In [203…
```
import pandas as pd
import numpy as np
```

In [204…
```
#dataset importing to pythong
data = pd.read_csv("netflix.csv")
```

In [205…
```
#table structure
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
```

In [206…
```
#data cleaning - Epxlodng the various columns
data['cast'] = data['cast'].str.split(',')
exploded_data = data.explode('cast')
exploded_data['country'] = exploded_data['country'].str.split(',')
exploded_data = exploded_data.explode('country')
exploded_data['listed_in'] = exploded_data['listed_in'].str.split(',')
exploded_data = exploded_data.explode('listed_in')
exploded_data['director'] = exploded_data['director'].str.split(',')
exploded_data = exploded_data.explode('director')
exploded_data['duration'] = exploded_data['duration'].str.split(' ').str[0]
exploded_data['duration'] = exploded_data['duration'].astype('float')

exploded_data
```

Out[206]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA	
1	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA	
...	...	...	...	...	...	...	...	...	...	
8806	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	March 2, 2019	2015	TV-14	
8806	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	March 2, 2019	2015	TV-14	
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14	
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14	

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14	

202065 rows × 12 columns

In [220…
```
def fill_date(date):
  if pd.isna(date):
    return '9999-01-01'
  return date

exploded_data['date_added'] = pd.to_datetime(exploded_data['date_added'], errors='coer
exploded_data['date_added'] = exploded_data['date_added'].dt.strftime('%Y-%m-%d')
exploded_data['date_added'] = exploded_data['date_added'].apply(fill_date)
exploded_data['date_added'] = pd.to_datetime(exploded_data['date_added'], errors='coer
exploded_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 202065 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   show_id       202065 non-null  object        
 1   type          202065 non-null  object        
 2   title         202065 non-null  object        
 3   director      202065 non-null  object        
 4   cast          202065 non-null  object        
 5   country       202065 non-null  object        
 6   date_added    201907 non-null  datetime64[ns]
 7   release_year  202065 non-null  int64         
 8   rating        202065 non-null  object        
 9   duration      202065 non-null  float64       
 10  listed_in     202065 non-null  object        
 11  description   202065 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(1), object(9)
memory usage: 20.0+ MB
```

In [219…
```
#data imputation
exploded_data['director'] = exploded_data['director'].fillna('unknown_director')
exploded_data['cast'] = exploded_data['cast'].fillna('unknown_cast')
exploded_data['country'] = exploded_data['country'].fillna('unknown_country')
exploded_data['duration'] = exploded_data['duration'].fillna(0.0)
exploded_data['rating'] = exploded_data['rating'].fillna('unknown_rating')
exploded_data['date_added'] = exploded_data['date_added'].fillna('date_added')

exploded_data
```

Out[219]:

	show_id	type	title	director	cast	country	date_added	release_year	ra
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	unknown_cast	United States	2021-09-25 00:00:00	2020	P
1	s2	TV Show	Blood & Water	unknown_director	Ama Qamata	South Africa	2021-09-24 00:00:00	2021	
1	s2	TV Show	Blood & Water	unknown_director	Ama Qamata	South Africa	2021-09-24 00:00:00	2021	
1	s2	TV Show	Blood & Water	unknown_director	Ama Qamata	South Africa	2021-09-24 00:00:00	2021	
1	s2	TV Show	Blood & Water	unknown_director	Khosi Ngema	South Africa	2021-09-24 00:00:00	2021	
...	...	...	...	...	...	...	...	...	
8806	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	2019-03-02 00:00:00	2015	T
8806	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	2019-03-02 00:00:00	2015	T
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	2019-03-02 00:00:00	2015	T
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	2019-03-02 00:00:00	2015	T

	show_id	type	title	director	cast	country	date_added	release_year	ra
8806	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	2019-03-02 00:00:00	2015	T

202065 rows × 12 columns

In [209…
```
#Q) what are the number of movies every director has directed and who is the most popu
number_of_movies_per_director = exploded_data.groupby('director')['title'].nunique().s
number_of_movies_per_director
```

Out[209]:

director	title
unknown_director	2634
Rajiv Chilaka	22
Jan Suter	18
Raúl Campos	18
Suhas Kadav	16
...	...
J. Lee Thompson	1
J. Michael Long	1
Songyos Sugmakanan	1
Smriti Keshari	1
Joaquín Mazón	1

5121 rows × 1 columns

dtype: int64

In [210…
```
#Q) who is the most popular actor ?
popular_Actor = exploded_data.groupby('cast')['title'].nunique().sort_values(ascending
popular_Actor
```

Out[210]:

cast	title
unknown_cast	825
Anupam Kher	39
Rupa Bhimani	31
Takahiro Sakurai	30
Julie Tejwani	28
...	...
João Côrtes	1
João Assunção	1
Joziah Lagonoy	1
Jozef Gjura	1
Ṣọpẹ́ Dìrísù	1

39297 rows × 1 columns

dtype: int64

In [211…

```
#Q) who is the most popular actor-director combination ?
popular_actor_director= exploded_data.groupby(['director','cast'])['title'].nunique().
popular_actor_director = popular_actor_director[(popular_actor_director['director']!='
 (popular_actor_director['cast'] != 'unknown_cast')]
popular_actor_director.head(10)
```

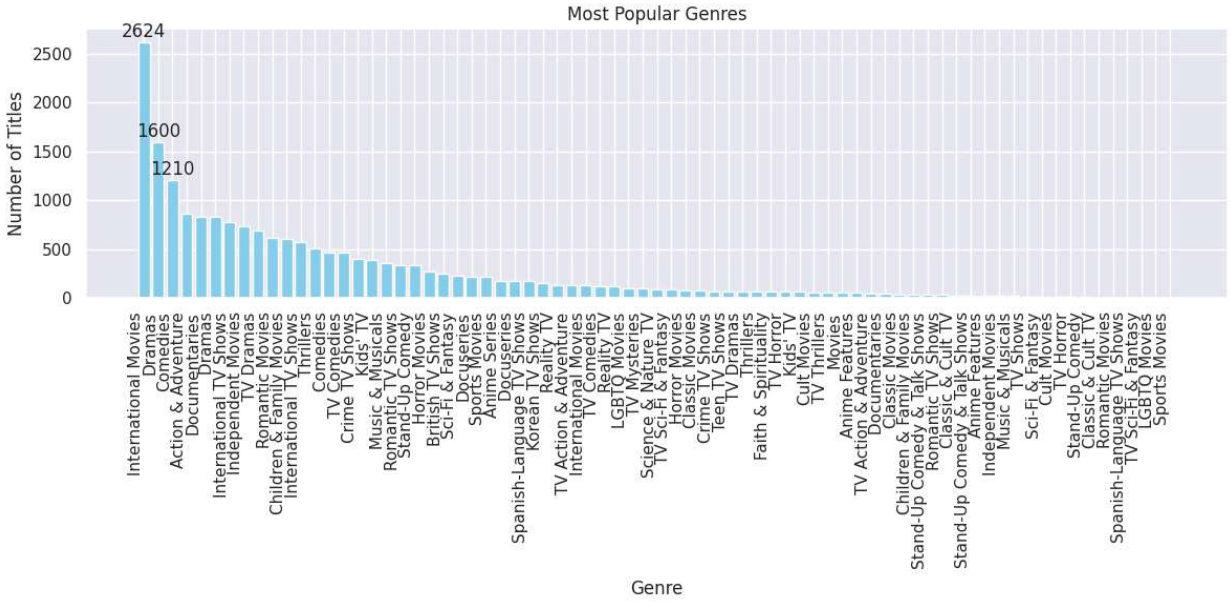
Out[211]:

	director	cast	title
2	Rajiv Chilaka	Rajesh Kava	19
3	Rajiv Chilaka	Julie Tejwani	19
4	Rajiv Chilaka	Rupa Bhimani	18
5	Rajiv Chilaka	Jigna Bhardwaj	18
7	Rajiv Chilaka	Vatsal Dubey	16
13	Rajiv Chilaka	Swapnil	13
17	Rajiv Chilaka	Mousam	13
66	Suhas Kadav	Saurav Chakraborty	8
86	Toshiya Shinohara	Satsuki Yukino	7
88	S.S. Rajamouli	Tamannaah Bhatia	7

