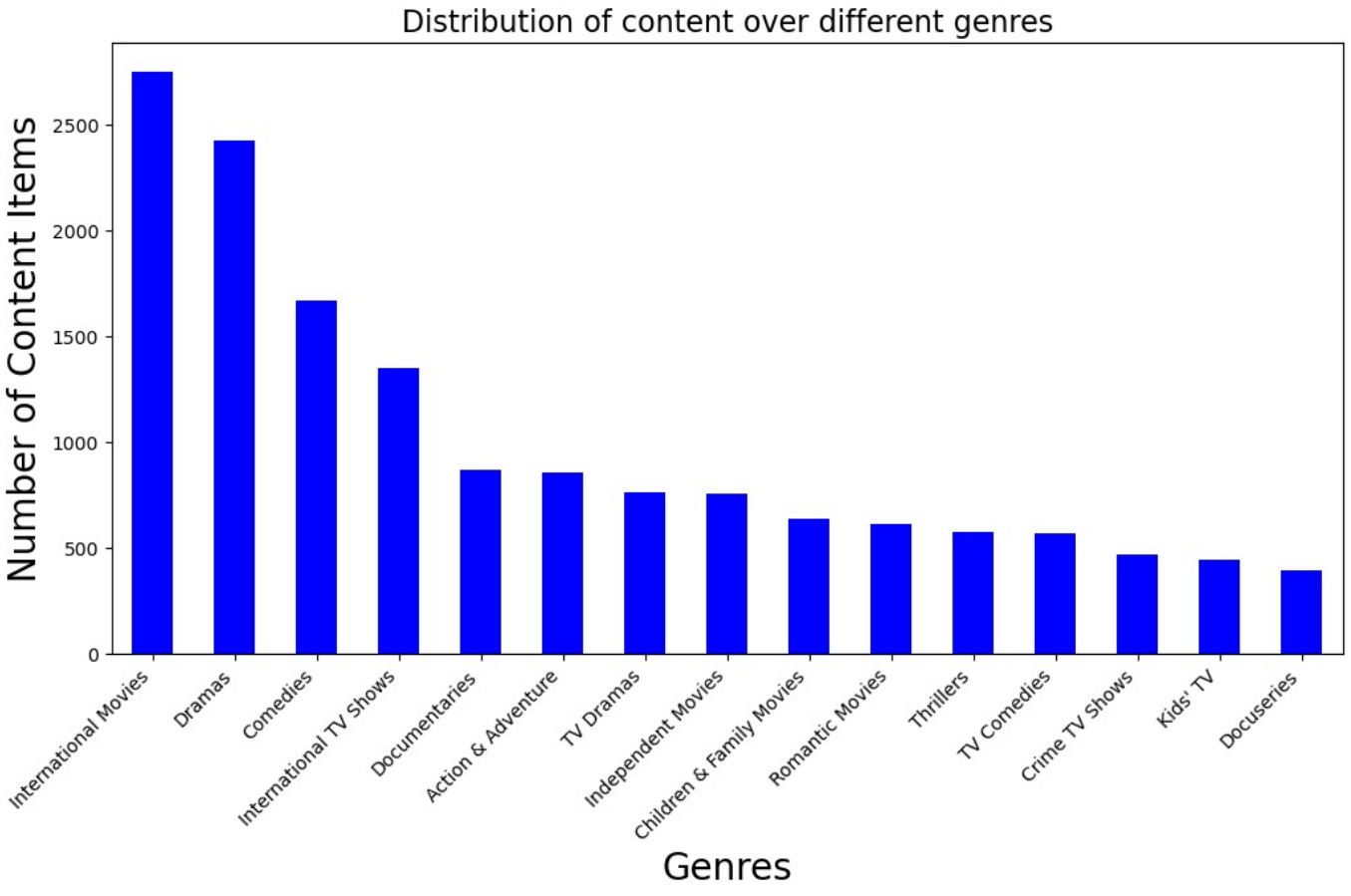
In [77]:
```
 # Split the "genres" column into multiple genres
df["genres"] = df["genres"].str.split(', ')
```

In [78]:
```
# Flatten the list of genres into a single series for counting
all_genres =df["genres"].explode()
```

In [79]:
```
genre_counts=all_genres.value_counts().head(15)
```

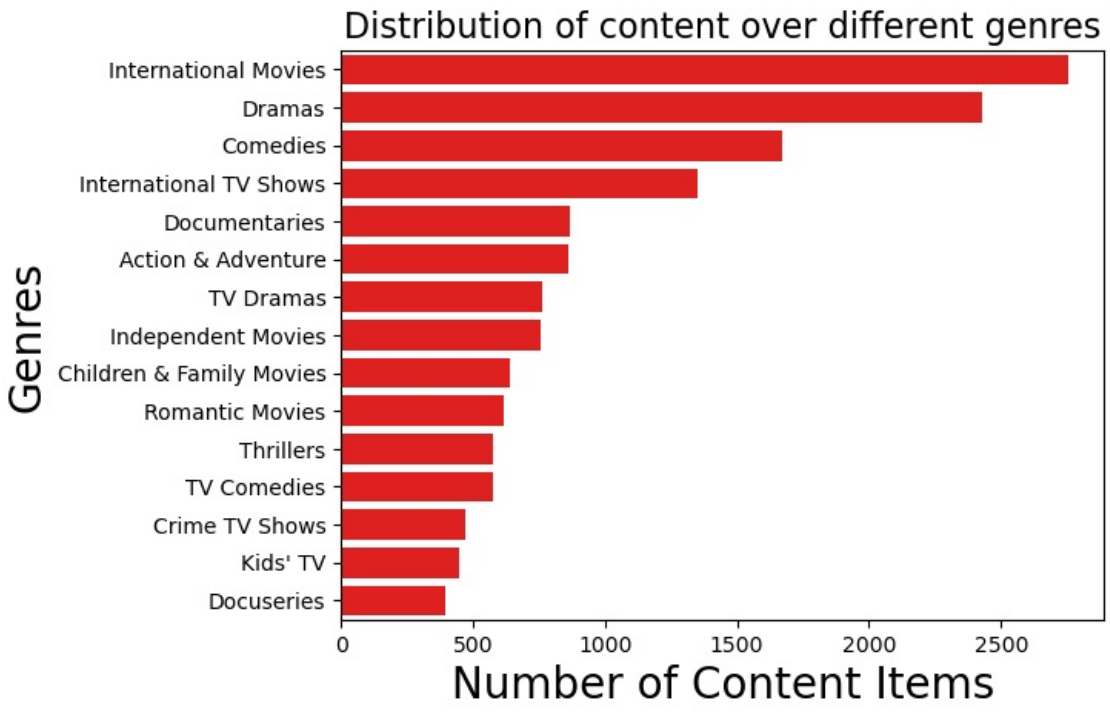
In [80]:
```
# We can check differnent distribution i am  using Matplotlib and Seaborn
```

In [81]:
```
# by using matplotlib
plt.figure(figsize=(12,6))
genre_counts.plot(kind="bar",color="b")
plt.title("Distribution of content over different genres",fontsize=16)
plt.xlabel("Genres",fontsize=20)
plt.ylabel("Number of Content Items",fontsize=20)
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [82]:
```
 # by using seaborn
sns.barplot(x=genre_counts.values,y=genre_counts.index,color='r')
plt.title("Distribution of content over different genres",fontsize=16)
plt.ylabel("Genres",fontsize=20)
plt.xlabel("Number of Content Items",fontsize=20)
```

