Business Case: Data Exploration and Visualization on Netflix

About NETFLIX :

Netflix is a leading global streaming platform that provides a wide variety of movies, TV shows, documentaries, and original content to millions of viewers worldwide. Founded in **1997** by **Reed Hastings and Marc Randolph** in the United States, Netflix started as a DVD rental service before evolving into a digital streaming service in 2007.

Introduction :

Netflix is one of the most popular media and video streaming platforms. Today, it operates in over 190 countries and offers entertainment in multiple languages and genres. They have over 10000 movies or tv shows available on their platform, as of mid-2021, they have over 222M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

Business Problem :

The business problem in this dataset focuses on understanding and optimizing Netflix's content strategy to improve user engagement and satisfaction. With thousands of movies and TV shows available, it becomes challenging for Netflix to identify what types of content attract and retain viewers the most. By analyzing details such as cast, directors, ratings, release year, and duration, Netflix can uncover patterns and insights about audience preferences and content performance. Ultimately, solving this business problem can lead to increased viewer retention, better investment in original content, and a stronger competitive position in the streaming market.

Analysis the Basic Metrics :

In [4]:

```
# Importing  the Libraries.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:

```
# Load the Dataset.

df=pd.read_csv('/content/netflix.csv')
```

In [ ]:

```
# To Check the Dataset.

df
```

Out[ ]:

	show_id	type	title	director	cast	country	date_added	release_y
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2
...	...	...	...	...	...	...	...	
8802	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...	United States	November 20, 2019	2
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2

	show_id	type	title	director	cast	country	date_added	release_y
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2

8807 rows × 12 columns

```
In [ ]: # To Check First Five Data.

        df.head()
```

Out[ ]:

	show_id	type	title	director	cast	country	date_added	release_year
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021

In [ ]:

```
# To Check the Shape of Data - It means how many rows and columns are present in

df.shape
```

Out[ ]:

```
(8807, 12)
```

In [ ]:

```
#  To Check the Data Types of all the attributes.

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 [ ]:

```
# To Check the Statistics.

df.describe(include=object).T
```

Out[ ]:

	count	unique	top	freq
show_id	8807	8807	s8807	1
type	8807	2	Movie	6131
title	8807	8807	Zubaan	1
director	6173	4528	Rajiv Chilaka	19
cast	7982	7692	David Attenborough	19
country	7976	748	United States	2818
date_added	8797	1767	January 1, 2020	109
rating	8803	17	TV-MA	3207
duration	8804	220	1 Season	1793
listed_in	8807	514	Dramas, International Movies	362
description	8807	8775	Paranormal activity at a lush, abandoned prope...	4

Data Cleaning :

In [ ]:

```
# To Check the Missing Values.

df.isna().sum()
```

Out[ ]:

	0
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

- Here the missing values are director, cast, country, date_added, rating and duration.

Director:

In [ ]:

```
df['director'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-3977420098.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['director'].fillna('Unknown', inplace=True)
```

Cast:

In [ ]:

```
df['cast'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-382634866.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['cast'].fillna('Unknown', inplace=True)
```

Country:

In [ ]:

```
df['country'].fillna('Unknown', inplace=True)
```

```
/tmp/ipython-input-2372192134.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['country'].fillna('Unknown', inplace=True)
```

Date_Added:

In [ ]:

```
df['date_added'].fillna(df['date_added'].mode()[0],inplace=True)
```

```
/tmp/ipython-input-41047033.py:1: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['date_added'].fillna(df['date_added'].mode()[0],inplace=True)
```

Rating:

In [ ]:

```
df['rating'].fillna(df['rating'].mode()[0],inplace=True)
```

```
/tmp/ipython-input-1316894312.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace metho
d.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['rating'].fillna(df['rating'].mode()[0],inplace=True)
```

Duration:

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

```
/tmp/ipython-input-9790679.py:1: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['duration'].fillna(df['duration'].mode()[0],inplace=True)
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
                0
     show_id    0
        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
```

In [ ]:

```
df.head()
```

Out[ ]:

	show_id	type	title	director	cast	country	date_added	release_year
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown	United States	September 25, 2021	2020
1	s2	TV Show	Blood & Water	Unknown	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Unknown	September 24, 2021	2021
3	s4	TV Show	Jailbirds New Orleans	Unknown	Unknown	Unknown	September 24, 2021	2021
4	s5	TV Show	Kota Factory	Unknown	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021

Non-Graphical Analysis:

- Non-graphical analysis helps in understanding the basic structure and distribution of data without using visualizations. We can perform this analysis using methods such as value_counts() and nunique() in Python (Pandas).

Value Counts:

In [ ]:

```
# Show how many Movies and TV Shows are available on Netflix.

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

Out[ ]:

```
           count
type
Movie       6131
TV Show     2676
```

dtype: int64

In [ ]:

```
# To check which countries produce the most content on Netflix.

df['country'].value_counts().head()
```

Out[ ]:

```
                  count
country
United States      2818
India               972
Unknown             831
United Kingdom      419
Japan               245
```

dtype: int64

In [ ]:

```
# The count of each rating category (like TV-MA, PG-13, TV-14, etc.).

df['rating'].value_counts().head()
```

Out[ ]:

```
         count
rating
TV-MA     3211
TV-14     2160
TV-PG      863
R          799
PG-13      490
```

dtype: int64

- The **value_counts()** function helps identify the most frequent occurrences of categorical data.

Unique Attributes:

```
In [ ]: # To check the number of unique directors.

        df['director'].value_counts().head()
```

Out[ ]:

director	count
Unknown	2634
Rajiv Chilaka	19
Raúl Campos, Jan Suter	18
Suhas Kadav	16
Marcus Raboy	16

dtype: int64

```
In [ ]: # Show how many unique actors/actresses appear in the dataset.

        df['cast'].nunique()
```

```
Out[ ]: 7693
```

```
In [ ]: # Show how many titles are represented.

        df['title'].nunique()
```

```
Out[ ]: 8807
```

```
In [ ]: # Show how many different years of release are represented.

        df['release_year'].nunique()
```

```
Out[ ]: 74
```

- The **nunique()** function tells how many unique values each column contains.

Queries:

1. How has the number of movies released per year changed over the last 20-30 years?

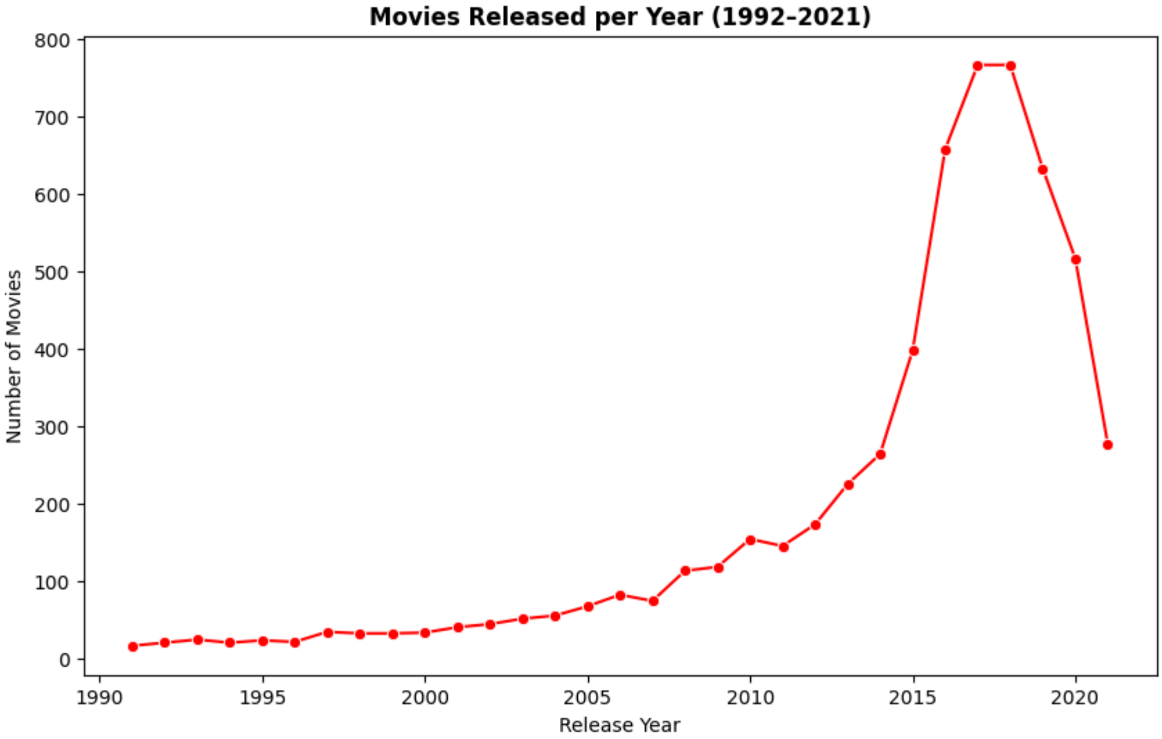
```
In [ ]: movies = df[df['type'] == 'Movie']
        yearly_count = movies['release_year'].value_counts().sort_index()
        yearly_count = yearly_count[yearly_count.index >= (yearly_count.index.max()-30)]
        yearly_count
```

Out[ ]:

	count
release_year	
1991	16
1992	20
1993	24
1994	20
1995	23
1996	21
1997	34
1998	32
1999	32
2000	33
2001	40
2002	44
2003	51
2004	55
2005	67
2006	82
2007	74
2008	113
2009	118
2010	154
2011	145
2012	173
2013	225
2014	264
2015	398
2016	658
2017	767
2018	767
2019	633
2020	517
2021	277

dtype: int64

```
In [ ]: plt.figure(figsize=(10,6))
        sns.lineplot(x=yearly_count.index,y=yearly_count.values,marker='o',color='red')
        plt.title('Movies Released per Year (1992-2021)',fontweight='bold',color='black'
        plt.xlabel('Release Year')
        plt.ylabel('Number of Movies')
        plt.show()
```



2. Comparison of tv shows vs. movies?

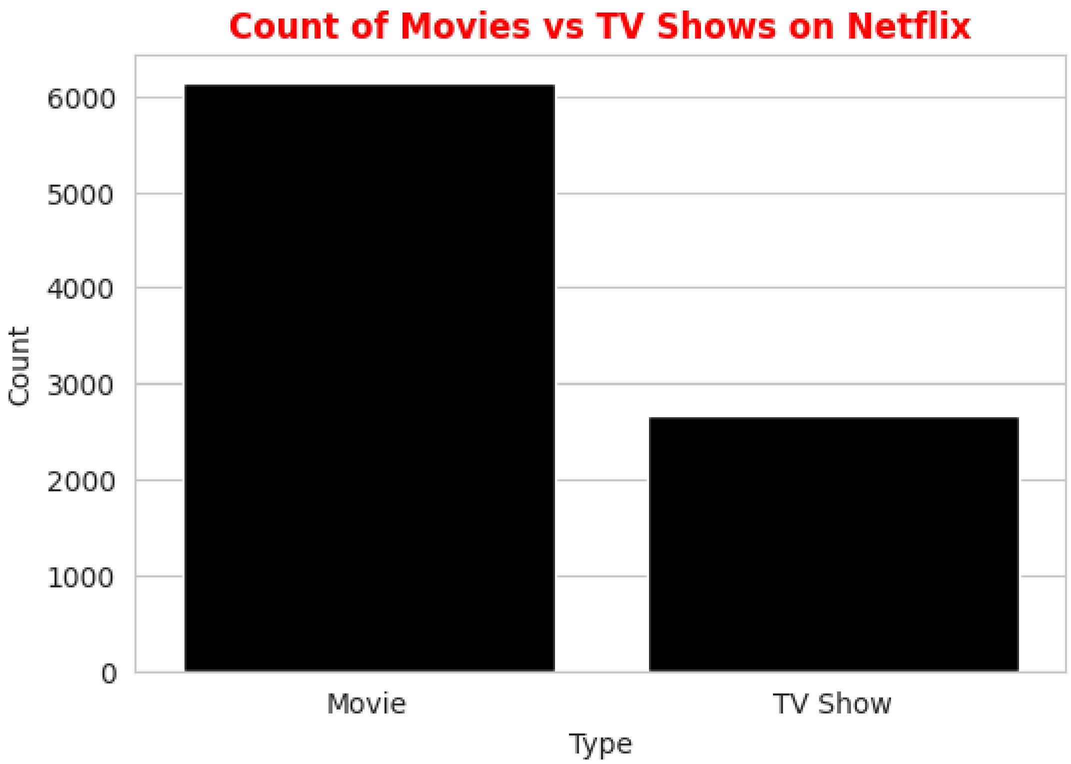
- **Compare Count of Movies vs TV Shows?**

```
In [ ]: type_count = df['type'].value_counts()
        print("Count of Movies vs TV Shows:\n")
        print(type_count)
```

```
Count of Movies vs TV Shows:

type
Movie      6131
TV Show    2676
Name: count, dtype: int64
```

```
In [ ]: plt.figure(figsize=(6,4))
        sns.countplot(data=df, x='type',color='black')
        plt.title('Count of Movies vs TV Shows on Netflix',fontweight='bold',color='red'
        plt.xlabel('Type')
        plt.ylabel('Count')
        plt.show()
```



- **Top 10 Countries Producing Movies vs TV Shows?**

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

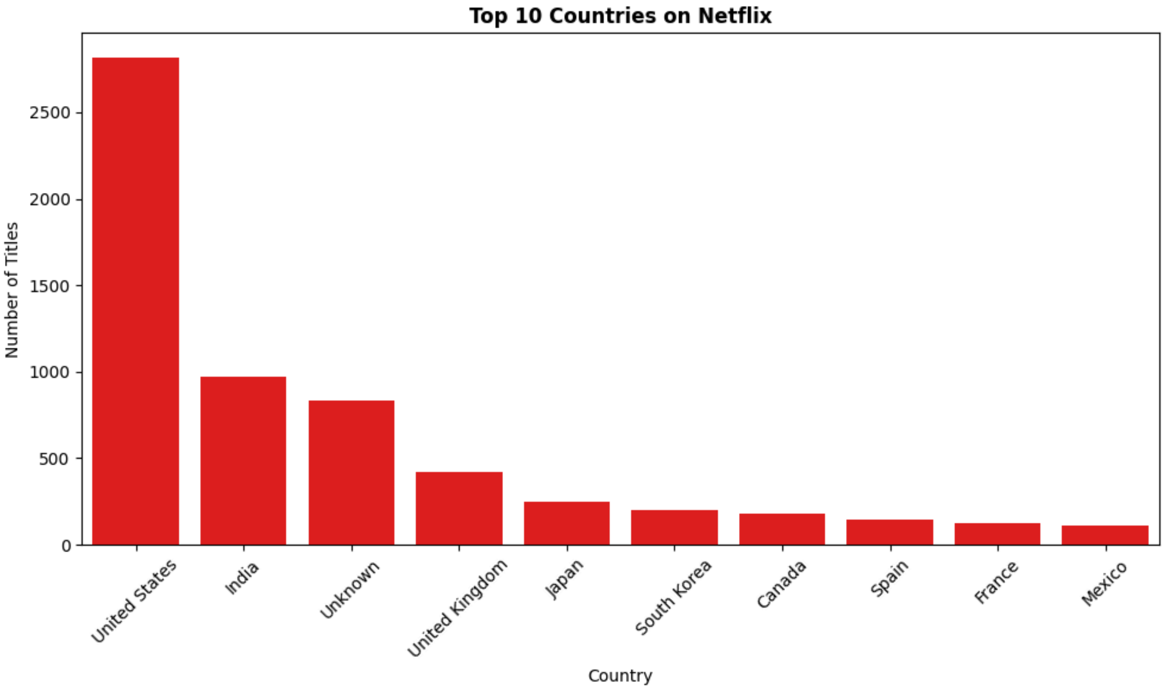
Out[ ]:

	count
country	
United States	2818
India	972
Unknown	831
United Kingdom	419
Japan	245
South Korea	199
Canada	181
Spain	145
France	124
Mexico	110

dtype: int64

In [ ]:
```
plt.figure(figsize=(10,6))
sns.barplot(x=top_countries.index, y=top_countries.values, color='red')
plt.title("Top 10 Countries on Netflix",fontweight='bold', color='black')
plt.xlabel("Country", color='black')
plt.ylabel("Number of Titles", color='black')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



3. What is the best time to launch a TV show?

```
In [ ]: tv_shows = df[df['type'] == 'TV Show']
        month_counts = tv_shows['date_added'].value_counts().sort_index().idxmax()
        month_counts
```

```
Out[ ]: 'July 6, 2021'
```

4. Analysis of actors/directors of different types of shows/movies?

- **Top Directors of Movies.**

```
In [ ]: movies = df[df['type'] == 'Movie']
        director_count = movies['director'].value_counts()
        top_directors = director_count[director_count.index != 'Unknown'].head(10)
        top_directors
```

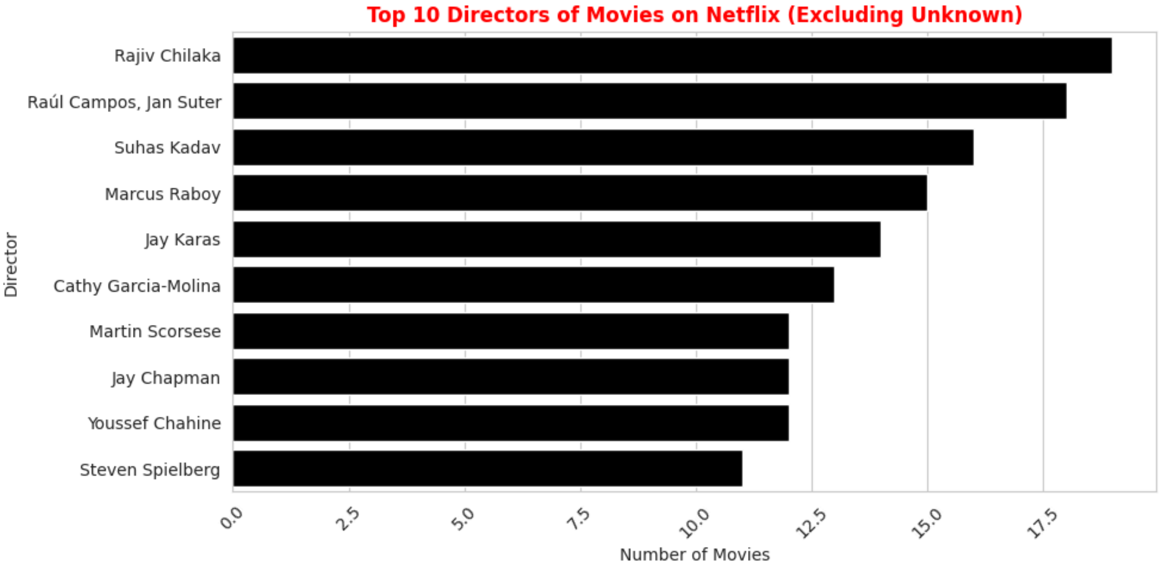
Out[ ]:

	count
director	
Rajiv Chilaka	19
Raúl Campos, Jan Suter	18
Suhas Kadav	16
Marcus Raboy	15
Jay Karas	14
Cathy Garcia-Molina	13
Martin Scorsese	12
Jay Chapman	12
Youssef Chahine	12
Steven Spielberg	11

dtype: int64

In [ ]:

```
plt.figure(figsize=(10,5))
sns.barplot(x=top_directors.values, y=top_directors.index,color='black')
plt.title('Top 10 Directors of Movies on Netflix (Excluding Unknown)',fontweight
plt.xlabel('Number of Movies')
plt.ylabel('Director')
plt.xticks(rotation=45)
plt.show()
```



5. Does Netflix has more focus on TV Shows than movies in recent years?

In [ ]:

```
df['year_wise']=df['date_added'].dt.year.astype('Int64')
year_wise=df.groupby(['year_wise','type']).size().tail(10)
year_wise
```

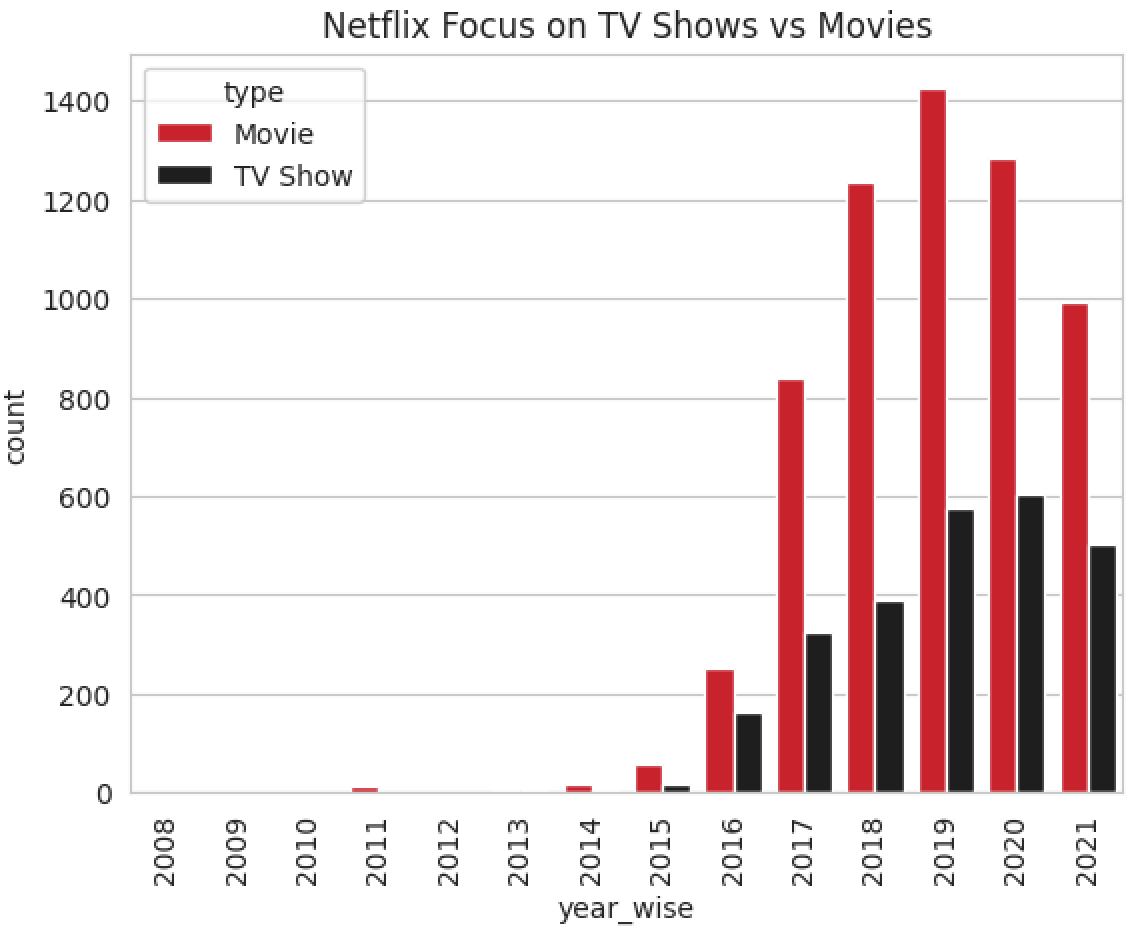
Out[ ]:

		0
year_wise	**type**	
2017	**Movie**	839
	TV Show	325
2018	**Movie**	1237
	TV Show	388
2019	**Movie**	1424
	TV Show	575
2020	**Movie**	1284
	TV Show	604
2021	**Movie**	993
	TV Show	505

dtype: int64

In [ ]:

```
palette = ["#E50914", "#221F1F"]
sns.countplot(x='year_wise', hue='type', data=df, palette=palette)
plt.xticks(rotation=90)
plt.title('Netflix Focus on TV Shows vs Movies')
plt.show()
```



6. Understanding what content is available in different countries?

In [ ]:

```
country_content=df['country'].value_counts().head()
country_content
```

Out[ ]:

	count
country	
United States	2818
India	972
Unknown	831
United Kingdom	419
Japan	245

dtype: int64

- **Country by type(movies/tvshows):**

In [ ]:

```
df[['country','type']].value_counts().head(10)
```

Out[ ]:

		count
country	**type**	
United States	**Movie**	2058
India	**Movie**	893
United States	**TV Show**	760
Unknown	**Movie**	440
	TV Show	391
United Kingdom	**TV Show**	213
	Movie	206
Japan	**TV Show**	169
South Korea	**TV Show**	158
Canada	**Movie**	122

dtype: int64

Visual Analysis:

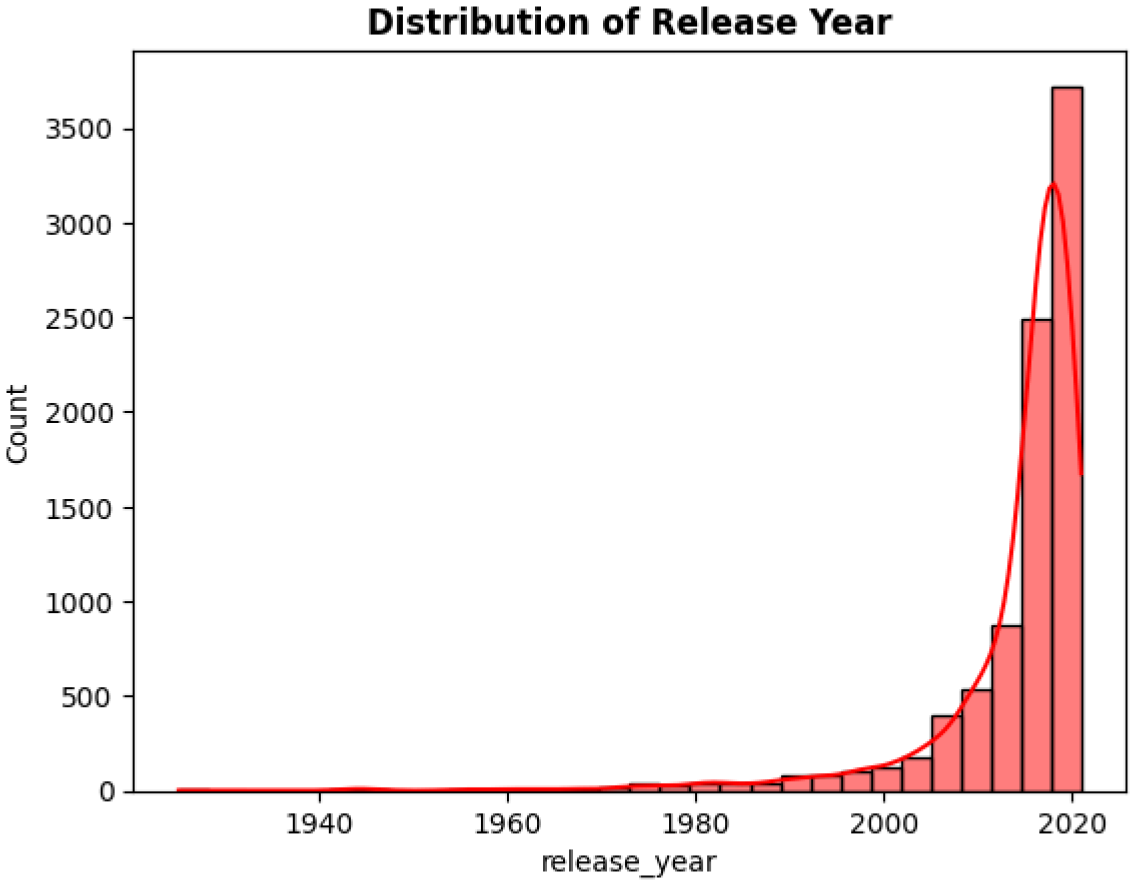
Univariate Analysis:

- Univariate = analyzing one variable at a time.

(a) For Continuous Variables.

- **Release_year:**

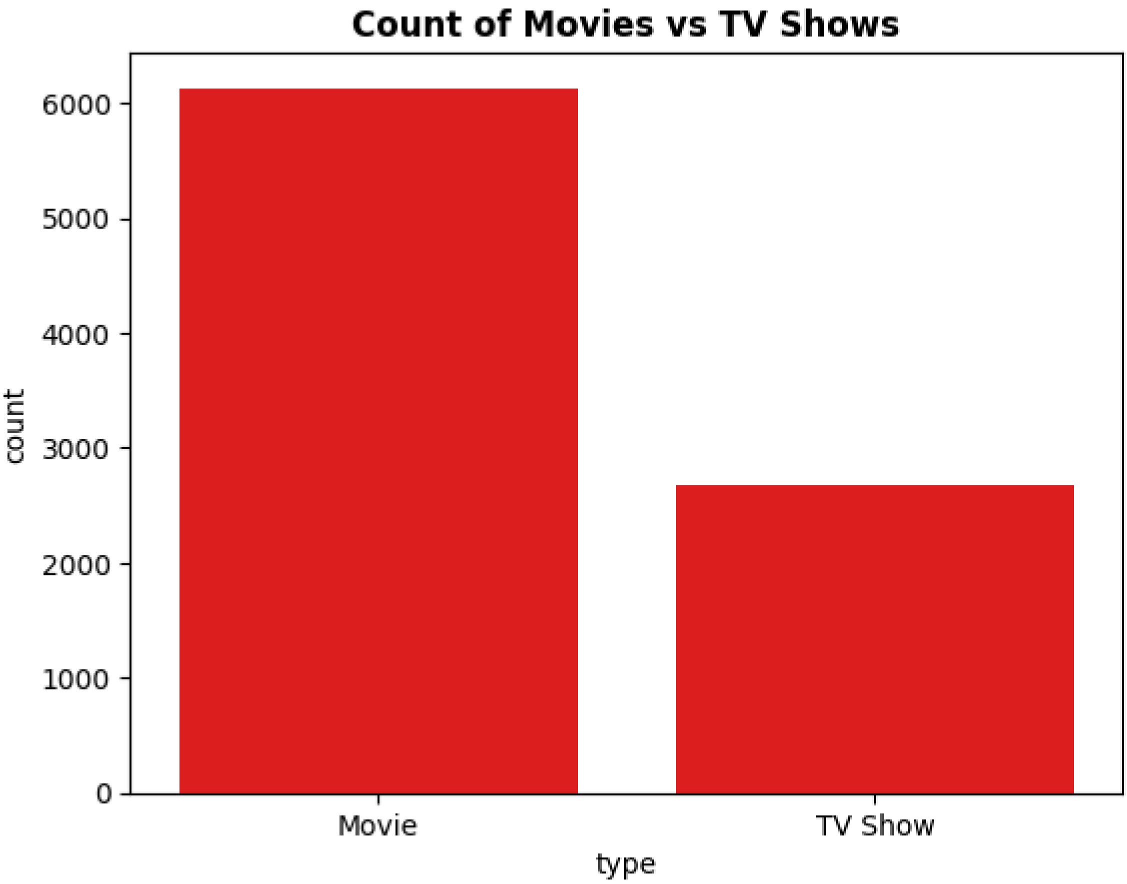
```
In [ ]: sns.histplot(df['release_year'], bins=30, kde=True,color='red')
        plt.title('Distribution of Release Year',fontweight='bold', color='black')
        plt.show()
```



(b) For categorical variable:

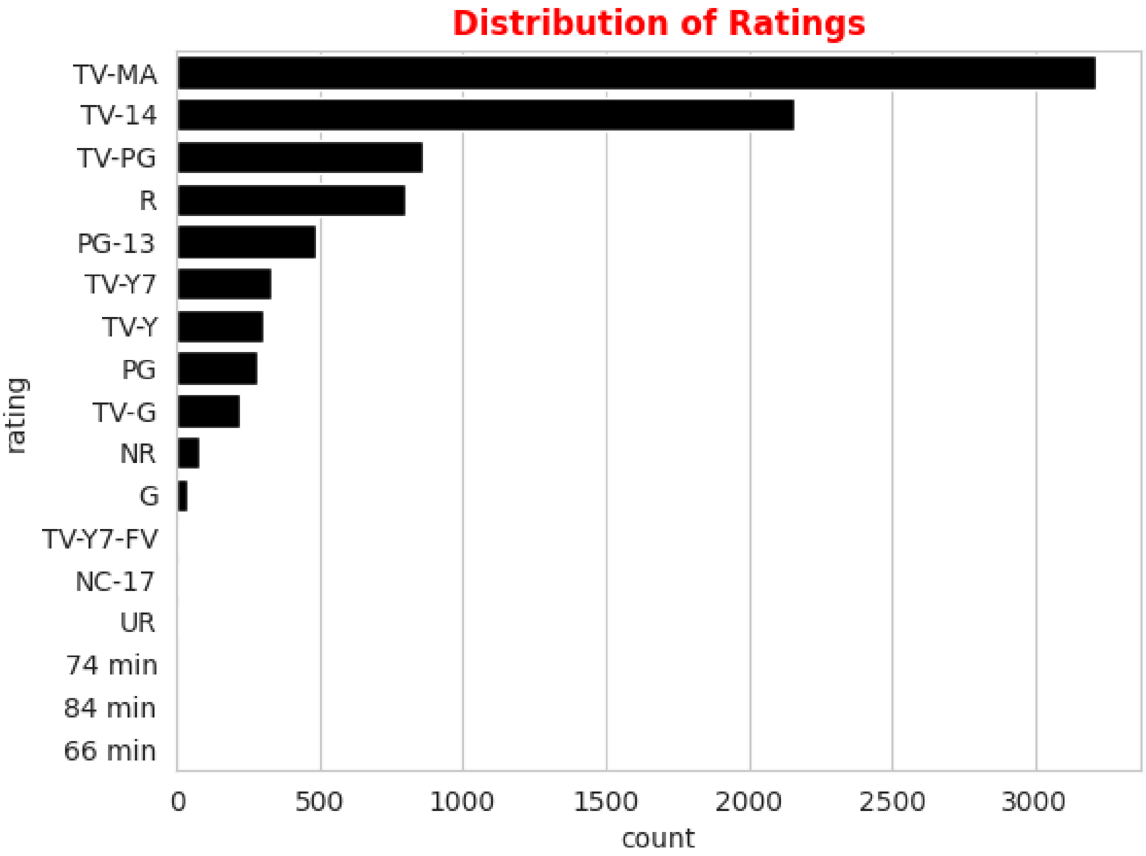
- **Type:**

```
In [ ]: sns.countplot(x='type', data=df,color='red')
        plt.title('Count of Movies vs TV Shows',fontweight='bold',color='black')
        plt.show()
```



- **Rating:**

In [ ]:
```
sns.countplot(y='rating',data=df, order=df['rating'].value_counts().index,color=
plt.title('Distribution of Ratings',fontweight='bold',color='red')
plt.show()
```



Bivariate Analysis:

- Bivariate = analyzing two variables together.

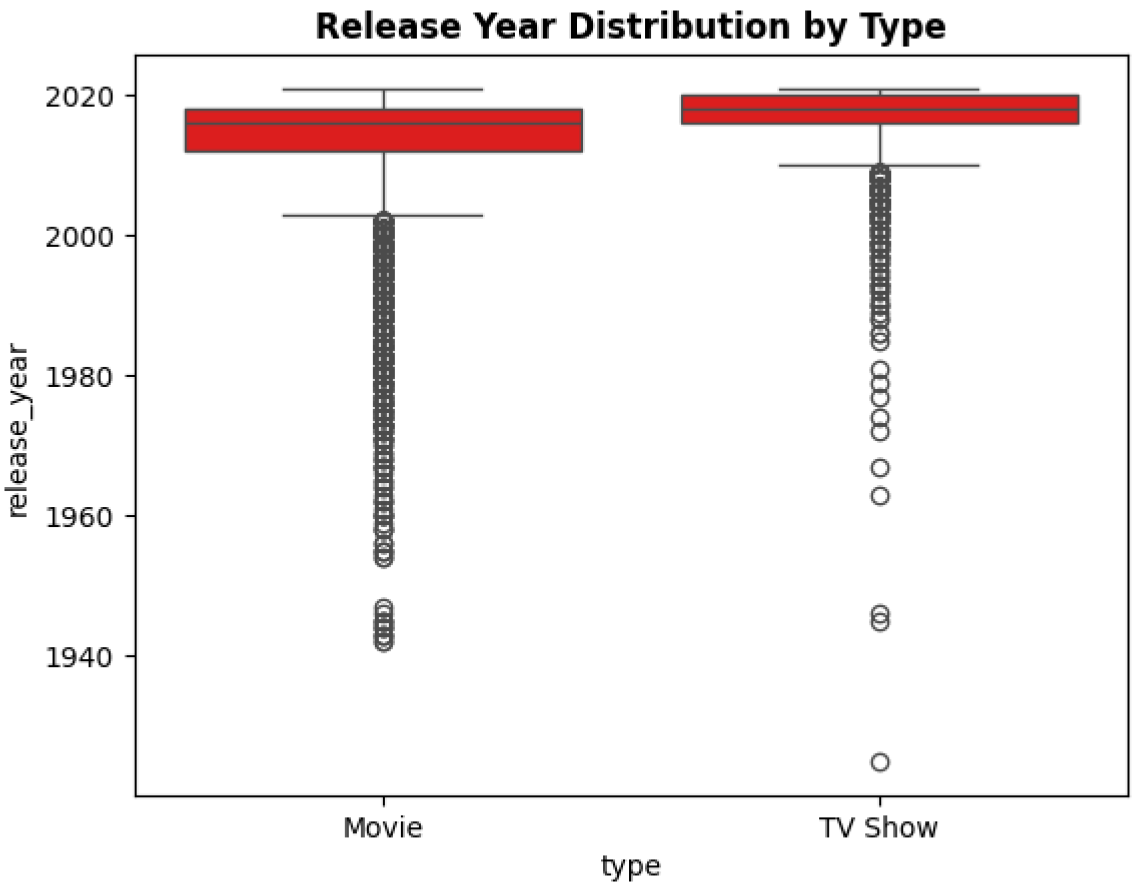
(a) Categorical vs Numerical:

- Used to compare how a numeric value changes across categories.

Type (Movie/TV Show) vs Release_year:

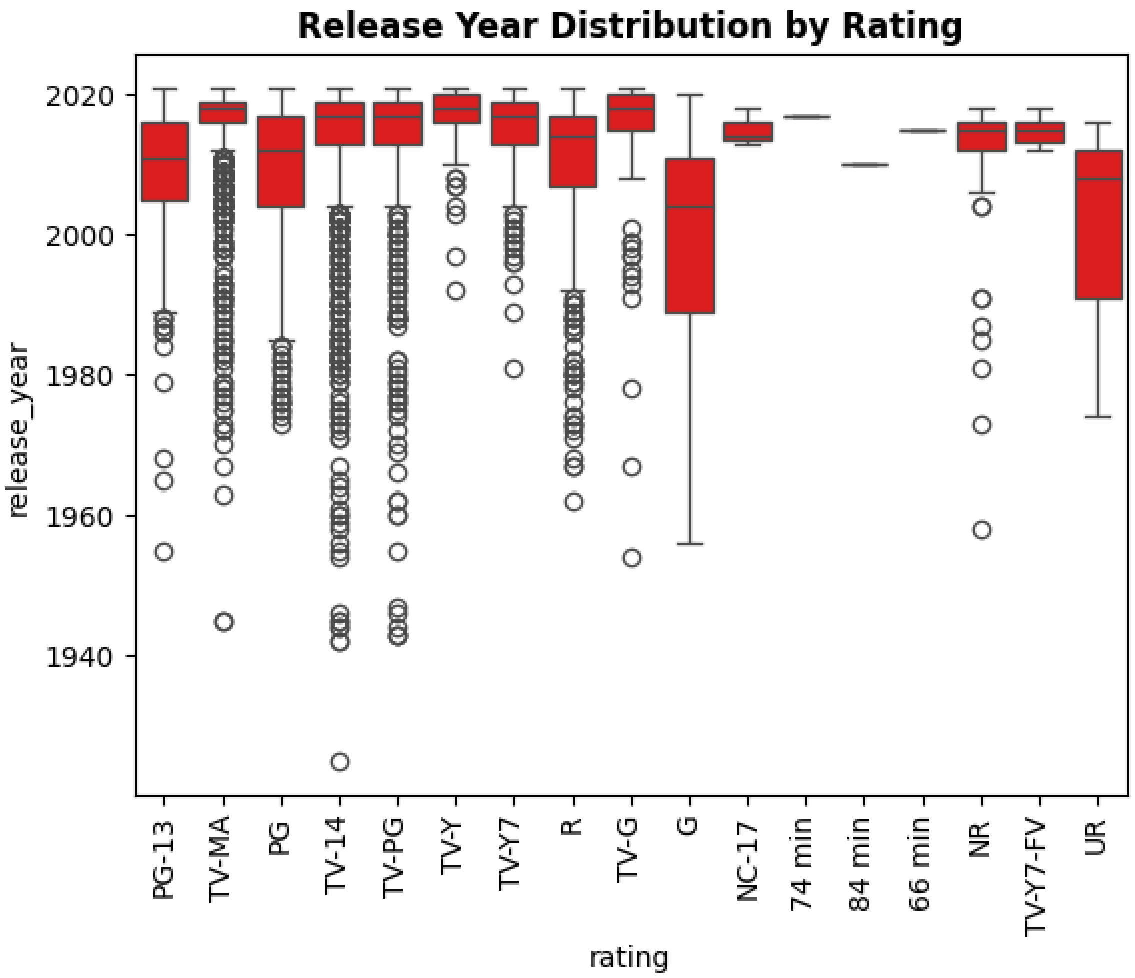
In [ ]:

```
sns.boxplot(x='type', y='release_year', data=df,color='red')
plt.title('Release Year Distribution by Type',fontweight='bold',color='black')
plt.show()
```



Rating vs Release_year:

```
In [ ]: sns.boxplot(x='rating', y='release_year', data=df,color='red')
        plt.xticks(rotation=90)
        plt.title('Release Year Distribution by Rating',fontweight='bold',color='black')
        plt.show()
```



📊 **Interpretation:** Shows the spread (median, quartiles, outliers) of a numerical variable across categories.

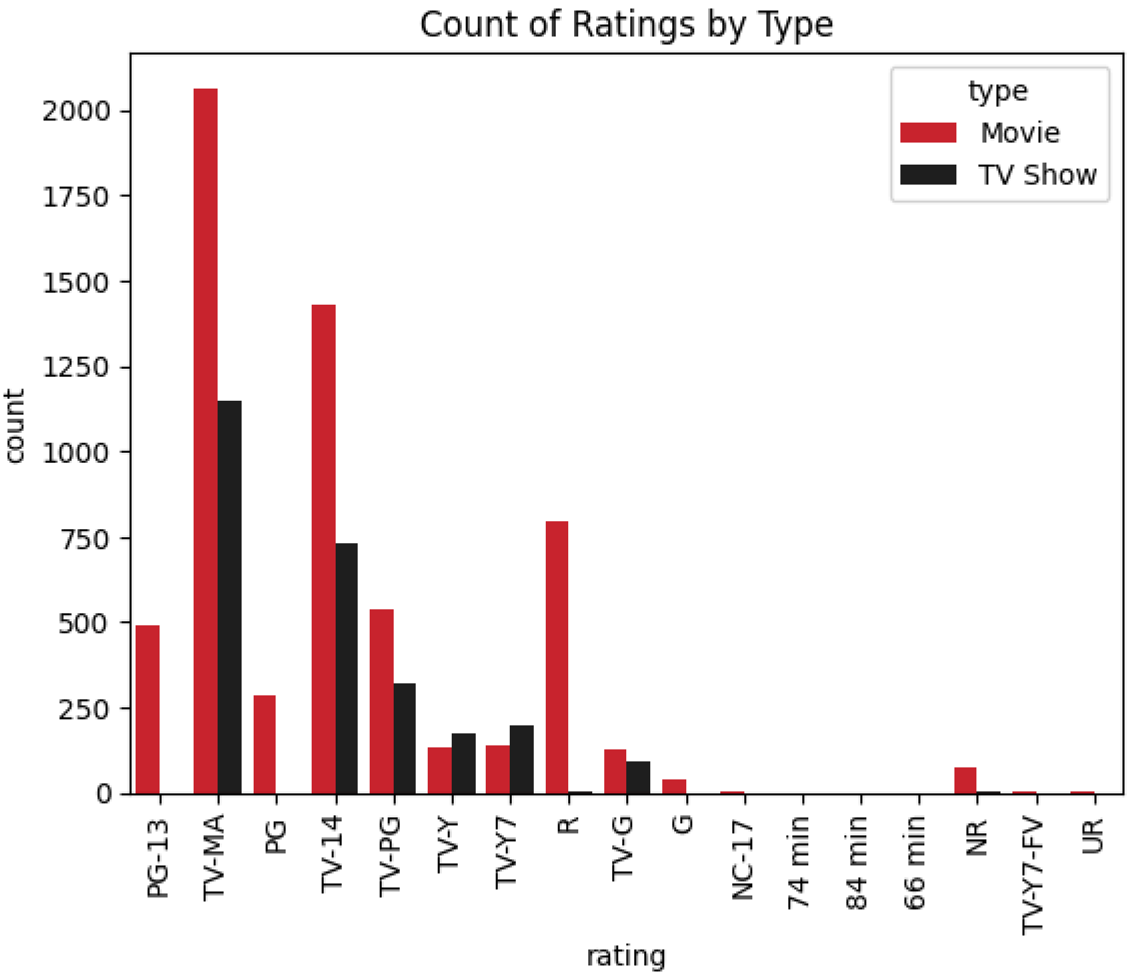
(b) Categorial vs Categorial:

- Used to see frequency relationships between two categories.

Type vs Rating:

In [ ]:

```
palette = ["#E50914", "#221F1F"]
sns.countplot(x='rating', hue='type', data=df, palette=palette)
plt.xticks(rotation=90)
plt.title('Count of Ratings by Type')
plt.show()
```



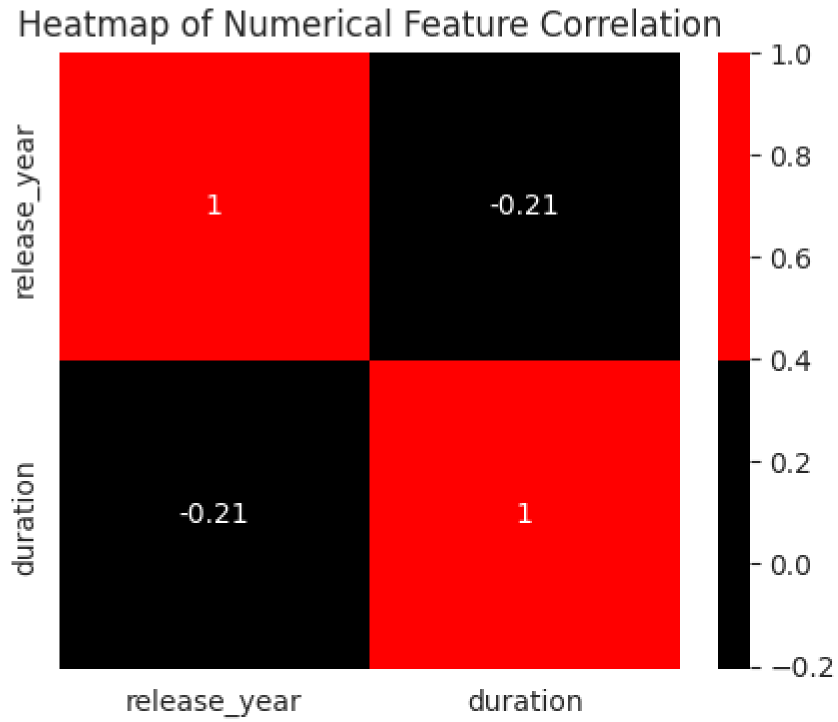
For correlation:

- **Heat Map Correlation:**

```
In [ ]: df['duration'] = df['duration'].astype(str).str.replace(' min', '', regex=False)
        df['duration'] = pd.to_numeric(df['duration'], errors='coerce')
        corr = df[['release_year', 'duration']].corr()
        print(corr)
```

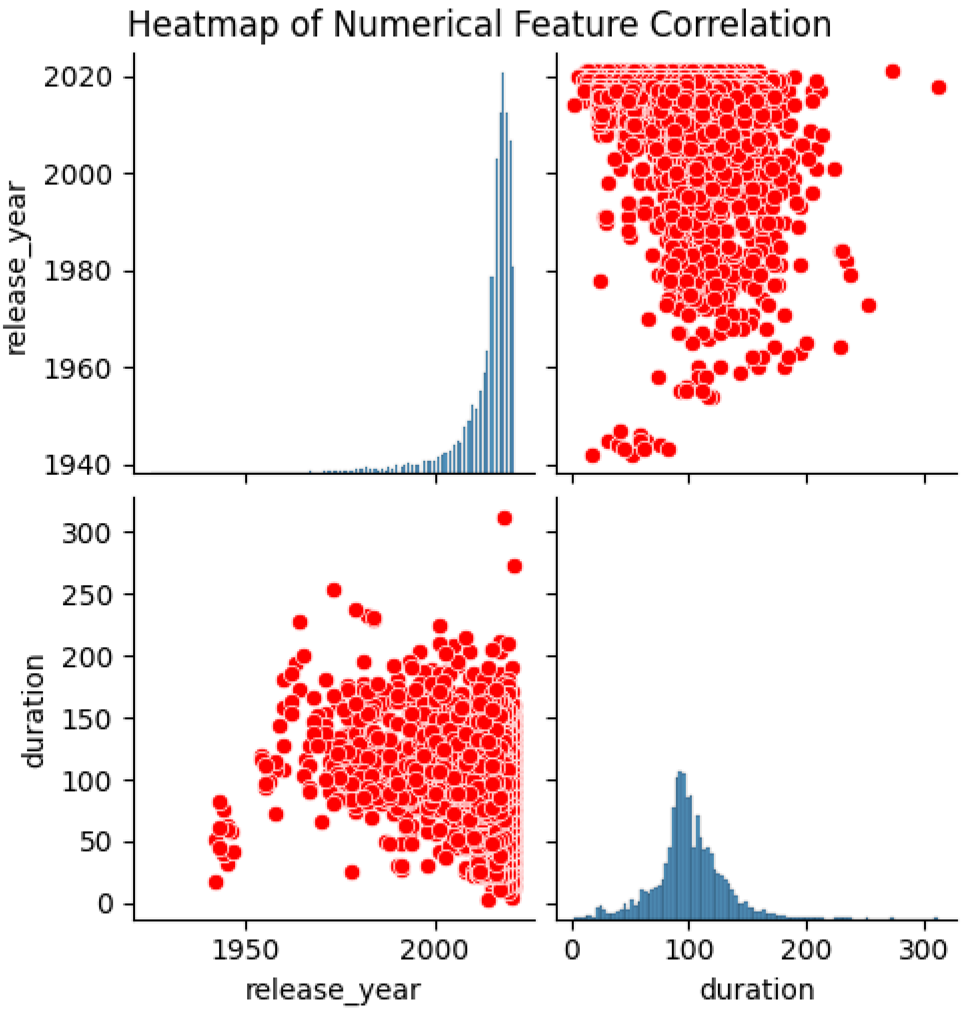
```
              release_year  duration
release_year      1.000000 -0.206285
duration         -0.206285  1.000000
```

```
In [ ]: plt.figure(figsize=(5,4))
        sns.heatmap(corr, annot=True, cmap=sns.color_palette(["black", "red"]))
        plt.title('Heatmap of Numerical Feature Correlation')
        plt.show()
```



- **Pair Plot Correlation:**

```
In [ ]: sns.pairplot(df[['release_year', 'duration']],plot_kws={'color': 'red'})
        plt.suptitle('Heatmap of Numerical Feature Correlation', y=1.02)
        plt.show()
```



Insights Based on Non-Graphical and Visual Analysis:

1. Comments on the Range of Attributes:

- The dataset contains a wide range of attributes describing Netflix titles — including show_id, type, title, director, cast, country, date_added, release_year, rating, duration, listed_in, and description.
- The release_year ranges from 1925 to 2021, showing that Netflix hosts both classic and modern content.
- The date_added attribute ranges across multiple years, indicating a continuous addition of titles to the platform.
- Country and Director fields have missing values, showing incomplete metadata for some titles.
- Rating covers categories like TV-MA, TV-14, PG, R, showing content suitable for different age groups.

2. Comments on the Distribution of Variables and Relationships Between Them:

- Most titles were released after 2010, showing Netflix's strong focus on recent content.

- Movies dominate the dataset with about 69% share, while TV Shows form ~31%.
- The duration variable is right-skewed — most movies last between 80–120 minutes.
- The correlation between release_year and duration is -0.20, showing that newer movies tend to be shorter.
- Country vs Type analysis shows that the United States and India produce the most Netflix content.

3. Comments for Each Univariate and Bivariate Plot:

Univariate (Single Variable):

- Movies dominate over TV Shows.
- Growth in releases after 2015, peak around 2018–2020.
- TV-MA and TV-14 most frequent; focus on adult content.
- U.S. leads, followed by India and the U.K.
- Most movies last around 90–120 minutes; few long outliers.

Bivariate Plots:

- Movies have a wide duration range; TV shows have grouped seasons.
- Ratings differ between Movies and TV Shows.
- Most titles are added close to their release year.
- Weak correlation — Netflix adds both old and new titles.
- Shows scattered clusters, no strong linear relation.

Outliers Check:

In [ ]:
```
df_num=df.describe()
df_num
```

Out[ ]:

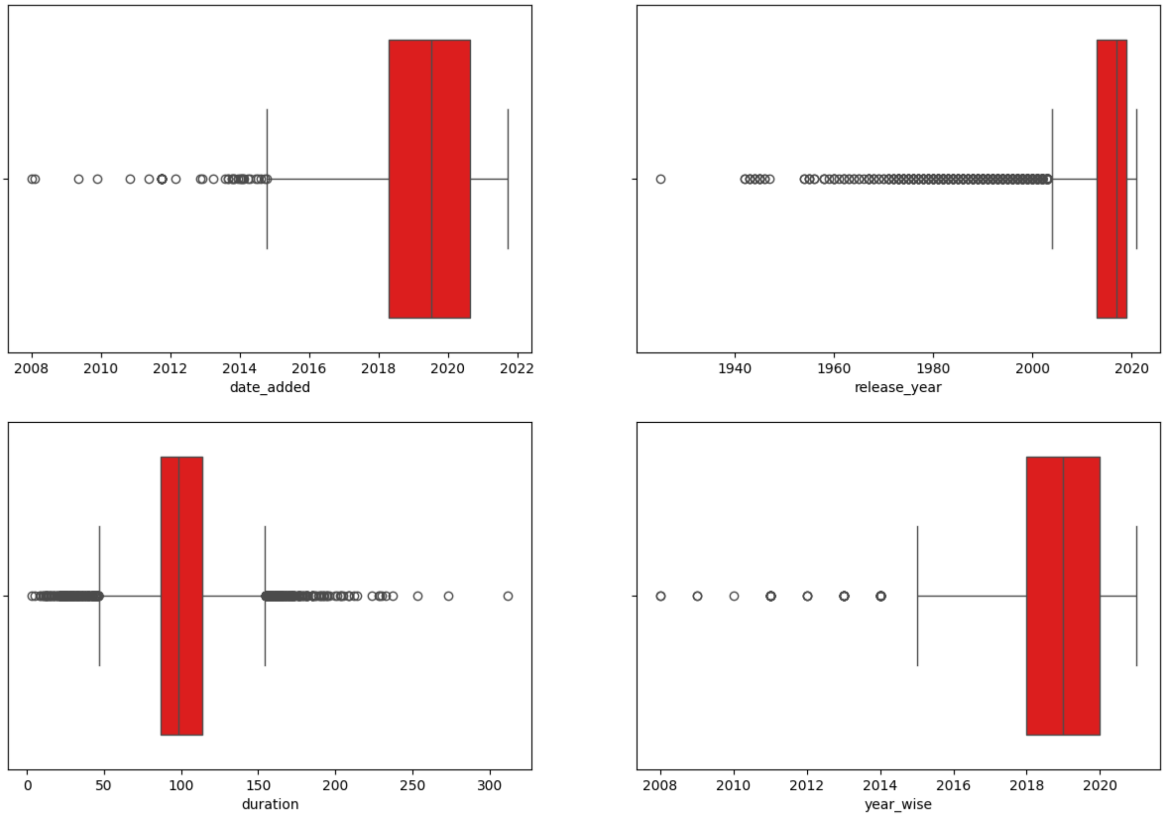
	date_added	release_year	duration	year_wise
count	8719	8807.000000	6128.000000	8719.0
mean	2019-05-23 07:53:40.139923968	2014.180198	99.577187	2018.889207
min	2008-01-01 00:00:00	1925.000000	3.000000	2008.0
25%	2018-04-20 00:00:00	2013.000000	87.000000	2018.0
50%	2019-07-12 00:00:00	2017.000000	98.000000	2019.0
75%	2020-08-25 00:00:00	2019.000000	114.000000	2020.0
max	2021-09-25 00:00:00	2021.000000	312.000000	2021.0
std	NaN	8.819312	28.290593	1.567513

In [ ]:
```
df_num.columns
```

Out[ ]:
```
Index(['date_added', 'release_year', 'duration', 'year_wise'], dtype='object')
```

In [ ]:

```
fig,axes=plt.subplots(nrows=2,ncols=2,figsize=(15,10))
axes=axes.flatten()
for i,j in enumerate(df_num.columns):
  sns.boxplot(x=df[j],ax=axes[i],color='red')
```



Remove Outliers:

- **Release_year:**

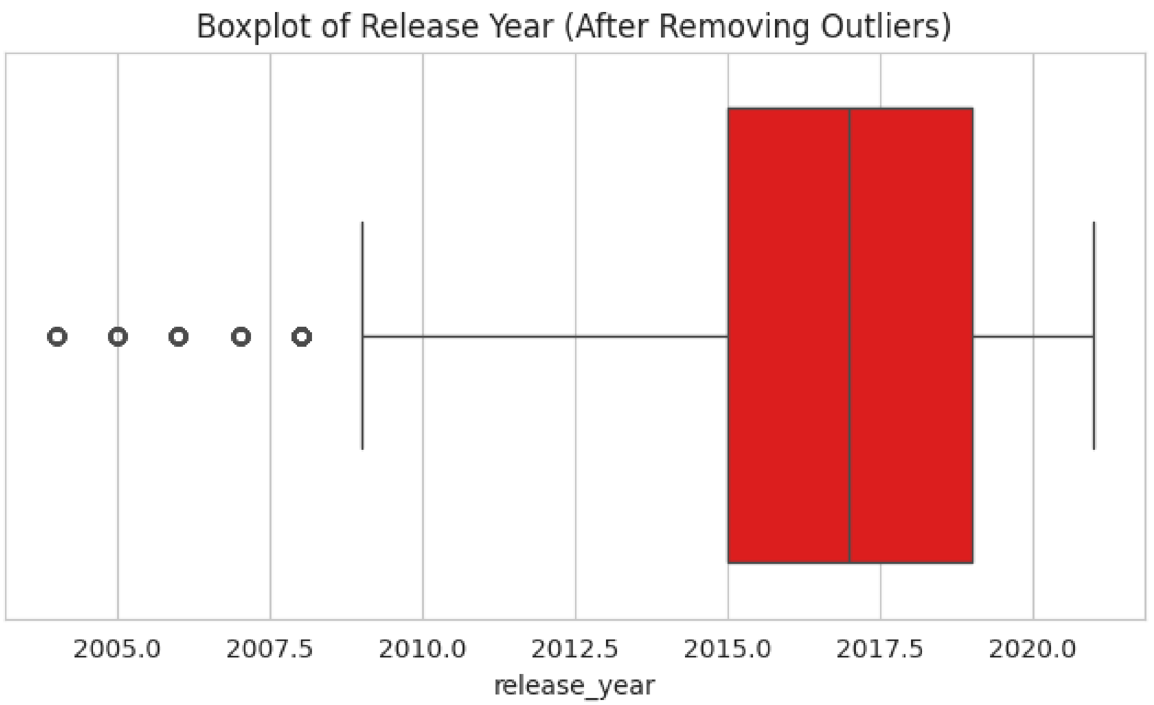
In [ ]:

```
Q1 = df['release_year'].quantile(0.25)
Q3 = df['release_year'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['release_year'] >= lower_bound) & (df['release_year'] <= upper_bound
print("Outliers removed. Remaining rows:", len(df))
```

```
Outliers removed. Remaining rows: 8088
```

In [ ]:

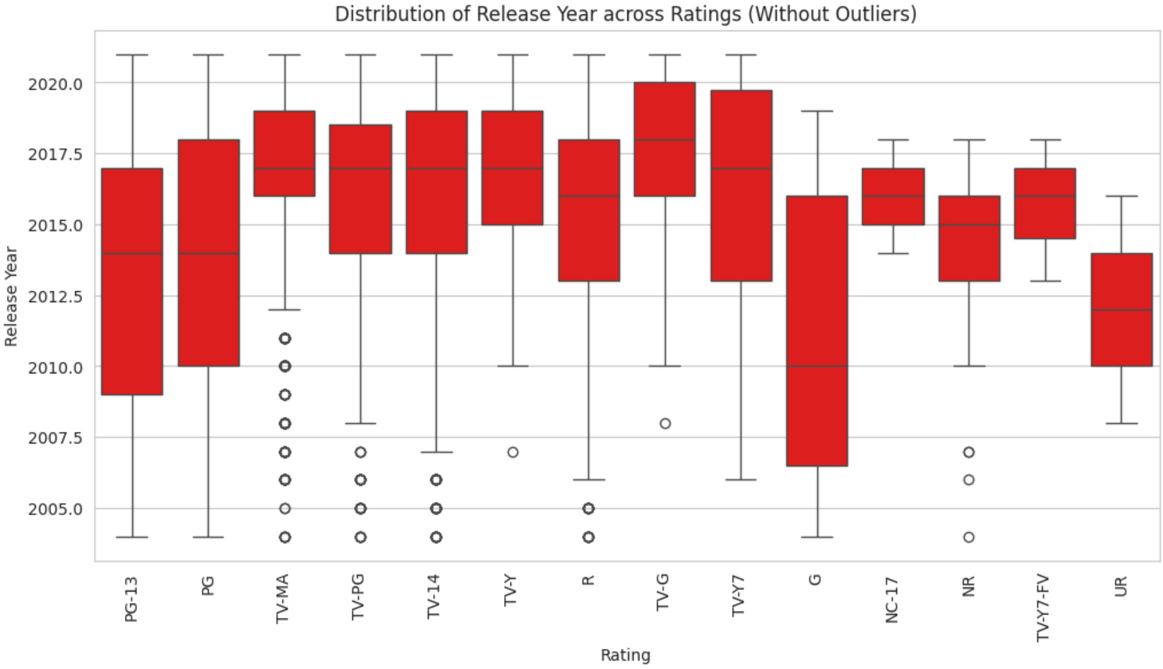
```
plt.figure(figsize=(8,4))
sns.boxplot(x=df['release_year'], color='red')
plt.title('Boxplot of Release Year (After Removing Outliers)', fontsize=12)
plt.show()
```



```
In [ ]: Q1 = df['duration'].quantile(0.25)
        Q3 = df['duration'].quantile(0.75)
        IQR = Q3 - Q1
        lower_limit = Q1 - 1.5 * IQR
        upper_limit = Q3 + 1.5 * IQR
        df= df[(df['duration'] >= lower_limit) & (df['duration'] <= upper_limit)]
```

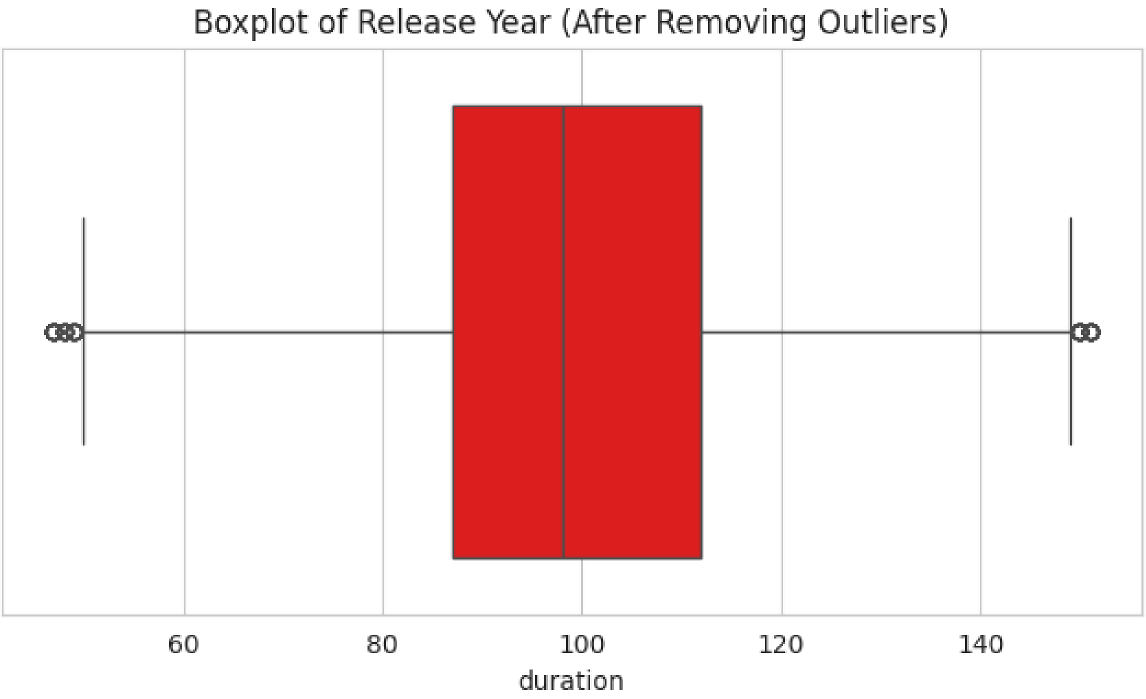
- **Release Year vs Rating:**

```
In [ ]: plt.figure(figsize=(12,6))
        sns.boxplot(x='rating', y='release_year', data=df, color='red')
        plt.title('Distribution of Release Year across Ratings (Without Outliers)')
        plt.xlabel('Rating')
        plt.ylabel('Release Year')
        plt.xticks(rotation=90)
        plt.show()
```



- **Duration:**

```
In [ ]: plt.figure(figsize=(8,4))
        sns.boxplot(x=df['duration'], color='red')
        plt.title('Boxplot of Release Year (After Removing Outliers)', fontsize=12)
        plt.show()
```



Business Insights:

✔ **Movies dominate Netflix’s content library** — showing that films are Netflix’s strongest category and attract the largest audience.

🎯 **Most content rated TV-MA or TV-14** — focus is mainly on mature and teenage audiences, suggesting a trend toward adult-oriented entertainment.

🌍 **Top-producing countries** — United States, India, and United Kingdom — Netflix relies heavily on English and Indian regional markets.

🎬 **Popular genres** — Dramas, Documentaries, and Comedies — viewers prefer emotional, real-life, and light-hearted content.

📈 **Low correlation between release year and duration** — suggests Netflix offers both short and long titles, ensuring variety.

💡 **Overall** — Netflix's library shows global diversity, mature audience targeting, and content variety across genres and durations.

Recommendations:

✔️ **Increase family and kids** content to attract wider age groups.

📺 **Invest more in TV Shows** since they keep users engaged for longer periods.

🌍 **Expand international collaborations** — especially with emerging markets like South Korea, Spain, and India.

📊 **Highlight short-duration content** for casual viewers who prefer quick entertainment.

💬 **Encourage user feedback and ratings** to improve personalized recommendations.

🛠️ **Fill missing details** (like directors or countries) in metadata for better search and categorization.

🎯 **Maintain balance between movies and TV shows** to serve both binge-watchers and short-term viewers.