In [212…

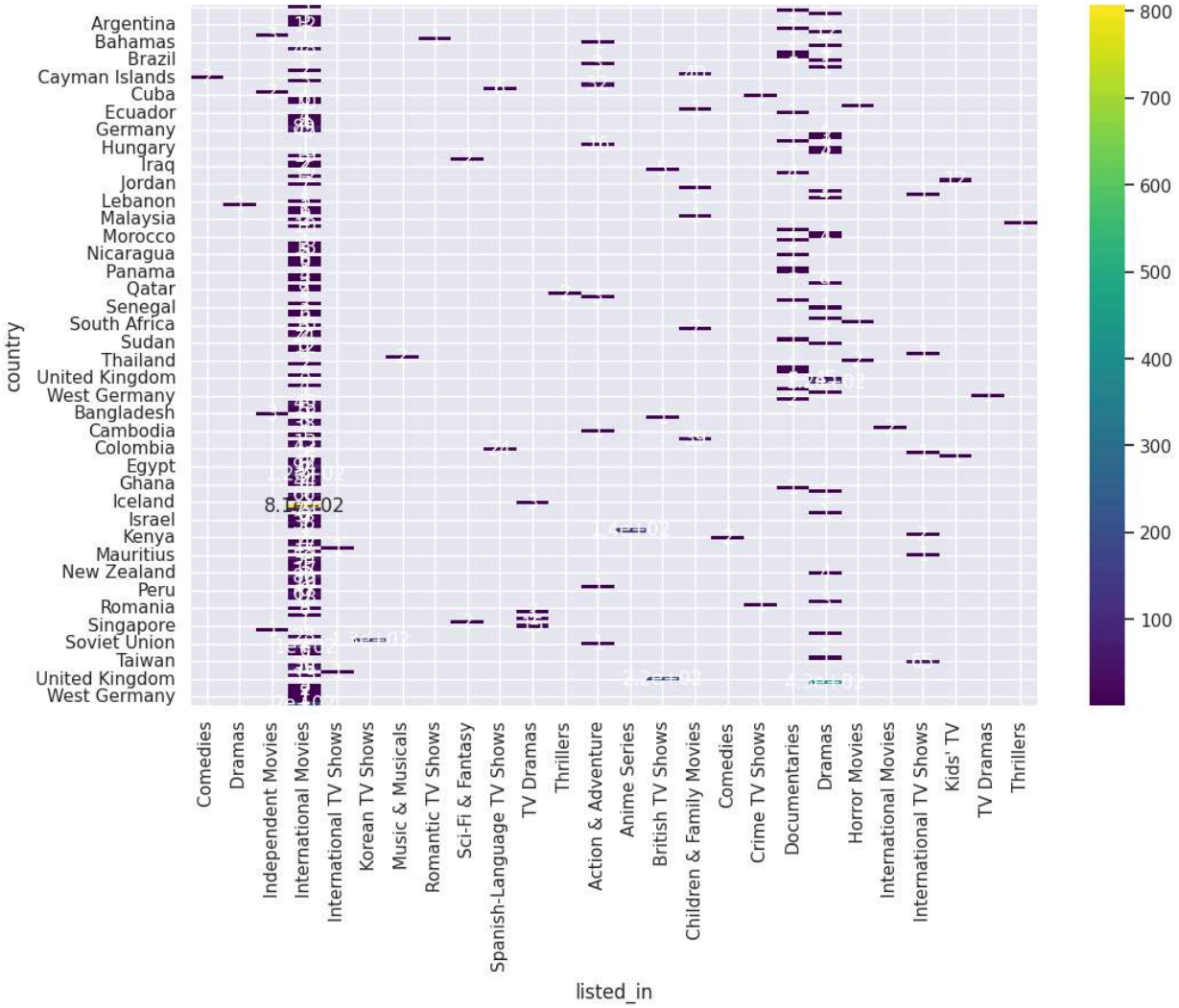
```
#Q) which is the most popular Genre
import matplotlib.pyplot as plt
popular_genre = exploded_data.groupby(['listed_in'])['title'].nunique().sort_values(as
```

```
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
plt.bar(popular_genre['listed_in'], popular_genre['title'], color='skyblue')
top_5_genre = popular_genre.nlargest(3, 'title')
# Customize the plot
plt.title('Most Popular Genres')
plt.xlabel('Genre')
plt.ylabel('Number of Titles')
plt.xticks(rotation=90, ha='right')  # Rotate x-axis labels for better readability
#adding values to the bars

for i, (genre, count) in enumerate(zip(top_5_genre['listed_in'], top_5_genre['title'])
        plt.annotate(str(count), xy=(i, count + 5), ha='center', va='bottom')
plt.tight_layout() # to prevent overlapping

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



In [213…

```
# Q) Country wise what is popular Genre
import seaborn as sns
import matplotlib.pyplot as plt

country_popular_genre = exploded_data.groupby(['country','listed_in'])['title'].nuniqu
country_popular_genre = country_popular_genre.drop_duplicates(subset='country',keep='f

