SCALER

NetFlix - Business Case Data Exploration and Visualisation

Submitted by :

Harsha Srinivas, Tanna
harshasrinivas.tanna@gmail.com
Scaler DSML - Morning TTS Feb 2023
Submitted on August 1, 2023

```
# downloading the csv file
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/original/netflix.csv
To: /content/netflix.csv
100% 3.40M/3.40M [00:00<00:00, 23.2MB/s]
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('netflix.csv')
```

```
# Q1 : Analysing Basic Metrics
```

```
# Q 1.1
# we're analysing Netflix dataset
# Netflix is a popular streaming service that offers a vast library of movies, TV shows, documentaries, and
# other content that users can watch on-demand over the internet on various devices.
# we can see that the data frame has 8807 rows × 12 columns
data
```

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

```
# Q 1.2
# various columns in the dataset
data.columns
```

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

```
# Q 1.3
data.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
```

```
# Q 1.4
# Since this is a movie dataset, you might be analyzing movie ratings,
# release dates, and genres to understand audience preferences.
# We're performing Exploratory Data Analysis (EDA) which includes summary, mean, & other averages on the columns
# checking for missing values and treating them
# Analysing the distribution of numerical data by plotting charts using seaborna and matplotlib
# Explore the distribution of categorical variables
# and also suggesting some recommendations and pattern indentification to Netflix
```

```
# Q 2 : missing value detection and filling NaN with 'Unknown'
# shape, data types, duplicate values, missing value detection, statistical summary
```

```
# Q 2.1
data.shape
```

```
(8807, 12)
```

```
# Q 2.2
# conversion of categorical attributes to 'category' (If required)
data.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
```

```
# Q 2.3
# conversion of categorical attributes to 'category' (If required)
data['type'] = data['type'].astype('category')
data['rating'] = data['rating'].astype('category')
data['listed_in'] = data['listed_in'].astype('category')
data.dtypes
```

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

```
# Q 2.4
# duplicate values
# no duplicate rows found
data[data.duplicated()]
```

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

```
# Q 2.5
# missing values
data.isnull()
```

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

8807 rows × 12 columns

```
# Q 2.6
# count of null values in each column
# we have null values only in director, cast, country, date_added, rating, duration columns
data.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
```

```
# Q 2.7
# Treatment of missing values
# replacing missing values in columns director, cast, country with 'Unknown'
# replacing missing values in date_added column with mode (most occuring value) of that column
data['director'].fillna('Unknown',inplace=True)
data['cast'].fillna('Unknown',inplace=True)
data['country'].fillna('Unknown',inplace=True)
data['date_added'].fillna(data['date_added'].mode(),inplace=True)
```

```
# Q 2.8
# Treatment of missing values for rating - movies
# replacing missing values in rating column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
rating_mode_movies = data[data['type'] == 'Movie']['rating'].mode()
data.loc[data['type'] == 'Movie']['rating'].fillna(rating_mode_movies,inplace=True)
```

```
<ipython-input-17-e26ca16f2831>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'Movie']['rating'].fillna(rating_mode_movies,inplace=True)
```

```
# Q 2.9
# Treatment of missing values for rating - tv shows
# replacing missing values in rating column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
rating_mode_tv_show = data[data['type'] == 'TV Show']['rating'].mode()
data.loc[data['type'] == 'TV Show']['rating'].fillna(rating_mode_tv_show,inplace=True)
```

```
<ipython-input-18-fca70b1a7a0c>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'TV Show']['rating'].fillna(rating_mode_tv_show,inplace=True)
```

```
# Q 2.10
# Treatment of missing values for duration - movies
# replacing missing values in duration column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
duration_mode_movie = data[data['type'] == 'Movie']['duration'].mode()
data.loc[data['type'] == 'Movie']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
<ipython-input-19-4447a441bea4>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'Movie']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
# Q 2.11
# Treatment of missing values for duration - tv shows
# replacing missing values in duration column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
duration_mode_movie = data[data['type'] == 'TV Show']['duration'].mode()
data.loc[data['type'] == 'TV Show']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
<ipython-input-20-ddf967aff735>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'TV Show']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
data.isnull().sum()
```

```
show_id          0
type             0
title            0
director         0
cast             0
country          0
date_added      10
release_year     0
rating           4
duration         3
listed_in        0
description      0
dtype: int64
```

```
# Q 2.12
# making the columns explode - cast column
data['cast'] = data['cast'].str.split(', ')
data = data.explode('cast')
```

```
# Q 2.13
# making the columns explode - listed_in column
data['listed_in'] = data['listed_in'].str.split(', ')
data = data.explode('listed_in')
```

```
data = data.reset_index(drop=True)
```

```
# Q 2.14
# Statistical Summary
data.describe()
```

	release_year
count	149512.000000
mean	2013.621482
std	9.156160
min	1925.000000
25%	2012.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

```
# Q 3
# Non-Graphical Analysis: Value counts and unique attributes
```

```
# Q 3.1
# types of content here we have only 2 types
```