```
plt.show()
```



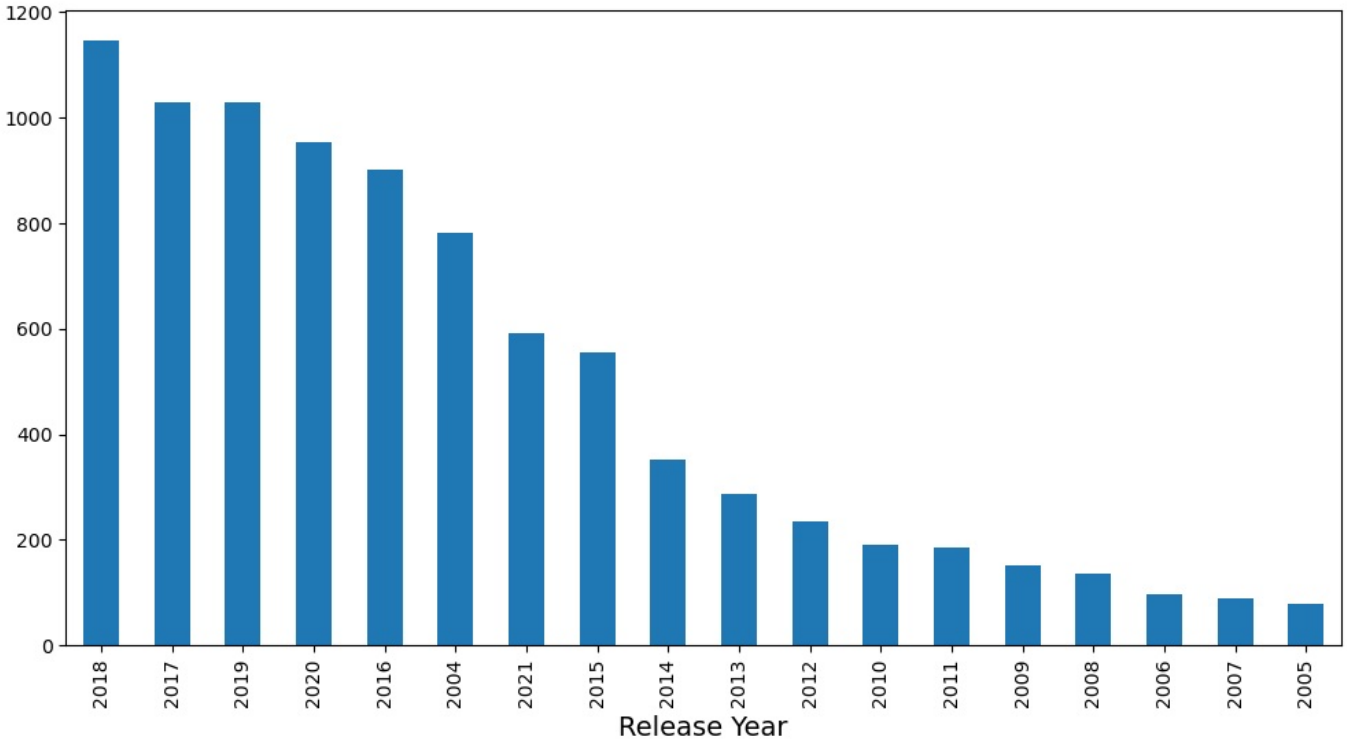
- **This bar graph shows that international Movies has largest content than the other geners in second position dramas and so on**

In [84]:
```
# It's my choice to use any of this four my analysis
```

. Visualize the distribution of content across release years.

In [87]:
```
release_year_count=df["release_year"].value_counts()
```

In [88]:
```
plt.figure(figsize=(12,6))
release_year_count.plot(kind="bar")
plt.xlabel('Release Year', fontsize=14)
plt.show()
```

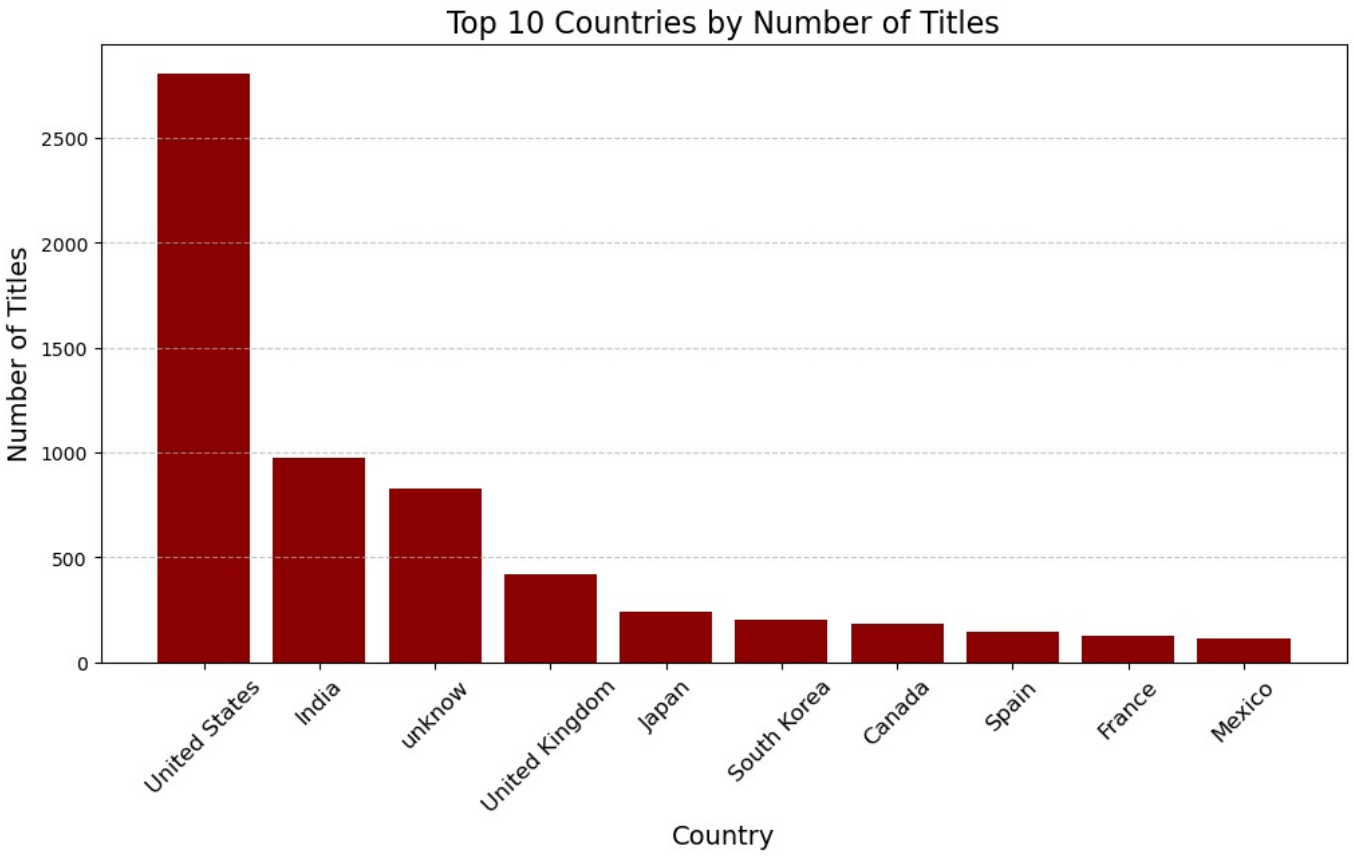


- **This shows that content Across release years are increases upto 2018 and then it start decresing**

- Explore the geographical distribution of content (if applicable).

In [92]:
```
country_counts=df['country'].value_counts().head(10)
```

In [93]:
```
plt.figure(figsize=(12, 6))
plt.bar(country_counts.index, country_counts.values, color='darkred')

# Add title and labels
plt.title('Top 10 Countries by Number of Titles', fontsize=16)
plt.xlabel('Country', fontsize=14)
plt.ylabel('Number of Titles', fontsize=14)
plt.xticks(rotation=45, fontsize=12)  # Rotate x-axis labels
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines for better readability

# Show the plot
plt.show()
```



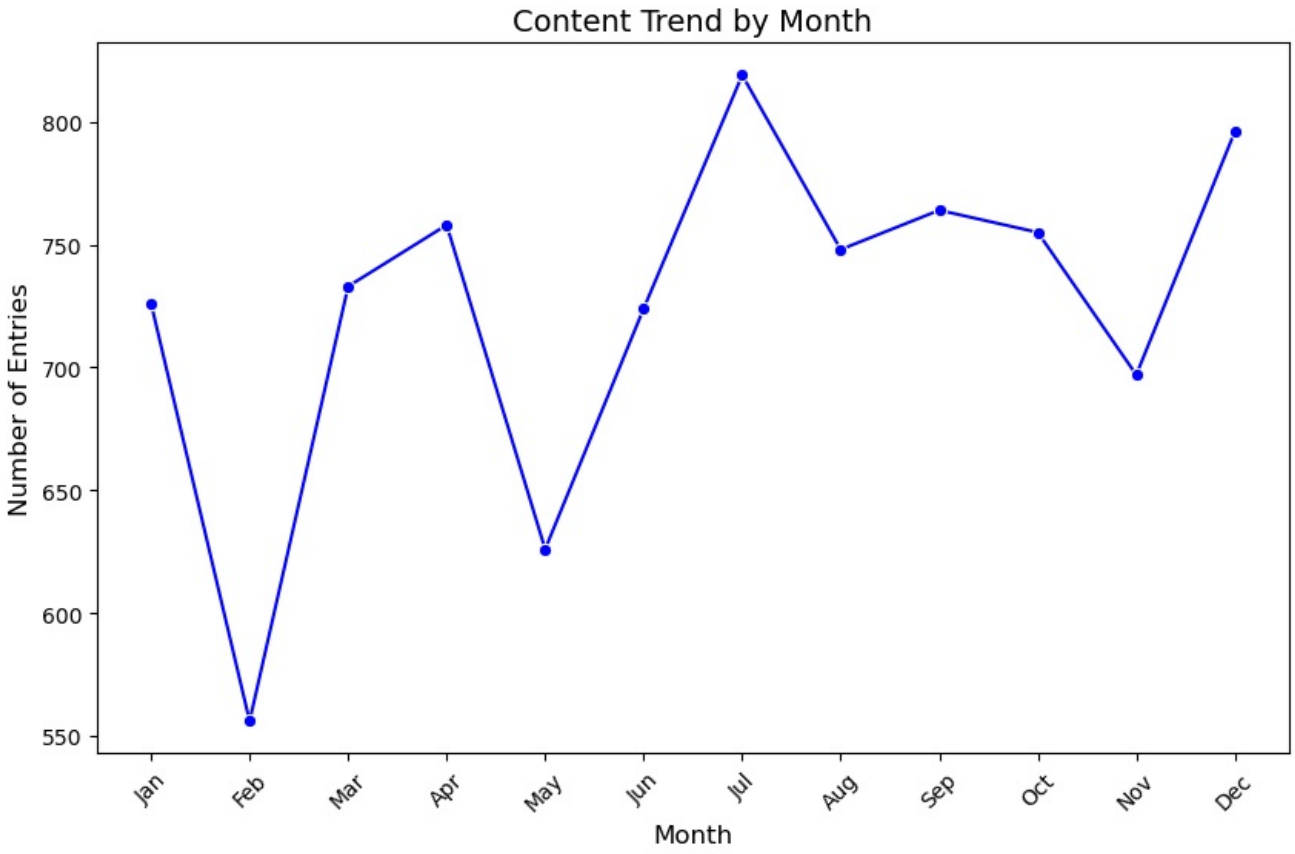
In [245…
```
country_counts_df = country_counts.reset_index()
country_counts_df.columns = ['Country', 'Count']  # Rename columns

# Create a choropleth map
fig = px.choropleth(
    country_counts_df,
    locations='Country',                # Country column
    locationmode='country names',       # Match country names
    color='Count',                      # Color by the number of titles
    title='Geographical Distribution of Netflix Content',
    color_continuous_scale='Viridis'    # Choose color scheme
)
fig.update_layout(
    width=1200,  # Set width
    height=800   # Set height
)


# Display the map
fig.show()
```

- **The United States has the highest amount of content available on Netflix, indicating its dominance in the platform's catalog**

Time Series Analysis:

In [119…
```
month_count = df['month_added'].value_counts().sort_index()
```

In [121…
```
# Create the plot
plt.figure(figsize=(10, 6))
sns.lineplot(x=month_count.index, y=month_count.values, marker='o', color='b')

# Add labels and title
plt.title('Content Trend by Month', fontsize=14)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Number of Entries', fontsize=12)
plt.xticks(ticks=range(1,13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'No

# Show the plot
plt.show()
```



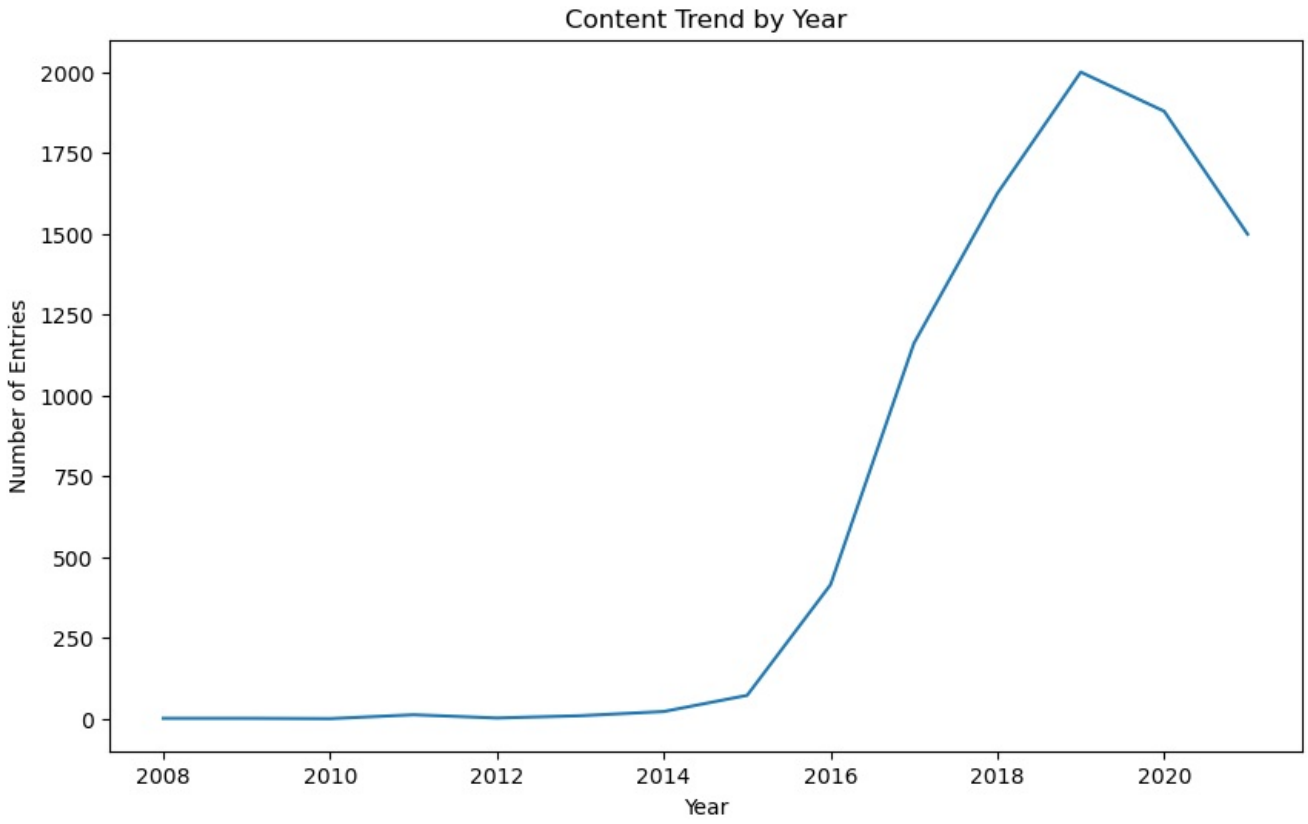
In [ ]:
```
# we make line trend with Plotly
```

In [123…
```
# Create the plot
fig = px.line(
    x=month_count.index,
    y=month_count.values,
    labels={'x': 'Month', 'y': 'Number of Entries'},
    title='Content Trend by Month'
)

# Customize the month labels to show month names instead of numbers
fig.update_xaxes(
    tickmode='array',
    tickvals=list(range(1, 13)),
    ticktext=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
)

# Show the plot
fig.show()
```

In [ ]:
```
# distribution by year
```

In [125…
```
yearly_counts = df['year_added'].value_counts().sort_index()
```

In [127…
```
# Create a line plot using Seaborn
plt.figure(figsize=(10,6))
sns.lineplot(x=yearly_counts.index, y=yearly_counts.values)

# Label the axes and set the title
plt.xlabel('Year')
plt.ylabel('Number of Entries')
plt.title('Content Trend by Year')

# Display the plot
plt.show()
```



In [129…
```
fig = px.line(
    x=yearly_counts.index,
```

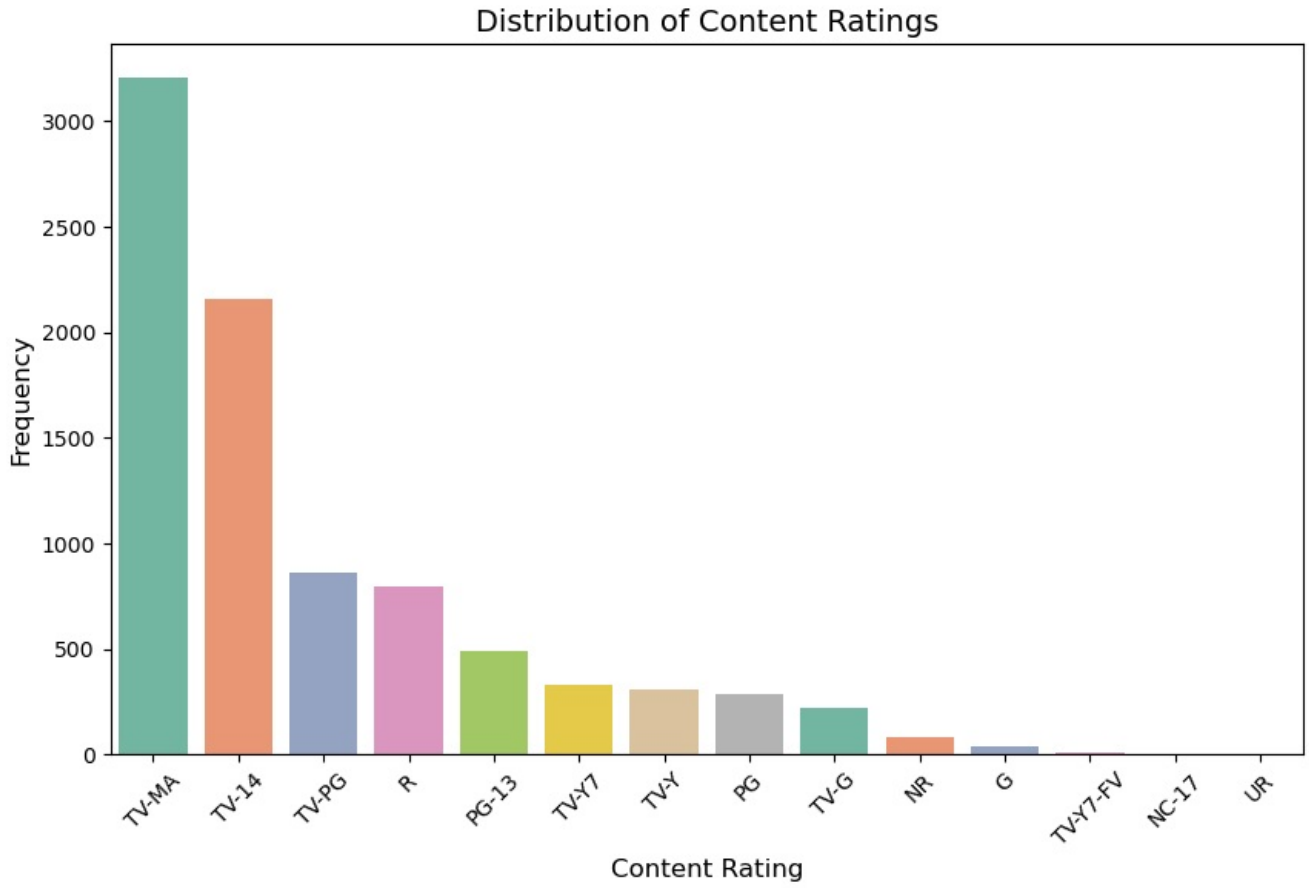
```
    y=yearly_counts.values,
    labels={'x': 'Year', 'y': 'Number of Entries'},
    title='Content Trend by Year'
)

# Show the plot
fig.show()
```

- **July shows a peak in content availability.**
- **February has the lowest number of content releases, indicating less content during that month.**

• Analyze the distribution of content ratings.

In [134…
```
rating_count = df["rating"].value_counts()
```

In [136…
```
# Create a bar plot for the value counts of ratings
plt.figure(figsize=(10, 6))
sns.barplot(x=rating_count.index, y=rating_count.values,palette='Set2')

# Add labels and title
plt.title('Distribution of Content Ratings', fontsize=14)
plt.xlabel('Content Rating', fontsize=12)
plt.ylabel('Frequency', fontsize=12)

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

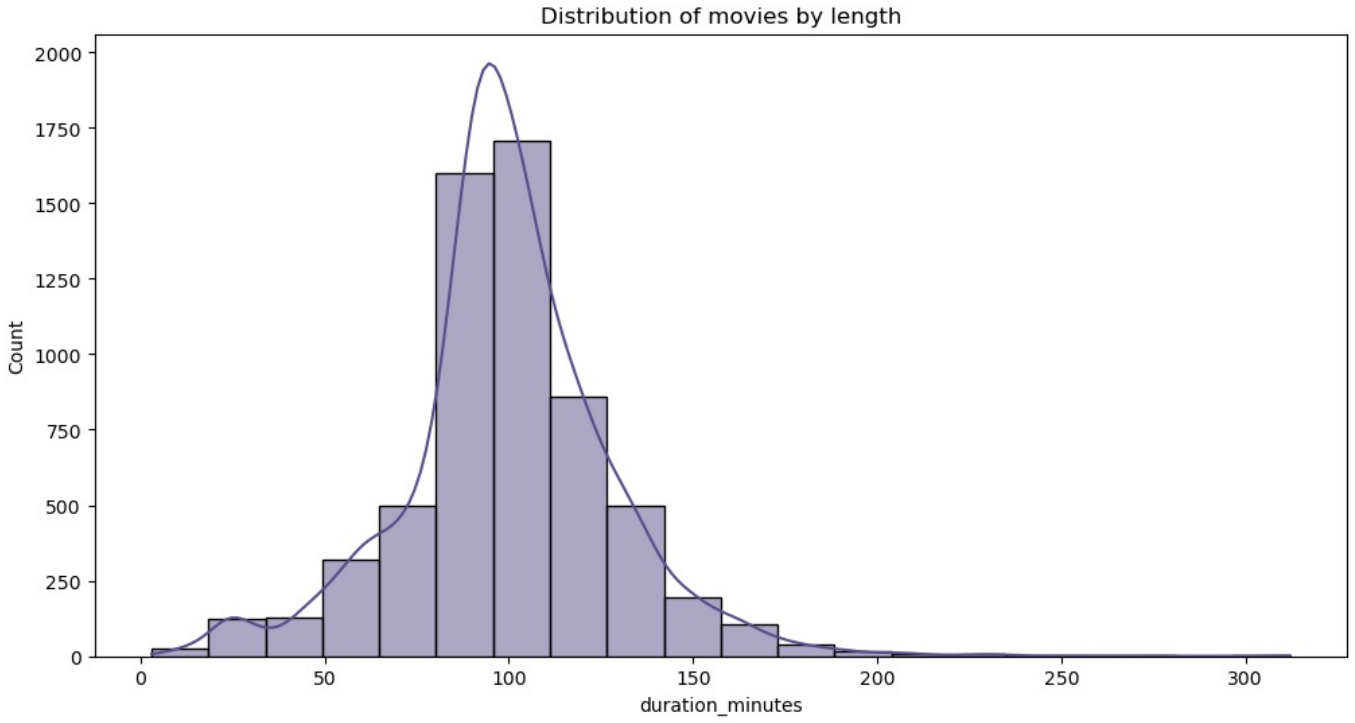


- **It shows that TV-MA has highest content rating**

Explore the length of movies or episodes and identify any trends.

