Submitting By: Mansi Sinha

About Netflix:

- Netflix is one of the most popular media and video streaming platforms.
- 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:

- Analyze the data and generate insights that could help Netflix in deciding which type of shows/movies to produce.
- How they can grow the business in different countries.

1. Defining Problem Statement and Analysing basic metrics

- How to Build & Requirements
- Import Python Libraries
- Data Understanding

How to Build & Requirements:

- We will perform data preprocessing and feature engineering on the dataset to handle missing values and create new features.
- Further, we will apply various descriptive statistics and data visualization techniques to identify underlying patterns and derive main insights.
- We will be using these libraries, tools, and modules in this buisness case - Pandas, Numpy, Matplotlib, Seaborn

Import Libraries:

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

Data Understanding:

- Load the dataset in a pandas dataframe and explore variables and their data types.

Insight 1:

- The dataset provided to you consists of a list of all the TV shows/movies available on Netflix:

1. Show_id: Unique ID for every Movie / Tv Show
2. Type: Identifier - A Movie or TV Show
3. Title: Title of the Movie / Tv Show
4. Director: Director of the Movie
5. Cast: Actors involved in the movie/show
6. Country: Country where the movie/show was produced
7. Date_added: Date it was added on Netflix
8. Release_year: Actual Release year of the movie/show
9. Rating: TV Rating of the movie/show
10. Duration: Total Duration - in minutes or number of seasons

```
### Load Dataset

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

```
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...
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...

Next steps: Generate code with df | View recommended plots

```
#info

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
```

```
# last five row

df.tail()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
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 political cartoonist, a crime reporter 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 living alone in a spooky town, a young g...
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 to survive in a world taken over by zo...
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	Dragged from civilian life, a former superhero...

```
# top 5 rows

df.head(5)
```

	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...

Next steps: Generate code with df   View recommended plots

2. 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.

- Data Information
- Data Cleaning
- Statistical Summary

Data Information

```
#shape: (rows/index/entries, col)

df.shape
```

```
(8807, 12)
```

```
#Columns:

df.keys() #note*: df.columns() this one can also used for extracting columns names!
```

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

```
#Explore variables, their data types, and total non-null values:

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.nunique()

# The function used to generate the output is df.nunique().
# This function calculates the number of unique entries for each column (or Series) in the DataFrame df.
# It returns a Series containing the count of unique values for each column.
# In this case, it provides the number of unique values for each column in the Netflix dataset.
```

```
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
```

```
df.describe()

#The df.describe() function provides a statistical summary of the numerical column release_year in the DataFrame df
```

	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.describe(include = object)
# To generate a statistical summary for columns of object type (such as strings) in the DataFrame df,
# you can use the include='object' parameter in the df.describe() function.
# This will provide summary statistics for all the object-type columns
```

	show_id	type	title	director	cast	country	date_added	rating	duration	listed_in	description
count	8807	8807	8807	6173	7982	7976	8797	8803	8804	8807	8807
unique	8807	2	8807	4528	7692	748	1767	17	220	514	8775
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV-MA	1 Season	Dramas, International Movies	Paranormal activity at a lush, abandoned prope...
freq	1	6131	1	19	19	2818	109	3207	1793	362	4

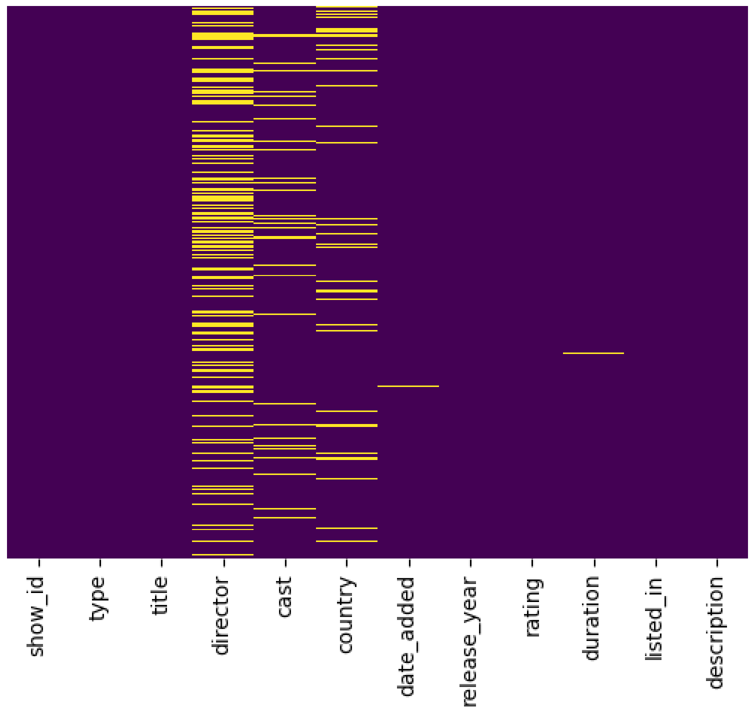
```
# Data Types of each Column

df.dtypes
```

```
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
```

Data Cleaning:

```
# Missing Value :

# count the missing value in each column

df.isna().sum()

     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


# Checking Percentage of Null values persent in each column

for i in df.columns:
  null_rate = (df[i].isna().sum() / len(df))* 100
  if null_rate > 0:
    print(f"{i}'s null rate: {round(null_rate, 2)}%")

     director's null rate: 29.91%
     cast's null rate: 9.37%
     country's null rate: 9.44%
     date_added's null rate: 0.11%
     rating's null rate: 0.05%
     duration's null rate: 0.03%


# plotting the heap map for null values

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap ='viridis')
plt.title('Null Values Check')
plt.savefig(fname = 'NullChecks_boxPlot10.png')
plt.show()
```

Null Values Check



```
# count of null values in each column

df.count()

     show_id         8807
     type            8807
     title           8807
     director        6173
     cast            7982
     country         7976
     date_added      8797
     release_year    8807
     rating          8803
     duration        8804
     listed_in       8807
     description     8807
     dtype: int64


df.isnull().sum()

     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
```

```
# Handling missing Value in these columns
# duration, rating, date_added, country, cast, director
```

```
df[df['duration'].isnull()]  #so we have indexes where duration have null values!!
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
5541	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	April 4, 2017	2017	74 min	NaN	Movies	Louis C.K. muses on religion, eternal love, gi...
5794	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	September 16, 2016	2010	84 min	NaN	Movies	Emmy-winning comedy writer Louis C.K. brings h...

```
index = df[df['duration'].isnull()].index
index
```

```
     Int64Index([5541, 5794, 5813], dtype='int64')
```

```
# Imputating missing values of duration from rating column
df.loc[index, 'duration'] = df.loc[index, 'rating']
```

```
# checking null in duration
df[df['duration'].isna()]
```

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description

```
# replacing wrong values of rating
df.loc[index, 'rating'] = None
```

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

```
     show_id            0
     type               0
     title              0
     director        2634
     cast             825
     country          831
     date_added        10
     release_year       0
     rating             7
     duration           0
     listed_in          0
     description        0
     dtype: int64
```

```
# missing values in columns of director, cast, country
# with Unknown director, cast, country respectively

df.fillna({'director':'unknown director'}, inplace = True)
df.fillna({'cast':'unknown cast'}, inplace = True)
df.fillna({'country':'unknown country'}, inplace = True)


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

```
     show_id          0
     type             0
     title            0
     director         0
     cast             0
     country          0
     date_added      10
     release_year     0
     rating           7
     duration         0
     listed_in        0
     description      0
     dtype: int64
```

```
# as above we can see we have still two columns with null values


df.dropna(subset = ['date_added','rating'], axis=0 ,inplace = True)


#FINAL CHECK OF NULL VALUES

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

```
     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
```

```
#Conversions of Datatypes


#Converting duration column from categorical variable to numerical to show number of seasons for TV shows and minute for movies

def duration(s):
  return int(s.split(" ")[0])

df['duration'] = df['duration'].apply(duration)


# Converting date_added column to show in datetime format

df['date_added'] = pd.to_datetime(df['date_added'].str.strip(), format= '%B %d, %Y')


#Adding columns day, month and year when a particular content was added on the platforms

df['day_added'], df['month_added'], df['year_added'] = df['date_added'].dt.day_name(), df['date_added'].dt.month_name(), df['date_added'].dt.year
df.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	unknown cast	United States	2021-09-25	2020	PG-13	90	Documentaries	As her father nears the end of his life, filmm...	Saturday	September	2021
					Ama										

Next steps: Generate code with df View recommended plots

Statistical summary

```
# Mean runtime for different types of content.
# Obsevation -> For movie mean duration is approx 100 minutes and
#            -> for TV show it is close to 2 seasons.

df.groupby('type')['duration']. mean()
```

```
type
Movie      99.584884
```

```
TV Show     1.751877
Name: duration, dtype: float64
```

```
# Median runtime for different types of content.
# Obsevation - > For movie mean duration is approx 98 minutes and for TV show it is 1 season.
# We can see the difference b/w mean and median for TV shows is significant.

df.groupby('type')[ 'duration' ].median()
```

```
type
Movie      98.0
TV Show     1.0
Name: duration, dtype: float64
```

```
# Longest movie (Considering duration) - > 312 minutes

movie_data = df[df['type'] == 'Movie']
movie_data[movie_data['duration'] == movie_data['duration']. max()]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
					Fionn Whitehead.						Dramas, International	In 1984, a young			

```
# Shortest movie (Considering duration) - > 3 minutes

movie_data[movie_data['duration']== movie_data['duration'].min()]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
3777	s3778	Movie	Silent	Limbert Fabian, Brandon	unknown cast	United States	2019-06-04	2014	TV-Y	3	Children & Family Movies, Sci-Fi	"Silent" is an animated short	Tuesday	June	2019

```
# Longest TV show

tv_data = df[df['type'] == 'TV Show']
tv_data[tv_data['duration'] == tv_data['duration'].max()]
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
548	s549	TV	Grey's	unknown	Ellen Pompeo, Sandra Oh,	United	2021-07-03	2020	TV-14	17	Romantic TV Shows, TV	Intern (and eventual resident)	Saturday	July	2021

```
# Statistical summary for data for movies

df[df['type'] == 'Movie'].describe()
```

	release_year	duration	year_added
count	6126.000000	6126.000000	6126.000000
mean	2013.120144	99.584884	2018.851126
std	9.681723	28.283225	1.561173
min	1942.000000	3.000000	2008.000000
25%	2012.000000	87.000000	2018.000000
50%	2016.000000	98.000000	2019.000000
75%	2018.000000	114.000000	2020.000000
max	2021.000000	312.000000	2021.000000

```
# Statistical summary for data for TV Shows

df[df['type'] == 'TV Shows'].describe()
```

	release_year	duration	year_added
count	0.0	0.0	0.0
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

3. Non-Graphical Analysis: Value counts and unique attributes

- Non-graphical analysis refers to the examination and interpretation of data without relying on visual representations such as charts or graphs.
- It involves using descriptive statistics, tables, and summary measures to understand the characteristics of the data.

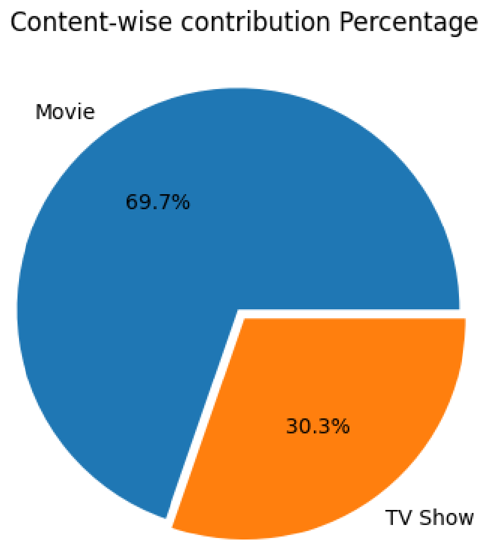
```
#Value counts with respective of each column

df.columns.value_counts()
```

```
     show_id         1
     type            1
     title           1
     director        1
     cast            1
     country         1
     date_added      1
     release_year    1
     rating          1
     duration        1
     listed_in       1
     description     1
     day_added       1
     month_added     1
     year_added      1
     dtype: int64
```

```
# Count of different types of shows present on platform.
# Observation - > Content on platform are either Movie or TV show.
show_type = df['type'].value_counts()
show_type
```

```
     Movie      6126
     TV Show    2664
     Name: type, dtype: int64
```

```
# Below pie chart shows the contentwise percentage.
# Observations - >
# Movies contribute to 69. 7% of the content on platform and 30.3% are TV shows.

plt.pie(show_type, labels = show_type.index, autopct= '%1.1f%% ', explode= [0.05, 0])
plt.title('Content-wise contribution Percentage')
plt.savefig(fname= 'piechart_content-wise37.png')
plt.show()
```



```
# Count of contents rating wise

df['rating'].value_counts()

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


# Ratings available for each content type and count for each rating.

df.groupby('type')['rating'].value_counts()

     type      rating
     Movie     TV-MA       2062
               TV-14       1427
               R            797
```

```
          TV-PG        540
          PG-13        490
          PG           287
          TV-Y7        139
          TV-Y         131
          TV-G         126
          NR            75
          G             41
          TV-Y7-FV       5
          NC-17          3
          UR             3
 TV Show  TV-MA       1143
          TV-14        730
          TV-PG        321
          TV-Y7        194
          TV-Y         175
          TV-G          94
          NR             4
          R              2
          TV-Y7-FV       1
 Name: rating, dtype: int64
```

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

```
 2018    1146
 2017    1030
 2019    1030
 2020     953
 2016     901
         ...
 1959       1
 1925       1
 1961       1
 1947       1
 1966       1
 Name: release_year, Length: 74, dtype: int64
```

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

```
 type     release_year
 Movie    2018            767
          2017            765
          2016            658
          2019            633
          2020            517
                         ...
 TV Show  1979              1
          1981              1
          1985              1
          1989              1
          1991              1
 Name: release_year, Length: 119, dtype: int64
```

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

```
type     month_added
Movie    July           565
         April          549
         December       547
         January        545
         October        545
         March          528
         August         518
         September      518
         November       498
         June           492
         May            439
         February       382
TV Show  December       265
         July           262
         September      251
         August         236
         June           236
         October        215
         April          214
         March          213
         November       207
         May            193
         January        192
         February       180
Name: month_added, dtype: int64
```

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

```
type     day_added
Movie    Friday       1565
         Thursday     1052
         Wednesday     905
         Tuesday       851
         Monday        627
         Sunday        569
         Saturday      557
TV Show  Friday        932
         Wednesday     382
         Tuesday       345
         Thursday      341
         Saturday      259
         Monday        223
         Sunday        182
Name: day_added, dtype: int64
```

```
df.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	unknown cast	United States	2021-09-25	2020	PG-13	90	Documentaries	As her father nears the end of his life, filmm...	Saturday	September	2021
					Ama										

Next steps: Generate code with df   View recommended plots

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

- Note: Pre-processing involves unnesting of the data in columns like Actor, Director, Country
- For continuous variable(s): Distplot, countplot, histogram for univariate analysis
- For categorical variable(s): Boxplot
- For correlation: Heatmaps, Pairplots

```
df.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
1	s2	TV Show	Blood & Water	unknown director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...	Friday	September	2021

Next steps: Generate code with df   View recommended plots

```
# Unnesting listed_in

df = df.assign(listed_in=df.listed_in.str.split(', '))
df = df.explode('listed_in',ignore_index=True)
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s2	TV Show	Blood & Water	unknown director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
1	s2	TV Show	Blood & Water	unknown director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2	TV Dramas	After crossing paths at a party, a Cape Town t...	Friday	September	2021
2	s2	TV Show	Blood & Water	unknown director	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	2	TV Mysteries	After crossing paths at a party, a Cape Town t...	Friday	September	2021

Next steps: Generate code with df View recommended plots

```
# Unnesting Cast field
df = df.assign(cast=df.cast.str.split(', '))
df = df.explode('cast', ignore_index=True)
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s2	TV Show	Blood & Water	unknown director	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
1	s2	TV Show	Blood & Water	unknown director	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
2	s2	TV	Blood &	unknown	Gail	South	2021-09-24	2021	TV-MA	2	International	After crossing paths at a party, a	Friday	September	2021

```
# Unnesting country

df = df.assign(country=df.country.str.split(', '))
df = df.explode('country', ignore_index=True)
df.head()
```

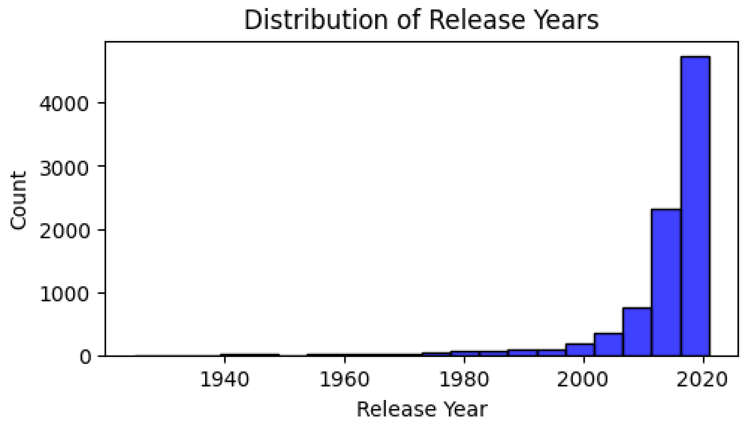
	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s2	TV Show	Blood & Water	unknown director	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
1	s2	TV Show	Blood & Water	unknown director	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
2	s2	TV	Blood &	unknown	Gail	South	2021-09-24	2021	TV-MA	2	International	After crossing paths at a party, a	Friday	September	2021

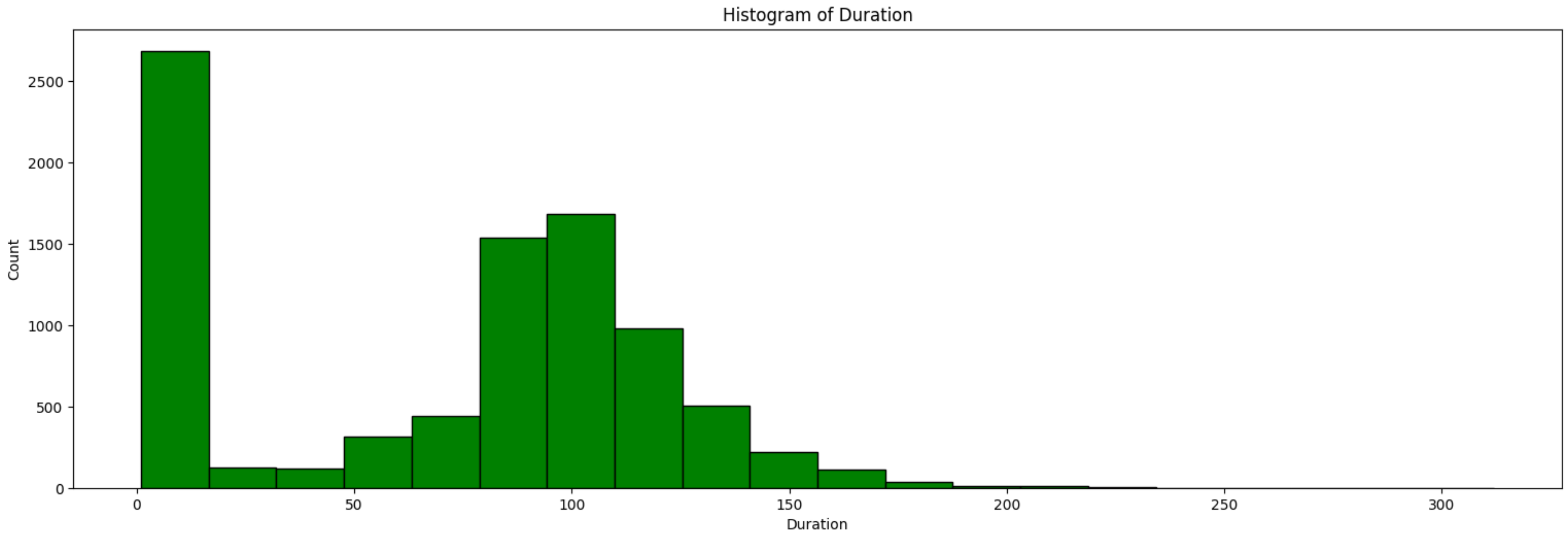
```
# Unnesting director
df = df.assign(director=df.director.str.split(', '))
df = df.explode('director', ignore_index=True)
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description	day_added	month_added	year_added
0	s2	TV Show	Blood & Water	unknown director	Ama Qamata	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
1	s2	TV Show	Blood & Water	unknown director	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA	2	International TV Shows	After crossing paths at a party, a Cape Town t...	Friday	September	2021
2	s2	TV	Blood &	unknown	Gail	South	2021-09-24	2021	TV-MA	2	International	After crossing paths at a party, a	Friday	September	2021

```
assert 'release_year' in df.columns
assert 'duration' in df.columns
```

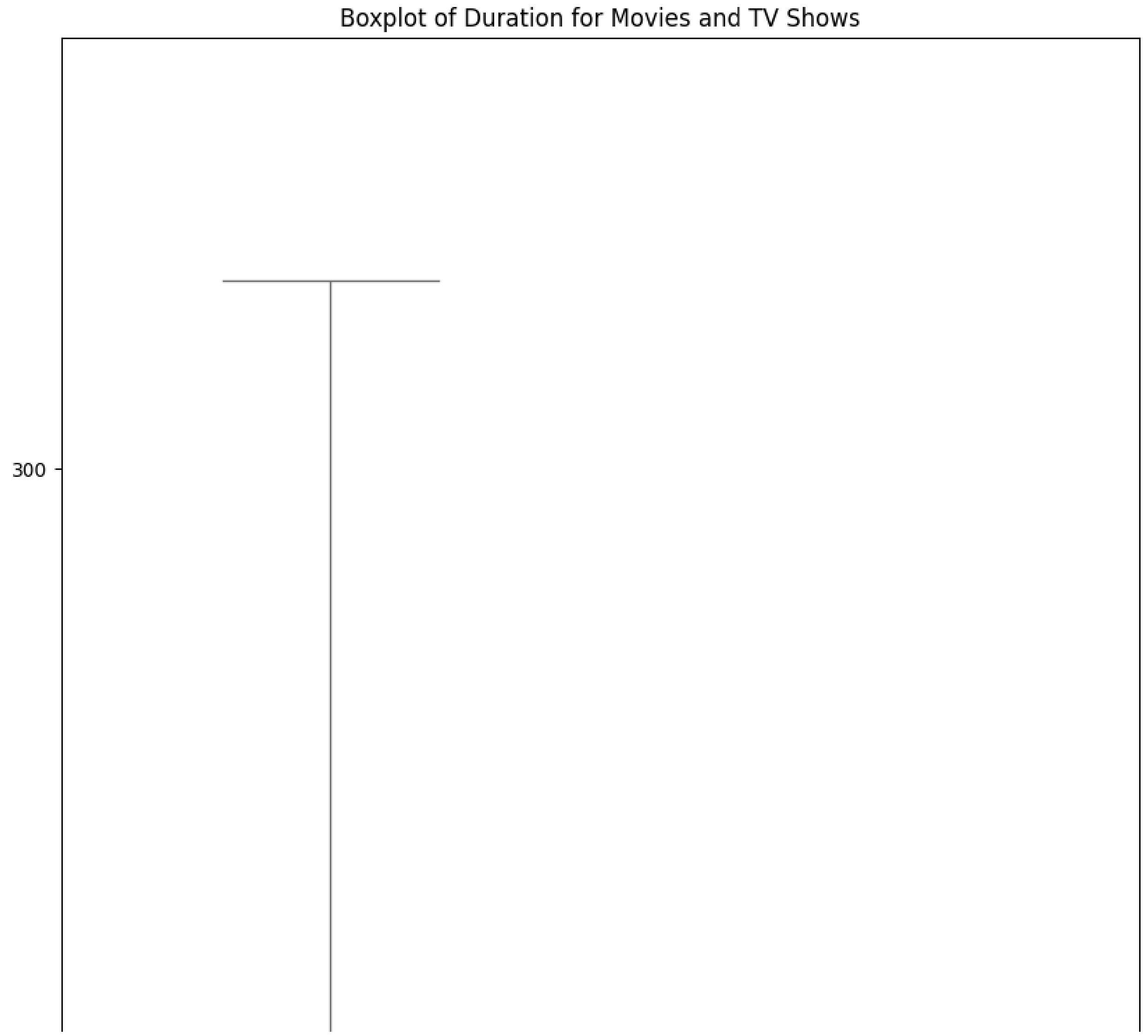
```
# Univariate Analysis for Continuous Variables
plt.figure(figsize=(12, 6))

# Displot for 'release_year'
plt.subplot(2, 2, 1)
sns.histplot(df['release_year'], kde=False, bins=20, color='blue')
plt.title('Distribution of Release Years')
plt.xlabel('Release Year')
plt.ylabel('Count')


plt.figure(figsize=(30, 10))
# Histogram for 'duration' (assuming 'duration' is a continuous variable)
plt.subplot(2, 2, 3)
plt.hist(df['duration'].dropna(), bins=20, color='green', edgecolor='black')
plt.title('Histogram of Duration')
plt.xlabel('Duration')
plt.ylabel('Count')

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

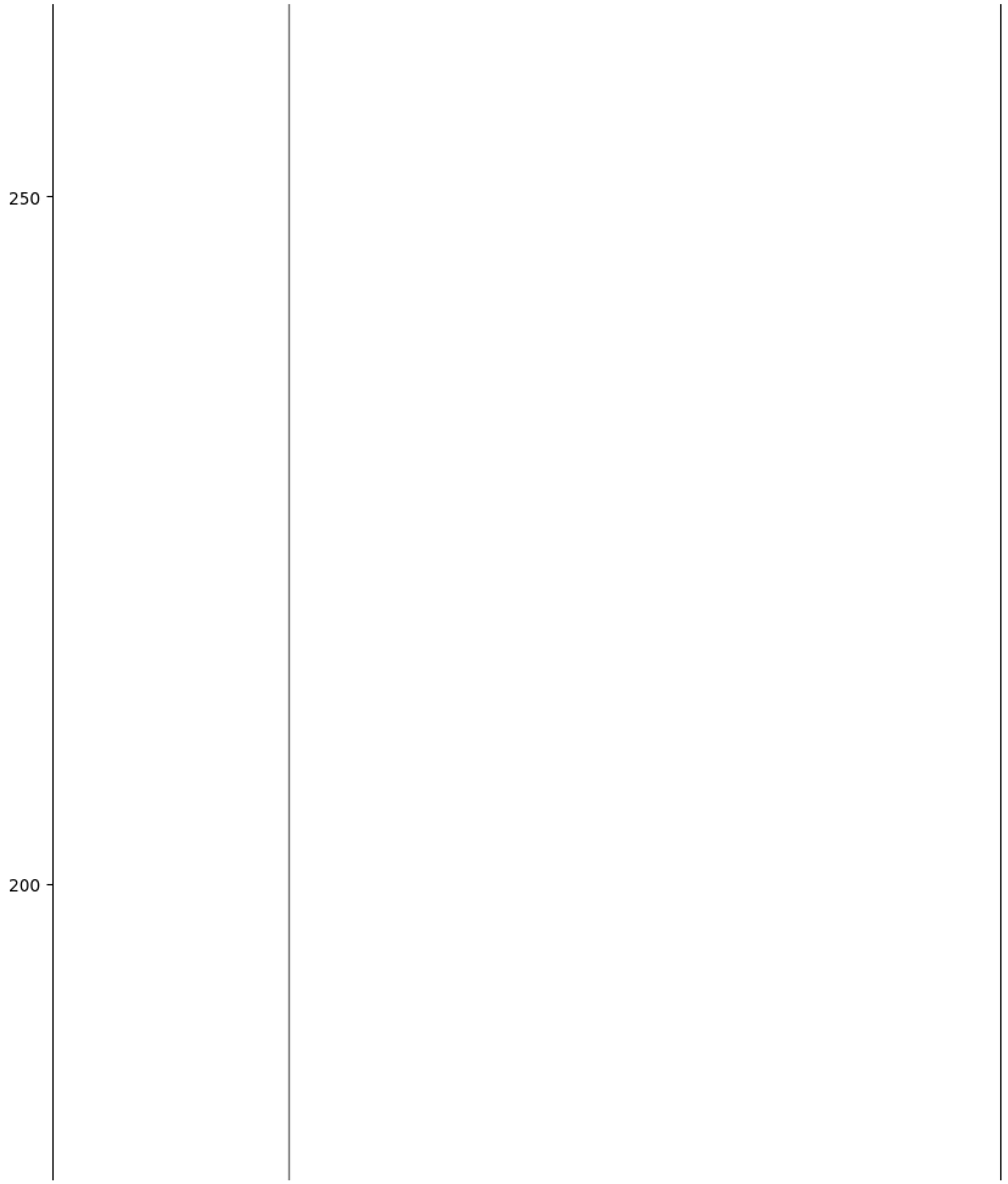

Distribution of Release Years
Count
Release Year
0
1000
2000
3000
4000
1940
1960
1980
2000
2020

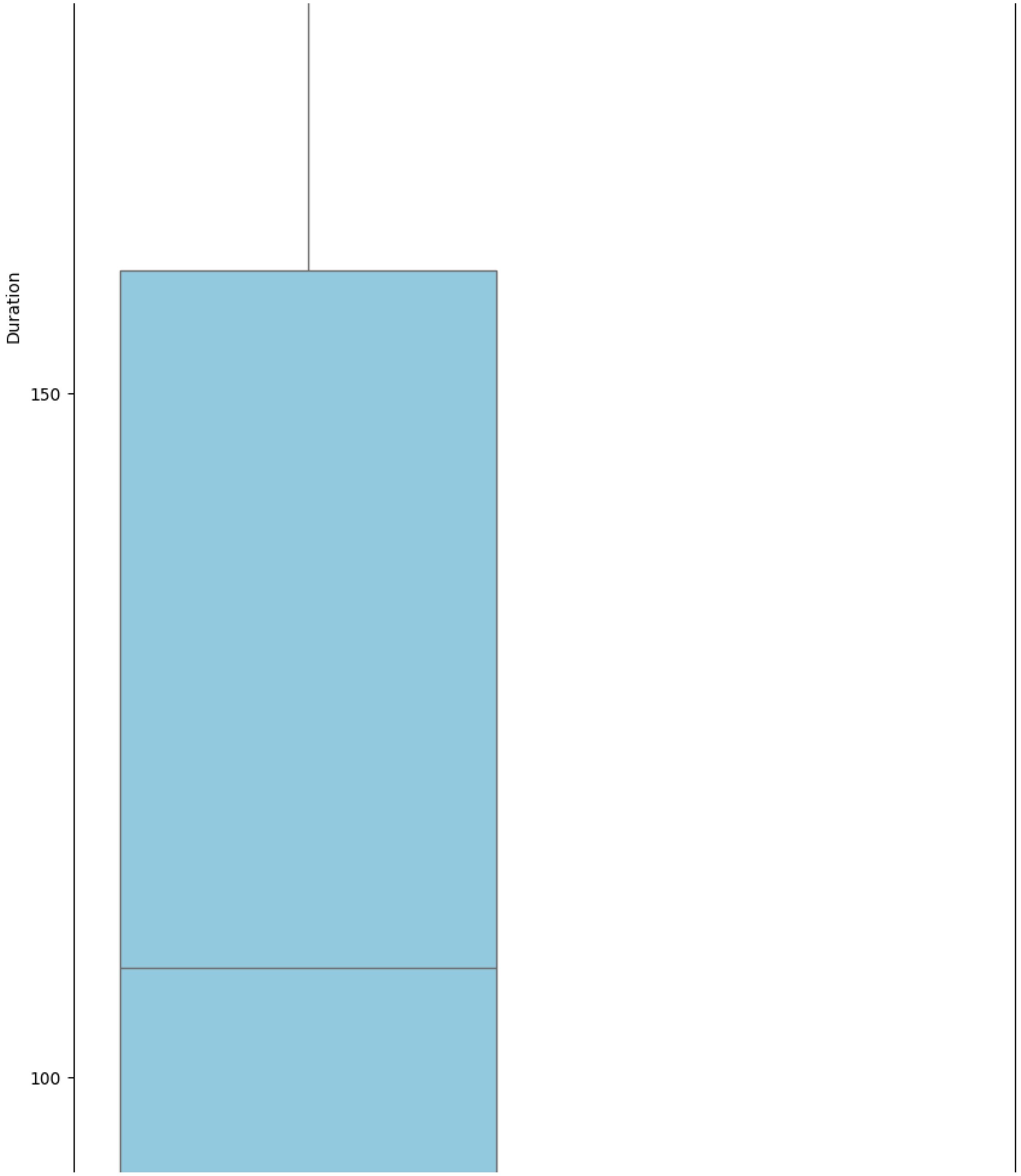

Histogram of Duration
Count
Duration
0
500
1000
1500
2000
2500
0
50
100
150
200
250
300

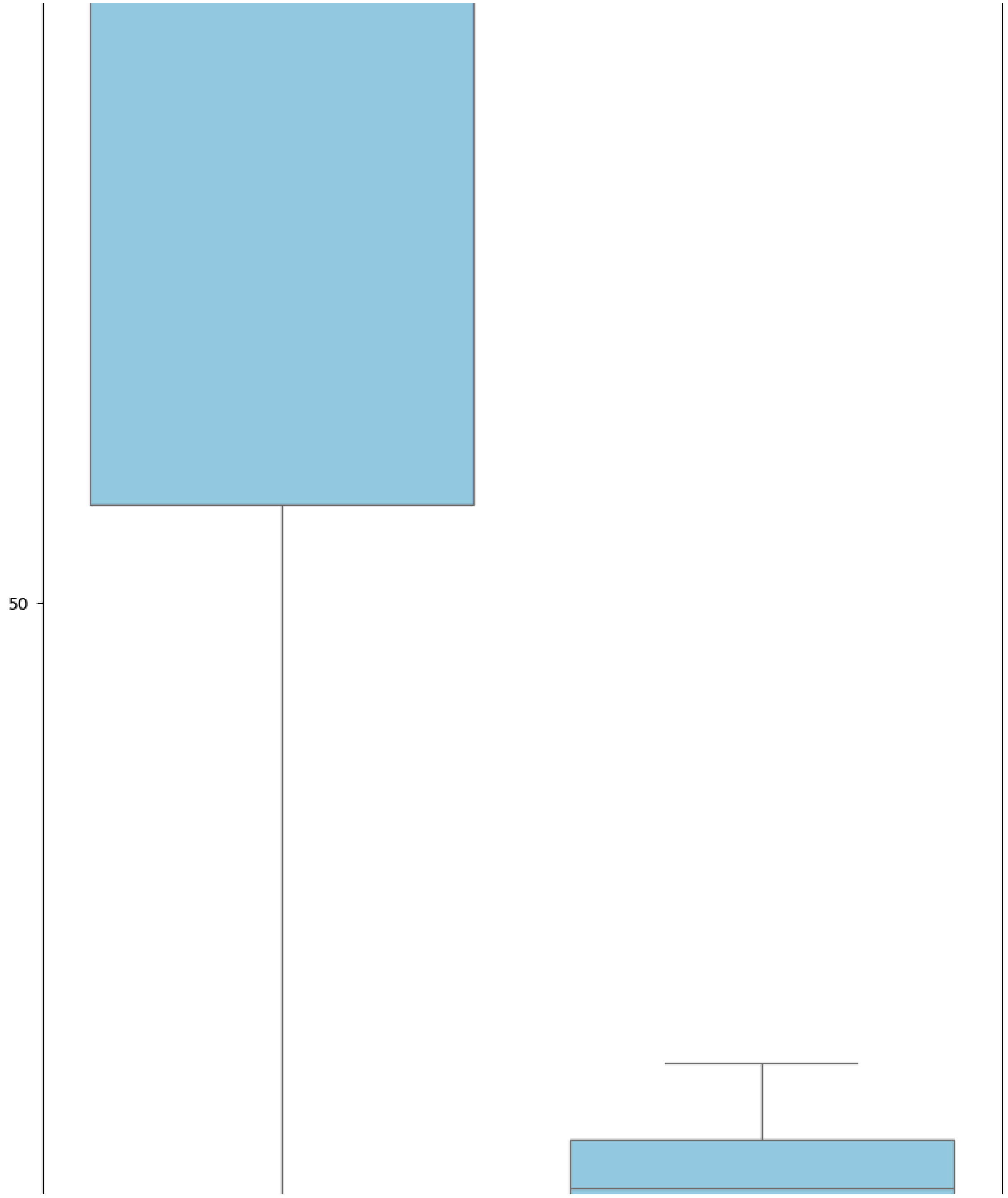
```
# Check for duplicate values in the 'type' column
duplicate_types = df['type'][df['type'].duplicated()]
print("Duplicate 'type' values:", duplicate_types)

# Drop duplicate rows based on the 'type' column
df_cleaned = df.drop_duplicates(subset=['type', 'duration'])

# Boxplot for Categorical Variable ('type') vs Numerical Variable ('duration')
plt.figure(figsize=(10, 50))
sns.boxplot(x='type', y='duration', data=df_cleaned, color='skyblue')
plt.title('Boxplot of Duration for Movies and TV Shows')
plt.xlabel('Type')
plt.ylabel('Duration')
plt.show()
```

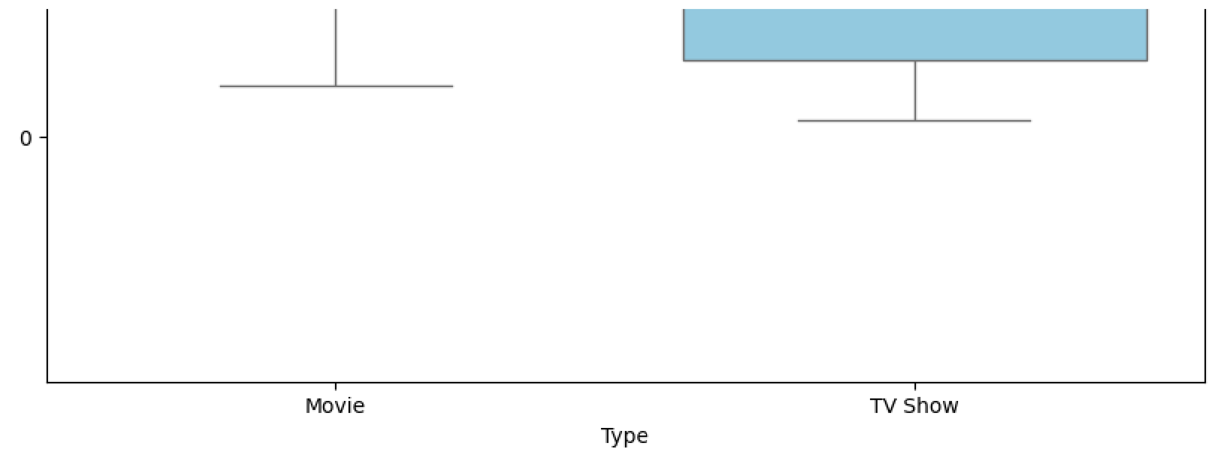
```
Duplicate 'type' values: 1       TV Show
1       TV Show
1       TV Show
1       TV Show
1       TV Show
         ...
8806      Movie
8806      Movie
8806      Movie
8806      Movie
8806      Movie
Name: type, Length: 64839, dtype: object
```




250
200

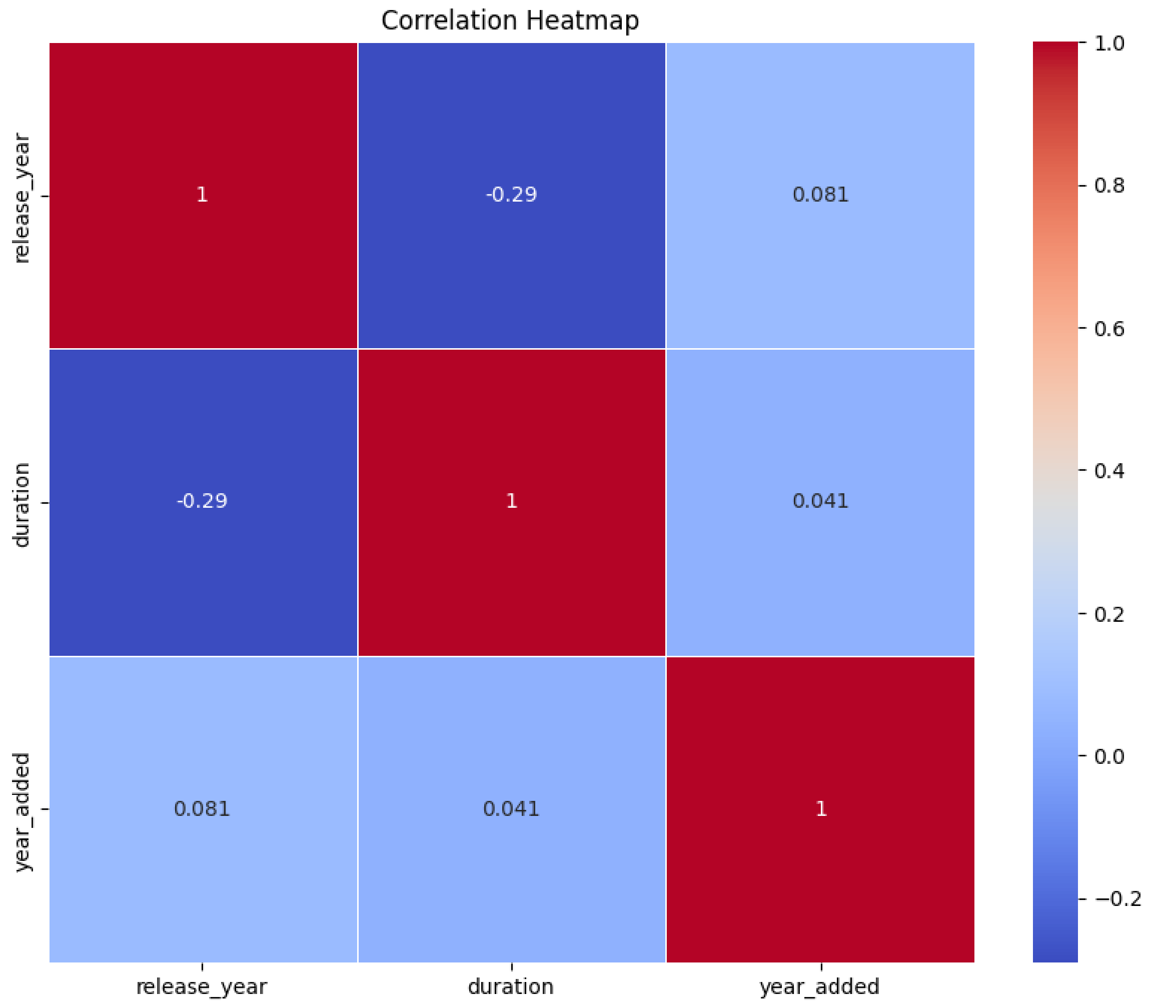

Duration
150
100


50

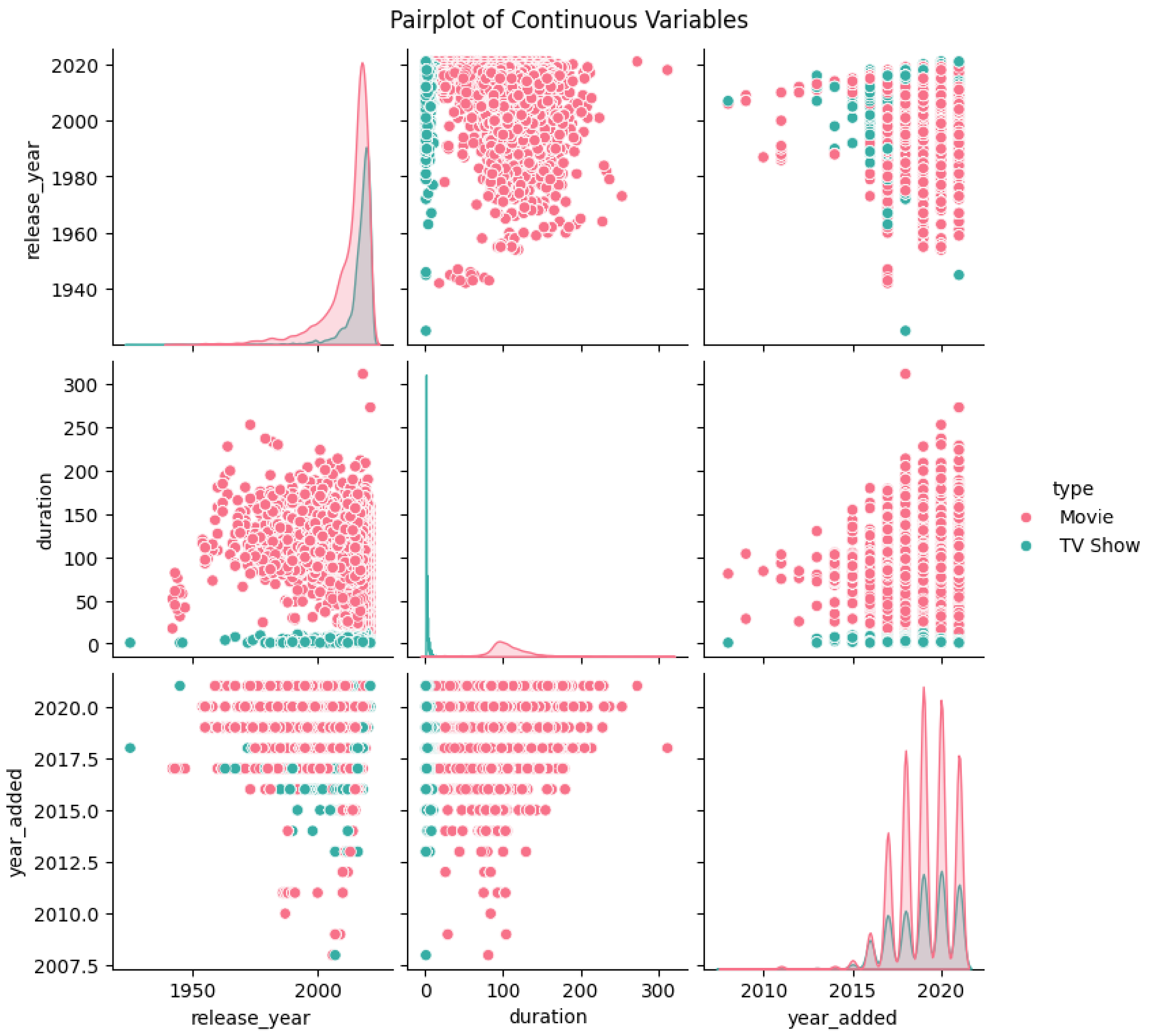


```
# Correlation Heatmap
correlation_matrix = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-247-79603ecb9b6b>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select or
  correlation_matrix = df.corr()
```



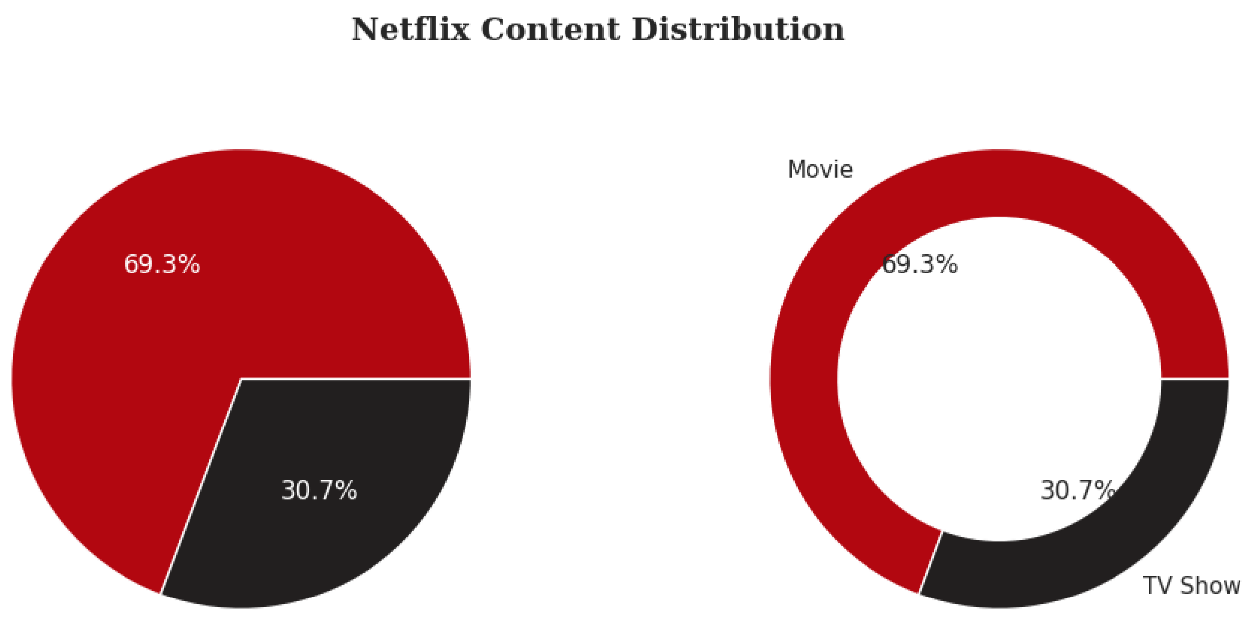
```
# Pairplot of Continuous Variables
sns.pairplot(df, hue='type', palette='husl')
plt.suptitle('Pairplot of Continuous Variables', y=1.02)
plt.show()
```



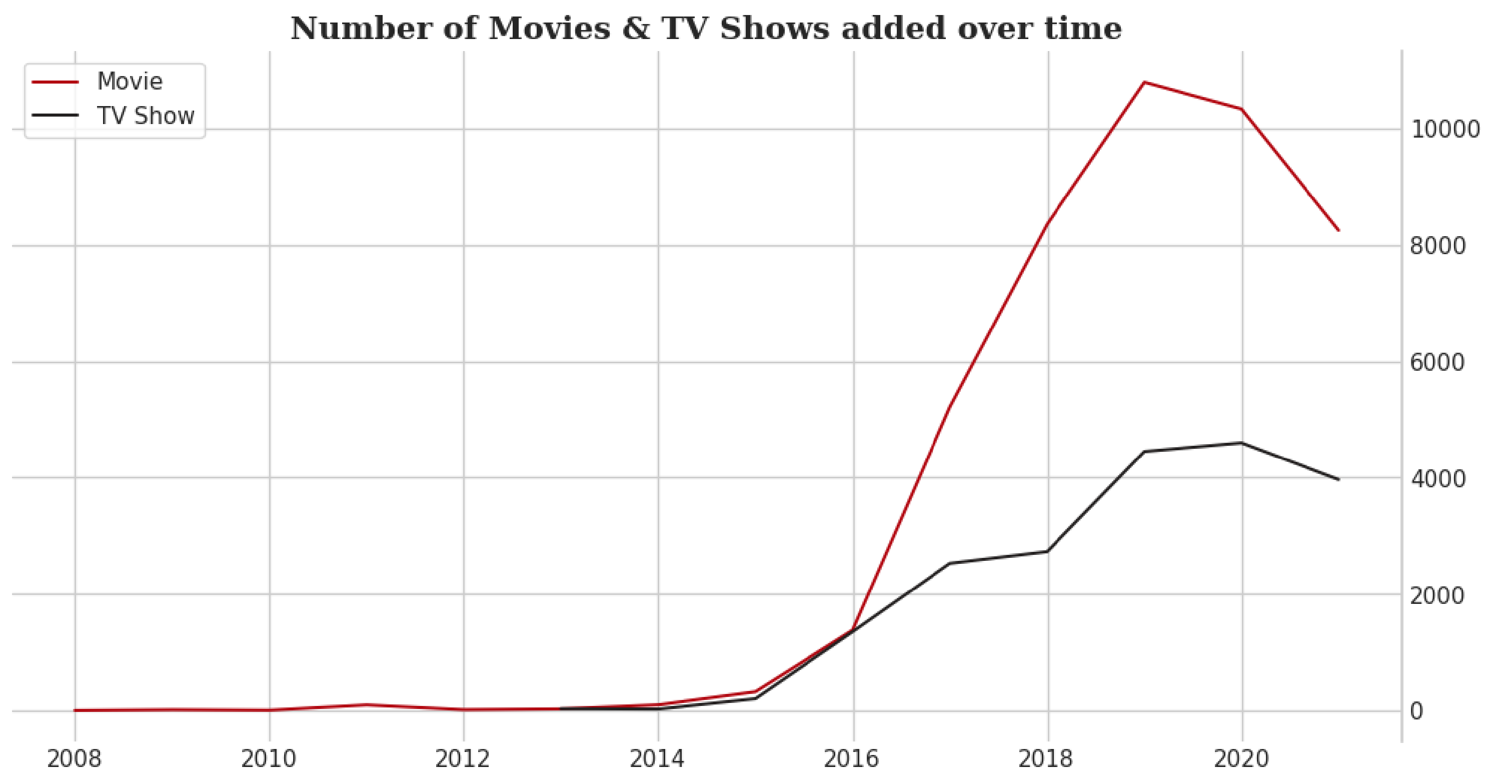
```
x = df['type'].value_counts()
x
```

```
    Movie      44938
    TV Show    19903
    Name: type, dtype: int64
```

```
#Set the style of seaborn
sns.set_theme(style="whitegrid")

# Create a figure and a set of subplots
fig, axs = plt.subplots(1, 2, figsize=(12, 5))

# Creating pie chart for count of movies
# Specify textprops in pie function
axs[0].pie(x.values, labels=x.index, colors=['#b20710','#221f1f'], autopct='%1.1f%%', textprops={'color':'white'})

# Creating donut chart for percentage distribution
axs[1].pie(x.values, labels=x.index, colors=['#b20710','#221f1f'], autopct='%1.1f%%', wedgeprops=dict(width=0.3))

# Adding title to the visual
fig.suptitle('Netflix Content Distribution', fontproperties={'family': 'serif', 'size': 15, 'weight': 'bold'})

plt.show()
```



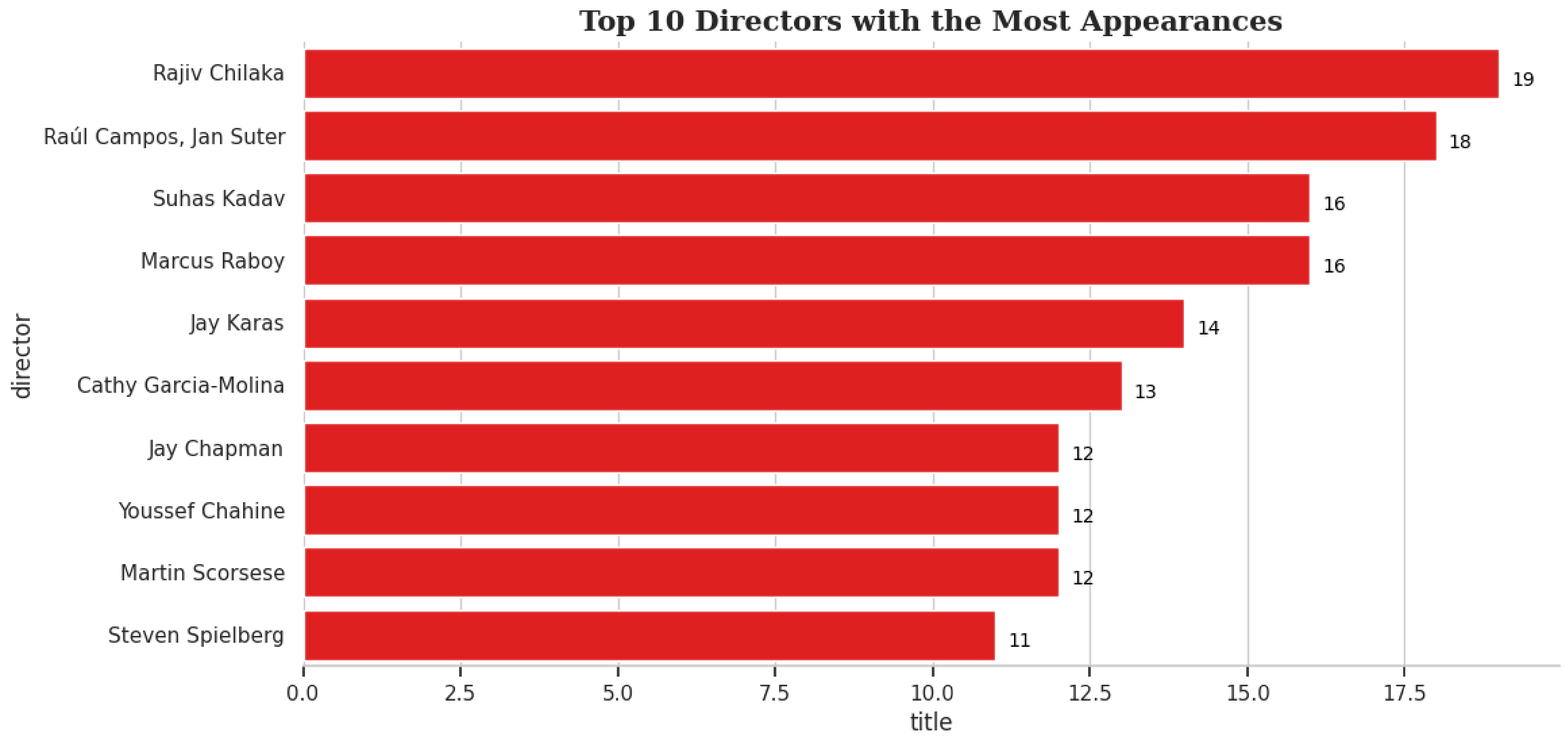
```
# Set the style of seaborn
sns.set_theme(style="whitegrid")

# Create a figure and a set of subplots
fig, ax = plt.subplots(figsize=(12, 6))

# Creating a dataframe for the plot
df_plot = pd.DataFrame(index=df['year_added'].sort_values().unique())

for type_ in df['type'].unique():
    temp_df = df[df['type'] == type_]['year_added'].value_counts().sort_index()
    df_plot[type_] = temp_df

# Plotting the line plot
df_plot.plot(kind='line', color=['#b20710','#221f1f'], ax=ax)

# Changing the y-axis position from left to right
ax.yaxis.tick_right()

# Removing the axis lines
sns.despine(ax=ax, top=True, right=False, left=True, bottom=True)

# Removing tick marks but keeping the labels
ax.tick_params(axis='both', length=0)

# Adding title to the visual
ax.set_title('Number of Movies & TV Shows added over time',
             {'font': 'serif', 'size': 15, 'weight': 'bold'})

plt.show()
```



```
# Directors with the Most Appearances
# Not Considering 'Unknown Director' s

df_1 = df[df['director'] != 'unknown director']
d_cnt = df_1.groupby('director')['title'].nunique().sort_values(ascending=False)[0:10].reset_index()
print(d_cnt)
```

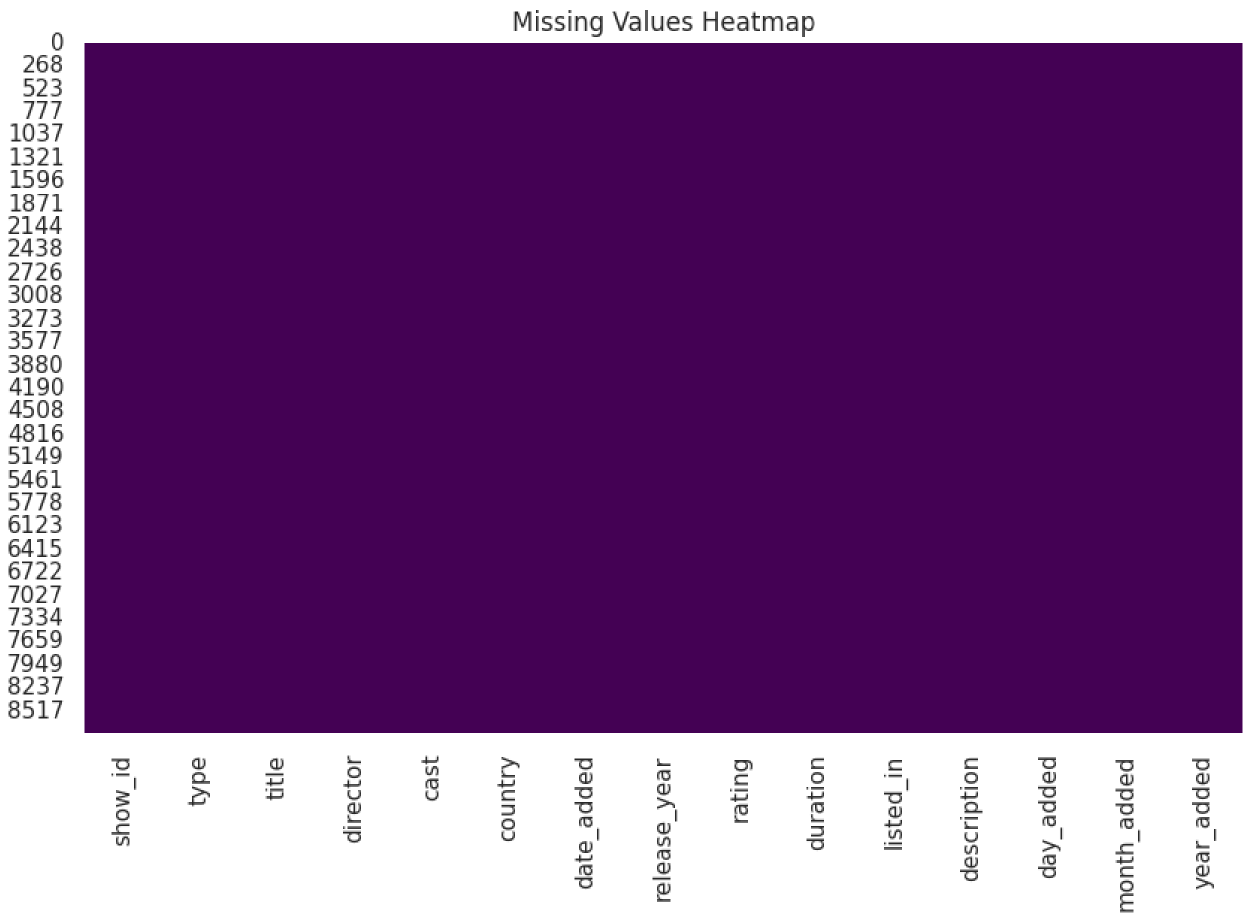
```
                    director  title
     0           Rajiv Chilaka     19
     1  Raúl Campos, Jan Suter     18
     2             Suhas Kadav     16
     3            Marcus Raboy     16
     4               Jay Karas     14
     5     Cathy Garcia-Molina     13
     6             Jay Chapman     12
     7          Youssef Chahine     12
     8         Martin Scorsese     12
     9        Steven Spielberg     11
```

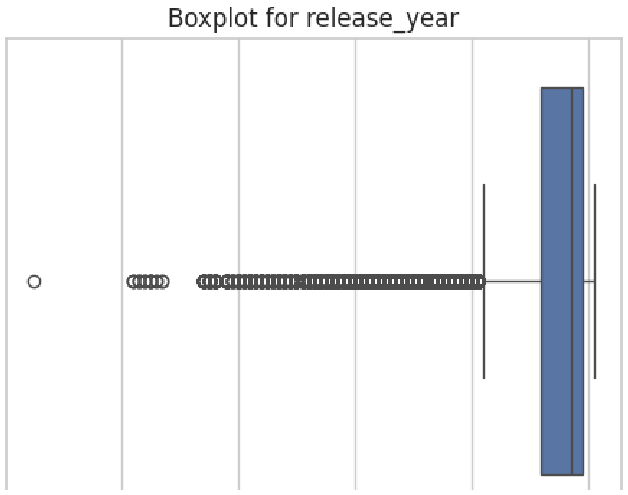
```
# Set the style of seaborn
sns.set_theme(style="whitegrid")

# Create a figure and a set of subplots
fig, ax = plt.subplots(figsize=(12, 6))

# Plotting the bar plot with a single color
barplot = sns.barplot(x='title', y='director', data=d_cnt, color='Red', ax=ax)

# Adding the text (actual count) on each bar
for i, v in enumerate(d_cnt['title']):
    ax.text(v + 0.2, i + .2, str(v), color='black', fontweight='light', fontsize=10)

# Changing the x-axis position from top to bottom
ax.xaxis.tick_bottom()

# Removing the axis lines
sns.despine(ax=ax, top=True, right=True, left=True, bottom=False)

# Adding title to the visual
ax.set_title('Top 10 Directors with the Most Appearances',
             {'font': 'serif', 'size': 15, 'weight': 'bold'})

plt.show()
```

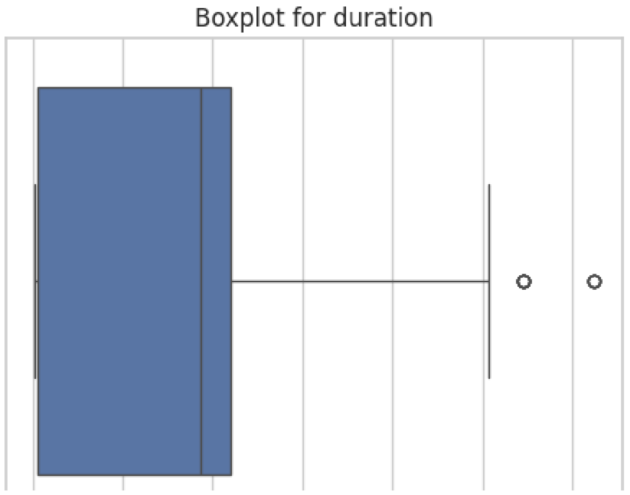


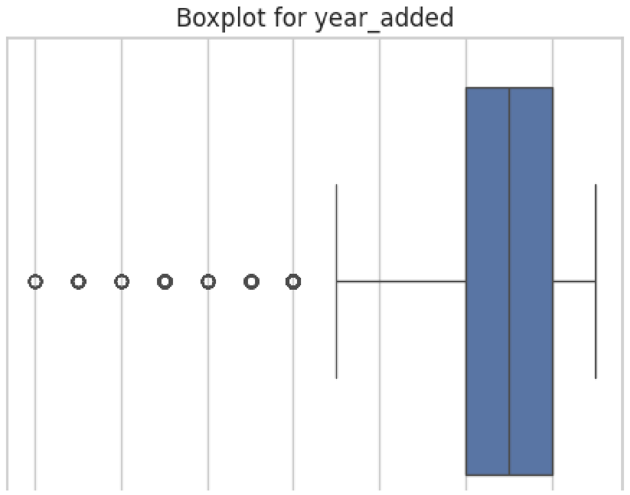
5. Missing Value & Outlier check

```
# Missing Value Check: Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False)
plt.title('Missing Values Heatmap')
plt.show()

# Outlier Check: Boxplot for Numerical Variables
numerical_columns = df.select_dtypes(include=['number']).columns

plt.figure(figsize=(14, 8))
for i, column in enumerate(numerical_columns, start=1):
    plt.subplot(2, 3, i)
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot for {column}')
    plt.title(f'Boxplot for {column}')

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


Missing Values Heatmap
0
268
523
777
1037
1321
1596
1871
2144
2438
2726
3008
3273
3577
3880
4190
4508
4816
5149
5461
5778
6123
6415
6722
7027
7334
7659
7949
8237
8517
show_id
type
title
director
cast
country
date_added
release_year
rating
duration
listed_in
description
day_added
month_added
year_added


Boxplot for release_year


Boxplot for duration


Boxplot for year_added

```
genres_tv_shows=df[df['type']=='TV Show']
genres_tv_shows['listed_in'].value_counts()
```

```
     Kids' TV                                                         1547
     Anime Series, International TV Shows                             1164
     Crime TV Shows, International TV Shows, TV Dramas                1101
     International TV Shows, TV Dramas                                1027
     Crime TV Shows, International TV Shows, Spanish-Language TV Shows  796
                                                                      ...
     Reality TV, Science & Nature TV, TV Action & Adventure              1
     Reality TV, TV Action & Adventure, TV Mysteries                     1
     Kids' TV, Reality TV, Science & Nature TV                           1
     Docuseries, Science & Nature TV, TV Comedies                        1
     Crime TV Shows, International TV Shows, Reality TV                  1
     Name: listed_in, Length: 235, dtype: int64
```

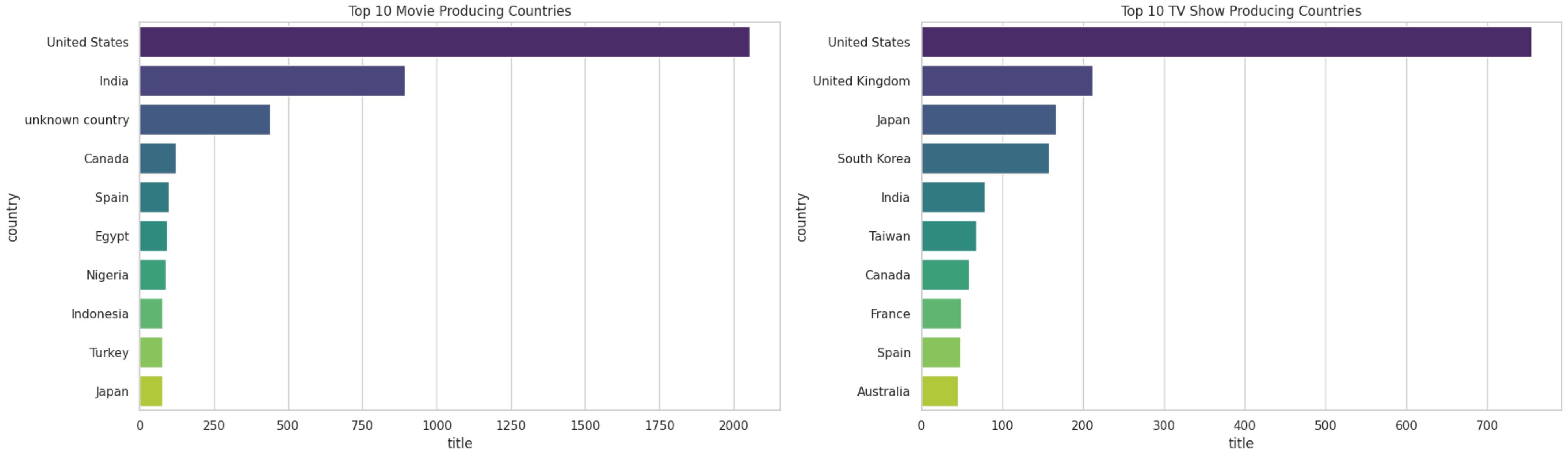
```
# Which genre movies are more popular or produced more

genres_movies=df[df['type']=='Movie'] # get the movie data from genre dataframe
genres_movies['listed_in'].value_counts()
```

```
     Dramas, International Movies                              3022
     Comedies, Dramas, International Movies                    2266
     Children & Family Movies, Comedies                        1956
     Dramas, Independent Movies, International Movies          1919
     Children & Family Movies                                  1725
                                                               ...
     Documentaries, Horror Movies                                 2
     Documentaries, LGBTQ Movies, Sports Movies                   2
     Anime Features, Documentaries                                1
     Classic Movies, Cult Movies, Documentaries                   1
     Documentaries, Faith & Spirituality, Music & Musicals        1
     Name: listed_in, Length: 278, dtype: int64
```

6. Insights based on Non-Graphical and Visual Analysis

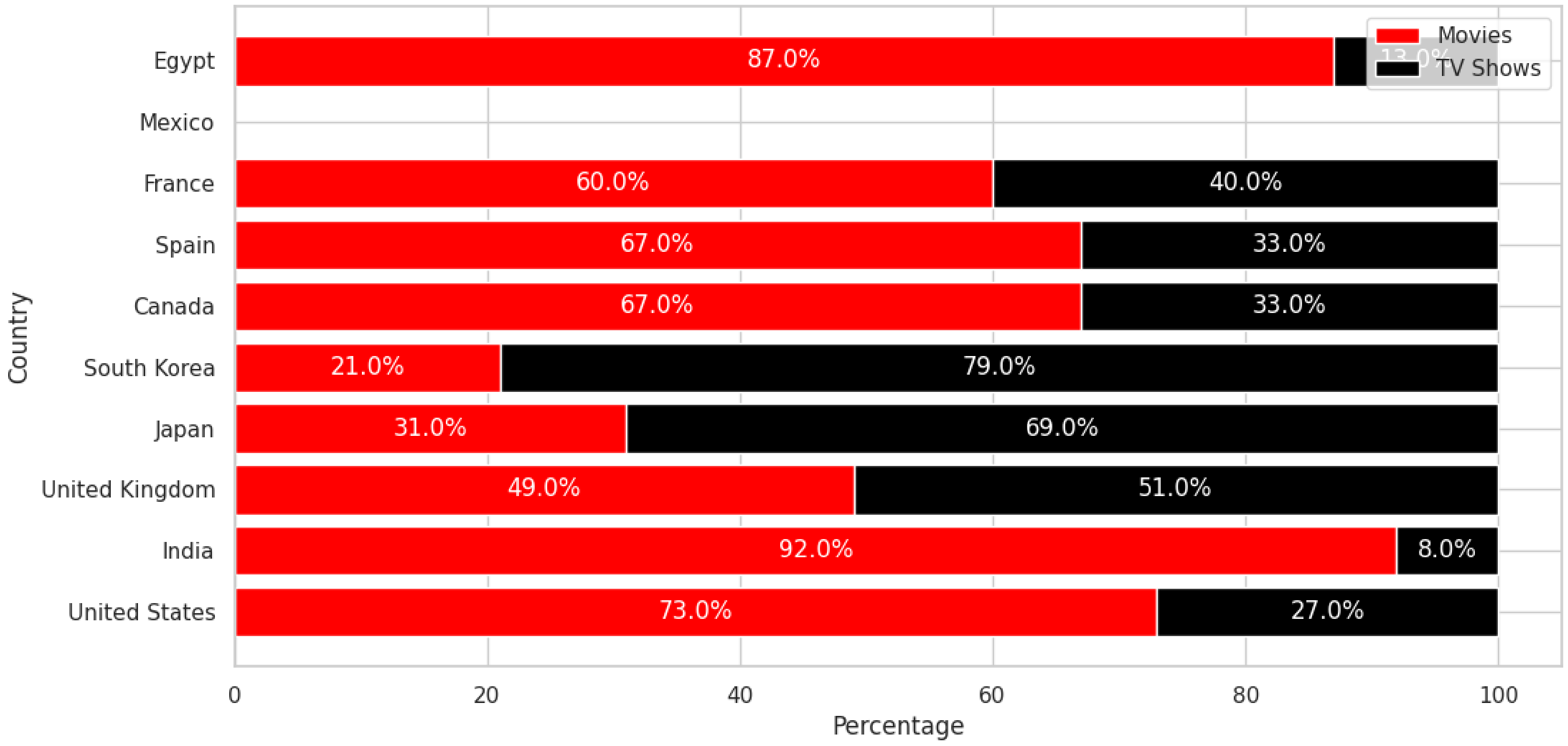
```python
#creating df for top 10 movies producing countries
df_movie = df[df['type'] == 'Movie']
df_movie = df_movie.groupby('country')['title'].nunique().sort_values(ascending = False).reset_index().loc[0:10]

#dropping unknown country column
df_movie = df_movie.drop(3)

#creating df for top 10 tv shows producing countries
df_tv = df[df['type'] == 'TV Show']
df_tv = df_tv.groupby('country')['title'].nunique().sort_values(ascending = False).reset_index().loc[0:10]

#dropping unknown country column
df_tv = df_tv.drop(1)


# Set the style of seaborn
sns.set_theme(style="whitegrid")

# Create a figure and a set of subplots
fig, axs = plt.subplots(1, 2, figsize=(20, 6))

# Plotting the bar plot for top 10 movie producing countries
sns.barplot(x='title', y='country', data=df_movie, ax=axs[0], palette='viridis')
axs[0].set_title('Top 10 Movie Producing Countries')

# Plotting the bar plot for top 10 TV show producing countries
sns.barplot(x='title', y='country', data=df_tv, ax=axs[1], palette='viridis')
axs[1].set_title('Top 10 TV Show Producing Countries')

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

```
<ipython-input-268-9de96eecd0a2>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='title', y='country', data=df_movie, ax=axs[0], palette='viridis')
<ipython-input-268-9de96eecd0a2>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='title', y='country', data=df_tv, ax=axs[1], palette='viridis')
```



```
# Create a figure and a set of subplots
fig, ax = plt.subplots(figsize=(12, 6))

# Plotting the stacked bar plot
bars1 = ax.barh(df_merge['country'], df_merge['Movie%'], color='red')
bars2 = ax.barh(df_merge['country'], df_merge['TV%'], left=df_merge['Movie%'], color='black')

# Adding percentages on bars
for bar in bars1:
    width = bar.get_width()
    ax.text(width/2, bar.get_y() + bar.get_height()/2, f'{width}%', ha='center', va='center', color='white')
for bar in bars2:
    width = bar.get_width()
    ax.text(bar.get_x() + width/2, bar.get_y() + bar.get_height()/2, f'{width}%', ha='center', va='center', color='white')

# Adding title to the visual
ax.set_title('Movie & TV Show Split for Top 10 Countries', color='white')

# Changing the x-axis and y-axis labels
ax.set_xlabel('Percentage')
ax.set_ylabel('Country')

# Adding a legend
ax.legend((bars1[0], bars2[0]), ('Movies', 'TV Shows'))

plt.show()
```

```
WARNING:matplotlib.text:posx and posy should be finite values
WARNING:matplotlib.text:posx and posy should be finite values
WARNING:matplotlib.text:posx and posy should be finite values
WARNING:matplotlib.text:posx and posy should be finite values
```



```
# Target Audience
```

```
#creating the relevant df
movie_rating = df.loc[df['type'] == 'Movie','rating'].value_counts().reset_index()
tv_rating = df.loc[df['type'] == 'TV Show','rating'].value_counts().reset_index()
#function for binning age groups
```