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

```
Movie      101692
TV Show     47820
Name: type, dtype: int64
```

```
# Q 3.2
# unique number of directors
data['director'].nunique()
```

```
4529
```

```
# Q 3.3
# unique number of movies
data[data['type'] == 'Movie']['show_id'].nunique()
```

```
6131
```

```
# Q 3.4
# unique number of tv shows
data[data['type'] == 'TV Show']['show_id'].nunique()
```

```
2676
```

```
# Q 3.5
# directors who directed most movies
data[data['type'] == 'Movie']['director'].value_counts().head(10)
```

```
Unknown                1204
Cathy Garcia-Molina     356
Youssef Chahine         288
Martin Scorsese         273
David Dhawan            270
Steven Spielberg        243
Kunle Afolayan          223
Toshiya Shinohara       204
Mae Czarina Cruz        198
Yılmaz Erdoğan          197
Name: director, dtype: int64
```

```
# Q 3.6
# directors who directed most tv shows
data[data['type'] == 'TV Show']['director'].value_counts().head(10)
```

```
Unknown                                          43417
Rob Seidenglanz                                    103
Kongkiat Komesiri                                   75
Danny Cannon                                        75
Pass Patthanakumjon                                 66
Noam Murro                                          63
Mateo Gil                                           60
Pantham Thongsang, Cheewatan Pusitsuksa             60
Srijit Mukherji, Vasan Bala, Abhishek Chaubey       57
Eli Roth                                            57
Name: director, dtype: int64
```

```
# Q 3.7
# directors who directed both movie and tv show
# There are no such directors
data[(data['type'] == 'Movie') & (data['type'] == 'TV Show')]['director'].value_counts().head(10)
```

```
Series([], Name: director, dtype: int64)
```

```
# Q 3.8
# Actors who've acted in most movies
data[data['type'] == 'Movie']['cast'].value_counts().head(10)
```

```
Unknown             807
Anupam Kher         117
Shah Rukh Khan       99
Naseeruddin Shah     92
Akshay Kumar         85
Paresh Rawal         79
Om Puri              78
Amitabh Bachchan     76
Boman Irani          72
Kareena Kapoor       67
Name: cast, dtype: int64
```

```
# Q 3.9
# Actors who've acted in most tv shows
```

```
data[data['type'] == 'TV Show']['director'].value_counts().head(10)
```

```
Unknown                                          43417
Rob Seidenglanz                                    103
Kongkiat Komesiri                                   75
Danny Cannon                                        75
Pass Patthanakumjon                                 66
Noam Murro                                          63
Mateo Gil                                           60
Pantham Thongsang, Cheewatan Pusitsuksa             60
Srijit Mukherji, Vasan Bala, Abhishek Chaubey       57
Eli Roth                                            57
Name: director, dtype: int64
```

```
# Q 3.10
# Countries with most movies
data[data['type'] == 'Movie']['country'].value_counts().head(10)
```

```
United States     28635
India             18817
Unknown            5708
Japan              2055
Nigeria            2014
United Kingdom     2013
Spain              1796
Philippines        1667
Indonesia          1660
Egypt              1651
Name: country, dtype: int64
```

```
# Q 3.11
# Countries with most tv shows
data[data['type'] == 'TV Show']['country'].value_counts().head(10)
```

```
United States     9916
Unknown           5437
Japan             4529
South Korea       3459
United Kingdom    3167
Spain             1574
Mexico            1469
Taiwan            1402
India              999
Canada             941
Name: country, dtype: int64
```

```
# Q 3.12
# Date on which Netflix adds more movies
data[data['type'] == 'Movie']['date_added'].value_counts().head(10)
```

```
January 1, 2020      2014
March 1, 2018        1511
December 31, 2019    1506
November 1, 2019     1370
October 1, 2018      1249
July 1, 2021         1130
September 1, 2021    1049
October 1, 2019      1017
November 1, 2018      972
June 2, 2021          880
Name: date_added, dtype: int64
```

```
# Q 3.13
# Date on which Netflix adds more tv shows
data[data['type'] == 'TV Show']['date_added'].value_counts().head(10)
```

```
July 6, 2021         767
June 19, 2021        486
July 1, 2017         414
August 1, 2017       338
April 14, 2021       316
January 1, 2016      308
December 15, 2016    296
November 1, 2019     286
November 15, 2019    282
September 1, 2017    276
Name: date_added, dtype: int64
```

```
# Q 3.14
# most frequent ratings for movies
data[data['type'] == 'Movie']['rating'].value_counts().head(10)
```

```
TV-MA    31482
TV-14    25001
```

```
R        15098
PG-13     9860
TV-PG     8133
PG        5955
TV-Y7     1600
TV-G      1508
TV-Y      1159
NR         978
Name: rating, dtype: int64
```

```
# Q 3.15
# most frequent ratings for tv shows
data[data['type'] == 'TV Show']['rating'].value_counts().head(10)
```

```
TV-MA       25213
TV-14       13643
TV-PG        3811
TV-Y7        2687
TV-Y         1248
TV-G          927
NR            155
R              54
TV-Y7-FV       24
66 min          0
Name: rating, dtype: int64
```

```
# Q 3.16
# mean of duration for movies
def time_fn(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(' ')
        return int(x[0])
    else:
        return None

mean_duration_of_movies = data[data['type'] == 'Movie']['duration'].apply(time_fn)
mean_duration_movies = mean_duration_of_movies.mean()
mean_duration_movies #in minutes
```

```
107.63875148737819
```

```
# Q 3.17
# average length of a season for tv shows

mean_duration_of_tv_shows = data[data['type'] == 'TV Show']['duration'].apply(time_fn)
mean_duration_tv_shows = mean_duration_of_tv_shows.mean()
mean_duration_tv_shows # average season length
```

```
1.915223755750732
```

```
# Q 3.18
# number of unique genres for movies
data[data['type'] == 'Movie']['listed_in'].value_counts()
```

```
Dramas                      20778
International Movies        20669
Comedies                    15197
Action & Adventure           8060
Independent Movies           6300
Children & Family Movies     5634
Romantic Movies              5221
Thrillers                    4821
Horror Movies                3069
Music & Musicals             2434
Sci-Fi & Fantasy             2294
Documentaries                1641
Sports Movies                1194
Classic Movies               1029
Cult Movies                   724
Anime Features                688
LGBTQ Movies                  679
Stand-Up Comedy               487
Faith & Spirituality          480
Movies                        293
Name: listed_in, dtype: int64
```

```
# Q 3.19
# number of unique genres for tv shows
data[data['type'] == 'TV Show']['listed_in'].value_counts()
```

```
International TV Shows          11319
TV Dramas                        7473
```

```
TV Comedies                     4482
Crime TV Shows                  4006
Kids' TV                        3252
Romantic TV Shows               2903
Anime Series                    2126
Spanish-Language TV Shows       1825
TV Action & Adventure           1788
British TV Shows                1424
TV Mysteries                    1088
Korean TV Shows                 1062
TV Sci-Fi & Fantasy              866
TV Horror                        819
Teen TV Shows                    742
Docuseries                       733
TV Thrillers                     602
Reality TV                       578
Stand-Up Comedy & Talk Shows     260
Classic & Cult TV                240
Science & Nature TV              130
TV Shows                         102
Name: listed_in, dtype: int64
```

```
# Q 3.20
# mean of difference between date_added and release for movies
def date_time_fn(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(',')
        return int(x[-1])
    else:
        return None
date_added_year_movies = data[data['type'] == 'Movie']['date_added'].apply(date_time_fn)
date_added_release_movies = data[data['type'] == 'Movie']['release_year'].apply(date_time_fn)
data['release_date_added_diff_movies'] = date_added_year_movies - date_added_release_movies
data['release_date_added_diff_movies'].mean() # in years
```

```
6.9227471187507374
```

```
# Q 3.21
# mean of difference between date_added and release for tv shows
def date_time_fn(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(',')
        return int(x[-1])
    else:
        return None
date_added_year_tv = data[data['type'] == 'TV Show']['date_added'].apply(date_time_fn)
date_added_release_tv = data[data['type'] == 'TV Show']['release_year'].apply(date_time_fn)
data['release_date_added_diff_tv'] = date_added_year_tv - date_added_release_tv
data['release_date_added_diff_tv'].mean() # in years
```

```
2.097436112626411
```

```
# Q 3.22
# data of movies that have been added after the maximum time gap
data[data['release_date_added_diff_movies'] == data['release_date_added_diff_movies'].max()].head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descript
												Frank Cap

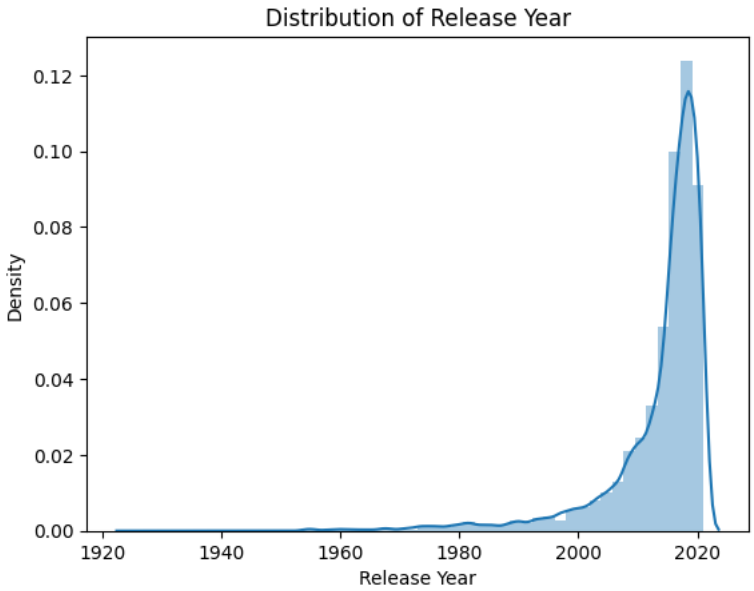
```
# Q 3.22
# data of tv shows that have been added after the maximum time gap
data[data['release_date_added_diff_tv'] == data['release_date_added_diff_tv'].max()].head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descript
76914	s4251	TV Show	Pioneers: First Women Filmmakers*	Unknown	Unknown	Unknown	December 30, 2018	1925	TV-14	1 Season	TV Shows	This colle restores from wo wh
				of		Fonda	States	2017				cor

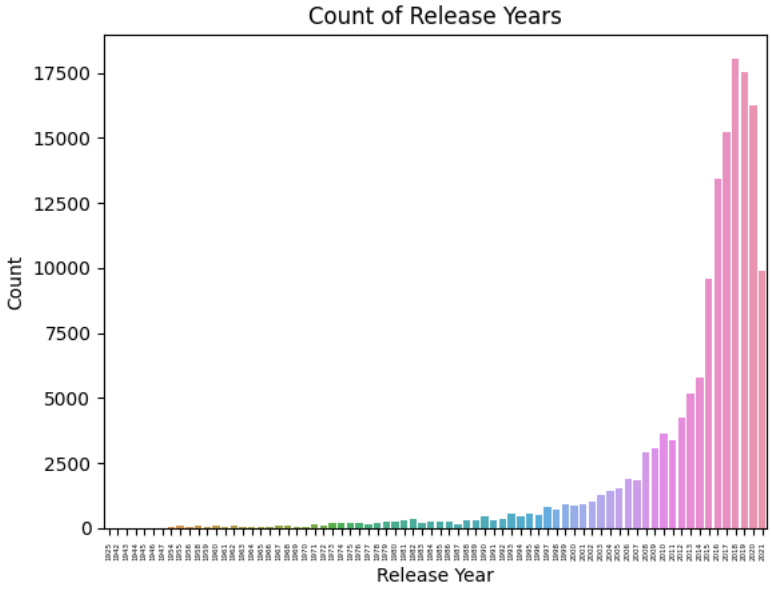
```
# Q 4
# Visual Analysis - Univariate, Bivariate after pre-processing of the data
# Pre-processing involves unnesting of the data in columns like Actor, Director, Country already done
# exploding the data
```

```
# Q 4.1
# distplot is used for continuous variable
# here in our dataset 'release_year' is one such variable
# the below plot shows the distribution of 'release_year' and
# insights into the frequency and spread of the release years in the Netflix dataset.
sns.distplot(data['release_year'],kde=True)
plt.title('Distribution of Release Year')
plt.xlabel('Release Year')
plt.ylabel('Density')
plt.show()
```

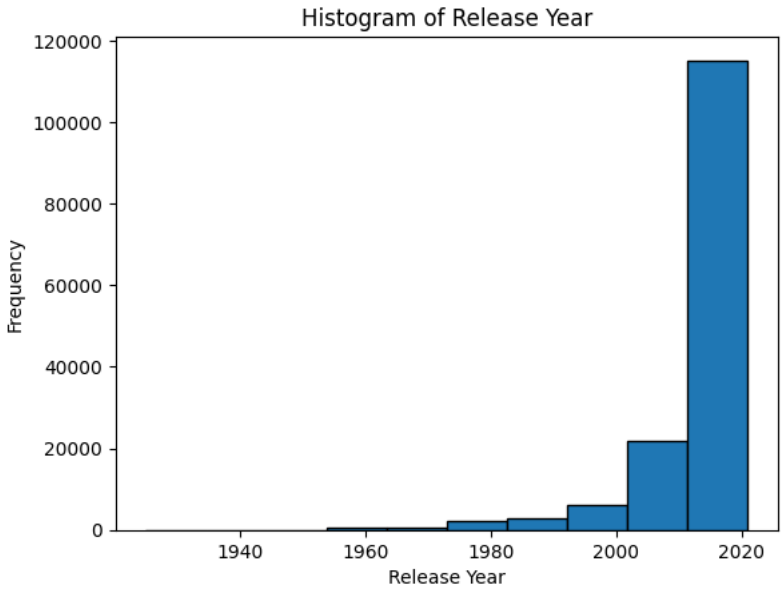
```
<ipython-input-51-7af0793690c2>:6: UserWarning:

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

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

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

  sns.distplot(data['release_year'],kde=True)
```




```
# Q 4.2
# Countplot for 'release_year'
# The below plot gives us insights into the number of movies/tv shows released vs year
# the year range is from 1925 to 2021 as per the dataset
# we can see more number of movies/tv shows are being produced in the recent years than that of the psat
sns.countplot(x='release_year', data=data)
plt.title('Count of Release Years')
plt.xlabel('Release Year')
plt.xticks(rotation = 90,fontsize = 4)
plt.ylabel('Count')
plt.show()
```

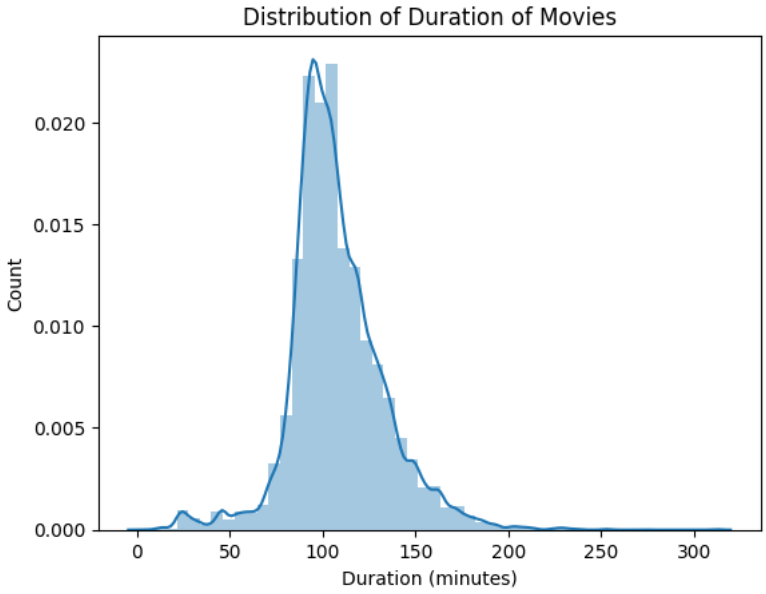



```
#  Q 4.3
# Histogram for 'release_year'
# Again we plot for number of releases vs year range
# here we divide the available year range into 10 bins and plot the histogram
# we can see the highest number of movies/tv shows is produced in the decade 2010-2020
plt.hist(data['release_year'], bins=10, edgecolor='k')
plt.title('Histogram of Release Year')
plt.xlabel('Release Year')
plt.ylabel('Frequency')
plt.show()
```




```
# Q 4.4
#  Distribution of duration of movies
# From the below plot we can see the how the duration of movies is distributed
# in minutes
# Observation : Most movies span for ~100 minutes
def date_time_fn(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(',')
        return int(x[-1])
    else:
        return None

duration_of_movies_extract = data[data['type'] == 'Movie']['duration'].apply(time_fn)
sns.distplot(x=duration_of_movies_extract)
plt.xlabel('Duration (minutes)')
plt.ylabel('Count')
plt.title('Distribution of Duration of Movies')
plt.show()
```

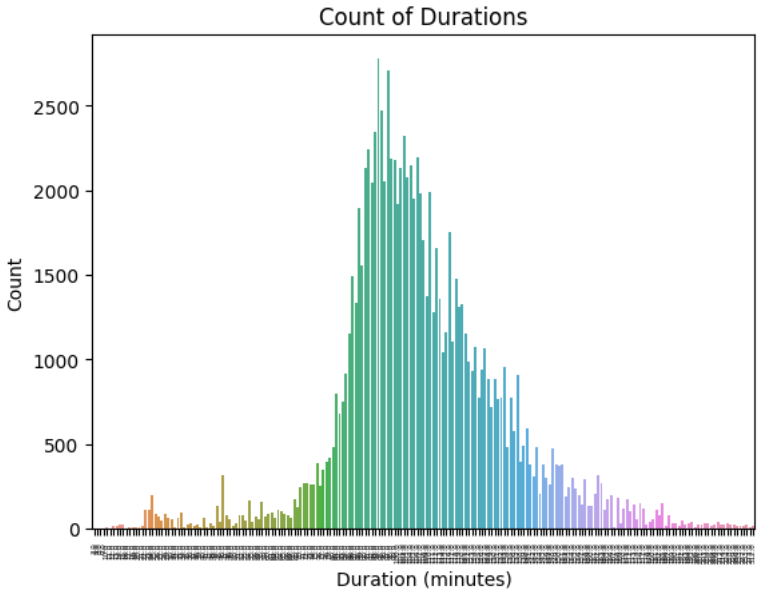
```
<ipython-input-54-0fa06057e879>:12: UserWarning:

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

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

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

  sns.distplot(x=duration_of_movies_extract)
```



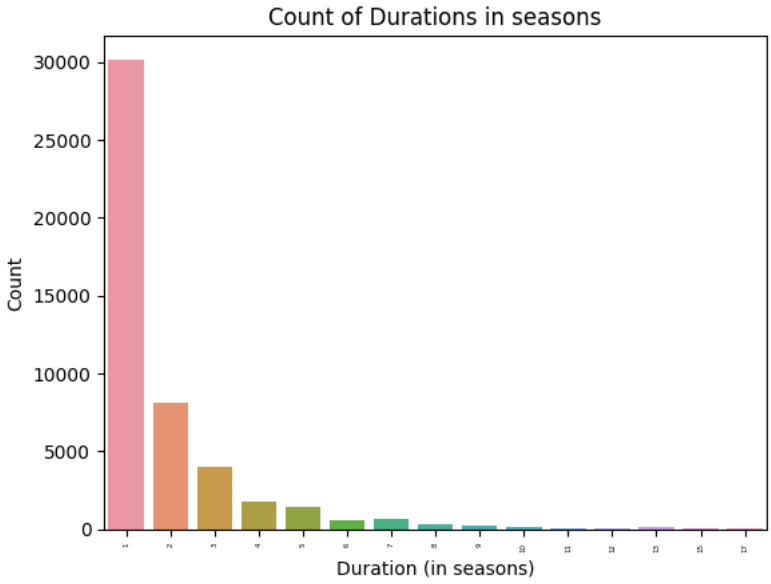

```
# Q 4.5
#  Distribution of duration of tv shows
# From the below plot we can see the how the duration of tv shows is distributed
# in seasons
# Observations : Most TV Shows span for ~ 1.25 seasons

def date_time_fn(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(',')
        return int(x[-1])
    else:
        return None
duration_of_tv_extract = data[data['type'] == 'TV Show']['duration'].apply(time_fn)
sns.distplot(x=duration_of_tv_extract)
plt.xlabel('Duration (in seasons)')
plt.ylabel('Count')
plt.title('Duration of TV Shows in seasons')
plt.show()
```

```
<ipython-input-55-ec50bb7cf258>:11: UserWarning:

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

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

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

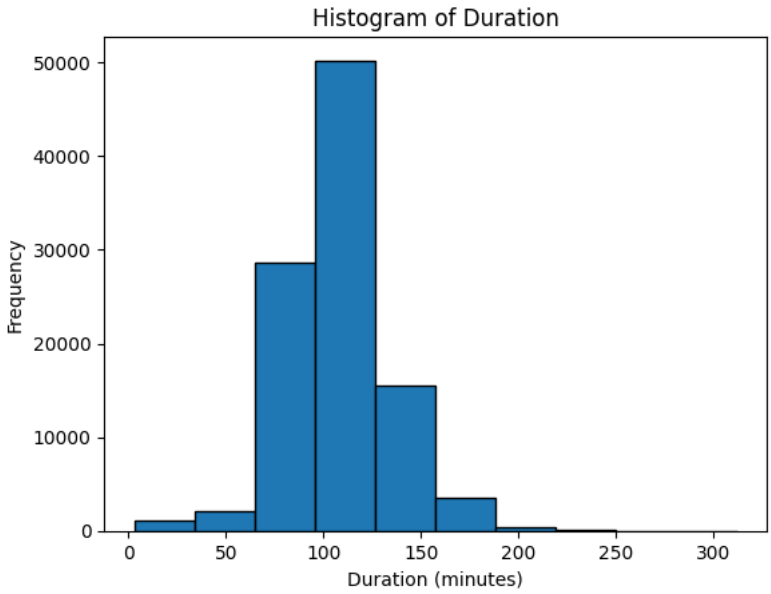
```
# Q 4.6
# Countplot for 'duration' for movies
# we can see most of the movies have a duration around the centre of the distribution
# This plot resembles a bell curve
sns.countplot(x=duration_of_movies_extract, data=data)
plt.title('Count of Durations')
plt.xlabel('Duration (minutes)')
plt.xticks(rotation = 90,fontsize = 4)
plt.ylabel('Count')
plt.show()
```



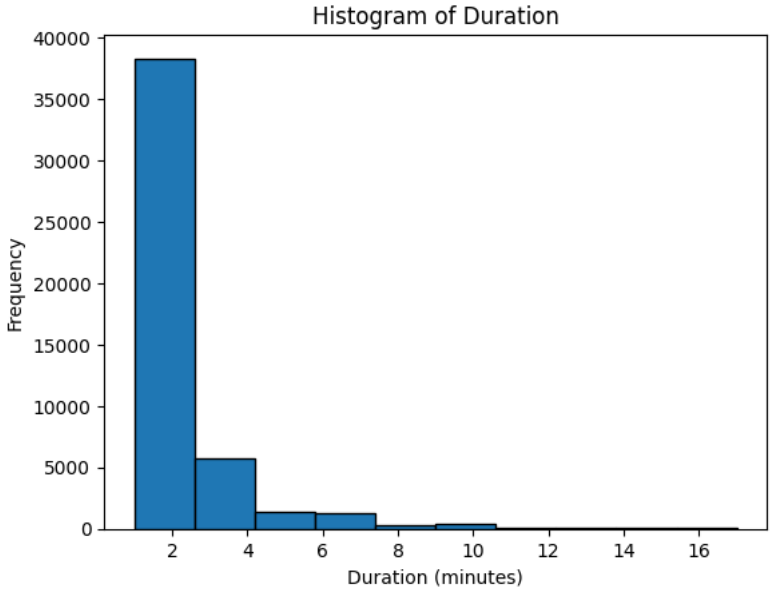

```
# Q 4.7
# Countplot for 'duration' of tv shows
# we can infer that most tv shows span for only 1 or 2 season
sns.countplot(x=duration_of_tv_extract, data=data)
plt.title('Count of Durations in seasons')
plt.xlabel('Duration (in seasons)')
plt.xticks(rotation = 90,fontsize = 4)
plt.ylabel('Count')
plt.show()
```



```
# Q 4.8
# Histogram for 'duration' for movies
# we can see most of the movies span for 100-125 minutes
plt.hist(duration_of_movies_extract, bins=10, edgecolor='k')
plt.title('Histogram of Duration')
plt.xlabel('Duration (minutes)')
plt.ylabel('Frequency')
plt.show()
```




```
# Q 4.9
# Histogram for 'duration' for movies
# we can see that most of the tv shows are of 1-2 seasons only
plt.hist(duration_of_tv_extract, bins=10, edgecolor='k')
plt.title('Histogram of Duration')
plt.xlabel('Duration (minutes)')
plt.ylabel('Frequency')
plt.show()
```

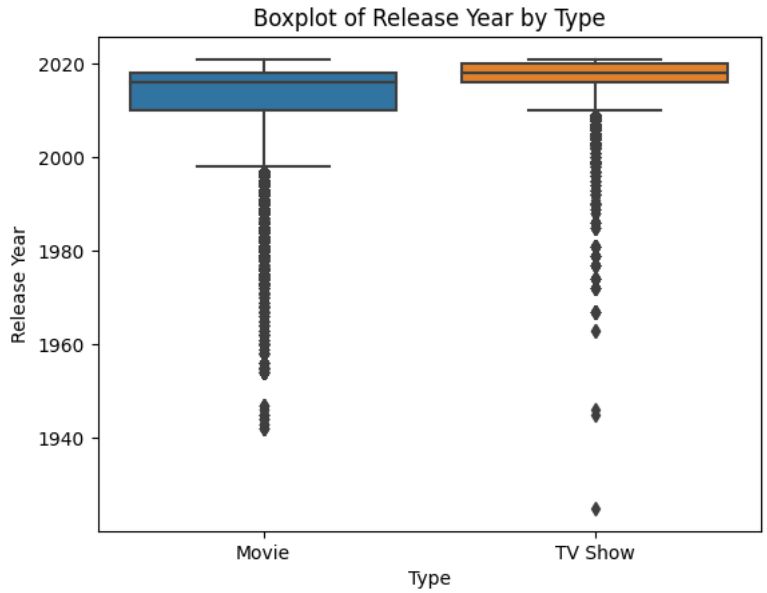



```
# 4.10
# conversion of categorical attributes to 'category'
data['type'] = data['type'].astype('category')
data['rating'] = data['rating'].astype('category')
data['listed_in'] = data['listed_in'].astype('category')
data.dtypes
```

```
show_id              object
type               category
title                object
director             object
cast                 object
country              object
date_added           object
```

```
release_year                      int64
rating                         category
duration                         object
listed_in                      category
description                      object
release_date_added_diff_movies    float64
release_date_added_diff_tv        float64
dtype: object
```

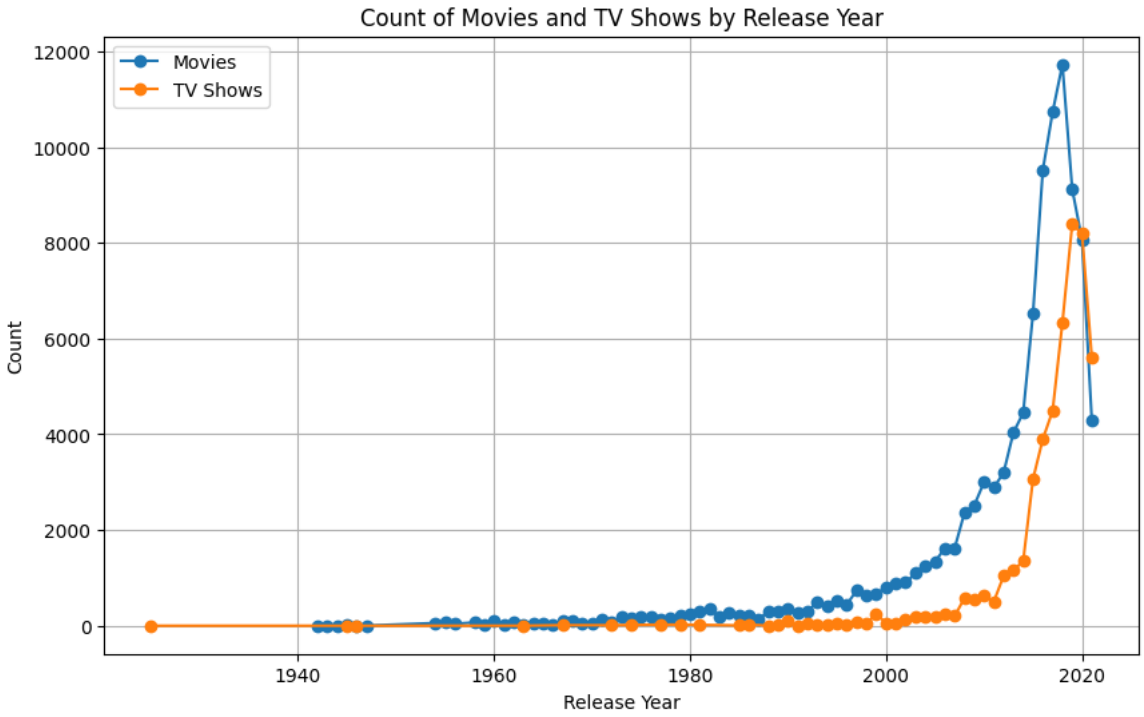
```
# Q 4.11
# Boxplot for 'type'
# TV Shows are being released more recently when relatively compared to movies
# we can also see the median line for tv shows being higher than that of movies
# which infer that most tv shows are produced in recent years than movies
sns.boxplot(x='type', y='release_year', data=data)
plt.title('Boxplot of Release Year by Type')
plt.xlabel('Type')
plt.ylabel('Release Year')
plt.show()
```




```
# Q 4.12
# Multiple boxplots for 'rating'
# movies/tv shows with G rating are more produced around the year 2000 and other content
# is more recent
sns.boxplot(x='rating', y='release_year', data=data)
plt.title('Boxplot of Release Year by Rating')
plt.xlabel('Rating')
plt.xticks(rotation=90)
plt.ylabel('Release Year')
plt.show()
```

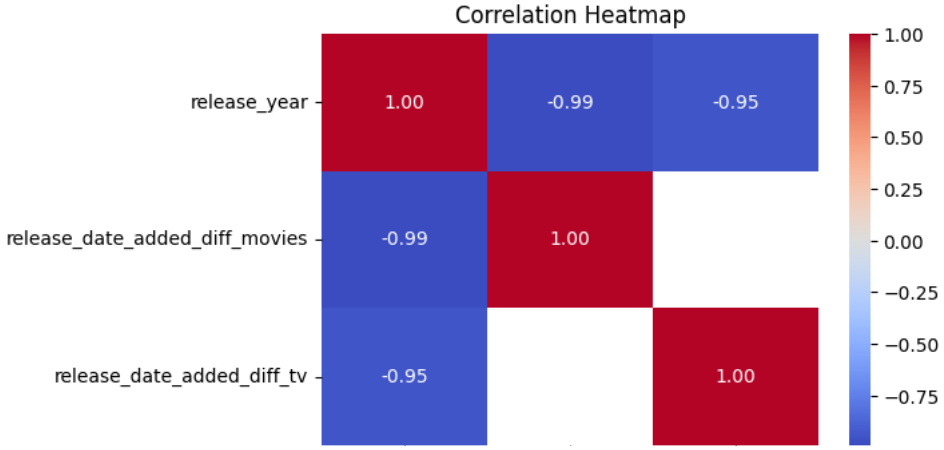
Boxplot of Release Year by Rating

```
# Q 4.13
# we can see the plots of both movies and tv shows and their count over the years
# Observation : Till 1980 both movies and tv shows are produced in almost same numbers after
# which movies took over
# and again in recent times i.e.,around the year 2020 both are produced in equal numbers

# Separate data for movies and TV shows
movies_df = data[data['type'] == 'Movie']
tv_shows_df = data[data['type'] == 'TV Show']

# Group data by 'release_year' and calculate counts for movies and TV shows
movies_count = movies_df['release_year'].value_counts().sort_index()
tv_shows_count = tv_shows_df['release_year'].value_counts().sort_index()

# Create the line plot
plt.figure(figsize=(10, 6))
plt.plot(movies_count.index, movies_count.values, label='Movies', marker='o')
plt.plot(tv_shows_count.index, tv_shows_count.values, label='TV Shows', marker='o')
plt.title('Count of Movies and TV Shows by Release Year')
plt.xlabel('Release Year')
plt.ylabel('Count')
plt.legend()
plt.grid(True)
plt.show()
```

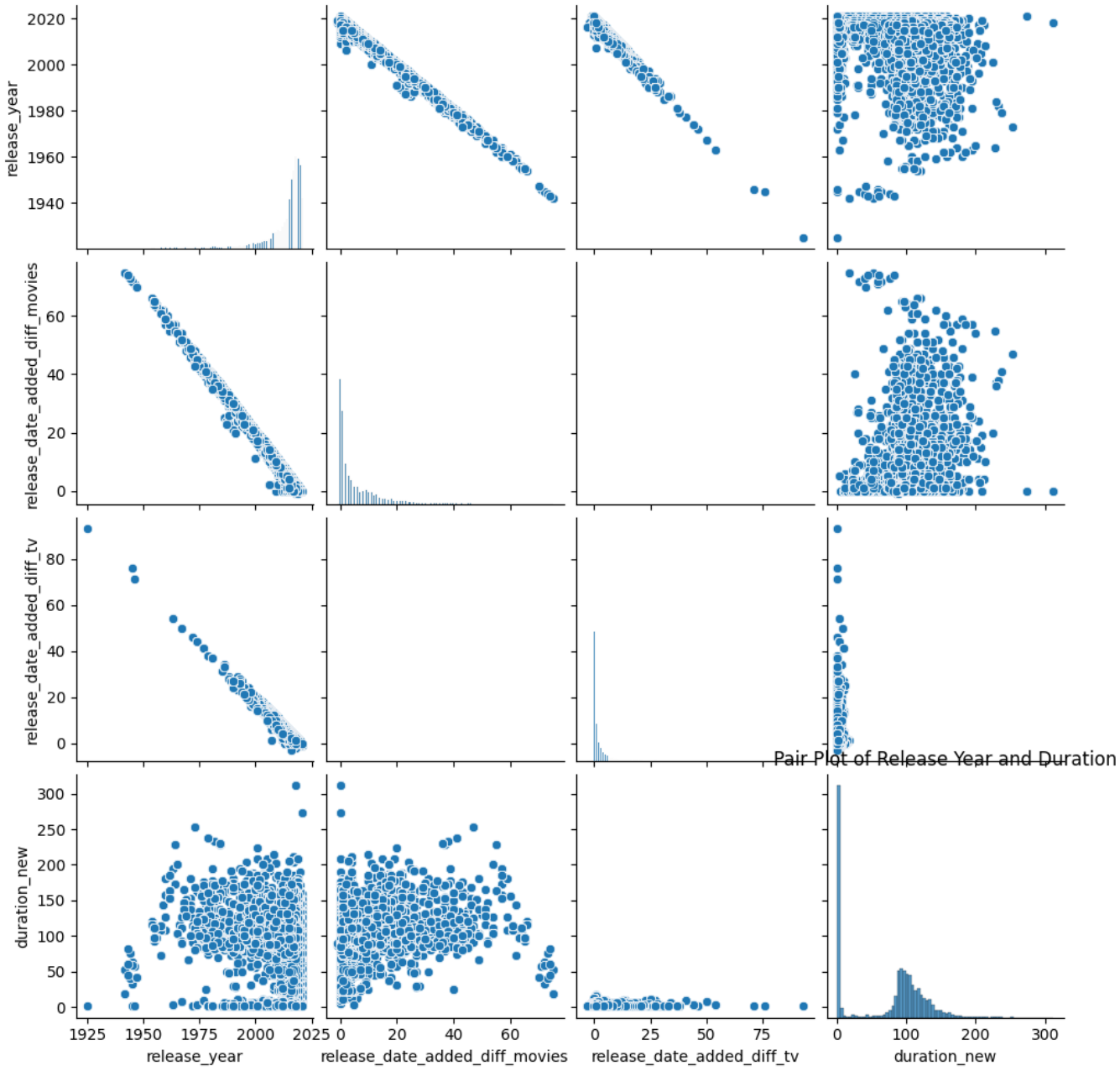



```
# Q 4.14
# Since the columns we used here are of same origin they have a very strong co-relation
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create the heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-64-f8d9cf1bdbc6>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a
  correlation_matrix = data.corr()
```

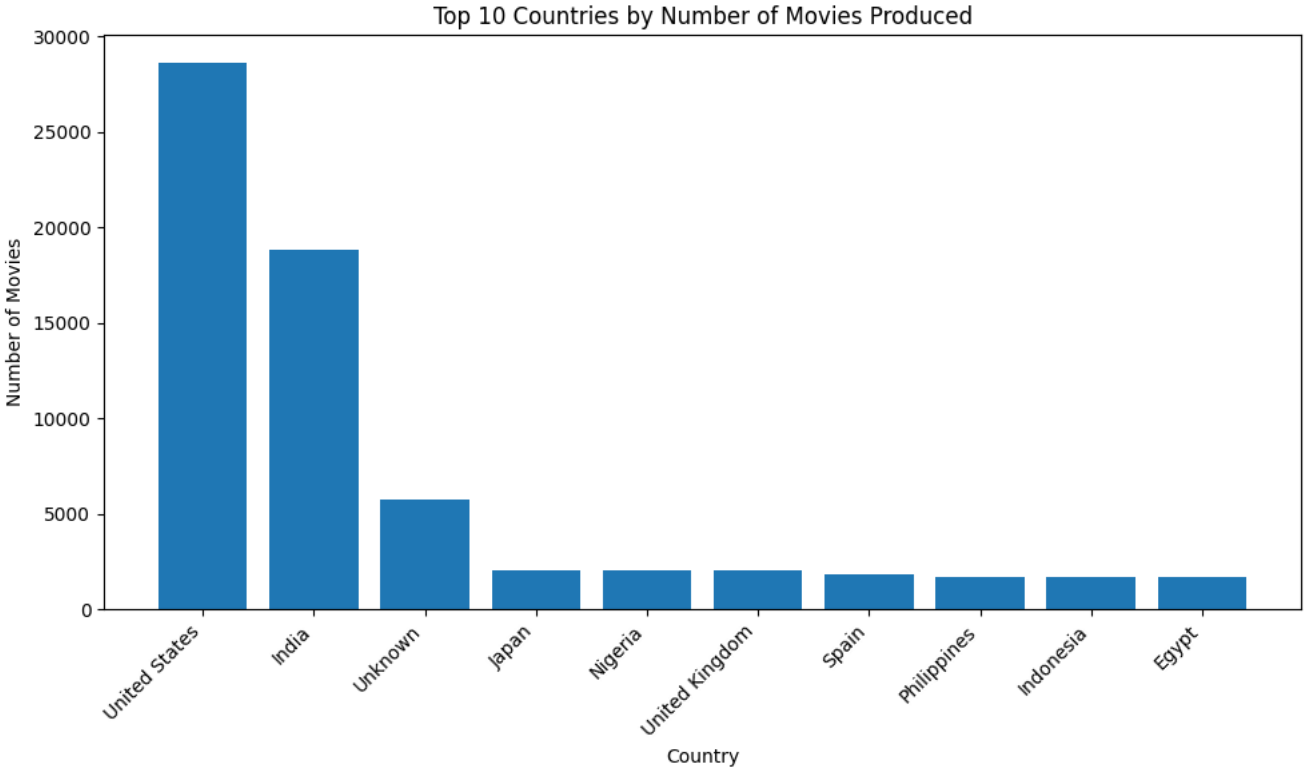



```
# Q 4.15
# All possible pair plots have been plotted here
# Create pair plot
sns.pairplot(data)
plt.title('Pair Plot of Release Year and Duration')
plt.show()
```

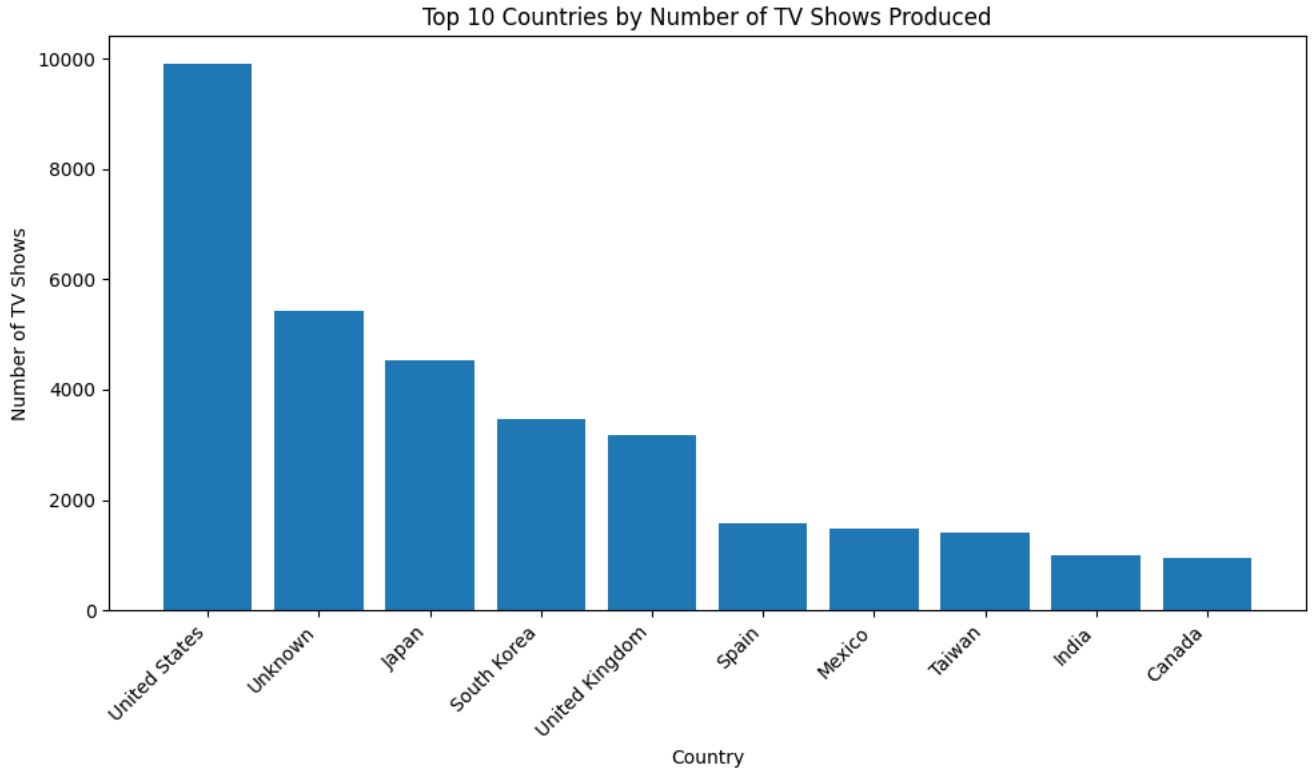


```
# Q 4.16
# word cloud of the column 'listen_in' i.e., genres
# Observation : This plot gives us insights into what genre is produced more as per the dataset
# International Movies, TV Shows are being produced more
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS

text = " ".join(i for i in data.listed_in)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure( figsize=(10,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

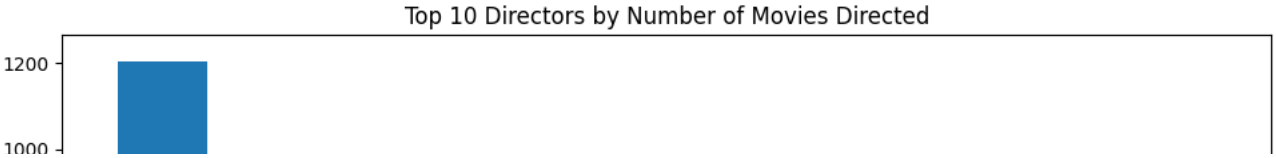

```
# Q 4.17
# Top 10 countries by number of movies
top_10_movies_by_country = data[data['type'] == 'Movie']['country'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_movies_by_country.index, top_10_movies_by_country.values)
plt.xlabel('Country')
plt.ylabel('Number of Movies')
plt.title('Top 10 Countries by Number of Movies Produced')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



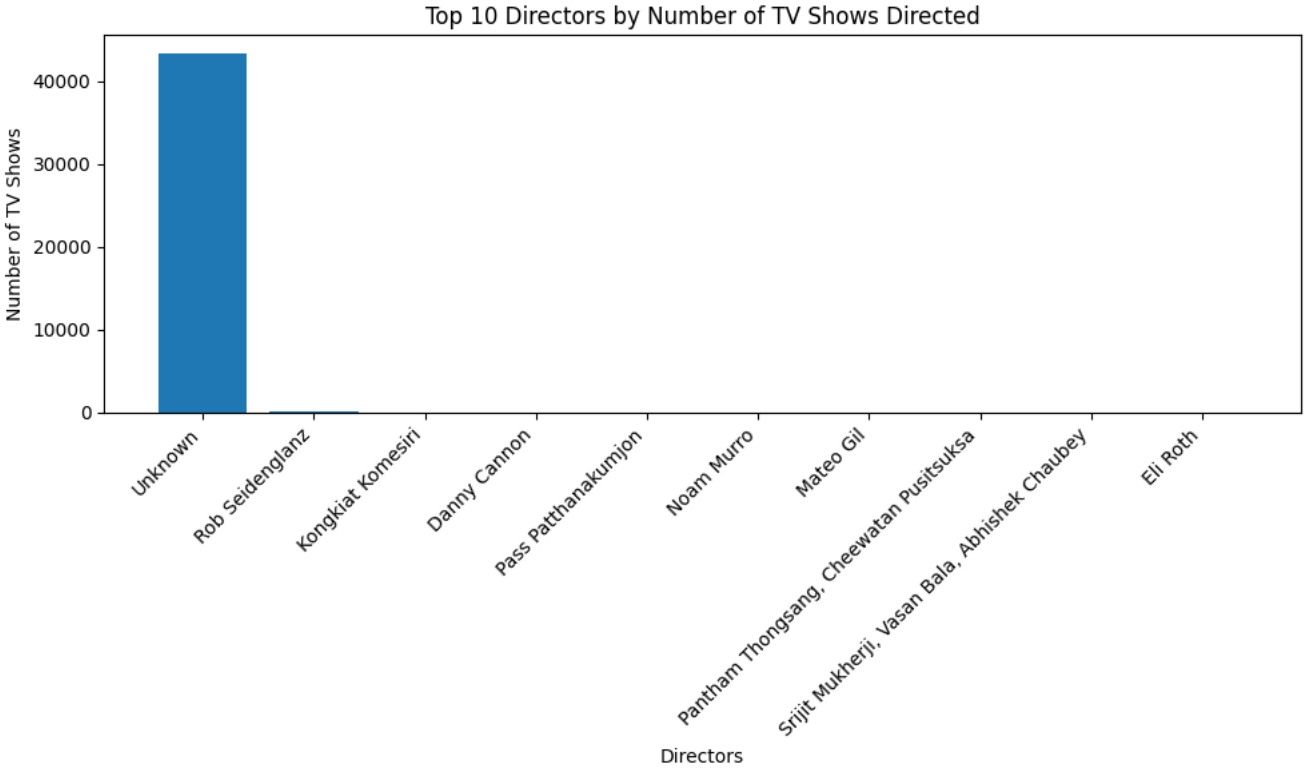
```
# Q 4.18
# Top 10 countries by number of tv shows
top_10_tv_shows_by_country = data[data['type'] == 'TV Show']['country'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_tv_shows_by_country.index, top_10_tv_shows_by_country.values)
plt.xlabel('Country')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Countries by Number of TV Shows Produced')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



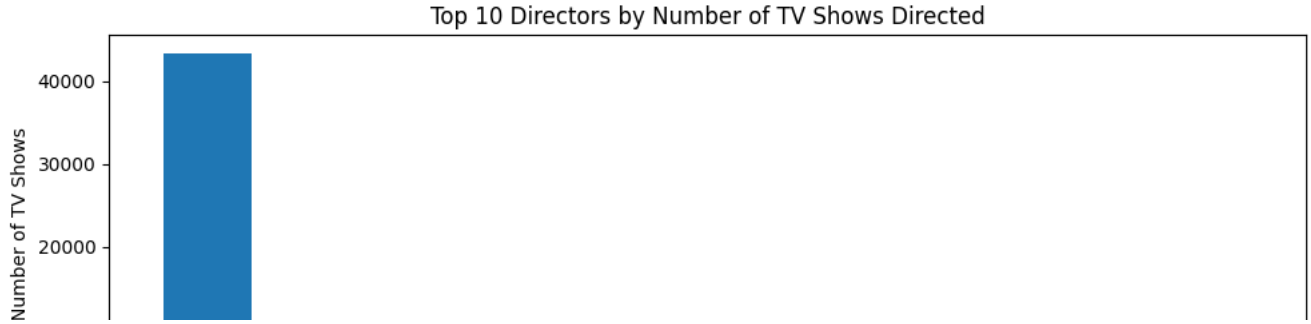

```
# Q 4.19
# Top 10 countries by number of movies
top_10_directors = data[data['type'] == 'Movie']['director'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_directors.index, top_10_directors.values)
plt.xlabel('Directors')
plt.ylabel('Number of Movies')
plt.title('Top 10 Directors by Number of Movies Directed')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



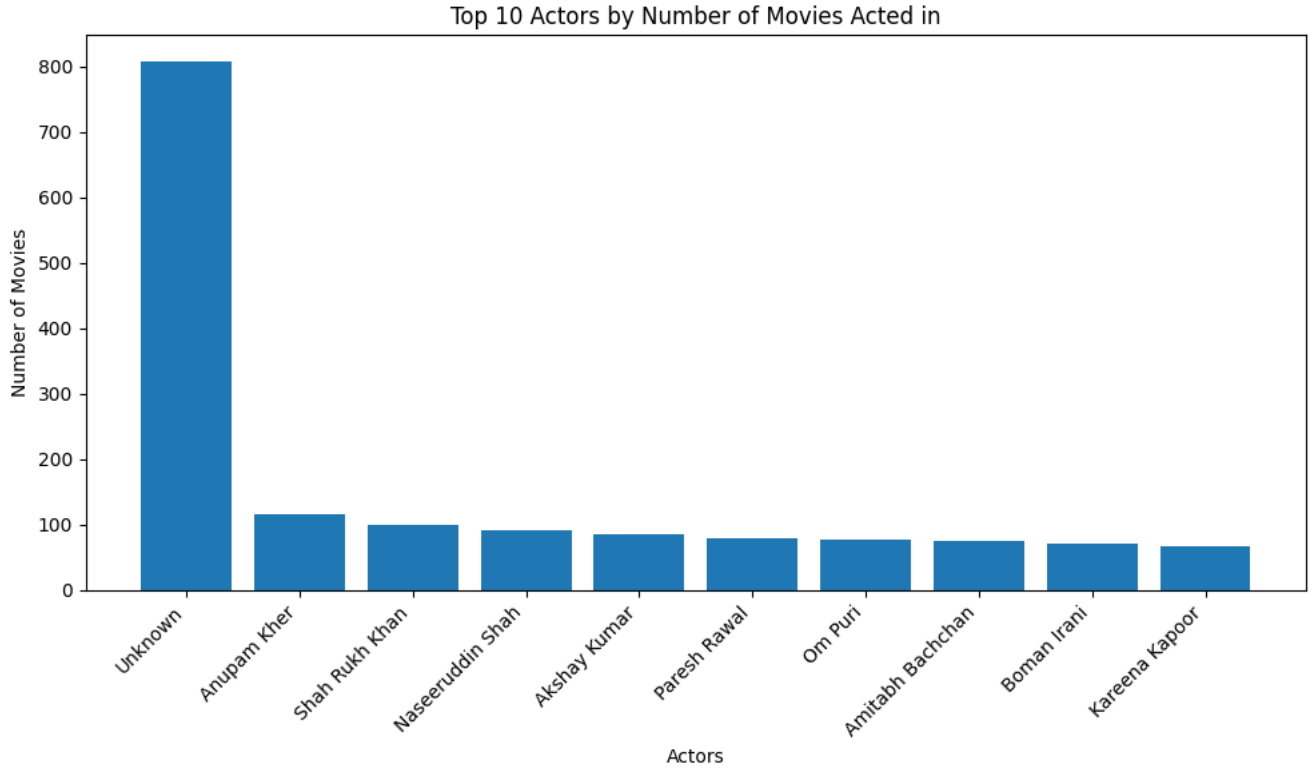

```
# Q 4.20
# Top 10 Directors by number of movies
top_10_directors_tv_shows = data[data['type'] == 'TV Show']['director'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_directors_tv_shows.index, top_10_directors_tv_shows.values)
plt.xlabel('Directors')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Directors by Number of TV Shows Directed')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
# Q 4.21
# Top 10 Directors by number of tv shows
top_10_directors_tv_shows = data[data['type'] == 'TV Show']['director'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_directors_tv_shows.index, top_10_directors_tv_shows.values)
plt.xlabel('Directors')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Directors by Number of TV Shows Directed')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

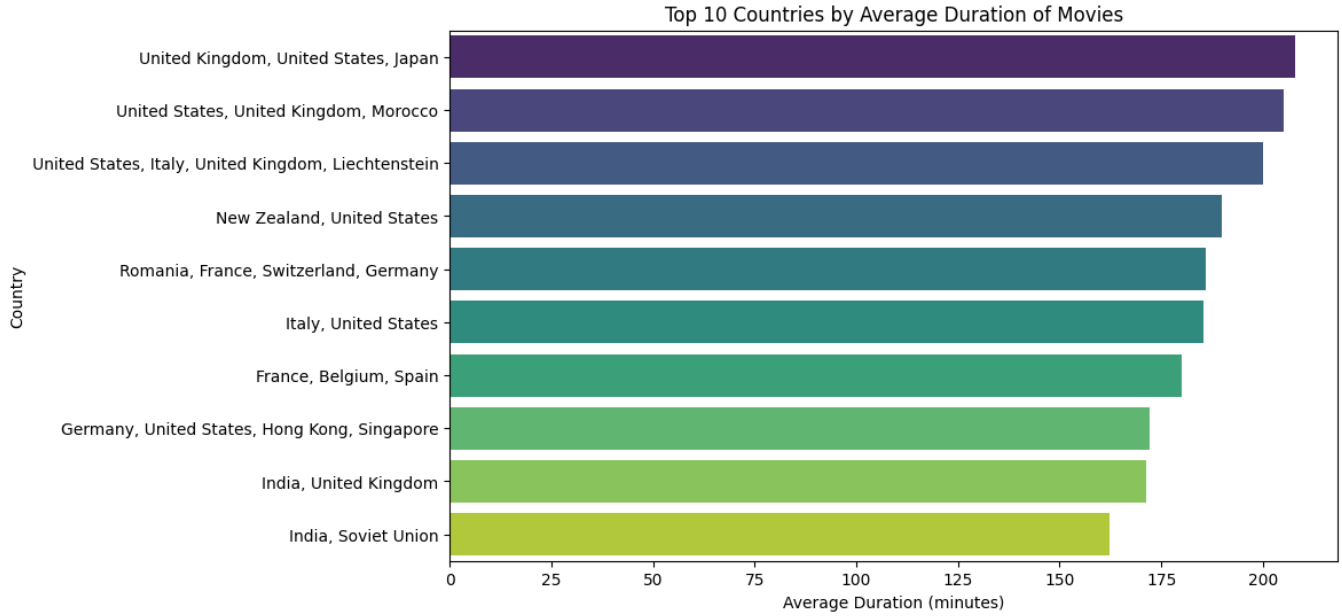



```
# Q 4.22
# Top 10 actors by number of movies
top_10_actors_by_movies = data[data['type'] == 'Movie']['cast'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_actors_by_movies.index, top_10_actors_by_movies.values)
plt.xlabel('Actors')
plt.ylabel('Number of Movies')
plt.title('Top 10 Actors by Number of Movies Acted in')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

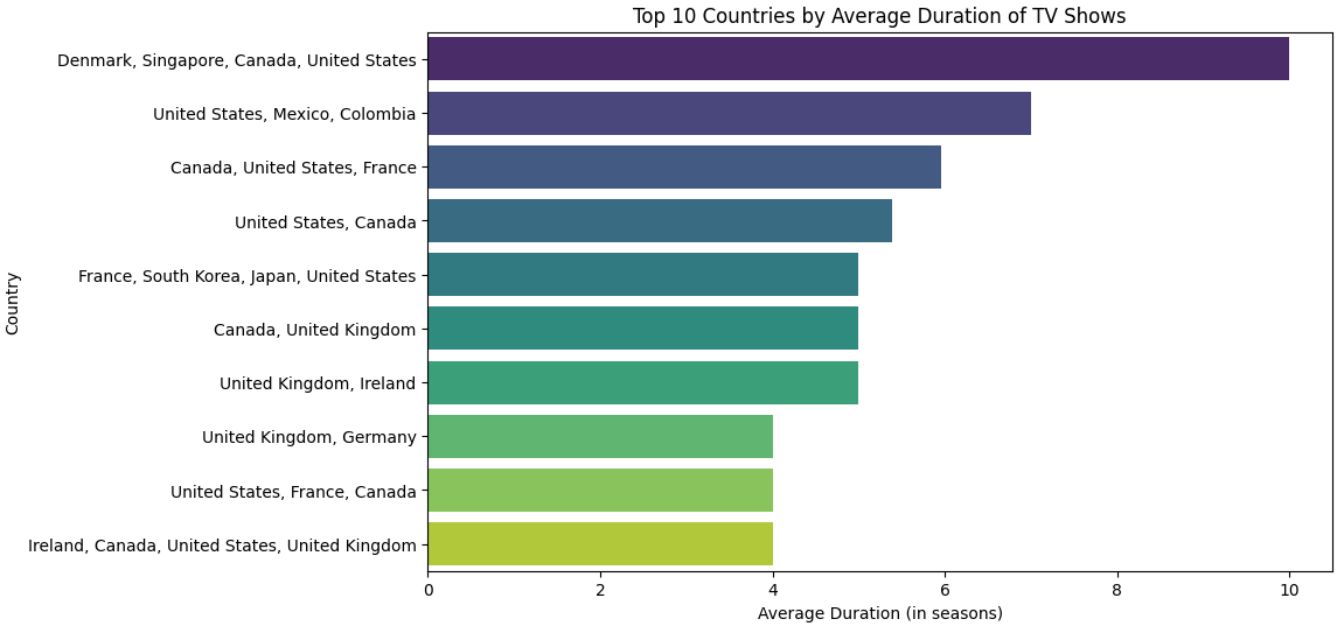



```
# Q 4.23
# Top 10 actors by number of tv shows
top_10_actors_by_tv_shows = data[data['type'] == 'TV Show']['cast'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_actors_by_tv_shows.index, top_10_actors_by_tv_shows.values)
plt.xlabel('Actors')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Actors by Number of TV Shows Acted in')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

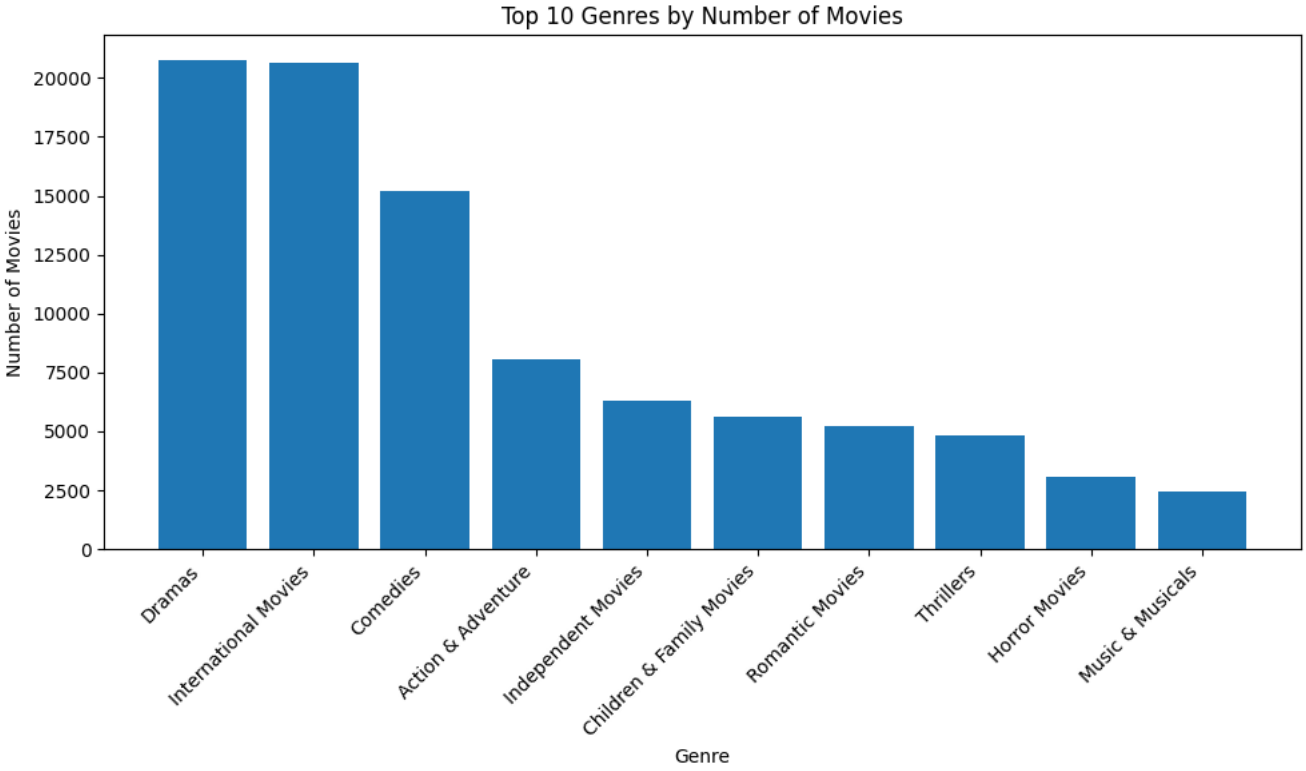

```
# Q 4.24
# Average duration of movies by country
def time_fn_split(x):
    if pd.notna(x):
        x = str(x)
        x = x.split(' ')
        return int(x[0])
    else:
        return None
data['duration_new'] = data['duration'].apply(time_fn_split)
avg_duration_of_movies_by_country = data[data['type'] == 'Movie'].groupby('country')['duration_new'].mean().sort_values(ascend
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_duration_of_movies_by_country.values, y=avg_duration_of_movies_by_country.index, palette='viridis')
plt.title('Top 10 Countries by Average Duration of Movies')
plt.xlabel('Average Duration (minutes)')
plt.ylabel('Country')
plt.show()
```



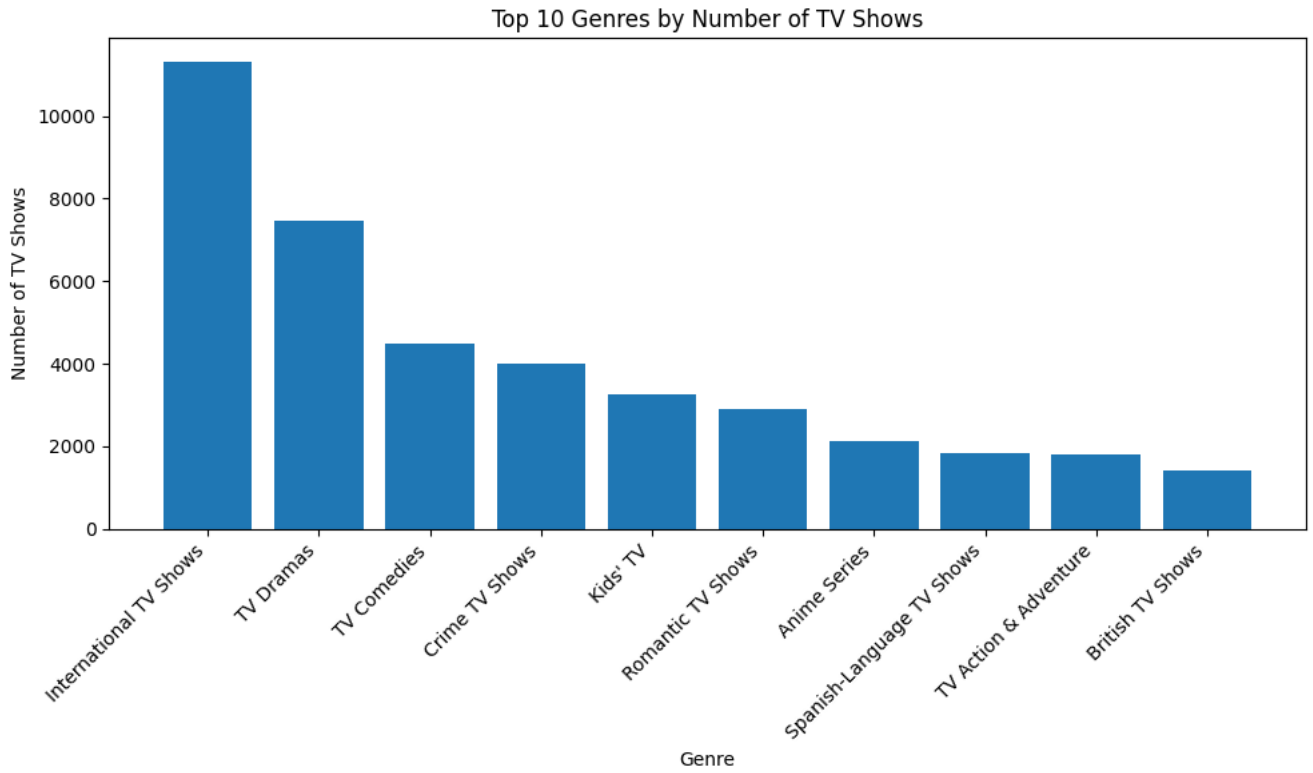

```
# Q 4.25
# Average duration of tv shows by country
avg_duration_of_tv_shows_by_country = data[data['type'] == 'TV Show'].groupby('country')['duration_new'].mean().sort_values(as
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_duration_of_tv_shows_by_country.values, y=avg_duration_of_tv_shows_by_country.index, palette='viridis')
plt.title('Top 10 Countries by Average Duration of TV Shows')
plt.xlabel('Average Duration (in seasons)')
plt.ylabel('Country')
plt.show()
```




```
# Q 4.26
# Top 10 genres by number of movies
top_10_genres_by_movies = data[data['type'] == 'Movie']['listed_in'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_genres_by_movies.index, top_10_genres_by_movies.values)
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.title('Top 10 Genres by Number of Movies')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
# Q 4.27
# Top 10 genres by number of movies
top_10_genres_by_tv_shows = data[data['type'] == 'TV Show']['listed_in'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.bar(top_10_genres_by_tv_shows.index, top_10_genres_by_tv_shows.values)
plt.xlabel('Genre')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Genres by Number of TV Shows')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
# Q 5
# Missing Value & Outlier check (Treatment optional)
```

```
# Q 5.1
# Number of missing values for each column
data.isnull().sum()
```

```
show_id                          0
type                             0
title                            0
director                         0
cast                             0
country                          0
date_added                     158
release_year                     0
rating                          67
duration                         3
listed_in                        0
description                      0
release_date_added_diff_movies   47820
release_date_added_diff_tv       101850
duration_new                     3
dtype: int64
```

```
# Q 5.2
# Treatment of missing values
# replacing missing values in columns director, cast, country with 'Unknown'
# replacing missing values in date_added column with mode (most occuring value) of that column
data['director'].fillna('Unknown',inplace=True)
data['cast'].fillna('Unknown',inplace=True)
data['country'].fillna('Unknown',inplace=True)
data['date_added'].fillna(data['date_added'].mode(),inplace=True)
```

```
# Q 5.3
# Treatment of missing values for rating - movies
# replacing missing values in rating column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
rating_mode_movies = data[data['type'] == 'Movie']['rating'].mode()
data.loc[data['type'] == 'Movie']['rating'].fillna(rating_mode_movies,inplace=True)
```

```
<ipython-input-143-34a5b99ac1c5>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'Movie']['rating'].fillna(rating_mode_movies,inplace=True)
```

```
# Q 5.4
# Treatment of missing values for rating - tv shows
# replacing missing values in rating column with mode (most occuring value) of that column
```

```
# we use the filter for column type = Movie or TV Show
rating_mode_tv_show = data[data['type'] == 'TV Show']['rating'].mode()
data.loc[data['type'] == 'TV Show']['rating'].fillna(rating_mode_tv_show,inplace=True)
```

```
<ipython-input-144-bfdcef16c8ba>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'TV Show']['rating'].fillna(rating_mode_tv_show,inplace=True)
```

```
# Q 5.5
# Treatment of missing values for duration - movies
# replacing missing values in duration column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
duration_mode_movie = data[data['type'] == 'Movie']['duration'].mode()
data.loc[data['type'] == 'Movie']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
<ipython-input-156-1227ebacfc96>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'Movie']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
# Q 5.6
# Treatment of missing values for duration - tv shows
# replacing missing values in duration column with mode (most occuring value) of that column
# we use the filter for column type = Movie or TV Show
duration_mode_movie = data[data['type'] == 'TV Show']['duration'].mode()
data.loc[data['type'] == 'TV Show']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
<ipython-input-155-626d52c672d4>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data.loc[data['type'] == 'TV Show']['duration'].fillna(duration_mode_movie,inplace=True)
```

```
# Q 5.7
# The number of NaN values reduced drastically
data.isnull().sum().head(11)
```

```
show_id           0
type              0
title             0
director          0
cast              0
country           0
date_added      158
release_year      0
rating           67
duration          3
listed_in         0
dtype: int64
```

```
# Q 5.8
# Outlier check
data.describe()
```

	release_year	release_date_added_diff_movies	release_date_added_diff_tv	duration_new
count	149512.000000	101692.000000	47662.000000	149509.000000
mean	2013.621482	6.922747	2.097436	73.823402
std	9.156160	10.338358	4.444949	53.619187
min	1925.000000	-1.000000	-3.000000	1.000000
25%	2012.000000	1.000000	0.000000	2.000000
50%	2017.000000	2.000000	0.000000	94.000000
75%	2019.000000	9.000000	2.000000	112.000000
max	2021.000000	75.000000	93.000000	312.000000

```
# Q 6
# Insights based on Non-Graphical and Visual Analysis
# Comments on the range of attributes
# Comments on the distribution of the variables and relationship between them
# Comments for each univariate and bivariate plot
```

```
# The insights and comments are mostly mentioned in the plots & comments in Q 4
# also I'm listing some observations here


# Q 6.1

# SOME OBSERVATIONS
# Stats like top 10 countries vs count of movies/tv shows, top 10 countries with most actors , top 10 countries with
# most directors is already plotted and the plot is self explanatory


# Q 7
# Business Insights
# 1. More number of movies are produced in USA followed by India
# 2. More number of TV Shows are produced in USA followed by Japan
# 3. Even African nations of Nigeria and Egypt come in top 10 countries by number of movies
# 4. In Asia India takes the top spot when it comes to movies but for TV Shows India is in 9th spot
# 5. In Europe UK takes the pole in both movies and TV Shows
# 6. The highest number of movies/tv shows is produced in the decade 2010-2020
# 7. Most movies span for 100 minutes
# 8. Most TV Shows span for 1-2 seasons
# 9. Movies/tv shows with 'G' rating are more produced around the year 2000 but now it is has become less when relatively
#    compared with other genres
# 10.Till 1980 both movies and tv shows are produced in almost same numbers after
#    which movies took over and again in recent times i.e.,around the year 2020 both are produced in equal numbers
# 11.International Movies, TV Shows genres are being produced more as per the word count plots


# Q 8
# 1. I'd advise Netflix to stop releasing content with rating 'G' as there are not many movies/tv shows of this rating
#     after 2000
# 2. Most number of movies span for 100 minutes and it is also evident from the Bell Curve for duration of movies vs count of
#     So, Netflix can release movies that have a duration of ~100 minutes
# 3. Similary for TV Shows the average span is just 1 season.So, I'd recommend Netflix to experiment with a TV Show for a
#     single season and it gets good reviews they can release subsequent seasons else they can drop that TV Show
# 4. Netflix should concentrate even on African countries of Nigeria & Egypt when it comes to releasing movies.
# 5. After 2020 there are equal number of releases for both movies and tv shows. So, Netflix should concentrate on
#    on both tv shows and movies equally after the COVID 19 pandemic.
```

Double-click (or enter) to edit