In [141…
```
# movies length
```

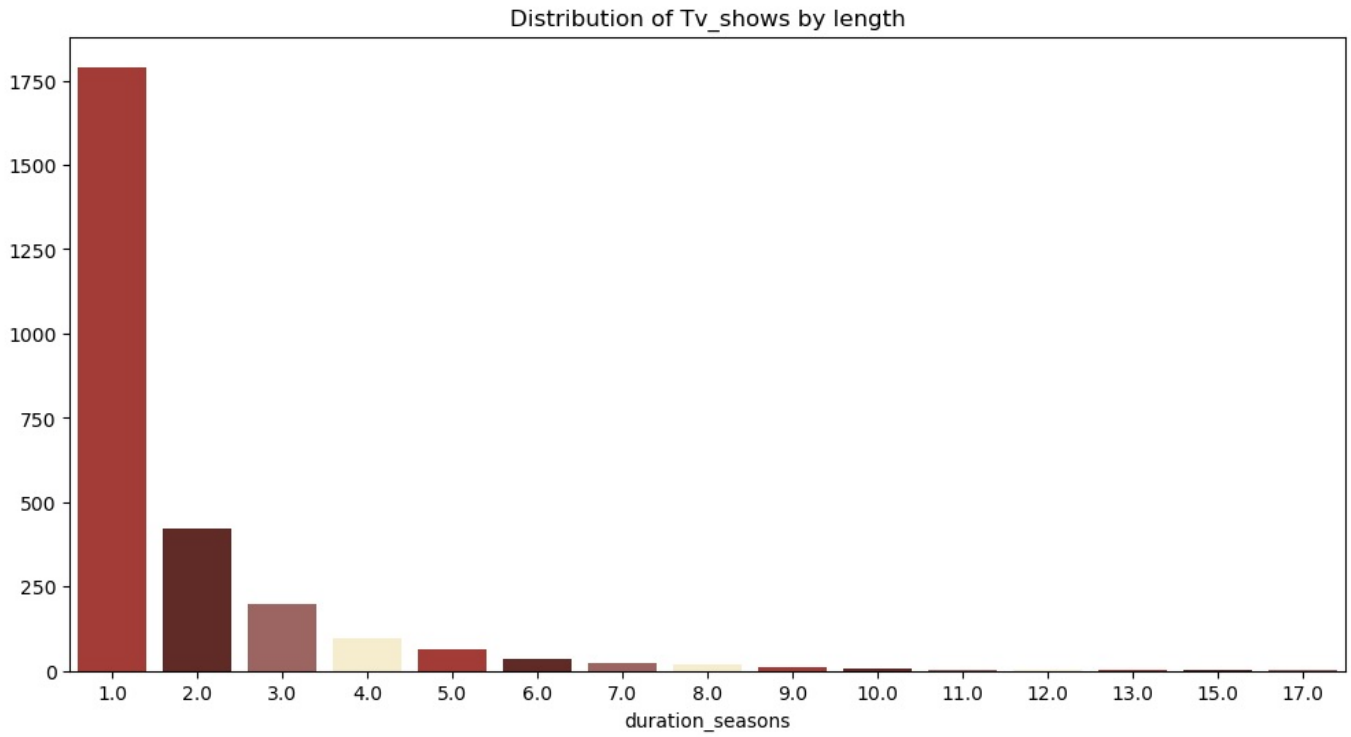
In [143…
```
plt.figure(figsize=(12,6))
sns.histplot(movie_data["duration_minutes"],bins=20,kde=True,color="#58508d")
plt.title("Distribution of movies by length")
plt.show()
```



In [145…
```
count_tv=tv_show_data["duration_seasons"].value_counts()
#"plt.xticks(ticks=range(1,13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', '
```

In [147…
```
plt.figure(figsize=(12,6))
```

```
sns.barplot(x=count_tv.index,y=count_tv.values,palette=["#b52b24","#6a211d","#a65c58","#fcf1c7"])
plt.title("Distribution of Tv_shows by length")
plt.show()
```



- **There are larger duration of movies but maxiumum movies are in 90 mins long and tv shows are maximum of 1 season**

Top Lists and Recommendations:

In [151…
```
# Identify and present top-rated movies or TV shows based on user ratings.
```

In [153…
```
rating_type_counts = df.groupby(['rating', 'type']).size().unstack(fill_value=0)
```

Breakdown:

- **df.groupby(['rating', 'type'])**
- Groups the dataset by the columns rating and type.
- Example: Groups all Movies and TV Shows by their respective ratings (like PG, R, TV-MA).
- **size()**
- Counts the number of occurrences (rows) in each group.
- Example: If TV-MA Movies have 50 entries and TV-MA TV Shows have 30 entries, this will count both.
- **unstack(fill_value=0)**
- Converts the grouped data into a pivot table.
- The rating values become rows.
- The type values (Movies and TV Shows) become columns.
- Any missing value (e.g., if a rating doesn't have a Movie or TV Show) is replaced with 0 using fill_value=0.

In [156…
```
movie_tv_counts = rating_type_counts.sum(axis=1).sort_values(ascending=False).head(10)
```

Breakdown

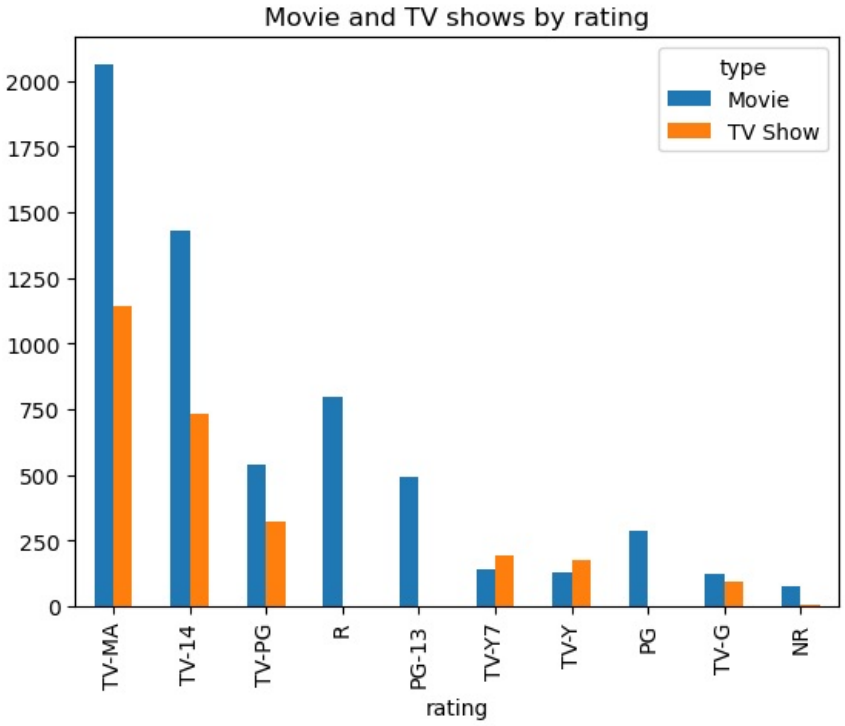
- **rating_type_counts.sum(axis=1):**
- Calculates the total count of Movies + TV Shows for each rating.

- axis=1 means the sum is done row-wise (across the Movie and TV Show columns).
- **.sort_values(ascending=False):**
- Sorts the ratings by their total count in descending order, showing the most popular ratings first.
- **.head(10):**
- Selects the top 10 ratings based on total counts.

In [159…
```
top_ratings = movie_tv_counts.index
filtered_rating_type_counts = rating_type_counts.loc[top_ratings]
```

In [161…
```
plt.figure(figsize=(12, 15))
filtered_rating_type_counts.plot(kind="bar")
plt.title("Movie and TV shows by rating")
plt.show()
```

```
<Figure size 1200x1500 with 0 Axes>
```



- **it shows that TV-MA rating are highest and movies are more rated then TV shows. It shows big difference by users rating and also shows that users prefer movies over TV shows.**

Genre Trends

Analyze trends in the popularity of different genres over time.

In [166…
```
trend=df[["genres","year_added"]]
```

In [168…
```
# Explode the genre list into separate rows
df_exploded = df.explode('genres')
```

In [170…
```
#Group by 'Year' and 'Genre', then count the occurrences
genre_count_by_year = df_exploded.groupby(['year_added', 'genres']).size().unstack(fill_value=0)
```

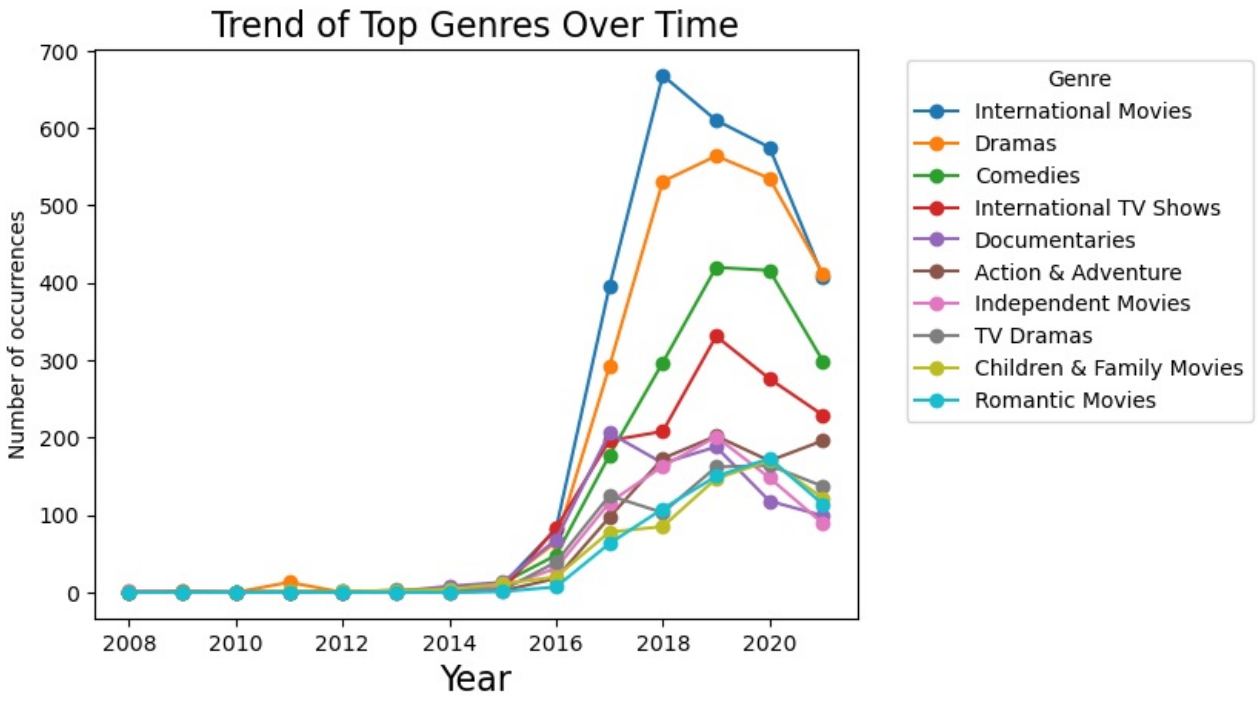
In [172…
```
#dentify top genres (e.g., top 5 genres based on total occurrences)
top_genres = genre_count_by_year.sum(axis=0).nlargest(10).index
```

In [174…
```
#Filter the dataset to only include these top genres
top_genre_trends = genre_count_by_year[top_genres]
```

In [176…
```
plt.figure(figsize=(16,100))
top_genre_trends.plot(kind='line', marker='o')
plt.title('Trend of Top Genres Over Time',fontsize=16)
plt.xlabel('Year',fontsize=16)
plt.ylabel('Number of occurrences')
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left')
#plt.tight_layout()  # Adjust layout to prevent clipping
```

```
plt.show()
```

```
<Figure size 1600x10000 with 0 Axes>
```

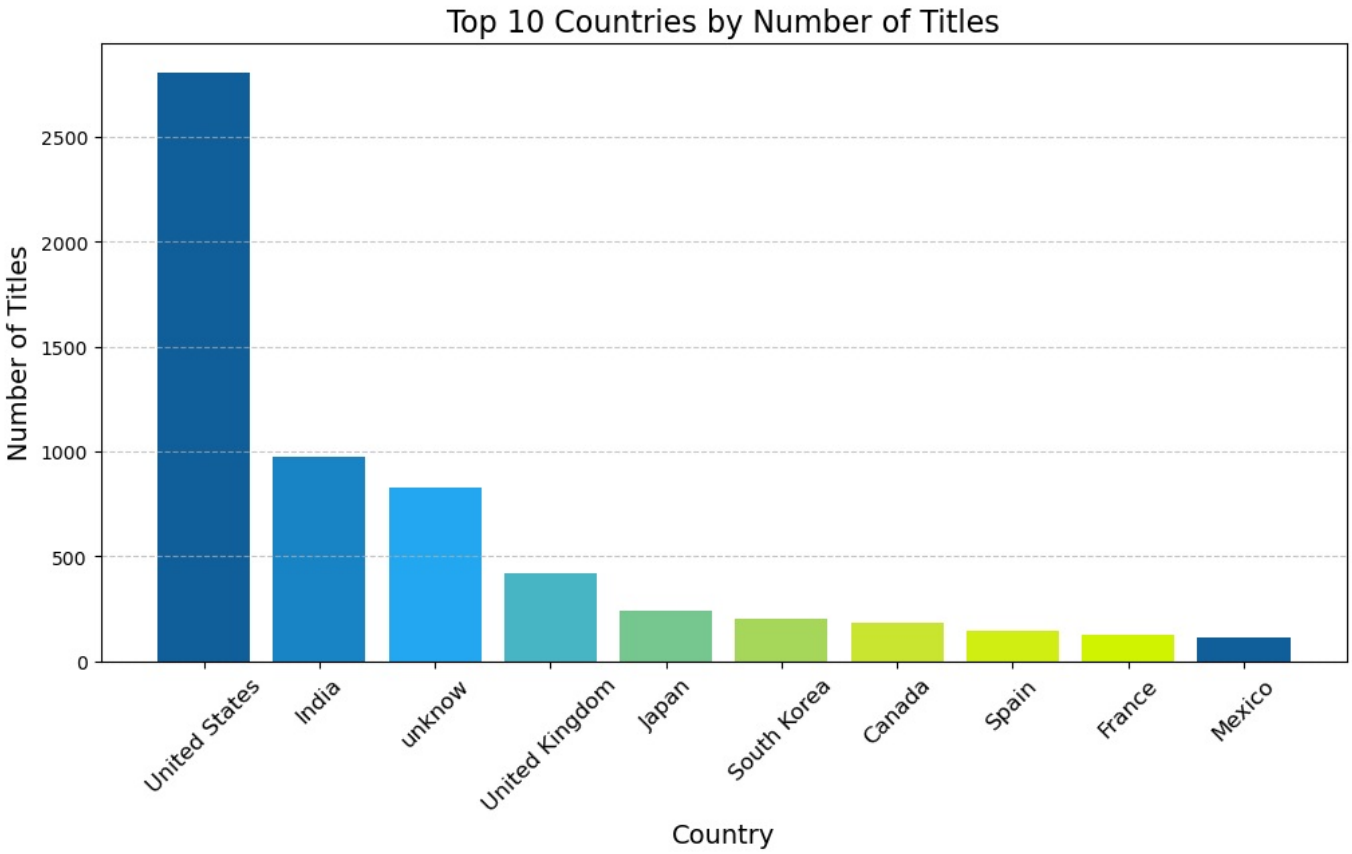


- **It shows that International Movies genre is most popular over time.**

Geographical Analysis:

In [180…
```
#Further explore the distribution of content across different countries and regions.
```

In [182…
```
# Count the number of entries per country
country_counts=df['country'].value_counts().head(10)

plt.figure(figsize=(12, 6))
plt.bar(country_counts.index, country_counts.values, color=["#115f9a", "#1984c5", "#22a7f0", "#48b5c4", "#76c68

# Add title and labels
plt.title('Top 10 Countries by Number of Titles', fontsize=16)
plt.xlabel('Country', fontsize=14)
plt.ylabel('Number of Titles', fontsize=14)
plt.xticks(rotation=45, fontsize=12)  # Rotate x-axis labels
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines for better readability

# Show the plot
plt.show()
```



- **It shows that united states has biggest content than other countries**

. Correlation Analysis:

In [185…
```
# Investigate potential correlations between variables (e.g., ratings and duration).
```

In [187…
```
df["rating"].unique()
```

Out[187…
```
array(['PG-13', 'TV-MA', 'PG', 'TV-14', 'TV-PG', 'TV-Y', 'TV-Y7', 'R',
       'TV-G', 'G', 'NC-17', 'NR', 'TV-Y7-FV', 'UR'], dtype=object)
```

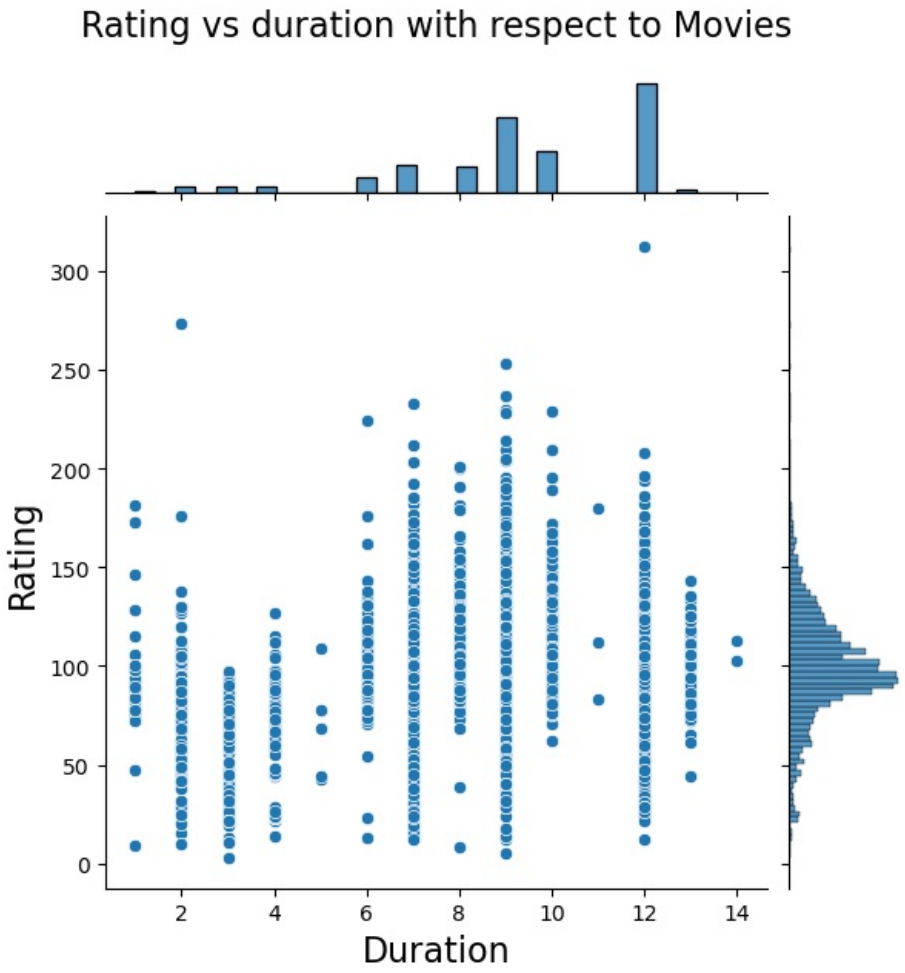
In [189…
```
rating_mapping = {
    'G': 1,
    'TV-G': 2,
    'TV-Y': 3,
    'TV-Y7': 4,
    'TV-Y7-FV': 5,
    'PG': 6,
    'TV-PG': 7,
    'PG-13': 8,
    'TV-14': 9,
    'R': 10,
    'NC-17': 11,
    'TV-MA': 12,
    'NR': 13,
    'UR': 14
}
```

In [191…
```
movie_data["rating_numeric"]= movie_data["rating"].map(rating_mapping)
```

In [193…
```
movie_data['duration'] = movie_data['duration'].str.replace(' min', '').astype(int)
```

In [195…
```
plt.figure(figure=(15,18))
sns.jointplot(data=movie_data,x="rating_numeric",y="duration",palette="Set2")
plt.title("Rating vs duration with respect to Movies",fontsize=16,y=1.25)
plt.xlabel("Duration",fontsize=16)
plt.ylabel("Rating",fontsize=16)
plt.show()
```

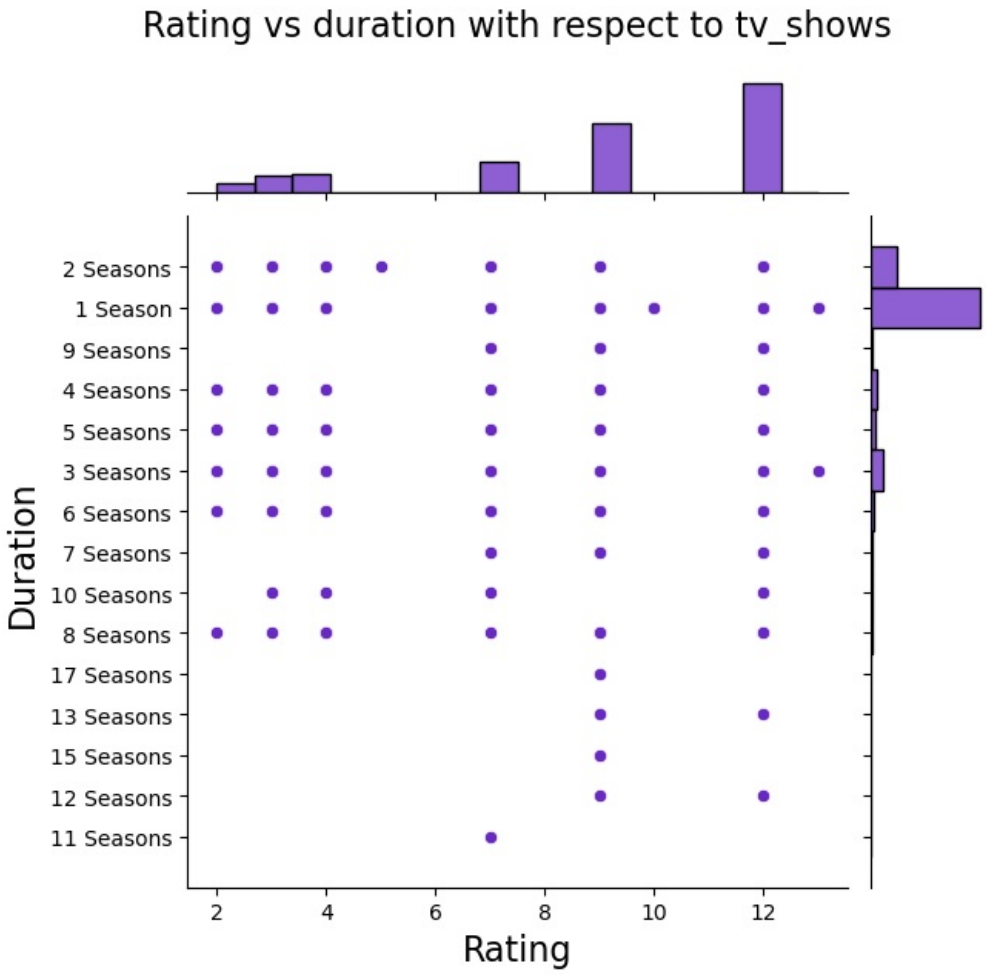
```
<Figure size 640x480 with 0 Axes>
```



In [197…
```
tv_show_data["rating_numeric"]= tv_show_data["rating"].map(rating_mapping)
```

In [201…
```
plt.figure(figure=(15,18))
sns.jointplot(data=tv_show_data,x="rating_numeric",y="duration",color="#6929c4")
plt.title("Rating vs duration with respect to tv_shows",fontsize=16,y=1.25)
plt.xlabel("Rating",fontsize=16)
plt.ylabel("Duration",fontsize=16)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



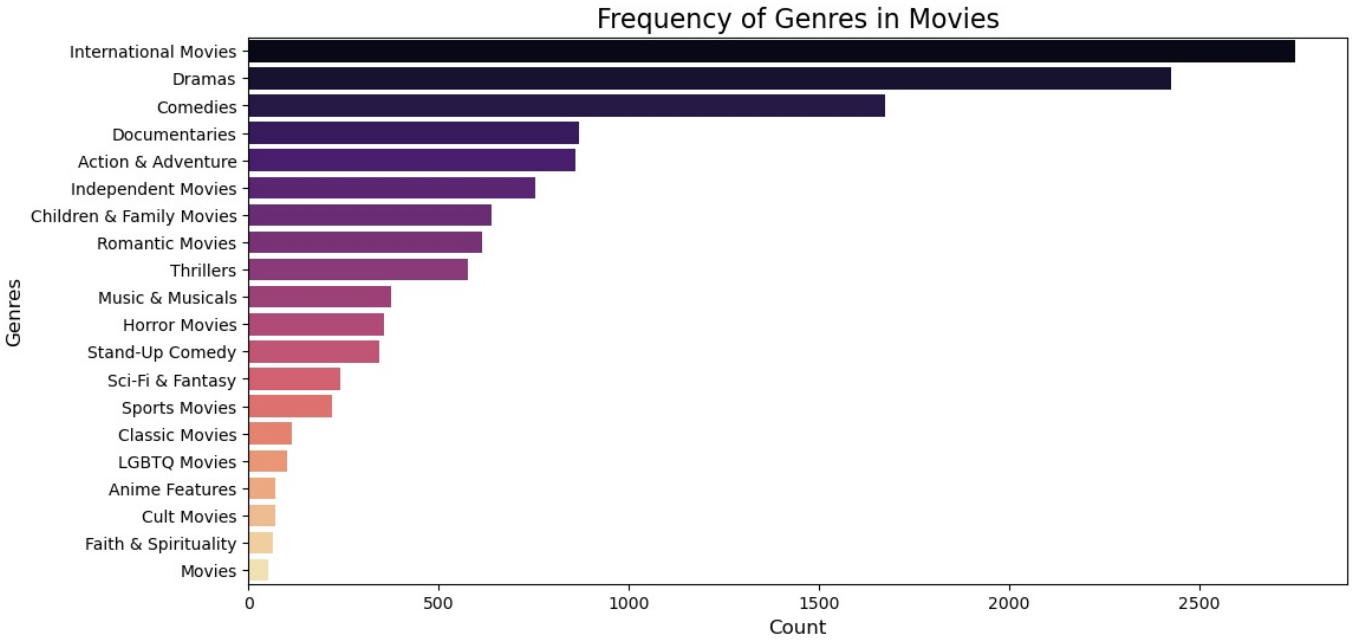
Content Variety:

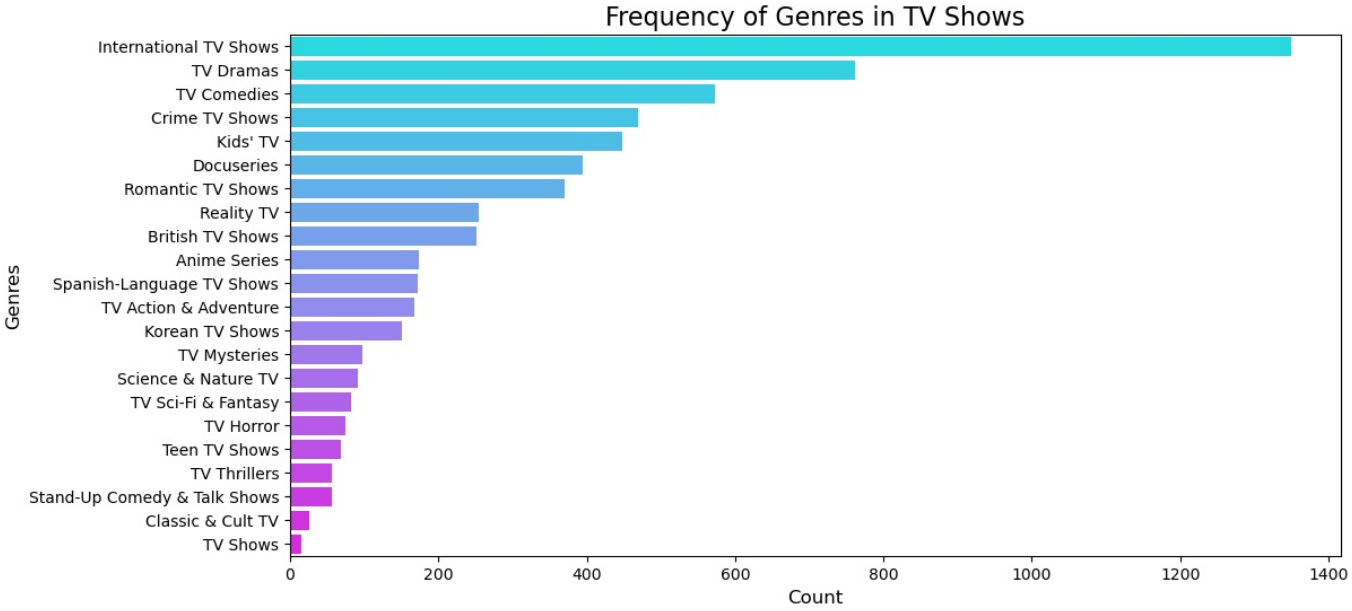
- Evaluate the diversity of content by analyzing the number of unique genres and categories.

In [205…
```
# we have to check all diversity of content with respect to movies and tv shows differenet
```

In [207…
```
if 'type' in df.columns:
    # Separate data for Movies and TV Shows
    movies = df[df['type'] == 'Movie']
    tv_shows = df[df['type'] == 'TV Show']

    # Count genres for Movies
```

```
movie_genres = movie_data['genres'].str.split(', ').explode().value_counts()

# Count genres for TV Shows
tv_genres = tv_show_data['genres'].str.split(', ').explode().value_counts()

# Plot for Movies
plt.figure(figsize=(12, 6))
sns.barplot(x=movie_genres.values, y=movie_genres.index, palette="magma")
plt.title("Frequency of Genres in Movies", fontsize=16)
plt.xlabel("Count", fontsize=12)
plt.ylabel("Genres", fontsize=12)
plt.show()

# Plot for TV Shows
plt.figure(figsize=(12, 6))
sns.barplot(x=tv_genres.values, y=tv_genres.index, palette="cool")
plt.title("Frequency of Genres in TV Shows", fontsize=16)
plt.xlabel("Count", fontsize=12)
plt.ylabel("Genres", fontsize=12)
plt.show()
```





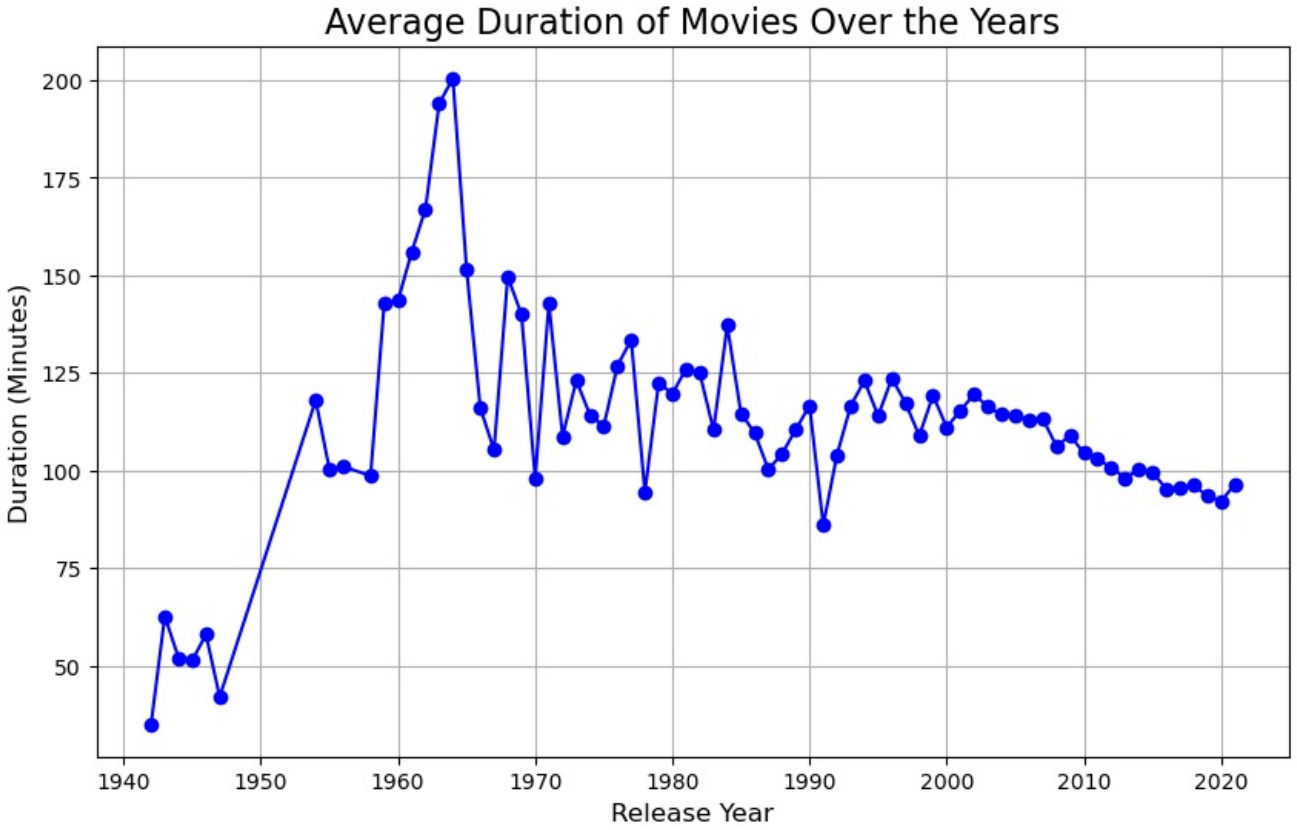
- **International movies make up a significant portion of the movie category, while international TV shows dominate the TV show category, highlighting Netflix's global content strategy.**

- Explore how the characteristics of content (e.g., duration, ratings) have evolved over the years.

In [210…
```
# --- Duration Trends Over Time ---
```

In [215…
```
#Average duration of Movies over the years
movie_duration_trends = movie_data.groupby('release_year')['duration'].mean()
```

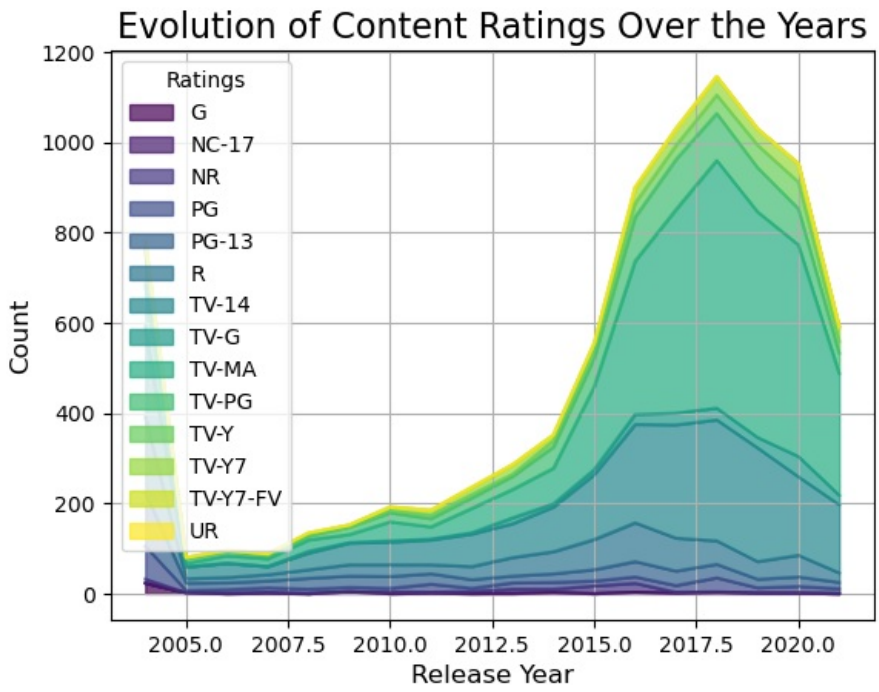
In [217…
```
plt.figure(figsize=(10, 6))
movie_duration_trends.plot(kind='line', color='blue', marker='o')
plt.title("Average Duration of Movies Over the Years", fontsize=16)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Duration (Minutes)", fontsize=12)
plt.grid()
plt.show()
```



- **The peak in movie duration between 1960 and 1966 indicates a trend towards longer films, reflecting a period of artistic and narrative experimentation in cinema.**

In [219…
```
# --- Rating Trends Over Time ---
# Count of Ratings Over the Years
rating_trends = df.groupby(['release_year', 'rating']).size().unstack(fill_value=0)

plt.figure(figsize=(12, 8))
rating_trends.plot(kind='area', stacked=True, alpha=0.7, colormap='viridis')
plt.title("Evolution of Content Ratings Over the Years", fontsize=16)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.legend(title="Ratings")
plt.grid()
plt.show()
```

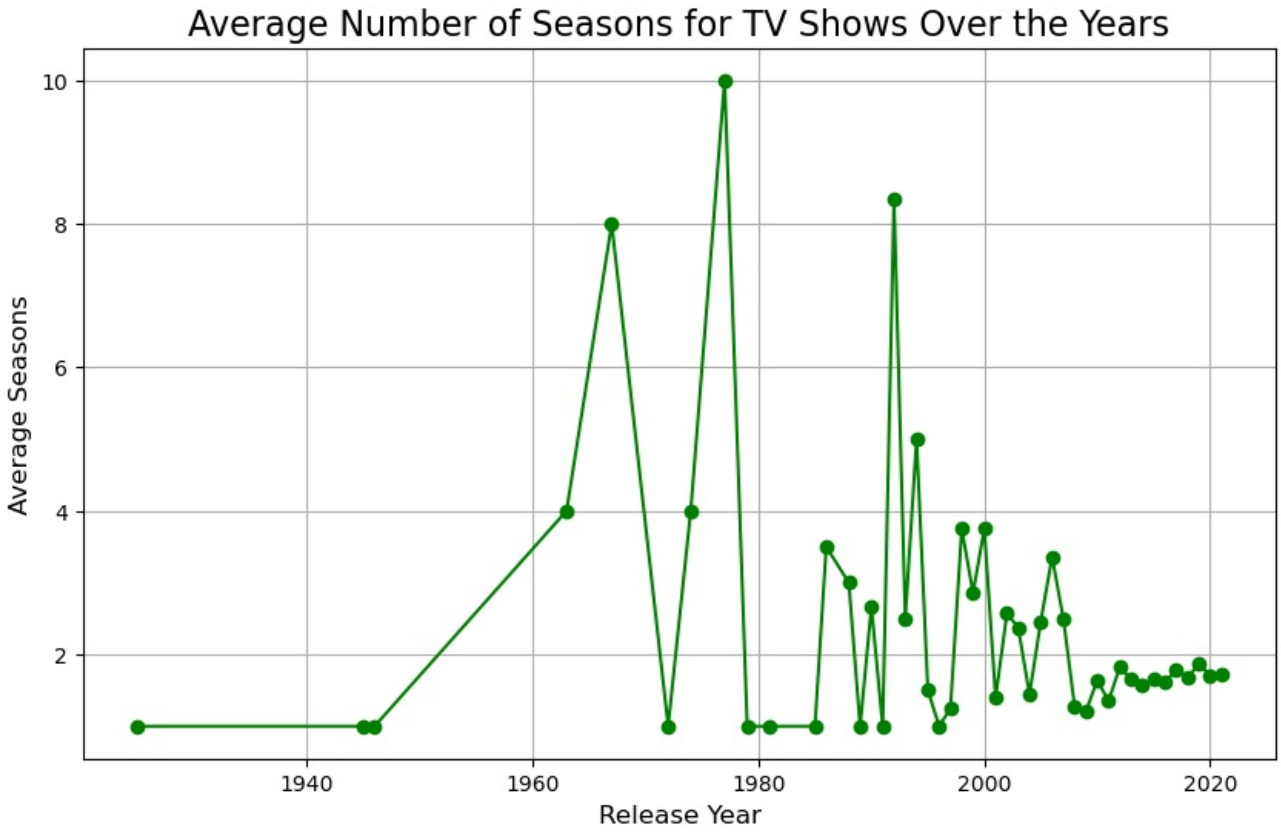
```
<Figure size 1200x800 with 0 Axes>
```



- **The increase in G, NC-17, and NR rated movies until 2018, followed by a gradual decline, reflects a peak in diverse content production, with a shift in focus or regulation after 2018.**

In [220…

```
# --- TV Shows: Seasons Over Time ---
tv_seasons_trends = tv_show_data.groupby('release_year')['duration_seasons'].mean()

plt.figure(figsize=(10, 6))
tv_seasons_trends.plot(kind='line', color='green', marker='o')
plt.title("Average Number of Seasons for TV Shows Over the Years", fontsize=16)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Average Seasons", fontsize=12)
plt.grid()
plt.show()
```

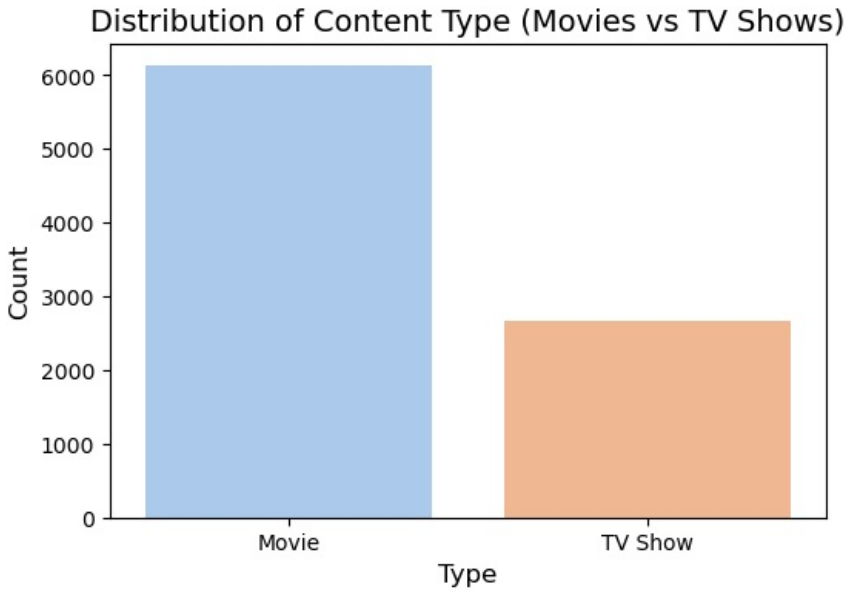


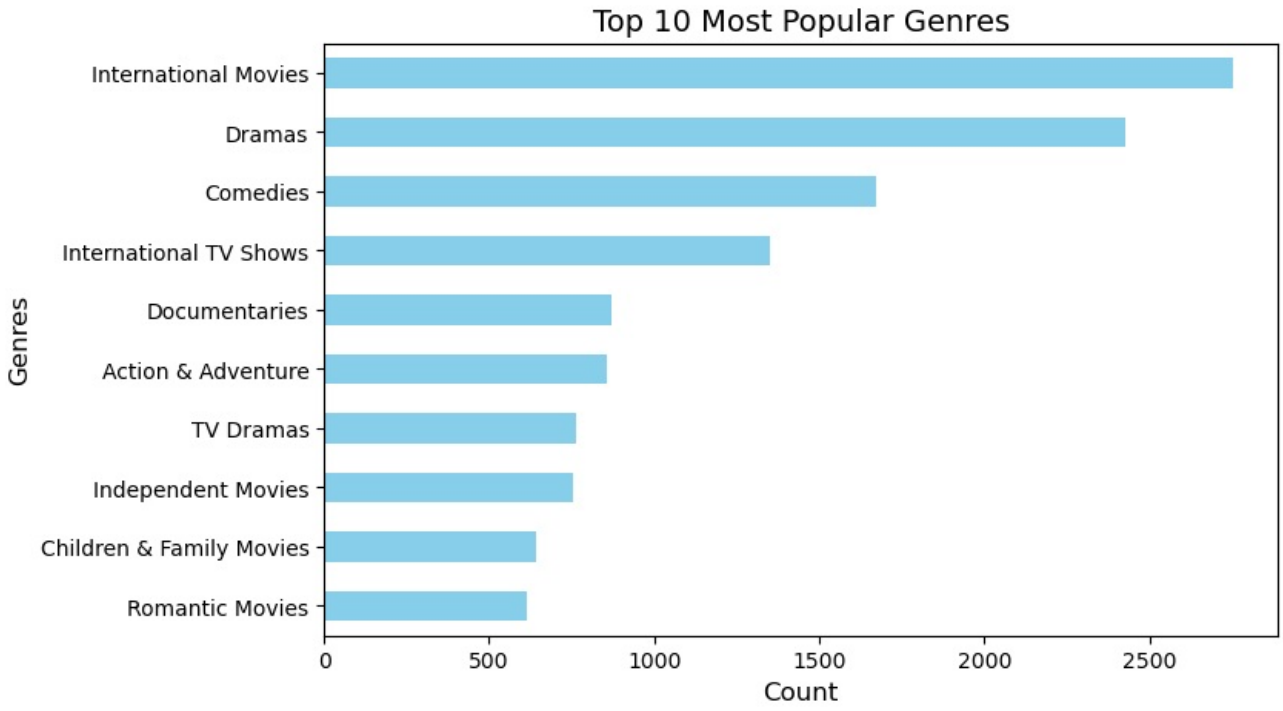
- **Between 1690 and 1980, shows with 4 or more seasons were common, but post-2000, the trend shifted to shorter series, likely due to changing audience preferences and the rise of streaming platforms.**

• Investigate whether certain genres or types of content are more popular among users.

In [225…
```
movie_genre_counts = movie_data['genres'].value_counts().head(5)
```

In [236…
```
# Visualization 1: Movies vs TV Shows Popularity
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='type', palette='pastel')
plt.title("Distribution of Content Type (Movies vs TV Shows)", fontsize=14)
plt.xlabel("Type", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()

# Visualization 2: Top 10 Most Popular Genres
df['listed_in'] = df['genres'].str.split(', ')  # Split genres into lists
genres = df.explode('genres')['genres'].value_counts().head(10)  # Flatten and count top genres

plt.figure(figsize=(8, 5))
genres.sort_values().plot(kind='barh', color='skyblue')
plt.title("Top 10 Most Popular Genres", fontsize=14)
plt.xlabel("Count", fontsize=12)
plt.ylabel("Genres", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```





- **Users prefer watching movies over TV shows, with a higher inclination towards international dramas, reflecting a global taste for diverse and engaging content.**

Insights:

- **International Movies** dominate in content, followed by **dramas**.
- Content increased up to **2018**, then began declining.
- The **United States** has the largest content share on Netflix.
- **July** shows the highest content availability, while **February** has the least.
- **TV-MA** is the most common content rating.
- Most movies are **90 minutes** long, while **TV shows** usually have just **1 season**.
- **Movies** are more highly rated than **TV shows**, and users prefer them over TV shows.
- **International Movies** are the most popular genre over time.
- **International TV shows** also dominate the TV category.
- A peak in movie duration between **1960-1966** indicates a period of artistic experimentation.
- **G, NC-17, and NR** rated movies peaked until 2018, with a decline afterward.
- Shows between **1690-1980** had 4+ seasons, but post-2000, shows became shorter.
- **Users** prefer movies, especially **international dramas**, over TV shows.

Key Trends and Patterns:

- **Steady Growth:** Content production increased significantly after 2010, peaking in the last five years.
- **Global Appeal:** Genres like "International Movies" and "TV Dramas" cater to diverse audiences.
- **Ratings Concentration:** Focus on content rated "TV-MA" or "TV-14" aligns with teenage and adult demographics.
- **Season Lengths:** TV shows tend to avoid longer runs, with a noticeable preference for shorter series.

Recommendations:

- **Diversify Content**: Continue focusing on **international content** as it is a dominant preference, especially in movies and TV shows.
- **Content Strategy by Region**: Consider increasing content from **countries with lower representation** to balance Netflix's global catalog.
- **Monthly Content Release Planning**: Focus on improving content release frequency in **February**, which has the least releases, to maintain consistent engagement throughout the year.
- **Explore Shorter Series**: Given the trend of **shorter series post-2000**, consider investing more in limited series or seasons with 1-3 seasons for flexibility.
- **Genre Focus**: Continue producing more **international dramas** and **movies** based on user preferences, keeping them at the forefront of content strategy.
- **Adjust Movie Length**: Given the longer movie duration peak between **1960-1966**, experiment with both long and short movie formats based on evolving user preferences.

Conclusion:

The insights show a clear preference for **international movies** and **TV-MA ratings**, as well as a general tendency toward shorter **TV shows** in recent years. The **United States** holds the largest content share on Netflix, with **July** being a peak month for content availability. A decline in diverse content production after 2018 and a shift to shorter series highlight changing trends. The recommendation is to leverage global tastes, balance regional content, and explore varying movie lengths while maintaining a strong focus on international genres and movies.

In [ ]: