NETFLIX DATA ANALYSIS -- BY SUHASINI MOTTANNAVAR

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

```
! wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv"
```

```
--2024-12-02 14:49:58--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 65.8.234.36, 65.8.234.174, 65.8.234.131, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|65.8.234.36|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: ‘netflix.csv.1’

netflix.csv.1       100%[===================>]   3.24M  --.-KB/s    in 0.08s

2024-12-02 14:49:58 (39.2 MB/s) - ‘netflix.csv.1’ saved [3399671/3399671]
```

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

Q1

Defining Problem Statement and Analysing basic metrics

```
df.head()
```

	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...

Feuds,

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
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
```

```
df.describe()
```

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

```
df.isnull().sum()
```

	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

```
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 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
```

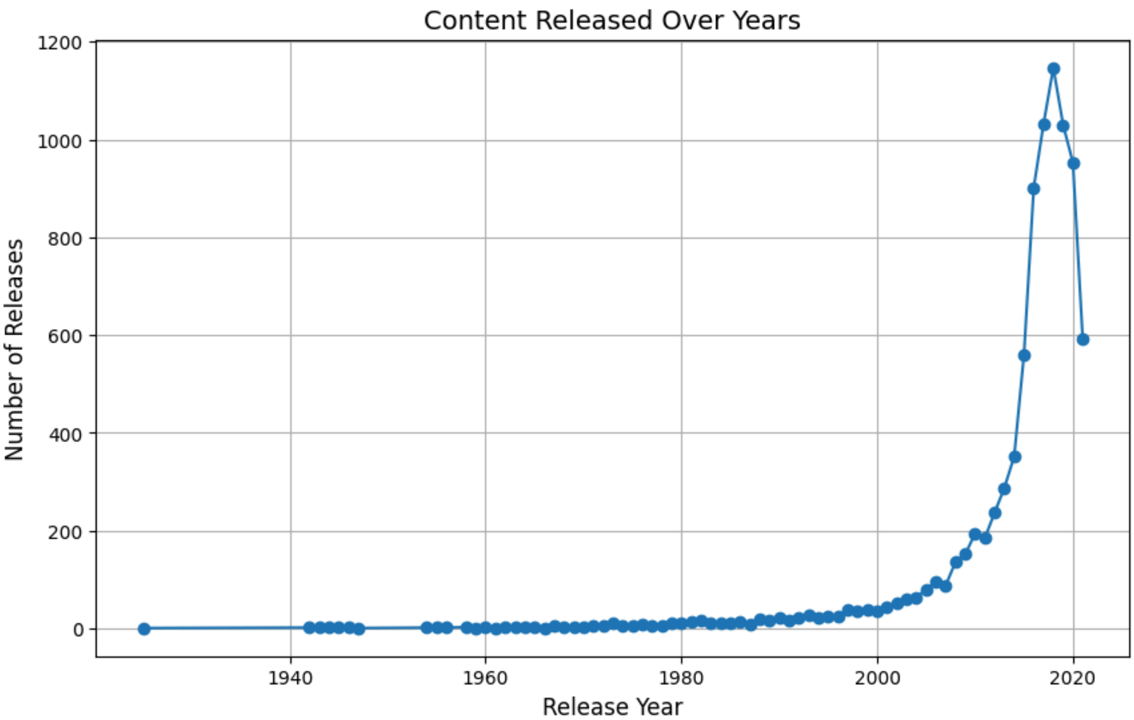
Double-click (or enter) to edit

```
df.value_counts()
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s10	Movie	The Starling	Theodore Melfi	Melissa McCarthy, Chris O'Dowd, Kevin Kline, Timothy Olyphant, Daveed Diggs, Skyler Gisondo, Laura Harrier, Rosalind Chao, Kimberly Quinn, Loretta Devine, Ravi Kapoor	United States	September 24, 2021	2021	PG-13	104 min	Comedies, Dramas	A woman adjusting to life after a loss contends with a feisty bird that's taken over her garden — and a husband who's struggling to find a way forward.
s655	Movie	#Selfie	Cristina Jacob	Flavia Hojda, Crina Semciuc, Olimpia Melinte, Sali Levent, Vlad Logigan, Alex Călin, Alina Chivulescu, Răzvan Vasilescu	Romania	June 21, 2021	2014	TV-MA	125 min	Comedies, Dramas, International Movies	Two days before their final exams, three teen girls make a seaside getaway to have the time of their lives.
s6548	Movie	Dabbe 6: The Return	Hasan Karacadağ	Sema Şimşek, Nilay Gök, Volkan Ünal, Fehmi Karaarslan, Elçin Atamgüç, Ömer Duran, Murat Seviş, Aybike Turan, Burak Çimen	Turkey	April 17, 2019	2015	TV-MA	153 min	Horror Movies, International Movies	A cardiologist tries to pinpoint the cause of her mother's sudden death as her sister, who witnessed it, claims malevolent demons are at play.
s6547	Movie	Daagdi Chaawl	Chandrakant Kanse	Ankush Chaudhari, Makrand Deshpande, Pooja Sawant, Sanjay Khapre, Yatin Karyekar, Kamlesh Sawant, Sandeep Gaikwad, Digvijay Rohidas	India	February 15, 2018	2015	TV-14	118 min	Action & Adventure, Dramas, International Movies	A simple man's peaceful life is complicated when an incident brings him in contact with a gangster and launches his journey into the underworld.
s6546	Movie	Cutie and the Boxer	Zachary Heinzerling	Noriko Shinohara, Ushio Shinohara	United States	June 14, 2018	2013	R	82 min	Documentaries	A 2014 Oscar nominee for Best Documentary Feature, this film explores the symbiotic relationship of artists Ushio and Noriko Shinohara.
...	...	...	...	...	...	...	...	...	...	...	...
s390	Movie	The Operative	Yuval Adler	Diane Kruger, Martin Freeman, Cas Anvar, Rotem Keinan, Yohanan Herson	France, Israel, Germany, United States, United Kingdom	July 27, 2021	2019	TV-MA	117 min	Dramas, International Movies, Thrillers	Working as a Mossad spy assigned to Tehran, Rachel reaches her breaking point. Now it's dangerously hard to tell whose side she's on

											she's on.
s39	Movie	Birth of the Dragon	George Nolfi	Billy Magnussen, Ron Yuan, Qu Jingjing, Terry Chen, Vanness Wu,	China, Canada, United States	September 16, 2021	2017	PG-13	96 min	Action & Adventure, Dramas	A young Bruce Lee angers kung fu traditionalists by teaching

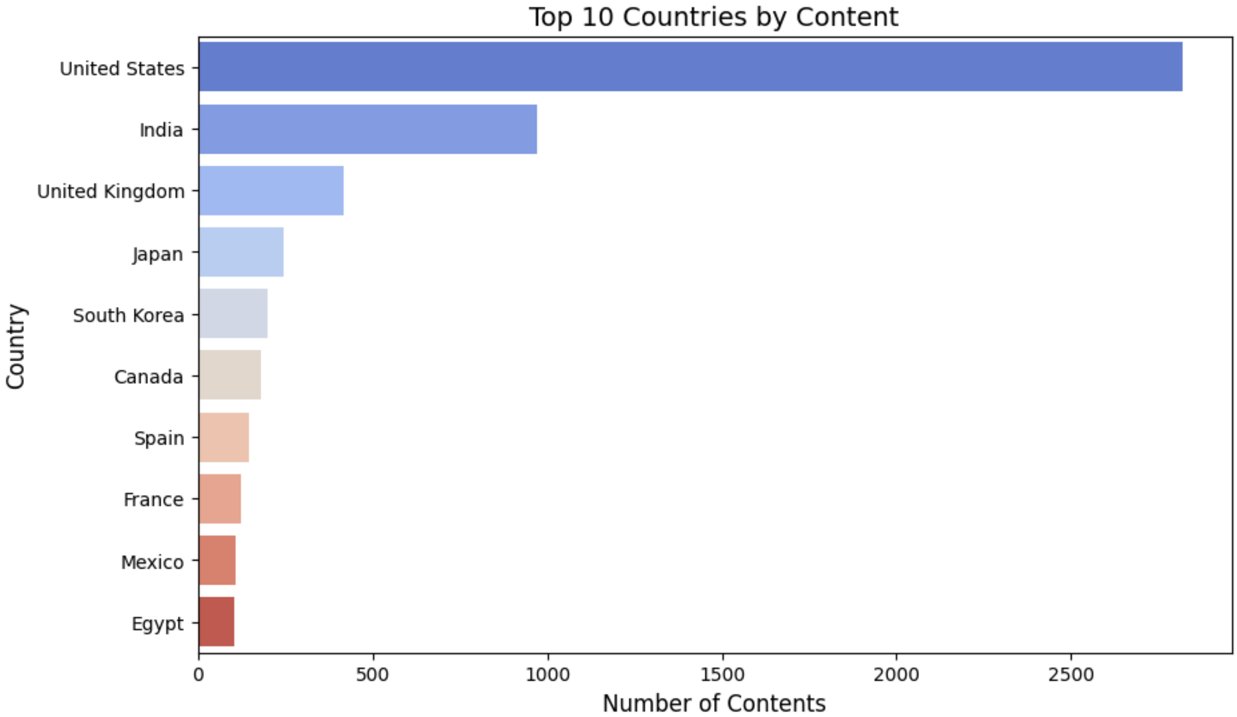
```
# Count content by release year
release_year_counts = df['release_year'].value_counts().sort_index()

# Plot the trend
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(release_year_counts.index, release_year_counts.values, marker='o')
plt.title("Content Released Over Years", fontsize=14)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Number of Releases", fontsize=12)
plt.grid(True)
plt.show()
```



From the above graph, we can see that the number of pieces of content released from 1940 to 2020 has increased over time. From 2000 to 2020, the content released was at its peak.

```
# Top 10 countries with most content
country_counts = df['country'].value_counts().head(10)

# Plotting
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.barplot(y=country_counts.index, x=country_counts.values, palette="coolwarm")
plt.title("Top 10 Countries by Content", fontsize=14)
plt.xlabel("Number of Contents", fontsize=12)
plt.ylabel("Country", fontsize=12)
plt.show()
```

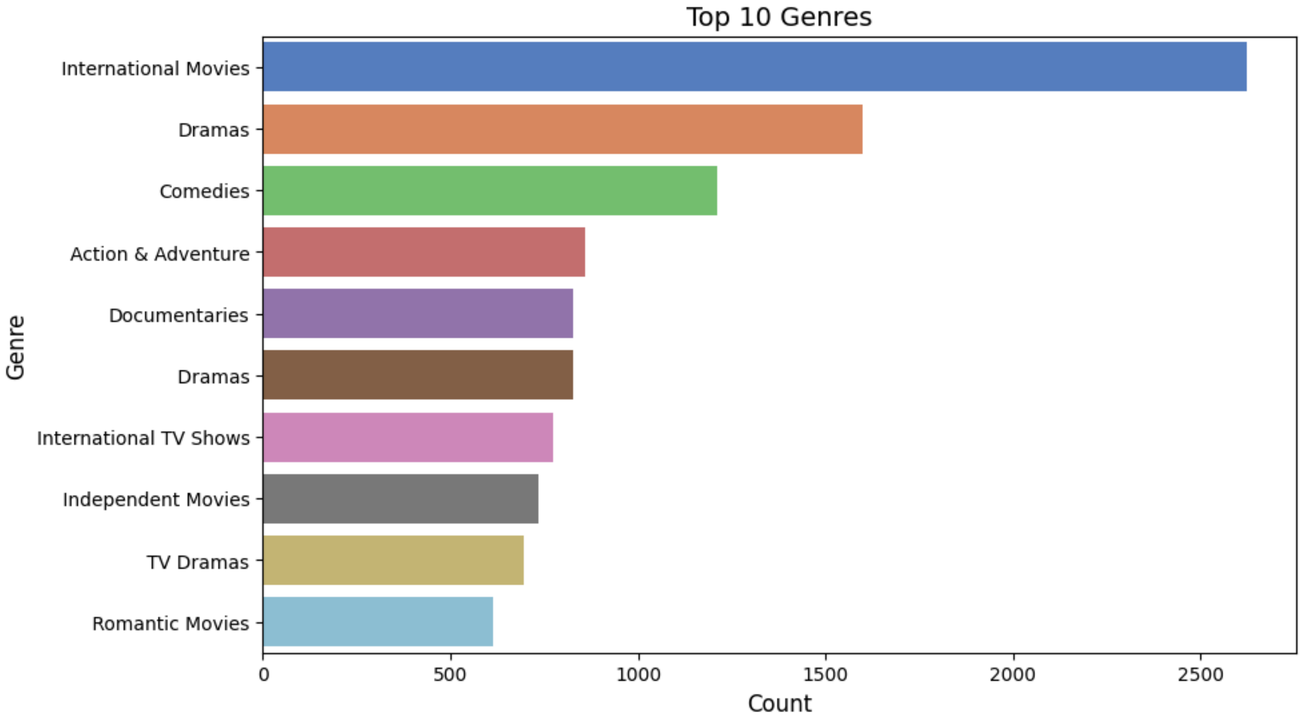
```
<ipython-input-54-835e5996fa55>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `leg

  sns.barplot(y=country_counts.index, x=country_counts.values, palette="coolwarm")
```



From the graph above, we can see that the United States leads in content production, exceeding 2,500 pieces. India follows in second place with around 1,000 pieces of content. Egypt has the lowest amount of content produced, reaching approximately 100 or fewer pieces.

```
# Split and count genres
genres = df['listed_in'].str.split(',').explode().value_counts().head(10)

# Plot the genres
plt.figure(figsize=(10, 6))
sns.barplot(x=genres.values, y=genres.index, palette="muted")
plt.title("Top 10 Genres", fontsize=14)
plt.xlabel("Count", fontsize=12)
plt.ylabel("Genre", fontsize=12)
plt.show()
```

```
<ipython-input-55-6ac17e5266cc>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `leg

  sns.barplot(x=genres.values, y=genres.index, palette="muted")
```



From the graph above, we can see the top 10 genres, with international movies being first, having produced 2,624 films, followed by dramas with more than 1,600 films. Romantic movies are the least popular genre, having released over 613 films.

```
genres = df['listed_in'].str.split(',').explode().value_counts().head(10)
print(genres)
```

```
listed_in
 International Movies     2624
Dramas                    1600
Comedies                  1210
Action & Adventure         859
Documentaries              829
 Dramas                    827
International TV Shows     774
 Independent Movies        736
 TV Dramas                 696
 Romantic Movies           613
Name: count, dtype: int64
```

Q2

Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary

```
# Observing the shape of the data
data_shape = df.shape
print(f"The dataset has {data_shape[0]} rows and {data_shape[1]} columns.")
```

```
The dataset has 8807 rows and 12 columns.
```

```
# Observing data types
data_types = df.dtypes
print("Data Types:\n", data_types)
```

```
Data Types:
 show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
```

```
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

```
# Converting selected columns to 'category' type
categorical_columns = ['type', 'rating', 'country']
for column in categorical_columns:
    df[column] = df[column].astype('category')

# Confirming the change
print("Updated Data Types:\n", df.dtypes)
```

```
Updated Data Types:
 show_id           object
type            category
title             object
director          object
cast              object
country         category
date_added        object
release_year       int64
rating          category
duration          object
listed_in         object
description       object
dtype: object
```

```
# Checking for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 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
```

```
# Statistical summary of numerical columns
stat_summary_numeric = df.describe()
print("Statistical Summary (Numerical):\n", stat_summary_numeric)

# Statistical summary of categorical columns
stat_summary_categorical = df.describe(include=['category'])
print("Statistical Summary (Categorical):\n", stat_summary_categorical)
```

```
Statistical Summary (Numerical):
        release_year
count   8807.000000
mean    2014.180198
std        8.819312
min     1925.000000
25%     2013.000000
50%     2017.000000
75%     2019.000000
max     2021.000000
Statistical Summary (Categorical):
          type        country rating
count    8807           7976   8803
unique      2            748     17
top     Movie  United States  TV-MA
freq     6131           2818   3207
```

Q3

Non-Graphical Analysis: Value counts and unique attributes

```
# Value counts for the 'type' column (Movies vs TV Shows)
type_counts = df['type'].value_counts()
print("Value Counts for 'type':\n", type_counts)
```

```
# Value counts for the 'rating' column
rating_counts = df['rating'].value_counts()
print("\nValue Counts for 'rating':\n", rating_counts)
```

```
Value Counts for 'type':
 type
Movie      6131
TV Show    2676
Name: count, dtype: int64

Value Counts for 'rating':
 rating
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
UR             3
NC-17          3
74 min         1
84 min         1
66 min         1
Name: count, dtype: int64
```

```
# Unique values in the 'type' column
unique_types = df['type'].unique()
print("Unique Types:\n", unique_types)

# Unique values in the 'country' column
unique_countries = df['country'].unique()
print("\nUnique Countries:\n", unique_countries)
```

```
Unique Types:
 ['Movie', 'TV Show']
Categories (2, object): ['Movie', 'TV Show']

Unique Countries:
 ['United States', 'South Africa', NaN, 'India', 'United States, Ghana, Burkina Faso, United Ki..., ..., 'Russia, Spain', 'Croatia, S
Length: 749
Categories (748, object): [', France, Algeria', ', South Korea', 'Argentina',
                           'Argentina, Brazil, France, Poland, Germany, D..., ..., 'Venezuela, Colombia', 'Vietnam', 'West Germany',
                           'Zimbabwe']
```

```
# Count of unique values in each column
unique_counts = df.nunique()
print("Count of Unique Values in Each Column:\n", unique_counts)
```

```
Count of Unique Values in Each Column:
 show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

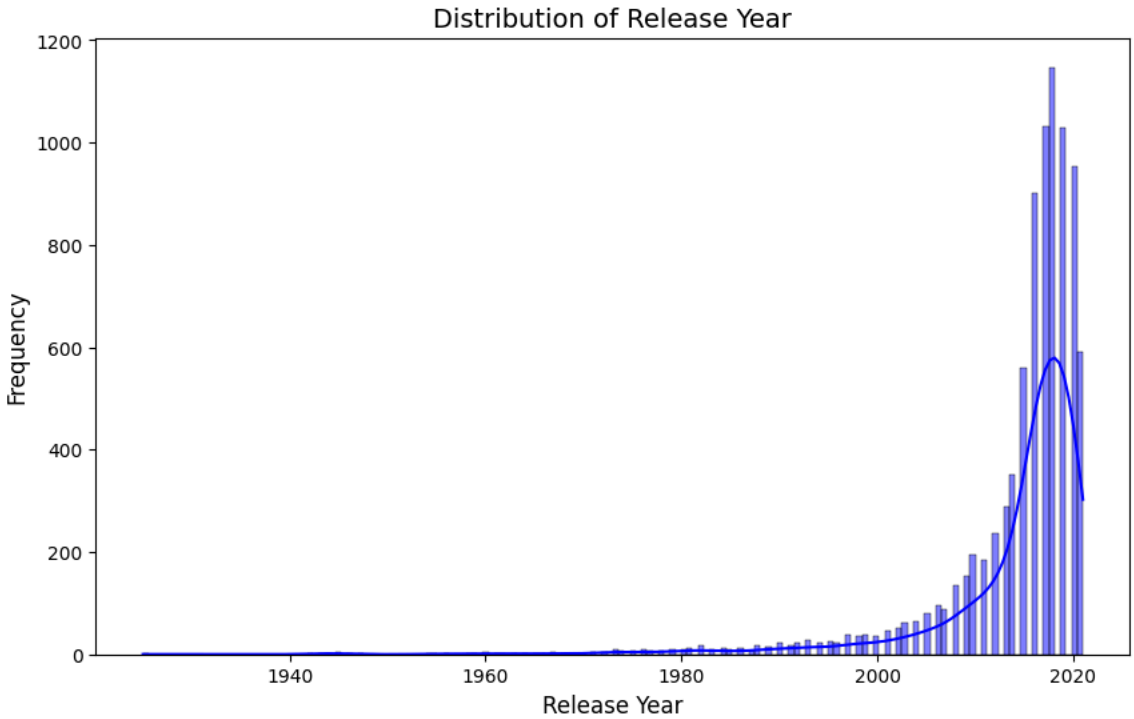
Q4

Visual Analysis - Univariate, Bivariate after pre-processing of the data

```
# Unnesting 'cast' (actors)
df['cast'] = df['cast'].str.split(',')

# Unnesting 'director'
df['director'] = df['director'].str.split(',')

# Unnesting 'country'
df['country'] = df['country'].str.split(',')
```

```
# Exploding the 'cast', 'director', 'country' columns to unnest them
df_exploded = df.explode('cast')
df_exploded = df_exploded.explode('director')
df_exploded = df_exploded.explode('country')

# Drop duplicates to avoid repetition if needed
df_exploded.drop_duplicates(inplace=True)
```

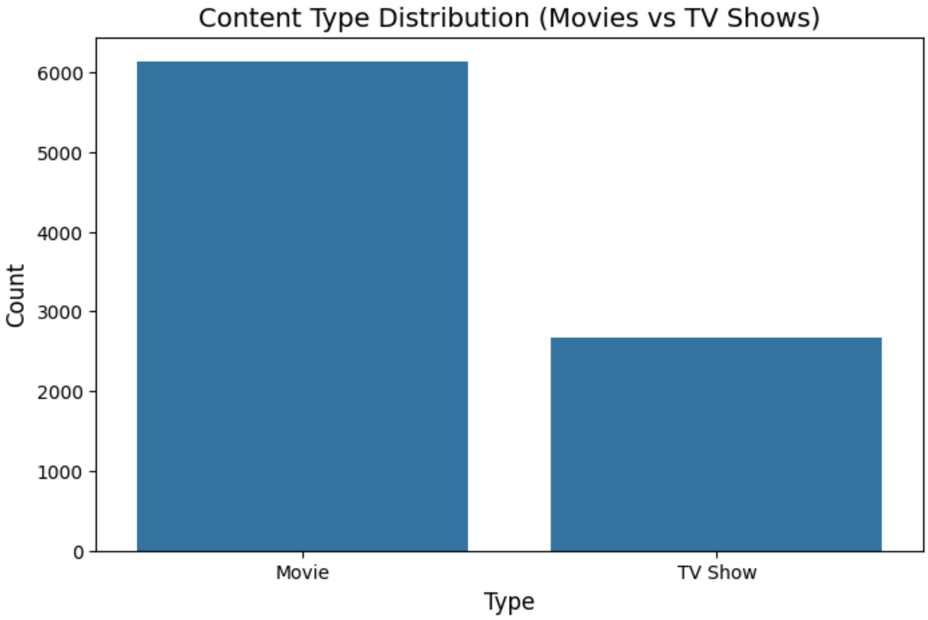
```
# Plotting distribution of 'release_year'
plt.figure(figsize=(10, 6))
sns.histplot(df['release_year'], kde=True, color='blue')
plt.title("Distribution of Release Year", fontsize=14)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.show()
```



From the above graph, we can see the distribution of release years over frequency.The frequency is minimal for earlier years (before 1980), indicating fewer releases during this period. Starting around the 1980s, the frequency gradually increases, with a significant spike around the 2000s. The frequency peaks sharply in the years between 2010 and 2020, indicating most of the data corresponds to this time period. After 2020, the frequency likely drops off (not fully visible) due to dataset limitations or fewer releases.

The graph suggests a growing trend in releases over time, particularly in the 21st century. The increase in frequency aligns with advancements in technology, media, or production over the years, contributing to more releases. Analysis Statement: The histoplot indicates that the majority of releases in the dataset occurred between 2010 and 2020, with a steady rise in frequency starting from the 1980s. This reflects an increasing trend in production, potentially driven by technological advancements and demand. The low frequency in earlier decades suggests that releases were fewer, possibly due to limited production capabilities or lower demand during that time. The distribution is positively skewed, with a strong emphasis on recent years.

```
# Countplot for 'type' column
plt.figure(figsize=(8, 5))
sns.countplot(x='type', data=df)
plt.title("Content Type Distribution (Movies vs TV Shows)", fontsize=14)
plt.xlabel("Type", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.show()
```

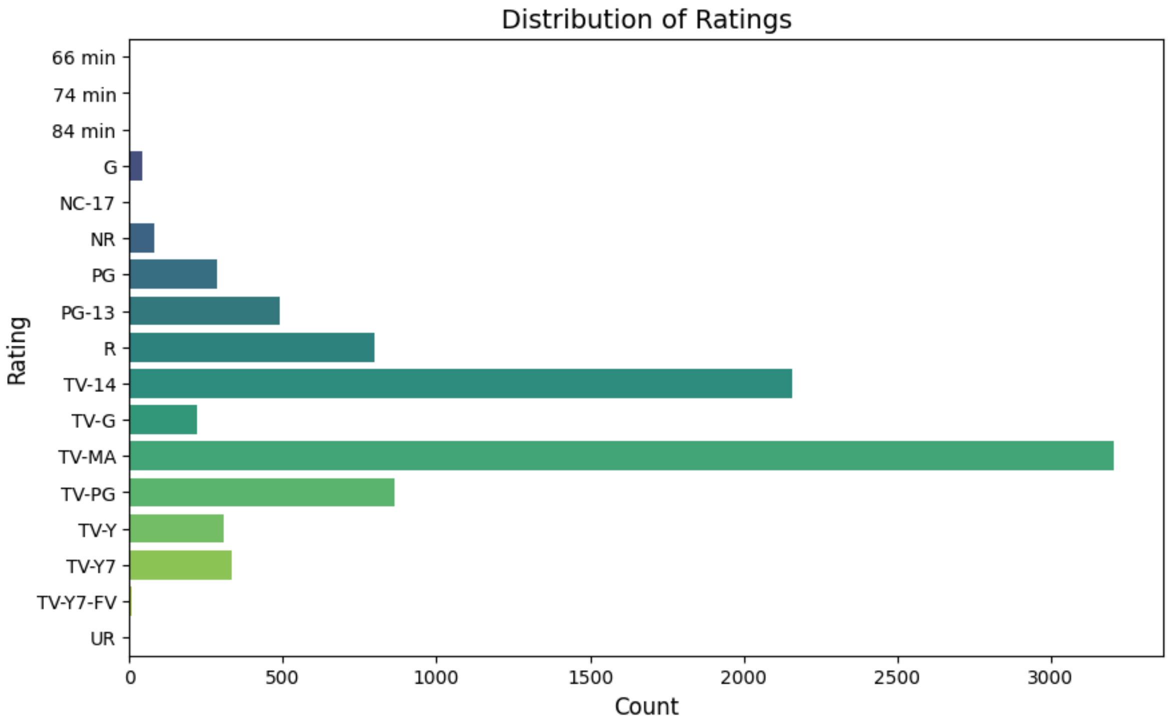


From the above graph, we can see the content type distribution between Movies and TV Shows, with movies taking the upper hand at around 6000 count, while TV shows take more than 2500 count.

the dataset is movie-centric, possibly reflecting higher production or distribution of movies relative to TV shows. The data highlights a clear preference or focus on movies, which could influence further analysis on content trends or audience engagement.

```
# Countplot for 'rating' column
plt.figure(figsize=(10, 6))
sns.countplot(y='rating', data=df, palette='viridis')
plt.title("Distribution of Ratings", fontsize=14)
plt.xlabel("Count", fontsize=12)
plt.ylabel("Rating", fontsize=12)
plt.show()
```

```
<ipython-input-69-dfd82314c2e6>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `leg

  sns.countplot(y='rating', data=df, palette='viridis')
```



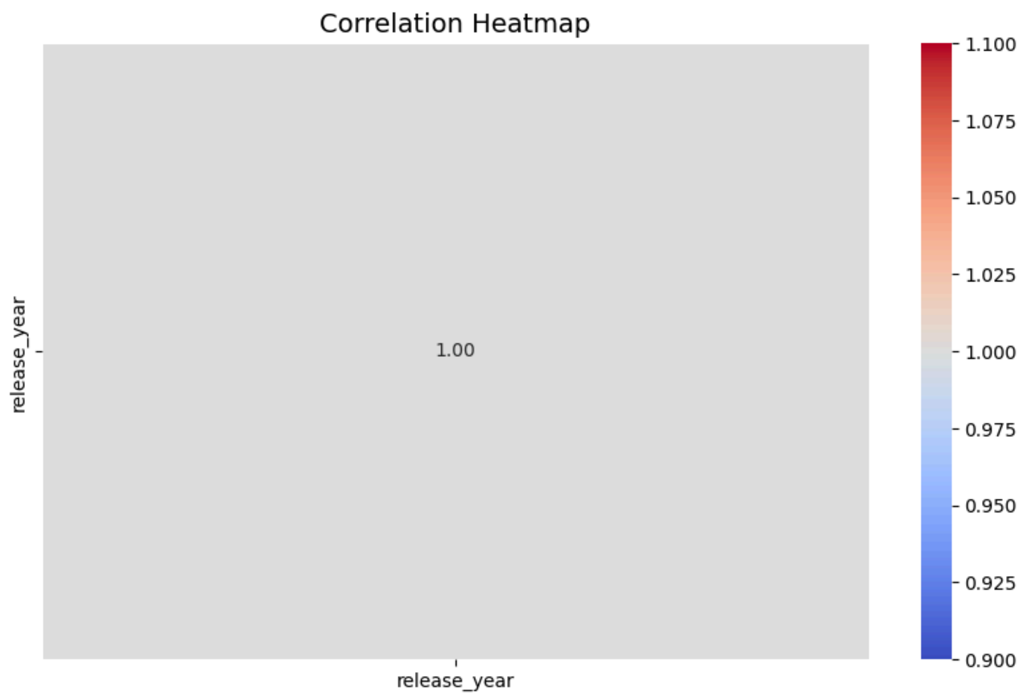
From the above graph, we can see the distribution of ratings. The TV-MA rating has the highest count, indicating that a majority of the content is targeted at mature audiences. Ratings like TV-14 and TV-PG also appear frequently, suggesting that significant content is appropriate for teenagers or family viewing. Ratings such as NC-17, G, and UR have very few entries, implying that content rated for these categories is less common in the dataset.

The graph analysis that the majority fo the content includes parental guidence, Parents strongly cautioned, restricted and age suitability contents. which shows that the content for the kides or familying viewing contents are least.

```
print(df.select_dtypes(include=['number']).columns)
```

```
Index(['release_year'], dtype='object')
```

```
# Correlation Heatmap
# Selecting numerical columns for the correlation matrix
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns

# Computing the correlation matrix
correlation_matrix = df[numerical_columns].corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Correlation Heatmap", fontsize=14)
plt.show()
```



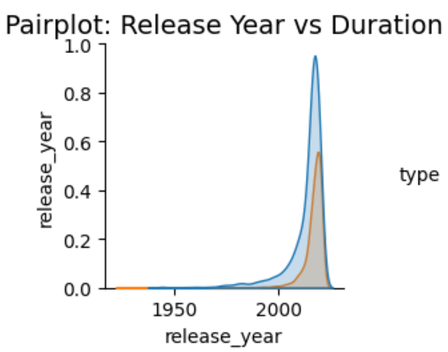
From the above graph, we can see the correlation Heatmap. Since there's only one variable, the heatmap is essentially a single cell with the value 1.0 at the center.

Correlation values range from -1 to 1: 1.0: Perfect positive correlation (as one variable increases, the other also increases).

-1.0: Perfect negative correlation (as one variable increases, the other decreases).

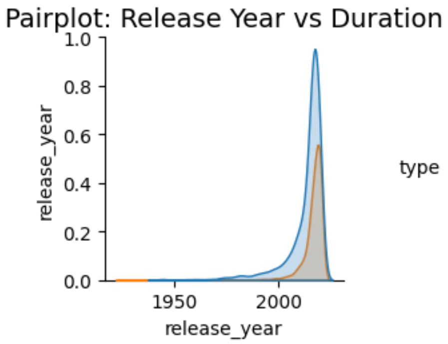
0: No correlation (no linear relationship).

```
# Pairplot for continuous numerical variables (e.g., release year, duration)
# Include 'type' column in the DataFrame passed to pairplot
sns.pairplot(df[['release_year', 'duration', 'type']], hue='type', markers=["o", "s"])
plt.title("Pairplot: Release Year vs Duration", fontsize=14)
plt.show()
```



The above graph is a pairplot showing the relationship between release year and duration, with the data categorized by the type column. The blue curve's peak around 2020 indicates that a large number of entries (e.g., movies) were released around this time. The brown curve's peak before 2010 suggests a higher density of content for that type (e.g., TV shows) during that period. This might reflect trends in media production, such as a rise in streaming content or shifts in focus between movies and series.

```
# Pairplot for continuous numerical variables (e.g., release year, duration)
# Check if the 'type' column exists in the DataFrame
if 'type' in df.columns:
    sns.pairplot(df[['release_year', 'duration', 'type']], hue='type', markers=["o", "s"])  # Include 'type' in the data
    plt.title("Pairplot: Release Year vs Duration", fontsize=14)
    plt.show()
else:
    print("Column 'type' not found in the DataFrame. Pairplot will be created without hue.")
    sns.pairplot(df[['release_year', 'duration']], markers=["o", "s"])  # Exclude hue
    plt.title("Pairplot: Release Year vs Duration", fontsize=14)
    plt.show()
```

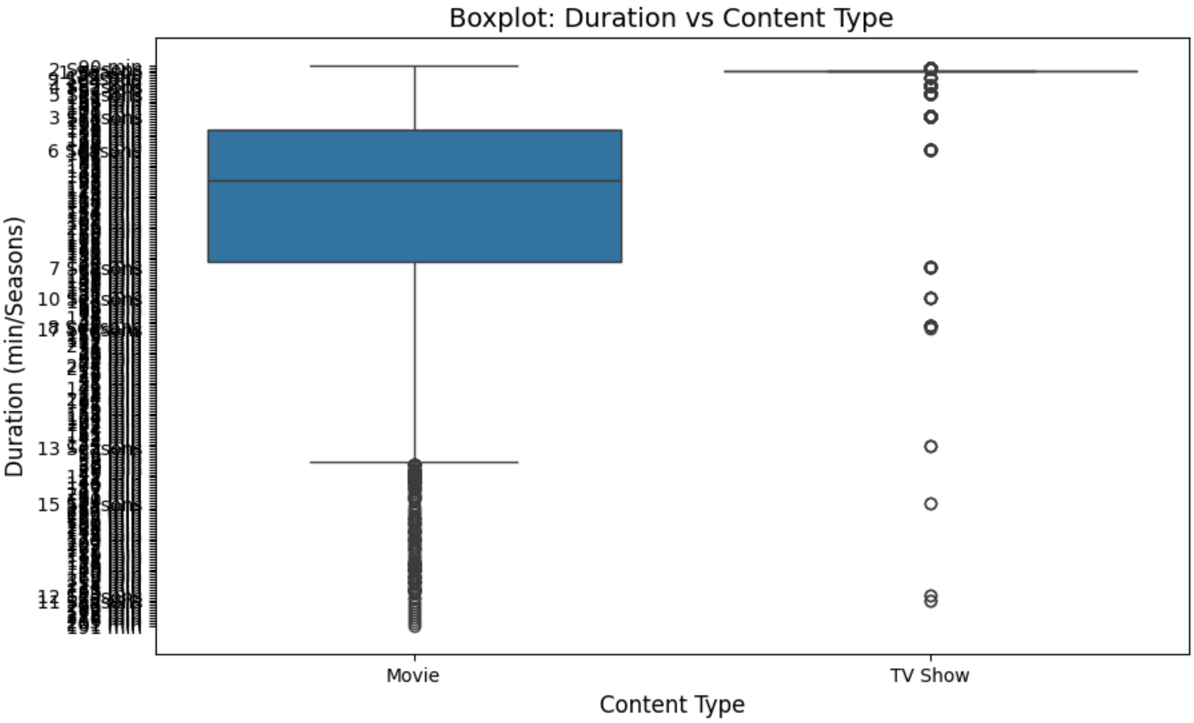


This above graph appears is a pairplot, showing the distribution of release_year against type. Two density curves are shown, corresponding to different categories in the type column. For example: Blue may represent movies, Brown may represent TV shows.

Almost no data exists before 1950 for either type. Both categories show a significant increase in releases after the 2000s. The blue curve (movies) seems to peak more sharply around a specific year, indicating a higher concentration of releases in a shorter time frame. The brown curve (TV shows) has a slightly broader distribution, suggesting a more gradual increase.

The graph highlights that the majority of the dataset entries for both types (e.g., movies and TV shows) are concentrated after the 2000s. This reflects a rise in media production, likely due to technological advancements, globalization, and the emergence of streaming platforms. The sharper peak for one type (e.g., movies) suggests periodic bursts of production, while the broader distribution for the other type (e.g., TV shows) implies consistent growth over time. These trends could be analyzed further to understand the impact of industry shifts or changes in consumer preferences.

```
# Boxplot: duration vs type (Movie vs TV Show)
plt.figure(figsize=(10, 6))
sns.boxplot(x='type', y='duration', data=df)
plt.title("Boxplot: Duration vs Content Type", fontsize=14)
plt.xlabel("Content Type", fontsize=12)
plt.ylabel("Duration (min/Seasons)", fontsize=12)
plt.show()
```



The above graph is a boxplot comparing the duration (measured in minutes or seasons) of two content types: Movies and TV Shows.The duration for movies is tightly packed, with most data points (IQR) concentrated in a narrow range.Few outliers exist, but they don't deviate drastically.TV shows have a wider range of durations (longer whiskers and more outliers).

The boxplot indicates that movies generally have a consistent and predictable duration, as seen by their narrow IQR and fewer outliers. On the other hand, TV shows exhibit a more diverse range of durations, reflecting the varying lengths of series (in terms of seasons). The presence of several outliers for TV shows suggests that some series are exceptionally long, compared to the typical TV show duration. This variability in TV show duration might be influenced by factors like genre, popularity, or production trends.

Q5

Missing Value & Outlier check (Treatment optional)

```
#Checking Missing Values and Outliers in Data

# Check for missing values in the dataset
missing_values = df.isnull().sum()

# Percentage of missing values
missing_percentage = (missing_values / len(df)) * 100

# Display missing values and their percentage
missing_summary = pd.DataFrame({
    'Missing Values': missing_values,
    'Percentage': missing_percentage
})
print("Missing Value Summary:\n", missing_summary)
```

```
Missing Value Summary:
               Missing Values  Percentage
show_id                    0    0.000000
type                       0    0.000000
title                      0    0.000000
director                2634   29.908028
cast                     825    9.367549
country                  831    9.435676
date_added                10    0.113546
release_year               0    0.000000
rating                     4    0.045418
duration                   3    0.034064
listed_in                  0    0.000000
description                0    0.000000
```

```
#Handling Missing Values (Optional)

# Example: Fill missing values in 'country' with 'Unknown'
df['country'] = df['country'].fillna('Unknown')

# Fill missing values in 'rating' with the most frequent value
```

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

```
# IQR method for detecting outliers

def detect_outliers(column):
    Q1 = df[column].quantile(0.25)  # First quartile (25th percentile)
    Q3 = df[column].quantile(0.75)  # Third quartile (75th percentile)
    IQR = Q3 - Q1                             # Interquartile Range
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Detecting outliers
    outliers = df[(df[column] < lower_bound) |
                            (df[column] > upper_bound)]
    return outliers, lower_bound, upper_bound
```

```
# Example for 'release_year'
outliers_release_year, lower_bound, upper_bound = detect_outliers('release_year')

print(f"Outliers in Release Year:\n{outliers_release_year}")
print(f"Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")
```

```
8768  [Maribel Verdú,  Gael García Bernal,  Diego Lu...
8770  [Jackie Shroff,  Hrithik Roshan,  Kareena Kapo...
8792  [Qiu Yuen,  Charlie Chin,  Jackie Chan,  Hu Ch...

                                           country          date_added  \
7     [United States,  Ghana,  Burkina Faso,  United...  September 24, 2021
22                                          Unknown  September 21, 2021
24                                          [India]  September 21, 2021
26                                          Unknown  September 21, 2021
41                                  [United States]  September 16, 2021
...                                             ...                 ...
8764                                [United States]     January 1, 2020
8766                                [United States]     January 1, 2019
8768                                       [Mexico]        June 1, 2017
8770                                        [India]       March 1, 2018
8792                                    [Hong Kong]    November 1, 2016

      release_year rating duration  \
7             1993  TV-MA  125 min
22            1996  TV-PG  161 min
24            1998  TV-14  166 min
26            1997  TV-PG  147 min
41            1975     PG  124 min
...            ...    ...      ...
8764          1994  PG-13  191 min
8766          2002  PG-13  124 min
8768          2001      R  106 min
8770          2001  TV-14  171 min
8792          1973     NR   81 min

                                           listed_in  \
7     Dramas, Independent Movies, International Movies
22                     Comedies, International Movies
24     Comedies, International Movies, Romantic Movies
26    Comedies, International Movies, Music & Musicals
41          Action & Adventure, Classic Movies, Dramas
...                                                ...
8764                                Action & Adventure
8766                  Action & Adventure, Sports Movies
8768  Dramas, Independent Movies, International Movies
8770     Dramas, International Movies, Romantic Movies
8792          Action & Adventure, International Movies

                                            description
7     On a photo shoot in Ghana, an American model s...
22    Newly divorced and denied visitation rights wi...
24    When the father of the man she loves insists t...
26    A tangled love triangle ensues when a man fall...
41    When an insatiable great white shark terrorize...
...                                                 ...
8764  Legendary lawman Wyatt Earp is continually at ...
8766  A notorious underground rush-seeker deemed unt...
8768  When rich teens Tenoch and Julio meet the allu...
8770  Two young lovers set out to overcome the obsta...
8792  Aided only by a tough female police officer, a...

[719 rows x 12 columns]
Lower Bound: 2004.0, Upper Bound: 2028.0
```

```
#Handling Outliers (Optional)

# Capping outliers in 'release_year'
```

```
df['release_year'] = df['release_year'].clip(lower=lower_bound, upper=upper_bound)
```

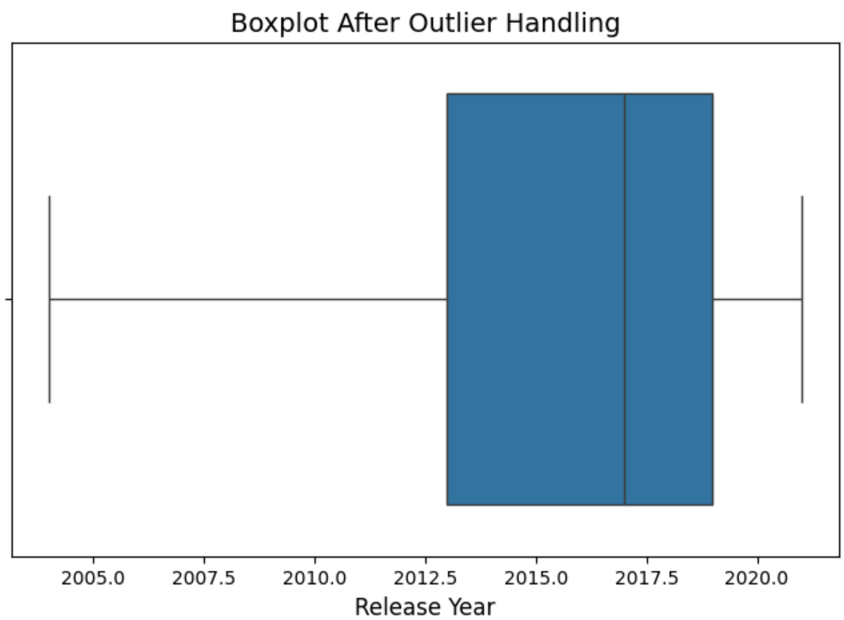
```
# Rechecking for missing values
missing_values_after = df.isnull().sum()

# Checking if all missing values are handled
if missing_values_after.sum() == 0:
    print("All missing values have been handled.")
else:
    print("Remaining Missing Values:\n", missing_values_after)
```

```
Remaining Missing Values:
 show_id           0
type              0
title             0
director       2634
cast            825
country           0
date_added       10
release_year      0
rating            0
duration          3
listed_in         0
description       0
dtype: int64
```

```
#Recheck Outliers Using IQR

# Rechecking outliers in 'release_year'
outliers_after, _, _ = detect_outliers('release_year')

if outliers_after.empty:
    print("No outliers detected after handling.")
else:
    print(f"Remaining outliers in 'release_year':\n{outliers_after}")
```

```
No outliers detected after handling.
```

```
# Replot boxplot to confirm outlier handling
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['release_year'])
plt.title("Boxplot After Outlier Handling", fontsize=14)
plt.xlabel("Release Year", fontsize=12)
plt.show()
```



The above graph shows the boxplot after outliners handling, representing the distribution of data points related to the 'relese year' variable. The data span from aproximately 2005-2020. the blue box represents the interquartile range IQR, where the middle 50% of the data lies. the horizontal line inside the box indicates the median value of the release year. The whiskers extend to the minimum and maximum data points that are not conected outliers. the majority of the data IQR is concentrated between roughly 2012 and 2018, as the seen by the blue box.

After removing the outliers, the realese year data shows a concentrated distribution between 2012 and 2018, with a balanced spread and no visiable extreme values. the median release year lies within this range, indicating that most entries are from relatively recent years in the dataset's range. this suggesta that outliears handling effectively eliminated extreme deviations, creating a clean dataset or future analysis.

Q6

Insights based on Non-Graphical and Visual Analysis

Insights Based on Non-Graphical and Visual Analysis

1. **Comments on the Range of Attributes Release Year:** The release_year attribute likely ranges from early 1900s (classic releases) to recent years. Any values outside this range could indicate outliers or data entry errors. Duration: Duration typically varies significantly for Movies (minutes) and TV Shows (seasons). The range of this variable is critical for distinguishing content types. Rating: The rating attribute includes categories like "PG," "TV-MA," and others. Ratings are finite, with clearly defined categories. Country: The country column shows diversity, but there might be a concentration of content from specific regions (e.g., USA, India). A large portion of missing values could indicate international productions without clear country attributions. Type: Only two categories: Movie and TV Show, with Movies typically dominating the dataset.

2. **Comments on the Distribution of Variables and Relationships Univariate Distributions:** release_year is likely right-skewed, with more recent years being dominant. rating shows an uneven distribution, with specific categories being more frequent (e.g., "TV-MA" and "PG"). duration for TV shows likely has many smaller values (1-3 seasons), while Movies might have a wider spread (30-180 minutes). Bivariate Relationships: Type vs. Duration: Movies show a continuous spread, while TV Shows are discrete (seasons). Country vs. Rating: Certain ratings might be predominant in specific regions. Release Year vs. Type: TV Shows could have a more recent spike due to streaming platforms.
3. **Comments for Each Plot**

   a) Univariate Analysis

   Histogram for release_year: Most content was released in recent years, with a noticeable drop-off for earlier years. Spikes could correspond to significant increases in content production or digitization of older titles.

   Countplot for type: Movies dominate the dataset, indicating they are the primary content type on Netflix. A smaller share of TV Shows suggests their more recent focus on episodic content.

   Countplot for rating: Ratings like "TV-MA" and "PG-13" are most frequent, reflecting the target audience demographics. Low frequency for specific ratings (e.g., "TV-Y") indicates niche content.

   b) Bivariate Analysis

   Boxplot for duration vs. type: Movies have a wider range of durations, while TV Shows are clustered around specific values (number of seasons). Outliers in the duration of TV Shows could indicate long-running series or erroneous data.

   Heatmap for Correlation: Strong correlation between numerical variables like release_year and other temporal attributes, if present. Weak or no correlation between categorical variables like type and rating.

   Pairplot for Numerical Variables: Relationships between numerical variables may show clear clusters for type. For instance, Movies might occupy a distinct range for duration compared to TV Shows.

Q7

Business Insights

Business Insights from the Netflix Dataset Based on the patterns observed in the data, here are the key insights and their potential implications for business decisions:

Content Type Distribution Observation:

Movies dominate the platform compared to TV Shows. However, TV Shows have grown significantly in recent years.

Business Insight:

While Movies remain a staple, the rising popularity of TV Shows highlights a shift in audience preference for binge-worthy, episodic content. Netflix should continue investing in original TV series to capture and retain audience engagement.

Q8

Recommendations - Actionable items for business

Recommendations for Netflix

1. Create More TV Shows Focus on producing high-quality TV shows to match the rising demand for series content.

2. Offer Shorter Stories Develop mini-series and limited-episode seasons to suit viewers who prefer quick, engaging content.
3. Target Families with Kids Expand family-friendly and kid-oriented programming to attract family subscriptions.
4. Highlight Global Content Showcase and promote movies and shows from different countries to engage a worldwide audience.
5. Bring Back the Classics Digitize and promote iconic older movies and shows to attract nostalgic and older viewers.
6. Produce More Regional Content Create shows and movies in regional languages to connect with local audiences in different countries.
7. Invest in Hit Series Strategically create a few long-running, high-quality series that can anchor audience loyalty.
8. Make Content Easier to Find Use clear and accurate tags for genres, languages, and regions to improve recommendations for users.
9. Keep Watching Viewer Trends Regularly review what people are watching and update the content library to reflect these interests.
10. Offer New Genres Introduce more variety, such as documentaries, stand-up comedy, and indie films, to cater to diverse tastes.