# Pivot the DataFrame
heatmap_data = country_popular_genre.pivot(index='country', columns='listed_in', value

sns.set(rc={'figure.figsize': (12, 8)})  # Adjust the figure size as needed
sns.heatmap(heatmap_data, annot=True, cmap='viridis')  # Customize the cmap and annota
plt.show()
```



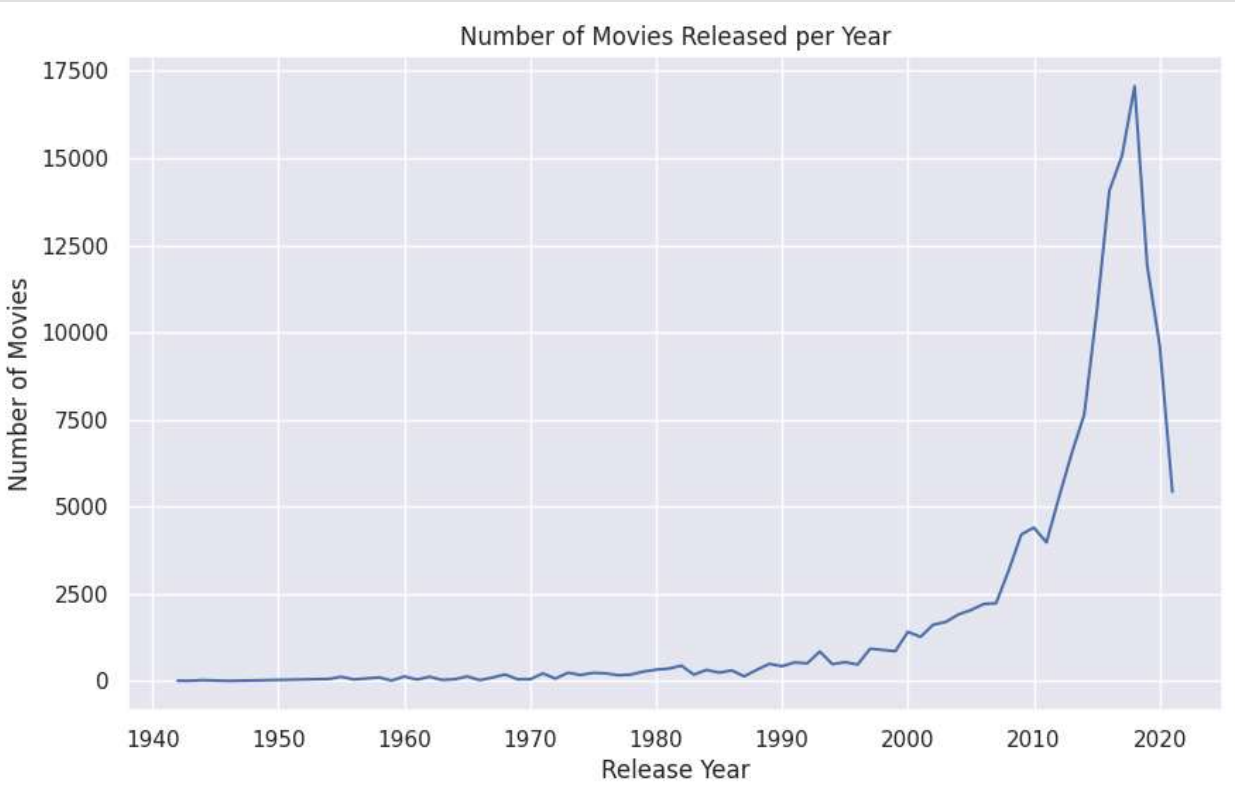
In [214…
```
# Q) Average runtime of movies.
avg_runtime_movies = exploded_data[exploded_data['type'] == 'Movie']['duration'].uniqu
avg_runtime_movies = np.round(avg_runtime_movies.mean(),3)
avg_runtime_movies
```

Out[214]: 109.859

In [215…
```
# Q) Average runtime of tv shows.
avg_runtime_movies = exploded_data[exploded_data['type'] == 'TV Show']['duration'].uni
avg_runtime_movies = np.round(avg_runtime_movies.mean(),3)
avg_runtime_movies
```

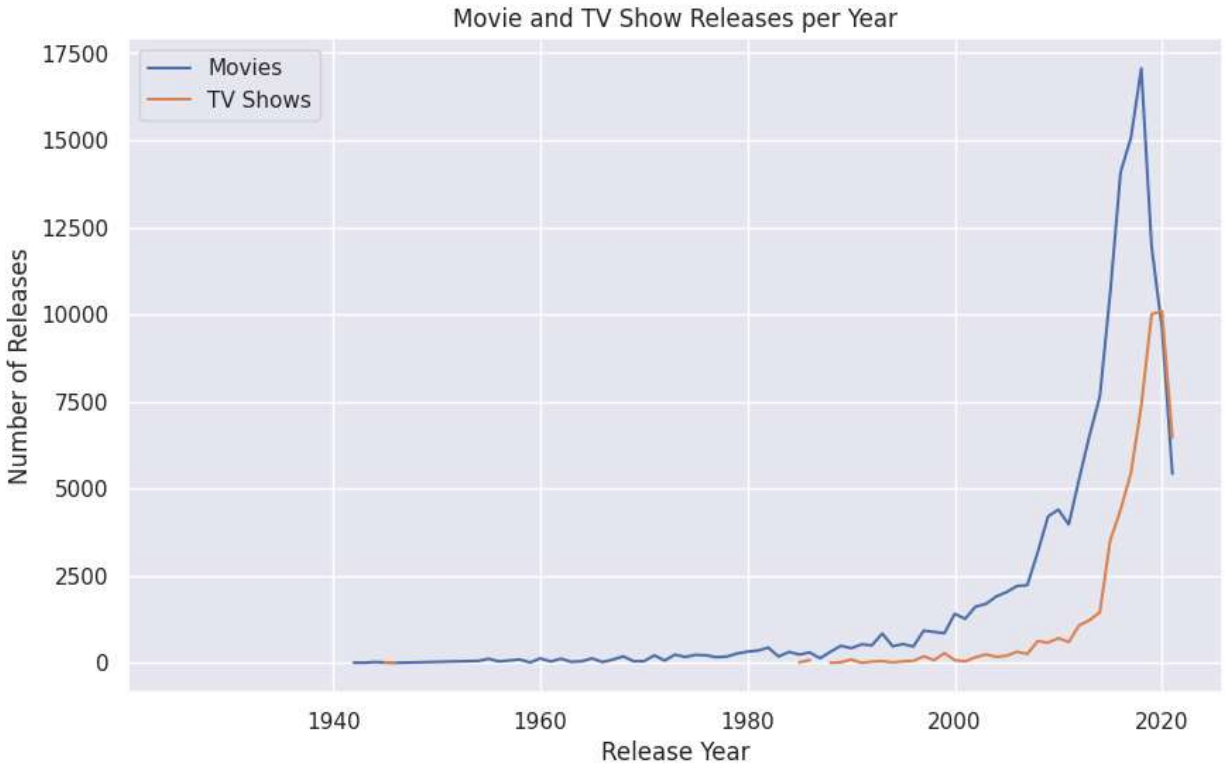
Out[215]: 8.2

In [216…
```
# Q)number of movies release year on year

movies_data = exploded_data[exploded_data['type'] == 'Movie']
movies_by_year = movies_data.groupby('release_year').size().reset_index(name='Movie_Co
plt.figure(figsize=(10, 6))
plt.plot(movies_by_year['release_year'], movies_by_year['Movie_Count'])
plt.title('Number of Movies Released per Year')
plt.xlabel('Release Year')
plt.ylabel('Number of Movies')
plt.grid(True)
plt.show()
```



In [217…

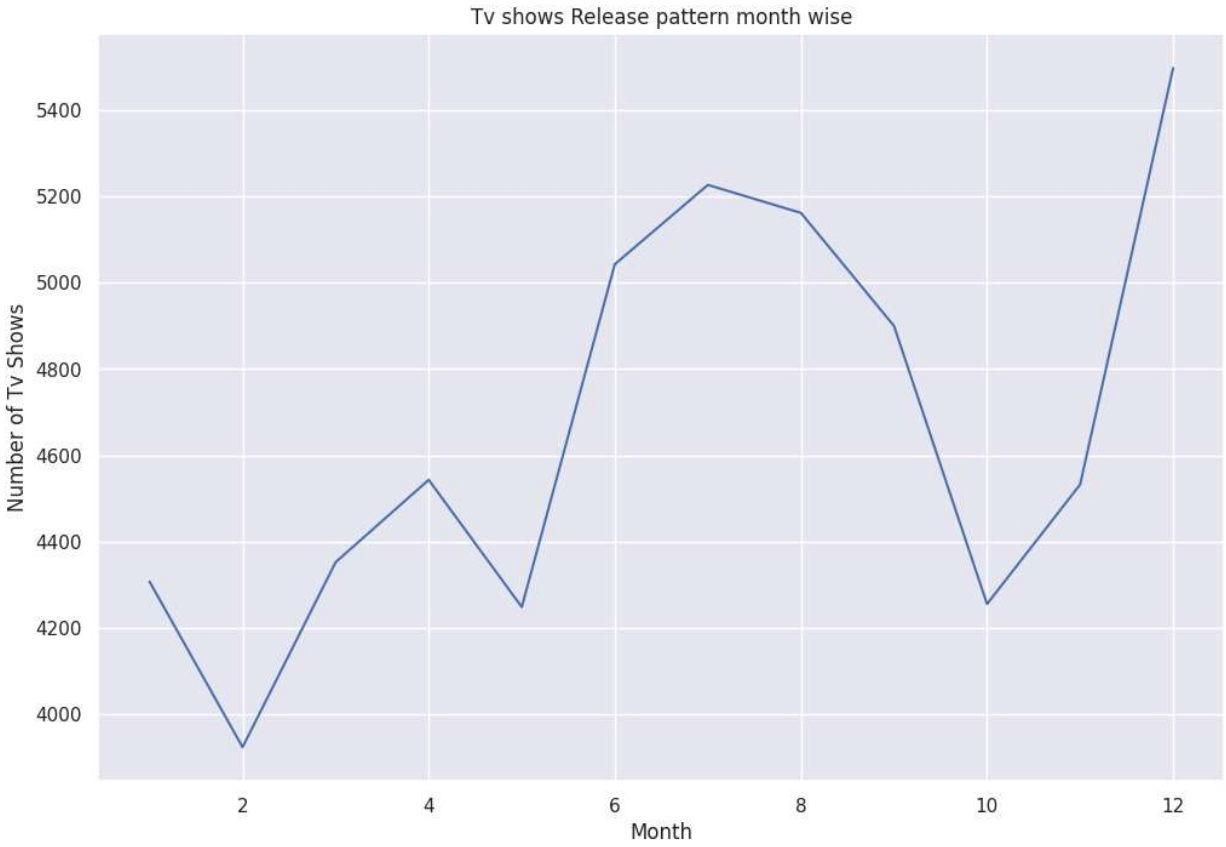
```
# Q) Comparison of Movies and Tv shows releases over the years
releases_by_type_year = exploded_data.groupby(['type', 'release_year']).size().reset_i
pivoted_data = releases_by_type_year.pivot(index='release_year', columns='type', value

plt.figure(figsize=(10, 6))
plt.plot(pivoted_data.index, pivoted_data['Movie'], label='Movies')
plt.plot(pivoted_data.index, pivoted_data['TV Show'], label='TV Shows')
plt.title('Movie and TV Show Releases per Year')
plt.xlabel('Release Year')
plt.ylabel('Number of Releases')
plt.legend()
plt.grid(True)
plt.show()
```



In [221…

```
# Q) Best time to launch tv show
tv_show_data = exploded_data[exploded_data['type'] == 'TV Show']
tv_show_data['Month'] = pd.to_datetime(tv_show_data['date_added']).dt.month


# Group by month and count movies
tvshows_by_month = tv_show_data.groupby('Month').size().reset_index(name='TVShow_Count

plt.plot(tvshows_by_month['Month'], tvshows_by_month['TVShow_Count'])
plt.xlabel('Month')
plt.ylabel('Number of Tv Shows')
plt.title('Tv shows Release pattern month wise')
plt.show()
```

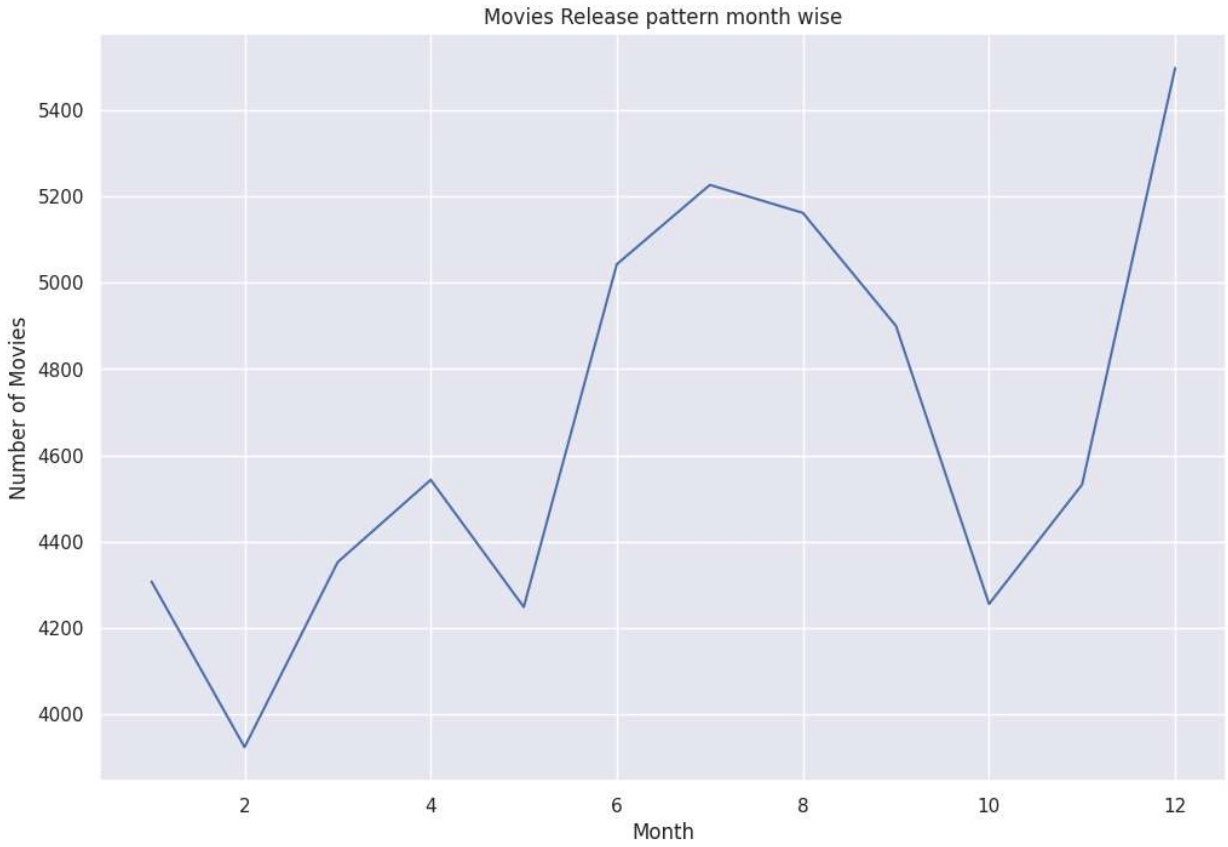
```
<ipython-input-221-cd7094e919de>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  tv_show_data['Month'] = pd.to_datetime(tv_show_data['date_added']).dt.month
```



In [222…

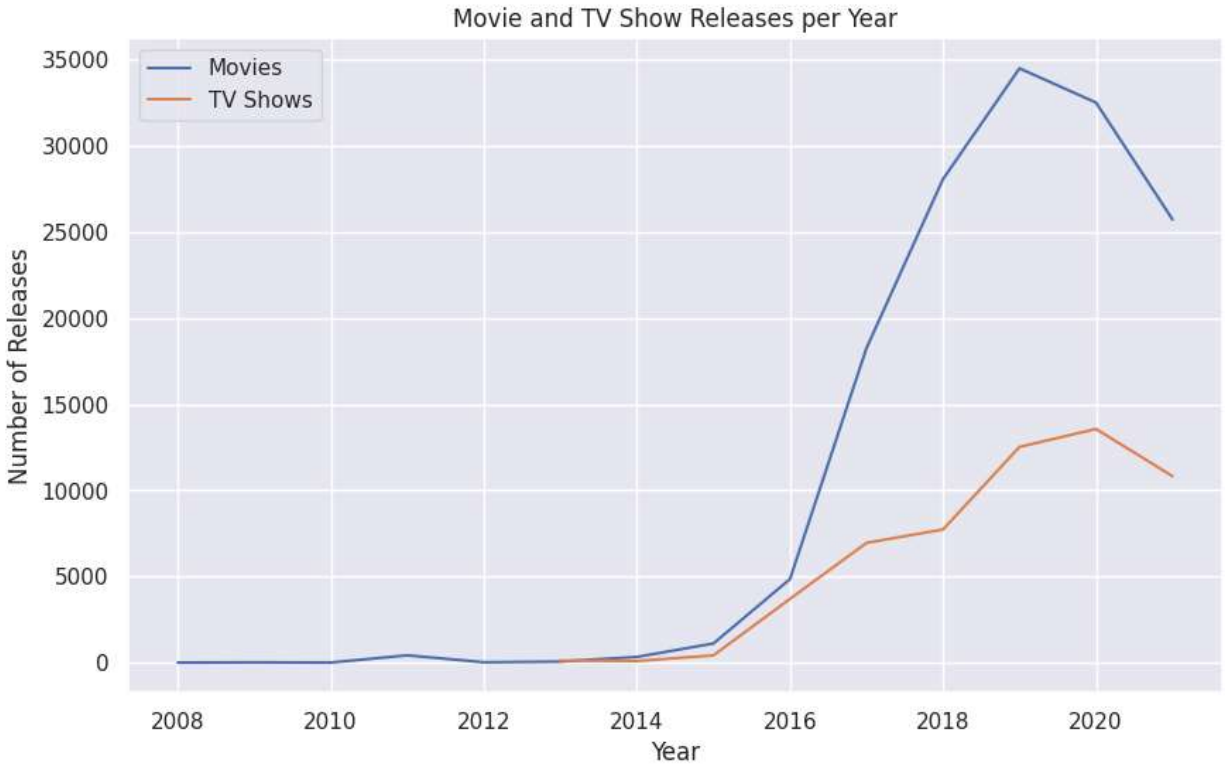
```
# Q) Best time to release Movies

movie_data = exploded_data[exploded_data['type'] == 'Movie']
movie_data['Month'] = pd.to_datetime(movie_data['date_added']).dt.month


# Group by month and count movies
movies_by_month = tv_show_data.groupby('Month').size().reset_index(name='Movie_Count')

plt.plot(movies_by_month['Month'], movies_by_month['Movie_Count'])
plt.xlabel('Month')
plt.ylabel('Number of Movies')
plt.title('Movies Release pattern month wise')
plt.show()
```

```
<ipython-input-222-3686622ccb72>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  movie_data['Month'] = pd.to_datetime(movie_data['date_added']).dt.month
```



In [224…

```
# Q) Does Netflix has more focus on TV Shows than movies in recent years

exploded_data['Year'] = pd.to_datetime(exploded_data['date_added']).dt.year

releases_by_type_year = exploded_data.groupby(['type', 'Year']).size().reset_index(nam
pivoted_data = releases_by_type_year.pivot(index='Year', columns='type', values='Count

plt.figure(figsize=(10, 6))
plt.plot(pivoted_data.index, pivoted_data['Movie'], label='Movies')
plt.plot(pivoted_data.index, pivoted_data['TV Show'], label='TV Shows')
plt.title('Movie and TV Show Releases per Year')
plt.xlabel('Year')
plt.ylabel('Number of Releases')
plt.legend()
plt.grid(True)
plt.show()
```



In [227…

```
# Q) Analyse which are the genres that are prefered by the popular actors.
# popular actors are defined as those who has acted in atleast 10 movies.
import seaborn as sns
movie_data = exploded_data[exploded_data['type'] == 'Movie']
popular_actors = movie_data['cast'].value_counts()[movie_data['cast'].value_counts() >

# Extract actor-genre pairs
actor_genre_pairs = movie_data[movie_data['cast'].isin(popular_actors)][['cast', 'list

# Count genre occurrences
genre_counts = pd.DataFrame(actor_genre_pairs, columns=['cast', 'listed_in']).groupby(

# Analyze genre preferences
print(genre_counts.idxmax(axis="columns"))
```

```
cast
 A.K. Hangal         International Movies
 Aakash Dabhade      International Movies
 Aamir Bashir        International Movies
 Aaron Abrams                  Thrillers
 Aaron Burns                Horror Movies
                            ...
Yılmaz Erdoğan       International Movies
Zac Efron              Independent Movies
Zeenat Aman          International Movies
Zoey Deutch                    Comedies
unknown_cast               Documentaries
Length: 3235, dtype: object
```

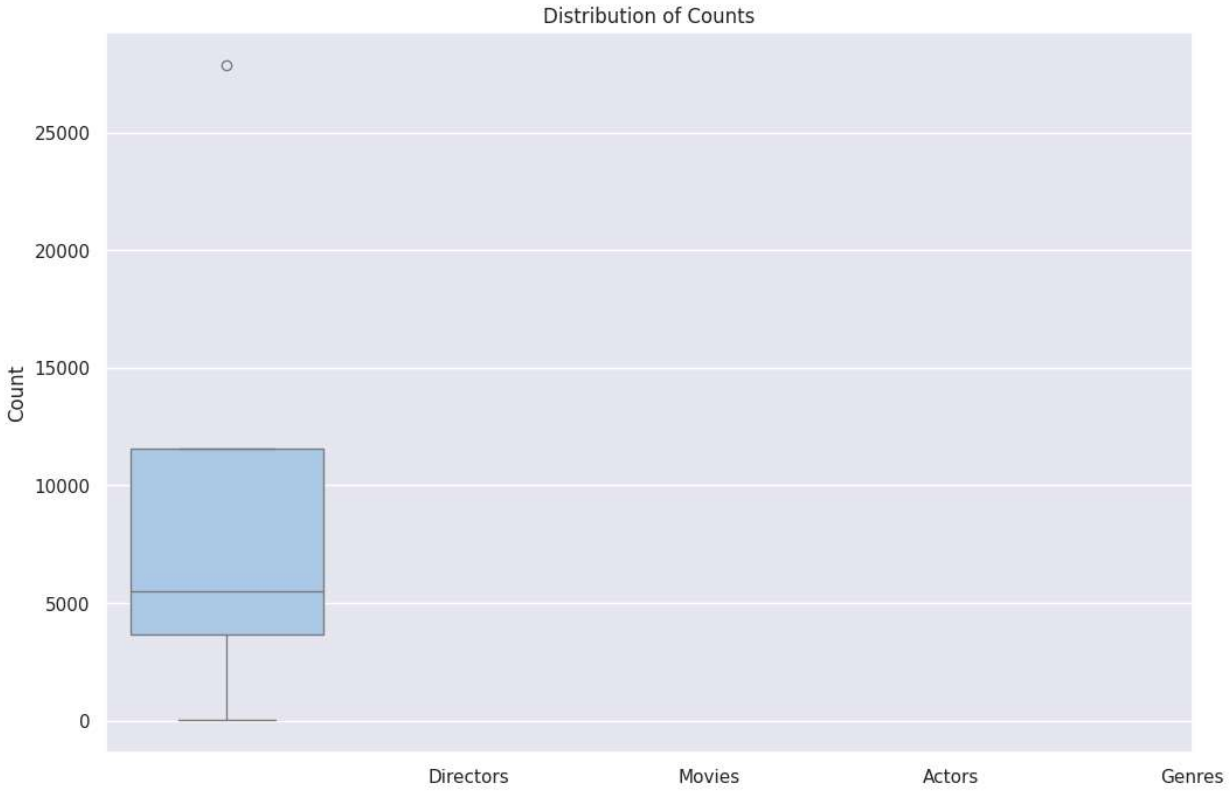
In [228…

```
# Q) Find the total number of Directors, Movies, Actors and Different generes present
import seaborn as sns
movie_data = exploded_data[exploded_data['type']=='Movie']
num_directors = movie_data['director'].nunique()
num_movies = movie_data['show_id'].nunique()
num_actors = movie_data['cast'].nunique()
```

```
num_genres = movie_data['listed_in'].nunique()
print("Number of directors:", num_directors)
print("Number of movies:", num_movies)
print("Number of actors:", num_actors)
print("Number of Genres:", num_genres)
data = [num_directors, num_movies, num_actors, num_genres]
sns.boxplot(data=data, palette="pastel")
plt.xticks([1, 2, 3, 4], labels=['Directors', 'Movies', 'Actors', 'Genres'])
plt.title('Distribution of Counts')
plt.ylabel('Count')

plt.show()
```

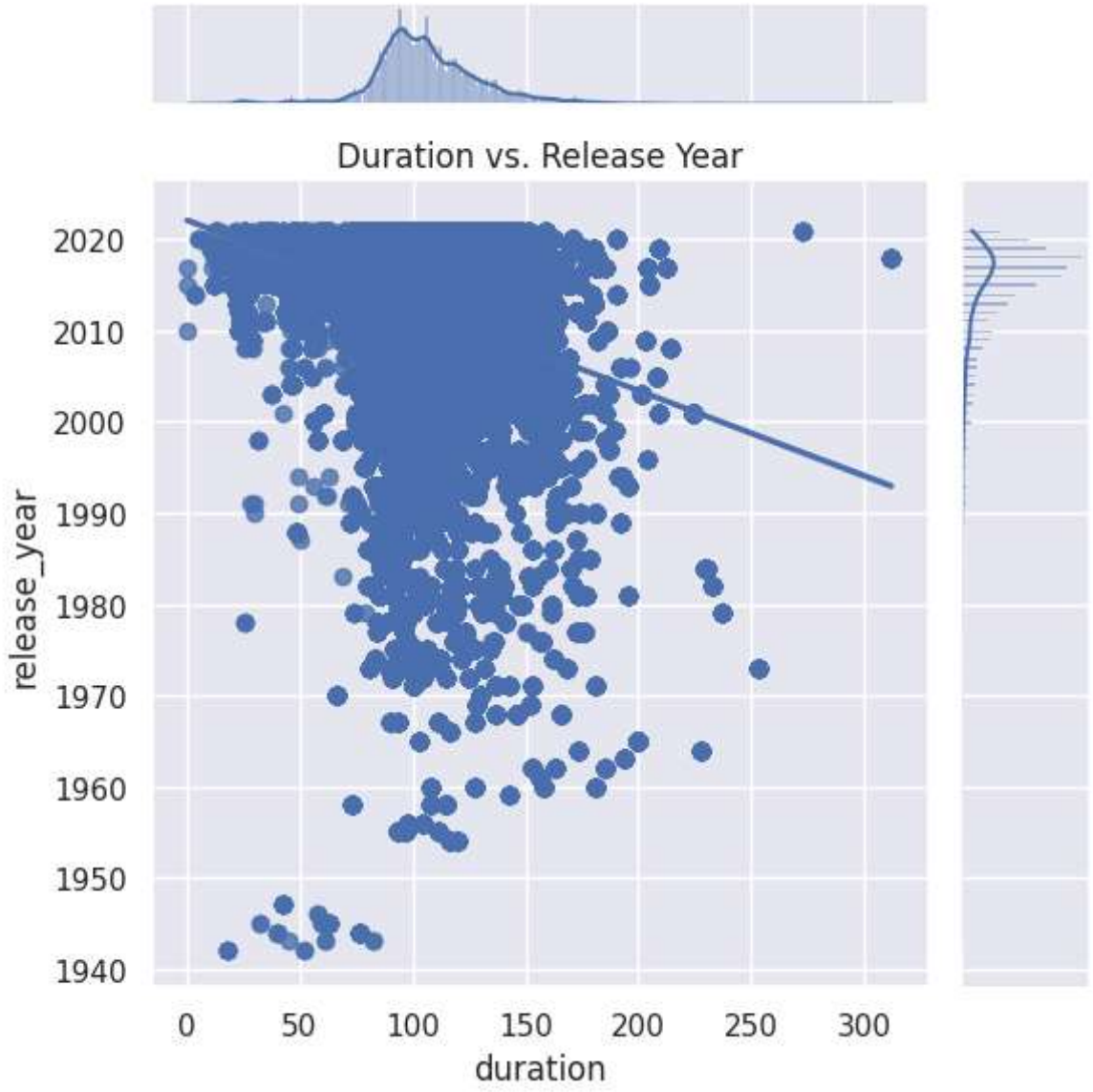
```
Number of directors: 4887
Number of movies: 6131
Number of actors: 27880
Number of Genres: 37
```

```
<ipython-input-228-6f1c5a3f5b72>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=data, palette="pastel")
```



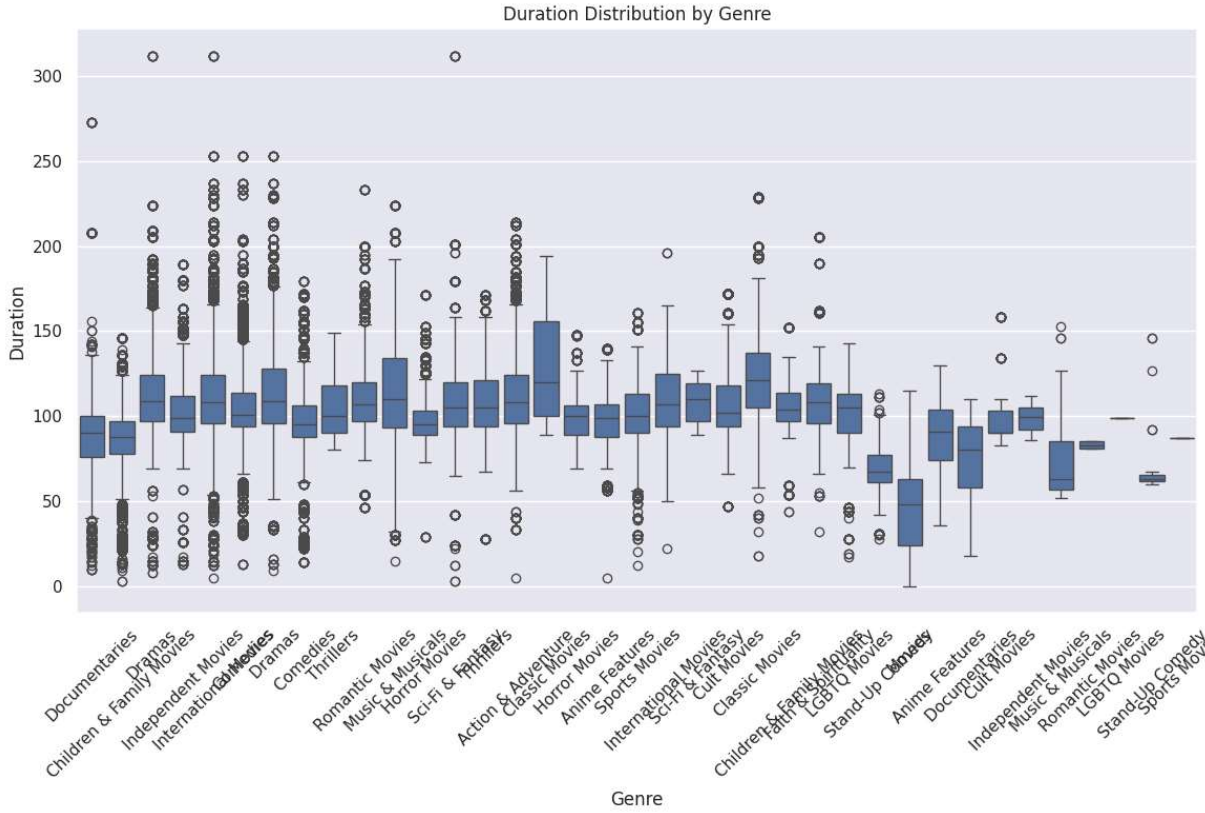
In [229…

```
# Q) Distribution of duration against various release year

movie_data = exploded_data[exploded_data['type']=='Movie']
sns.jointplot(x='duration', y='release_year', data=movie_data, kind='reg')
plt.title('Duration vs. Release Year')
plt.tight_layout()
plt.show()
```



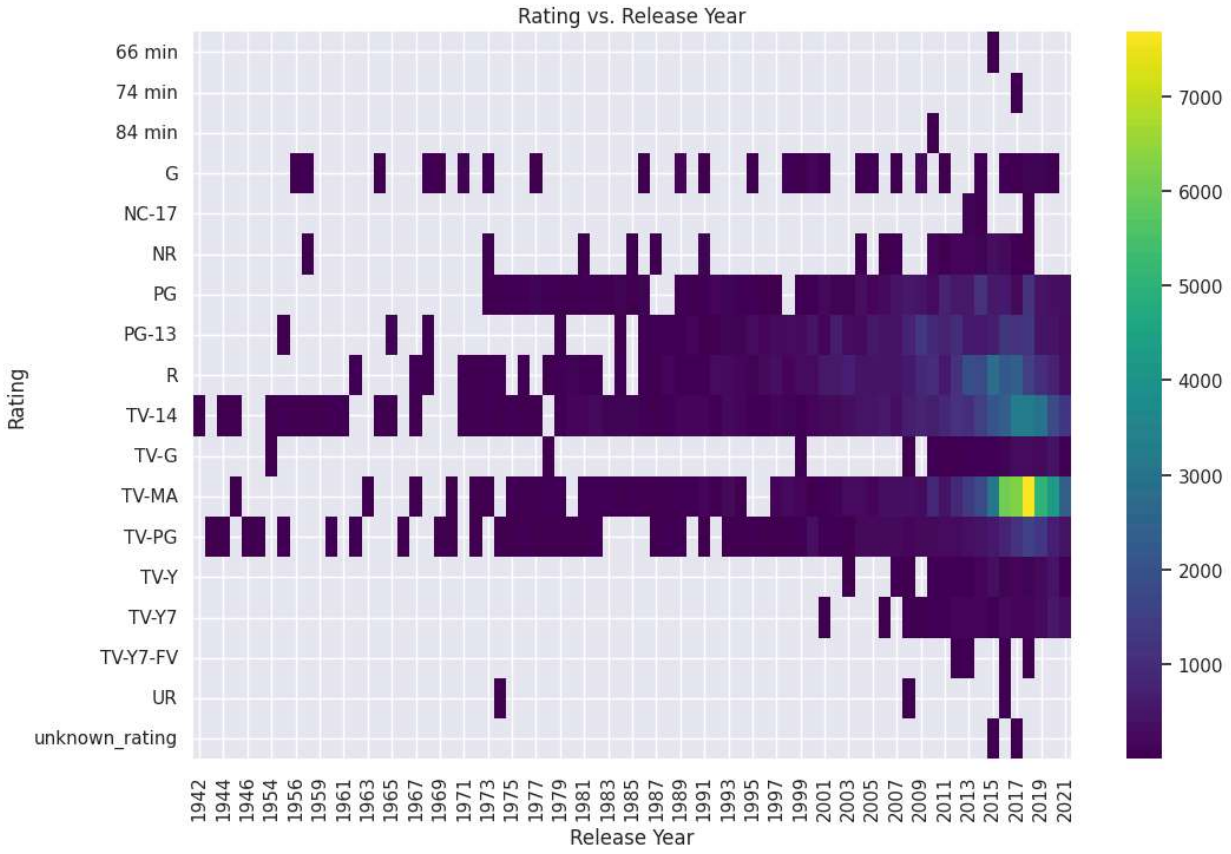
In [230…

```
# q) Find the relation between different genres and the duration of the movies
movie_data = exploded_data[exploded_data['type']=='Movie']

sns.boxplot(x='listed_in', y='duration', data=movie_data)
plt.title('Duration Distribution by Genre')
plt.xlabel('Genre')
plt.ylabel('Duration')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



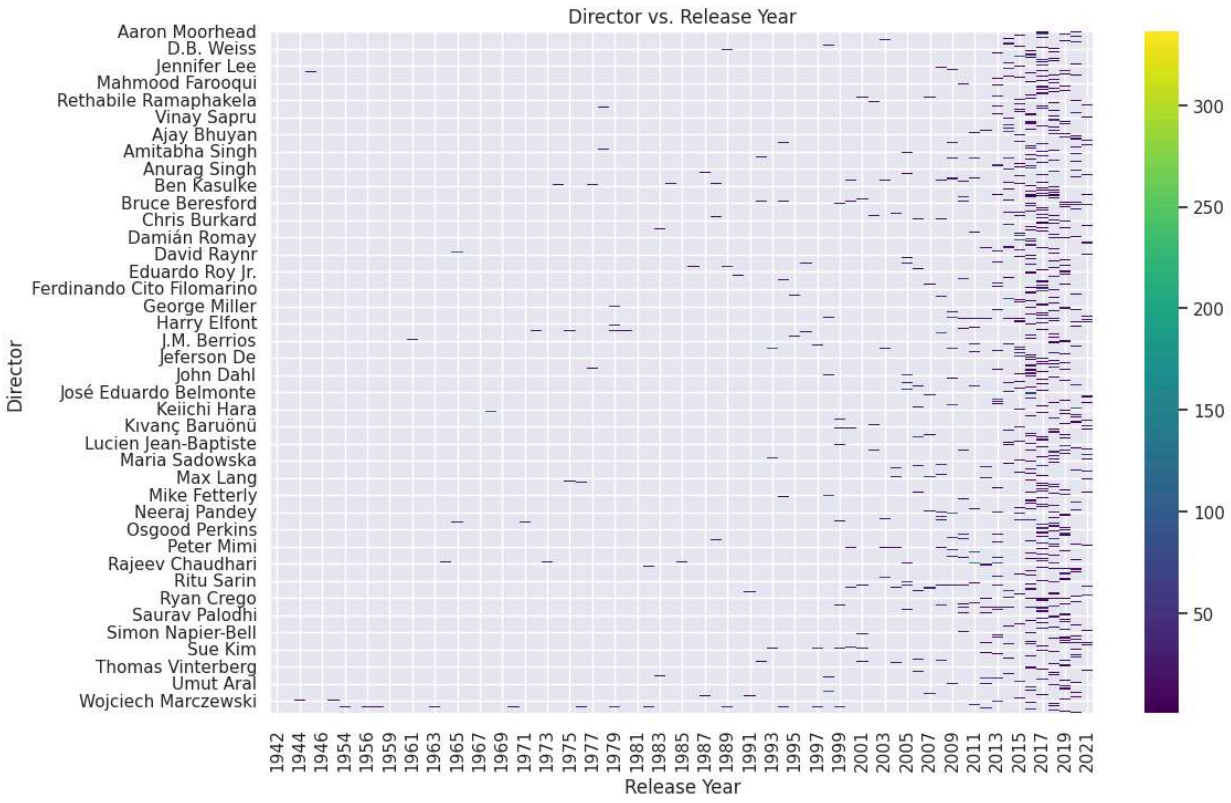
In [231…

```
# Q) How the rating has been changed over years
movie_data = exploded_data[exploded_data['type']=='Movie']

grouped_data = movie_data.groupby(['rating', 'release_year']).size().reset_index(name=
pivoted_data = grouped_data.pivot(index='rating', columns='release_year', values='Cour
sns.heatmap(pivoted_data, cmap='viridis', annot=False)
plt.title('Rating vs. Release Year')
plt.xlabel('Release Year')
plt.ylabel('Rating')
plt.show()
```



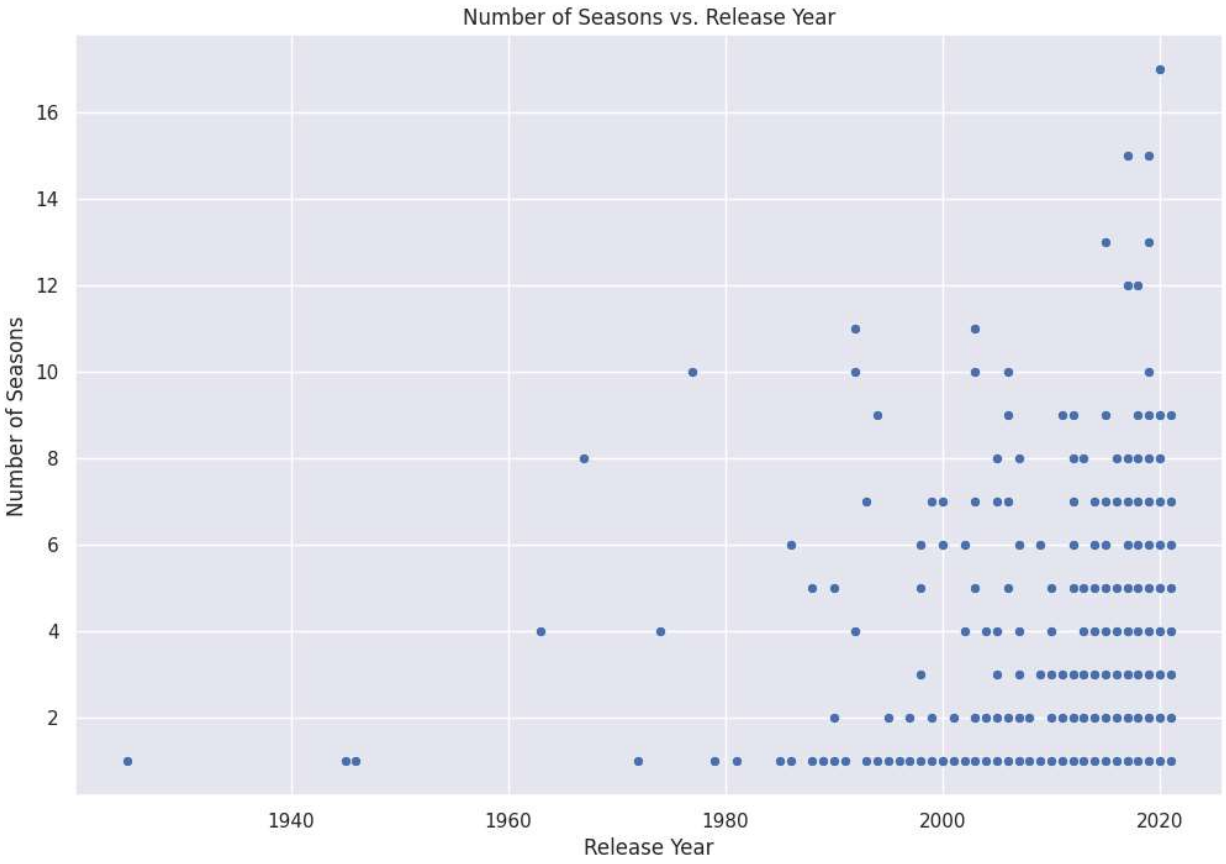
In [232…

```
# Q) Productivity of directors across the years

movie_data = exploded_data[exploded_data['type']=='Movie']

grouped_data = movie_data.groupby(['director', 'release_year']).size().reset_index(nam
pivoted_data = grouped_data.pivot(index='director', columns='release_year', values='Co
sns.heatmap(pivoted_data, cmap='viridis', annot=False)
plt.title('Director vs. Release Year')
plt.xlabel('Release Year')
plt.ylabel('Director')
plt.show()
```



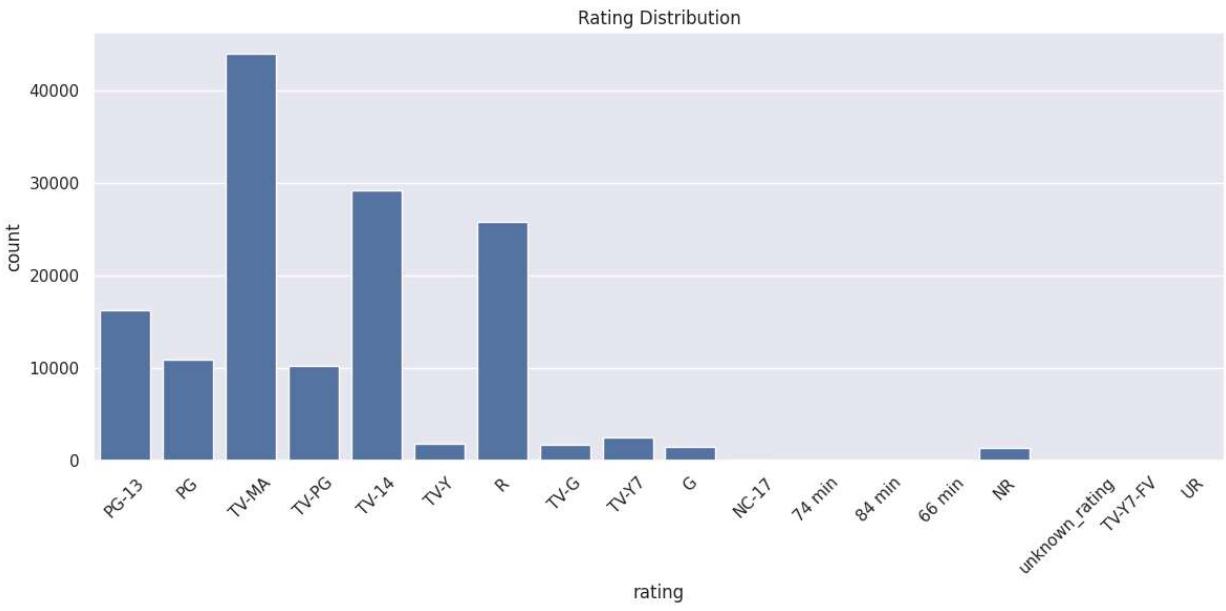
In [233…

```
# Q)  For TV shows, examine the relationship between the number of seasons and release
tv_shows = exploded_data[exploded_data['type'] == 'TV Show']
sns.scatterplot(x='release_year', y='duration', data=tv_shows)
plt.title('Number of Seasons vs. Release Year')
plt.xlabel('Release Year')
plt.ylabel('Number of Seasons')
plt.show()
```



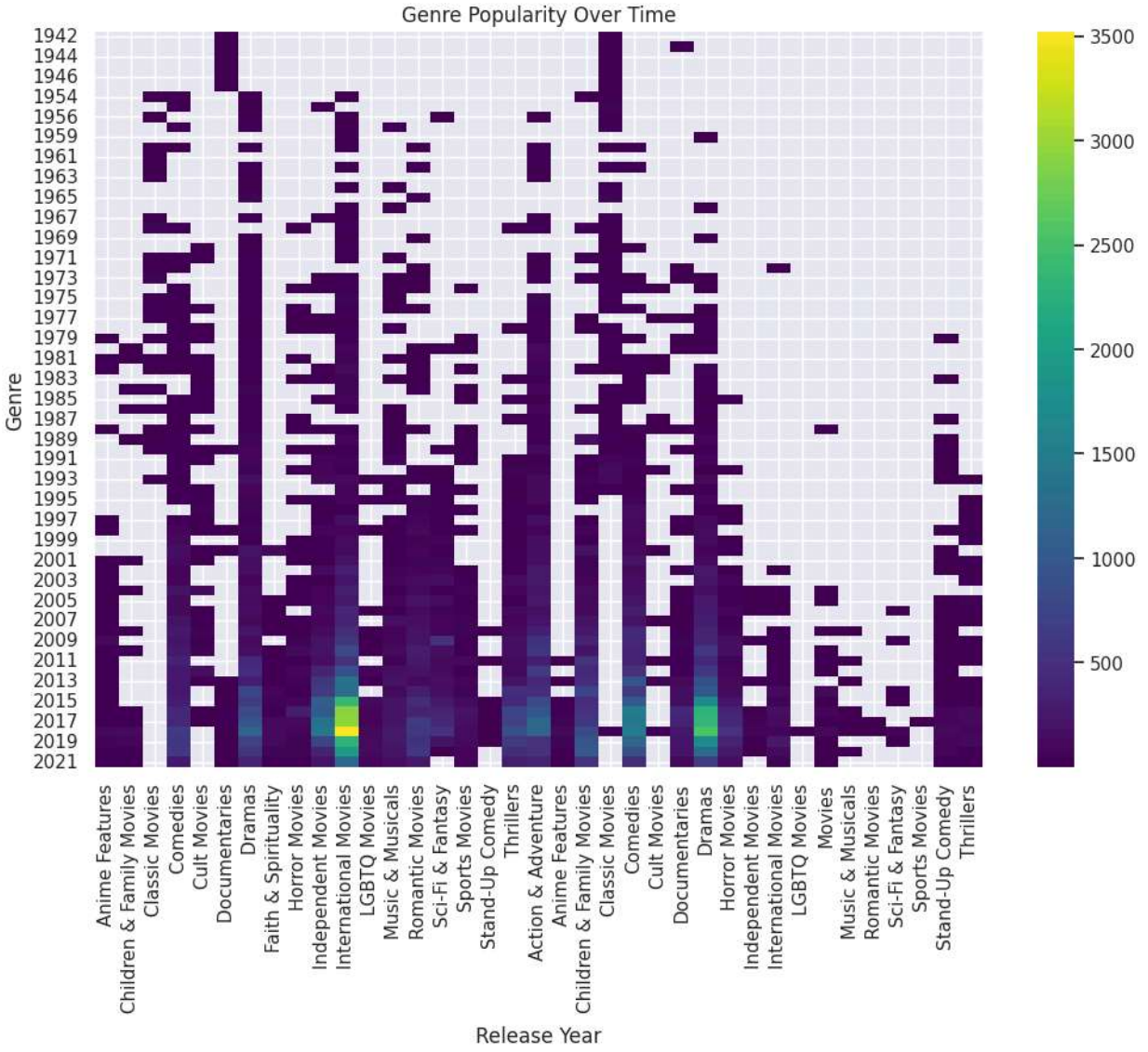
In [243…

```
# Q) What are the popular ratings for the movies released over the years
movie_data = exploded_data[exploded_data['type'] == 'Movie']
rating_counts = movie_data['rating'].value_counts()
plt.figure(figsize=(12, 6))

#plt.subplot(1, 2, 2)
sns.countplot(x='rating', data=movie_data)
plt.title('Rating Distribution')
plt.xticks(rotation=45)

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



In [245…

```
# Q) popularity of genres over time
movie_data = exploded_data[exploded_data['type'] == 'Movie']

grouped_data = movie_data.groupby(['listed_in', 'release_year']).size().reset_index(na

pivoted_data = grouped_data.pivot(index='release_year', columns='listed_in', values='C
sns.heatmap(pivoted_data, cmap='viridis', annot=False)
plt.title('Genre Popularity Over Time')
plt.xlabel('Release Year')
plt.ylabel('Genre')
plt.show()
```



In [ ]: