In [248…

```python
import numpy as np
import pandas as pd
```

In [250…

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

Defining Problem Statement and Analysing basic metrics:

In [253…

```python
df.head()
```

Out[253…

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

We can see there are some nested columns in our dataset: 'cast', 'director', 'listed_in', 'description', 'country'

In [255…

```
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
```

Our Data set have some nulls that needs handling: 'director', 'cast', 'country', 'date_added', 'rating', 'duration'

In [258…

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

Out[258…

| | show_id | type | title | director | cast | country | date_added | rating | du |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 8807 | 8807 | 8807 | 6173 | 7982 | 7976 | 8797 | 8803 | |
| **unique** | 8807 | 2 | 8807 | 4528 | 7692 | 748 | 1767 | 17 | |
| **top** | s1 | Movie | Dick Johnson Is Dead | Rajiv Chilaka | David Attenborough | United States | January 1, 2020 | TV-MA | 1 S |
| **freq** | 1 | 6131 | 1 | 19 | 19 | 2818 | 109 | 3207 | |

We can see the number of unique values in our data set, also for top and freq details we need filter and update our data set

In [261…

```
#As for now, 'description' is not required, we'll drop that column from our data se
df.drop('description', inplace = True, axis = 1)
```

In [263…

```
#Percentage of null values in df
round(df.isna().sum()/len(df) * 100,2)
```

```
Out[263…  show_id          0.00
          type             0.00
          title            0.00
          director        29.91
          cast             9.37
          country          9.44
          date_added       0.11
          release_year     0.00
          rating           0.05
          duration         0.03
          listed_in        0.00
          dtype: float64
```

```
In [265…  #Also we need to take care of the Dtype for 'date_added':
          df['date_added'] = pd.to_datetime(df['date_added'],exact = False)
```

Handling the Nested columns: 'director', 'cast', 'country', 'date_added', 'rating', 'duration'

```
In [268…  # Handling 'Cast' Nesting:
          def spit(x):
              return pd.Series(x.split(','))
          df_c = pd.DataFrame(df['cast'].dropna().apply(spit))

          #using Merge:
          #df_c['title'] = df['title']
          # df_c = df_c.melt(id_vars = 'title')
          # df_c.drop('variable', inplace = True, axis = 1)

          # Using Stack:
          df_c = df_c.stack().rename(index = df['title']).reset_index().drop('level_1', axis
          df_c.columns = ['title','cast']
          df_c['cast'] = df_c['cast'].apply(lambda x : x.strip())
          df_c.shape
```

```
Out[268…  (64126, 2)
```

```
In [269…  # Handling 'director' Nesting:
          df_d = pd.DataFrame(df['director'].dropna().apply(spit))
          df_d = df_d.stack().rename(index = df['title']).reset_index().drop('level_1', axis
          df_d.columns = ['title','director']
          df_d['director'] = df_d['director'].apply(lambda x : x.strip())
          df_d.shape
```

```
Out[269…  (6978, 2)
```

```
In [270…  # Handling 'listed_in' Nesting:
          df_l = pd.DataFrame(df['listed_in'].dropna().apply(spit))
          df_l = df_l.stack().rename(index = df['title']).reset_index().drop('level_1', axis
          df_l.columns = ['title','listed_in']
          df_l['listed_in'] = df_l['listed_in'].apply(lambda x : x.strip())
          df_l.shape
```

```
Out[270…  (19323, 2)
```

In [271…

```
# Handling 'country' Nesting:
df_co = pd.DataFrame(df['country'].dropna().apply(spit))
df_co = df_co.stack().rename(index = df['title']).reset_index().drop('level_1', axi
df_co.columns = ['title','country']
df_co['country'] = df_co['country'].apply(lambda x : x.strip())
df_co.shape
```

Out[271…

```
(10019, 2)
```

In [272…

```
# Merging All & Creating Final_df:
a = df_c.merge(df_d,how = 'outer', on = 'title')
a = a.merge(df_l, how = 'outer', on = 'title')
df_f = a.merge(df_co, how = 'outer', on = 'title')
df_f = df_f.merge(df[['show_id', 'type', 'title', 'date_added',
       'release_year', 'rating', 'duration']], how = 'left', on = 'title')
df_f.shape
```

Out[272…

```
(202065, 11)
```

In [275…

```
df_f.head()
```

Out[275…

| | title | cast | director | listed_in | country | show_id | type | date_added | release_yea |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Blood & Water | Ama Qamata | NaN | International TV Shows | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **1** | Blood & Water | Ama Qamata | NaN | TV Dramas | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **2** | Blood & Water | Ama Qamata | NaN | TV Mysteries | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **3** | Blood & Water | Khosi Ngema | NaN | International TV Shows | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **4** | Blood & Water | Khosi Ngema | NaN | TV Dramas | South Africa | s2 | TV Show | 2021-09-24 | 202 |

In [280…

```
df_f.isna().sum()
```

Out[280…

```
title               0
cast             2149
director        50643
listed_in           0
country         11897
show_id             0
type                0
date_added        158
release_year        0
rating             67
duration            3
dtype: int64
```

we will be required to handle all the nulls inside the data set

In [283…

```
df_f.describe(include = 'object')
```

Out[283…

| | title | cast | director | listed_in | country | show_id | type | rating | duration |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 202065 | 199916 | 151422 | 202065 | 190168 | 202065 | 202065 | 201998 | 202062 |
| **unique** | 8807 | 36439 | 4993 | 42 | 123 | 8807 | 2 | 17 | 220 |
| **top** | Kahlil Gibran's The Prophet | Liam Neeson | Martin Scorsese | Dramas | United States | s7165 | Movie | TV-MA | 1 Season |
| **freq** | 700 | 161 | 419 | 29806 | 59350 | 700 | 145917 | 73915 | 35035 |

In [285…

```
#there are 3 null rows in duration column, which we will be dropping, as it will no
df_f.dropna(subset = ['duration'], inplace = True)
```

In [287…

```
#handling the duration column:
# inside, for every tv show duration is in seasons, while for every Movie, duration
# hence we will be trimming out data from the values as follows:
#we will just be converting the text into int, for further processing

df_f['duration'] = df_f['duration'].apply(lambda x: x.split()[0]).astype('int')
```

In [289…

```
#Finding out the director favorite cast:
a = df_f.groupby(['director','cast'])['title'].nunique()
fill_c= a.groupby('director').idxmax().apply(lambda x: x[1]).rename('casts').reset_

#replacing the cast null with director favorite cast:
df_f = df_f.merge(fill_c, on = 'director', how = 'left')
df_f['cast'].fillna(df_f['casts'],inplace = True)
df_f.drop('casts', axis = 1, inplace = True)
```

In [291…

```
#Finding out director with whom Cast has worked the most:
a = df_f.groupby(['cast','director'])['title'].nunique()
fill_d = a.groupby('cast').idxmax().apply(lambda x: x[1]).rename('directors').reset

#replacing the director null values with cast favorite director:
```

```
df_f = df_f.merge(fill_d, on = 'cast', how = 'left')
df_f['director'].fillna(df_f['directors'],inplace = True)
df_f.drop('directors', axis = 1, inplace = True)
```

In [292…

```
#finding the country director worked in the most:
b = df_f.groupby(['country','director'])['title'].nunique()
fill_cc = b.groupby('director').idxmax().apply(lambda x: x[1]).rename('countries').

#replacing null countries with the director proximate countries
df_f = df_f.merge(fill_cc, on = 'director', how = 'left')
df_f['country'].fillna(df_f['countries'],inplace = True)
df_f.drop('countries', axis = 1, inplace = True)
```

In [293…

```
#finding the country cast worked in the most:
b = df_f.groupby(['country','cast'])['title'].nunique()
fill_cc = b.groupby('cast').idxmax().apply(lambda x: x[1]).rename('countries').rese

#replacing null countries with the cast proximate countries
df_f = df_f.merge(fill_cc, on = 'cast', how = 'left')
df_f['country'].fillna(df_f['countries'],inplace = True)
df_f.drop('countries', axis = 1, inplace = True)
```

In [294…

```
#handling date added null instances, replacing them with the mode of date added:
df_f['date_added'].fillna(df_f['date_added'].value_counts().index[0], inplace = Tru
```

In [297…

```
#finding the rating director had the most:
b = df_f.groupby(['rating','director'])['title'].nunique()
fill_cc = b.groupby('director').idxmax().apply(lambda x: x[1]).rename('ratings').re

#replacing null ratings with the director proximate ratings
df_f = df_f.merge(fill_cc, on = 'director', how = 'left')
df_f['rating'].fillna(df_f['ratings'],inplace = True)
df_f.drop('ratings', axis = 1, inplace = True)
```

In [298…

```
df_f.isna().sum()
```

Out[298…

```
title               0
cast             1959
director        32916
listed_in           0
country          4255
show_id             0
type                0
date_added          0
release_year        0
rating             23
duration            0
dtype: int64
```

In [303…

```
#Filling the rest of the data with the unknown strings and the ratings with the mos
df_f['cast'].fillna('Unknown Cast', inplace = True)
df_f['director'].fillna('Unknown Director', inplace = True)
df_f['country'].fillna('Unknown Country', inplace = True)
```

```
df_f['rating'].fillna(df_f['rating'].value_counts().index[0], inplace = True)
```

In [305…

```
df_f.isna().sum()
```

Out[305…

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

All the null values have been rectified, and the data set is read for the further exploration.

In [308…

```
#Non Graphical Exploration:
# exploration for different types of categorical columns:
print(df_f[df_f['type']== 'Movie'].groupby('country')['title'].nunique().rename('co
```

```
           country  counts
662  United States    2749
```

Most Movies were released in United States as 2749

In [311…

```
print(df_f[df_f['type']== 'TV Show'].groupby('country')['title'].nunique().rename('
```

```
           country  counts
681  United States     938
```

Most TV shows were released in United States as 938

In [314…

```
#Most popular director cast pair:
a = df_f.groupby(['director','cast'])['title'].nunique()
fill_c= a.groupby('director').idxmax().apply(lambda x: x[1]).rename('casts').reset_
fill_c.merge(a, on = 'director', how = 'left').sort_values(by = 'title', ascending
```

Out[314…

| | director | casts | title |
|---|---|---|---|
| **16357** | Hiroyuki Seshita | Takahiro Sakurai | 28 |
| **35685** | Rajiv Chilaka | Julie Tejwani | 27 |
| **35691** | Rajiv Chilaka | Julie Tejwani | 24 |
| **35692** | Rajiv Chilaka | Julie Tejwani | 22 |

These are the top 5 most famous director cast pair

In [317…

```
# Directors with most number of movies:
df_f.groupby(['director'])['title'].nunique().reset_index().sort_values(by = 'title
```

Out[317…

| | director | title |
|---|---|---|
| **1742** | Hiroyuki Seshita | 82 |
| **2958** | Masahiko Murata | 57 |
| **466** | Atsuko Ishizuka | 54 |
| **3390** | Noriyuki Abe | 48 |
| **123** | Akiva Schaffer | 48 |

Overall 'Hiroyuki Seshita' has done the most number of movies, i.e 82

In [320…

```
# Director who directed max number of movies over time:
a = df_f.groupby(['director','release_year'])['title'].nunique().reset_index().sort
a[a['director'] != 'Unknown Director'].sort_values(by = ['release_year','title'], a
```

Out[320…

| | director | release_year | title |
|---|---|---|---|
| **5758** | Kristian Mercado | 2021 | 10 |
| **2842** | Dennis Dugan | 2021 | 10 |
| **5767** | Krysia Plonka | 2021 | 10 |
| **3747** | Greg Rankin | 2021 | 8 |
| **311** | Akiva Schaffer | 2021 | 8 |
| **1163** | Atsuko Ishizuka | 2021 | 6 |
| **9670** | Tensai Okamura | 2021 | 6 |
| **4103** | Hiroyuki Seshita | 2021 | 6 |
| **2014** | Chiaki Kon | 2021 | 6 |
| **1773** | Byron Howard | 2021 | 6 |

'Kristian Mercado' | 'Dennis Dugan' | 'Krysia Plonka' are the most famous director in 2021, with total of 10 titles each, movies with these actors may seem profitable.

In [323…

```
# Actor who acted in most number of movies over time:
a = df_f.groupby(['cast','release_year'])['title'].nunique().reset_index().sort_val
#.reset_index().sort_values(by = 'title', ascending = False)[1:6]
a[a['cast'] != 'Unknown Cast'].head(10)
```

Out[323…

| | cast | release_year | title |
|---|---|---|---|
| **17070** | Fortune Feimster | 2021 | 11 |
| **32026** | London Hughes | 2021 | 10 |
| **12835** | David Spade | 2021 | 10 |
| **55483** | Vincent Tong | 2019 | 8 |
| **26693** | Julie Tejwani | 2018 | 8 |
| **3302** | Andrea Libman | 2018 | 8 |
| **53148** | Tiffany Haddish | 2019 | 7 |
| **27927** | Kathleen Barr | 2019 | 7 |
| **43263** | Radhika Apte | 2018 | 7 |
| **1740** | Alessandro Juliani | 2019 | 7 |

'Fortune Feimster' & 'London Hughes'&'David Spade' are the 3 actors with the most number of moves in 2021, that is 10, 8, 8 respectively. We may use movies by these actors to for see profits.

In [335…

```
# Genre distribution over Movies:
a = df_f.groupby(['listed_in', 'type'])['title'].nunique().reset_index().sort_value
a[a['type']== 'Movie'][:5]
```

Out[335…

| | listed_in | type | title |
|---|---|---|---|
| **16** | International Movies | Movie | 2752 |
| **12** | Dramas | Movie | 2427 |
| **7** | Comedies | Movie | 1674 |
| **10** | Documentaries | Movie | 869 |
| **0** | Action & Adventure | Movie | 859 |

'International Movies' followed by 'Dramas' hold the max number of title across all movies.

In [337…

```
# Genre distribution over TV shows:
a = df_f.groupby(['listed_in', 'type'])['title'].nunique().reset_index().sort_value
a[a['type']== 'TV Show'][:5]
```

```
Out[337…
```

| | listed_in | type | title |
|---|---|---|---|
| **17** | International TV Shows | TV Show | 1351 |
| **34** | TV Dramas | TV Show | 763 |
| **33** | TV Comedies | TV Show | 581 |
| **8** | Crime TV Shows | TV Show | 470 |
| **18** | Kids' TV | TV Show | 451 |

'International TV Shows' & 'TV Dramas' holds the corresponding max number of title records.

```
In [340…
```

```python
# various genre of data, sorted according to date added on netflix:
x = df_f.copy()
x['year'] = df_f['date_added'].dt.year
x.groupby(['listed_in', 'year','type'])['title'].nunique().reset_index().sort_value
```

```
Out[340…
```

| | listed_in | year | type | title |
|---|---|---|---|---|
| **105** | Dramas | 2021 | Movie | 412 |
| **139** | International Movies | 2021 | Movie | 408 |
| **61** | Comedies | 2021 | Movie | 299 |
| **147** | International TV Shows | 2021 | TV Show | 229 |
| **7** | Action & Adventure | 2021 | Movie | 196 |
| **276** | TV Dramas | 2021 | TV Show | 137 |
| **37** | Children & Family Movies | 2021 | Movie | 122 |
| **267** | TV Comedies | 2021 | TV Show | 118 |
| **197** | Romantic Movies | 2021 | Movie | 114 |
| **330** | Thrillers | 2021 | Movie | 112 |

In year 2021 Dramas and International movies topped the table.

```
In [343…
```

```python
# Rating corresponding to movies and TV shows:
x.groupby(['rating','year','type'])['title'].nunique().reset_index().sort_values(by
```

Out[343…

| | rating | year | type | title |
|---|---|---|---|---|
| **112** | TV-MA | 2021 | Movie | 256 |
| **113** | TV-MA | 2021 | TV Show | 233 |
| **74** | TV-14 | 2021 | Movie | 200 |
| **56** | R | 2021 | Movie | 190 |
| **46** | PG-13 | 2021 | Movie | 146 |
| **75** | TV-14 | 2021 | TV Show | 126 |
| **39** | PG | 2021 | Movie | 58 |
| **130** | TV-PG | 2021 | Movie | 58 |
| **159** | TV-Y7 | 2021 | Movie | 45 |
| **160** | TV-Y7 | 2021 | TV Show | 42 |

In the year 2021, 256 movies and 233 TV Shows were rated 'TV-MA'.

In [346…

```
# Average Duration of Movies in top 3 countries, over years:
y = x.copy()
z = y['country'].value_counts().index[:3]
y[(y['type'] == 'Movie') & (y['country'].isin(z))].groupby(['year','country'])['dur
```

Out[346…

| | year | country | duration |
|---|---|---|---|
| **26** | 2021 | India | 125.093826 |
| **27** | 2021 | United Kingdom | 114.791178 |
| **28** | 2021 | United States | 106.445798 |
| **23** | 2020 | India | 132.419621 |
| **24** | 2020 | United Kingdom | 108.371285 |
| **25** | 2020 | United States | 102.048157 |
| **20** | 2019 | India | 128.230139 |
| **21** | 2019 | United Kingdom | 107.489760 |
| **22** | 2019 | United States | 101.756165 |
| **17** | 2018 | India | 130.357647 |

In year 2021, India top the charts for the average of movies duration time.

In [349…

```
# Average no of series in top 3 countries, over years:
y = x.copy()
z = y['country'].value_counts().index[:3]
y[(y['type'] == 'TV Show') & (y['country'].isin(z))].groupby(['year','country'])['d
```

Out[349…

| | year | country | duration |
|---|---|---|---|
| **22** | 2021 | United States | 3.156274 |
| **21** | 2021 | United Kingdom | 2.493274 |
| **20** | 2021 | India | 1.096413 |
| **19** | 2020 | United States | 2.998622 |
| **18** | 2020 | United Kingdom | 2.408419 |
| **17** | 2020 | India | 1.189702 |
| **16** | 2019 | United States | 2.550303 |
| **15** | 2019 | United Kingdom | 2.342289 |
| **14** | 2019 | India | 1.173554 |
| **13** | 2018 | United States | 2.552072 |

In year 2021, United States top the charts for the average of number of seasons in series.

In [352…

```
# Importing visualization libraries:
import seaborn as sns
import matplotlib.pyplot as plt
```

In [414…

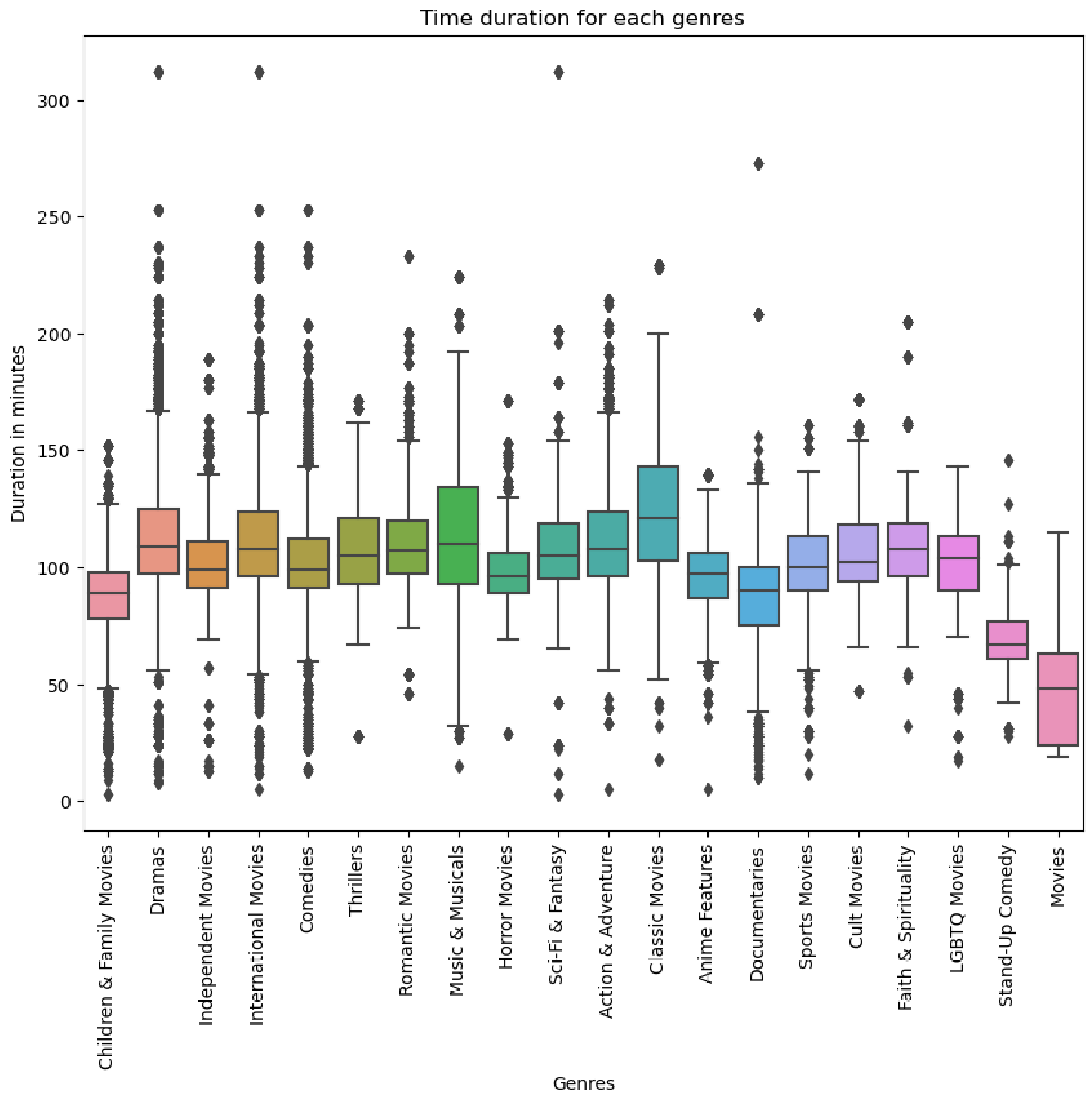
```
df_f.head()
```

Out[414…

| | title | cast | director | listed_in | country | show_id | type | date_added | release_ye |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Blood & Water | Ama Qamata | Unknown Director | International TV Shows | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **1** | Blood & Water | Ama Qamata | Unknown Director | TV Dramas | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **2** | Blood & Water | Ama Qamata | Unknown Director | TV Mysteries | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **3** | Blood & Water | Khosi Ngema | Unknown Director | International TV Shows | South Africa | s2 | TV Show | 2021-09-24 | 202 |
| **4** | Blood & Water | Khosi Ngema | Unknown Director | TV Dramas | South Africa | s2 | TV Show | 2021-09-24 | 202 |

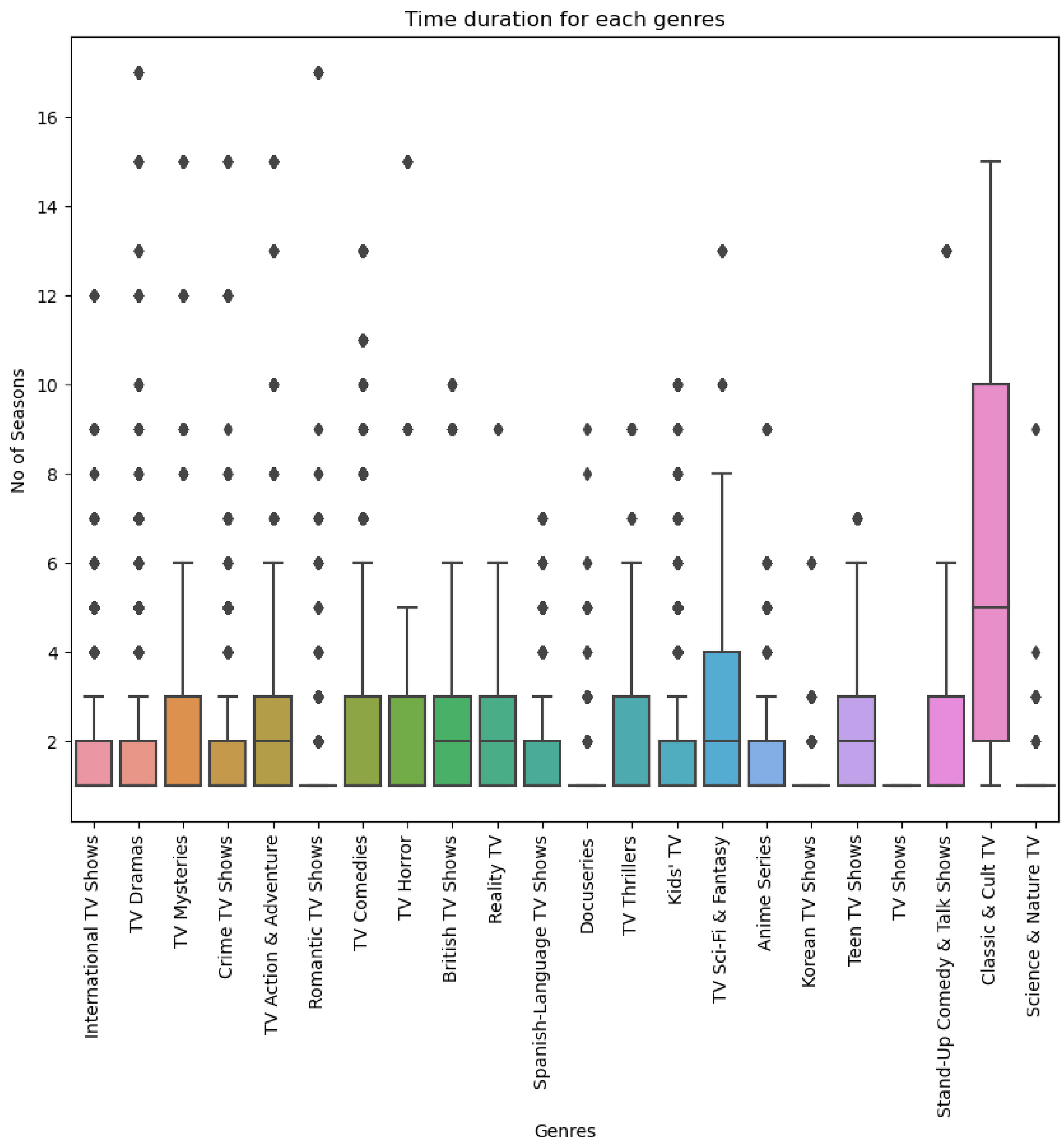
In [427…

```
plt.figure(figsize = (10,8))
sns.boxplot(data = df_f[df_f['type']== 'Movie'],x = 'listed_in', y = 'duration')
plt.xticks(rotation = 90)
plt.xlabel('Genres')
plt.ylabel('Duration in minutes')
plt.title('Time duration for each genres')
plt.show()
```



it can be seen that classic movies has the upper hand in duration in movies

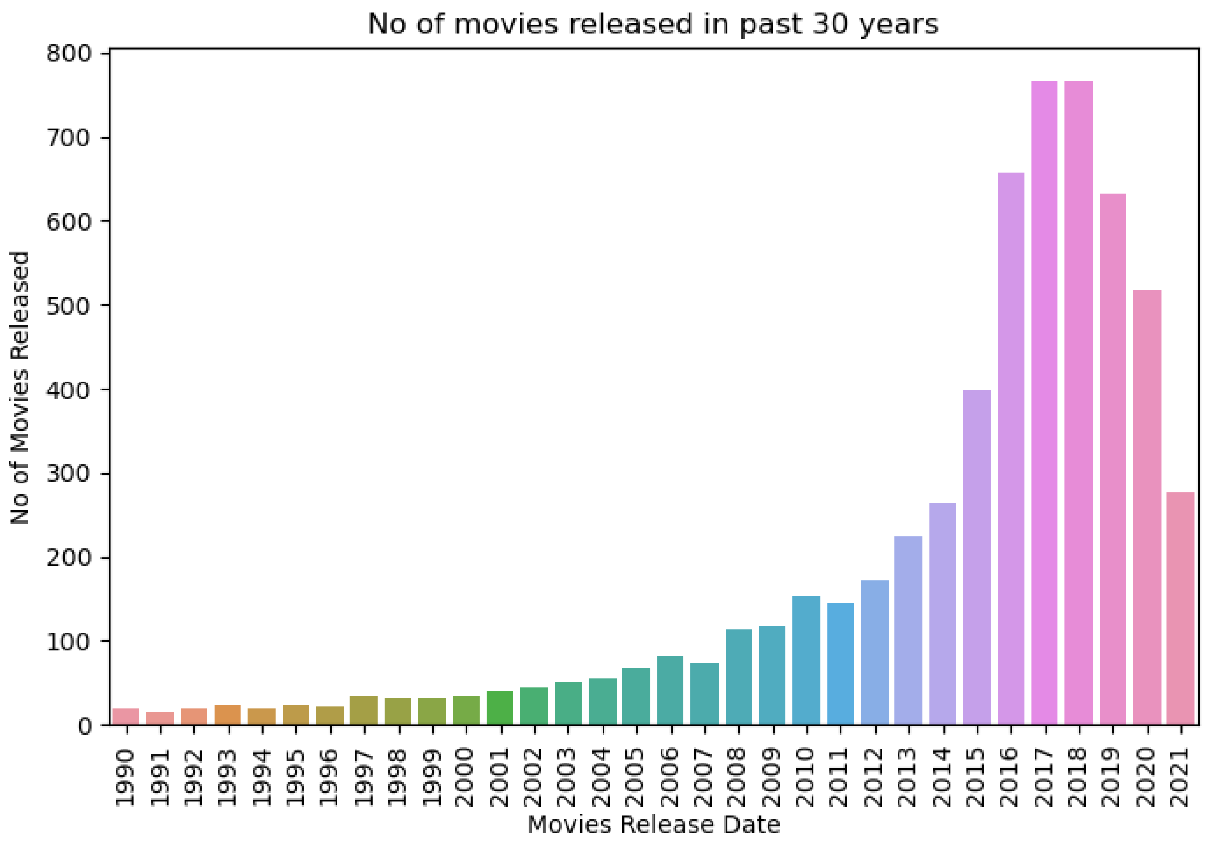
In [429…
```
plt.figure(figsize = (10,8))
sns.boxplot(data = df_f[df_f['type']== 'TV Show'],x = 'listed_in', y = 'duration')
plt.xticks(rotation = 90)
plt.xlabel('Genres')
plt.ylabel('No of Seasons')
plt.title('Time duration for each genres')
plt.show()
```



We can see a lot of out liners in our data corresponding to the International TV shows and TV dramas, while classic & cult tv owns the market

In [354…

```
# How has the number of movies released per year changed over the last 20-30 years?
plot = df_f[(df_f['type'] == 'Movie') & (df_f['release_year'] >= 1990)]
a = plot.groupby('release_year')['title'].nunique().rename('counts').reset_index()
plt.figure(figsize = (8,5))
sns.barplot(data = a, x = 'release_year', y = 'counts')
plt.xticks(rotation = 90)
plt.xlabel('Movies Release Date')
plt.ylabel('No of Movies Released')
plt.title('No of movies released in past 30 years')
plt.show()
```

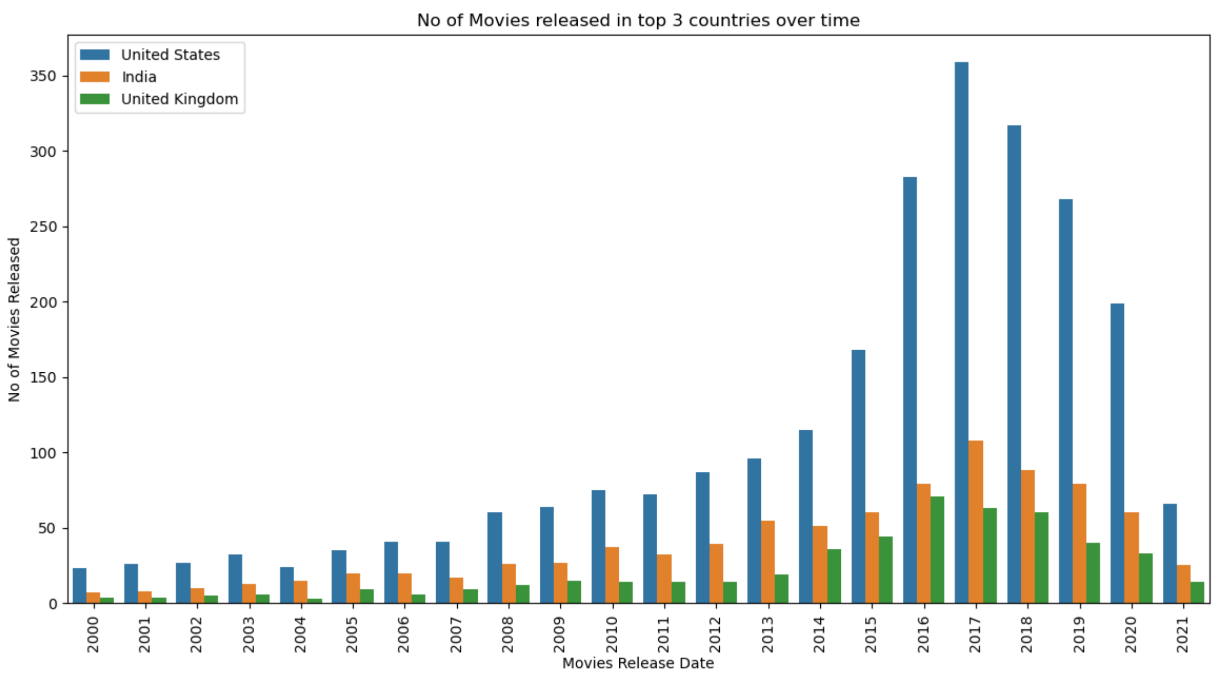


It can be seen that the plot is rightly skewed or positive skewed, also with the time, no of movies released increased, and followed with major fall in years after 2018, that might be seen during the covid pandemic years.

In [357…

```
# No of movies released in top 3 different countries over time period:
top_3_countries = df_f['country'].value_counts()[:3].index
plot = df_f[(df_f['type'] == 'Movie') & (df_f['release_year'] >= 2000) & (df_f['cou
plot = plot.groupby(['release_year','country'])['title'].nunique().rename('density'

plt.figure(figsize = (14,7))
sns.barplot(data = plot, x = 'release_year',y = 'density', hue = 'country',hue_orde
plt.xticks(rotation = 90)
plt.xlabel('Movies Release Date')
plt.ylabel('No of Movies Released')
plt.title('No of Movies released in top 3 countries over time')
plt.legend(loc = 'upper left')
plt.show()
```

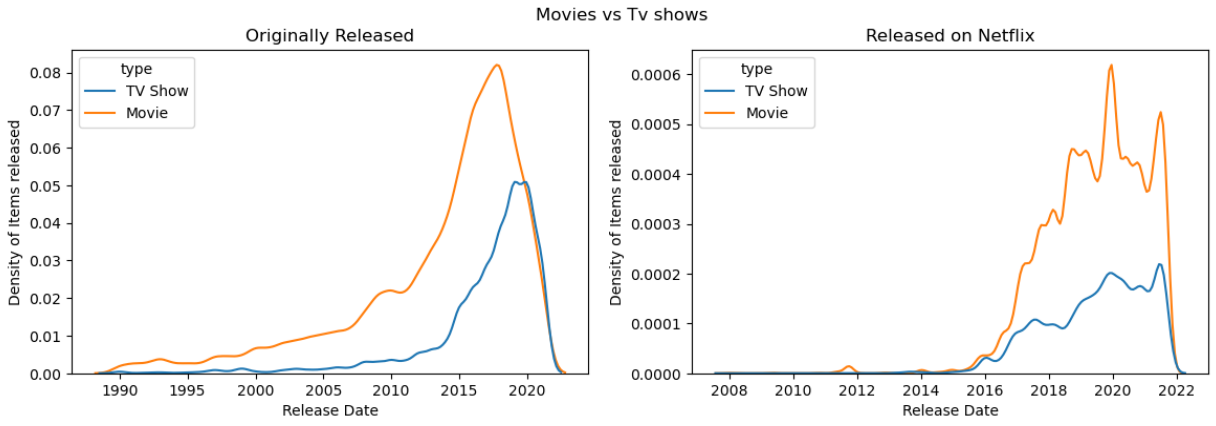


It shows the distribution of number of movies from the top 3 countries, and overall we can see that always US has been dominating the market.

In [360…

```
# Comparison of tv shows vs. movies.
# Movies vs Tv shows when they were released:
fig, axs = plt.subplots(1, 2, figsize=(14, 4))
plt.suptitle('Movies vs Tv shows')
plt.subplot(1,2,1)
sns.kdeplot(data = df_f[df_f['release_year'] >= 1990], x = 'release_year', hue = 't
plt.xlabel('Release Date')
plt.ylabel('Density of Items released')
plt.title('Originally Released')

plt.subplot(1,2,2)
sns.kdeplot(data = df_f, x = 'date_added', hue = 'type', )
plt.xlabel('Release Date')
plt.ylabel('Density of Items released')
plt.title('Released on Netflix')
plt.show()
```



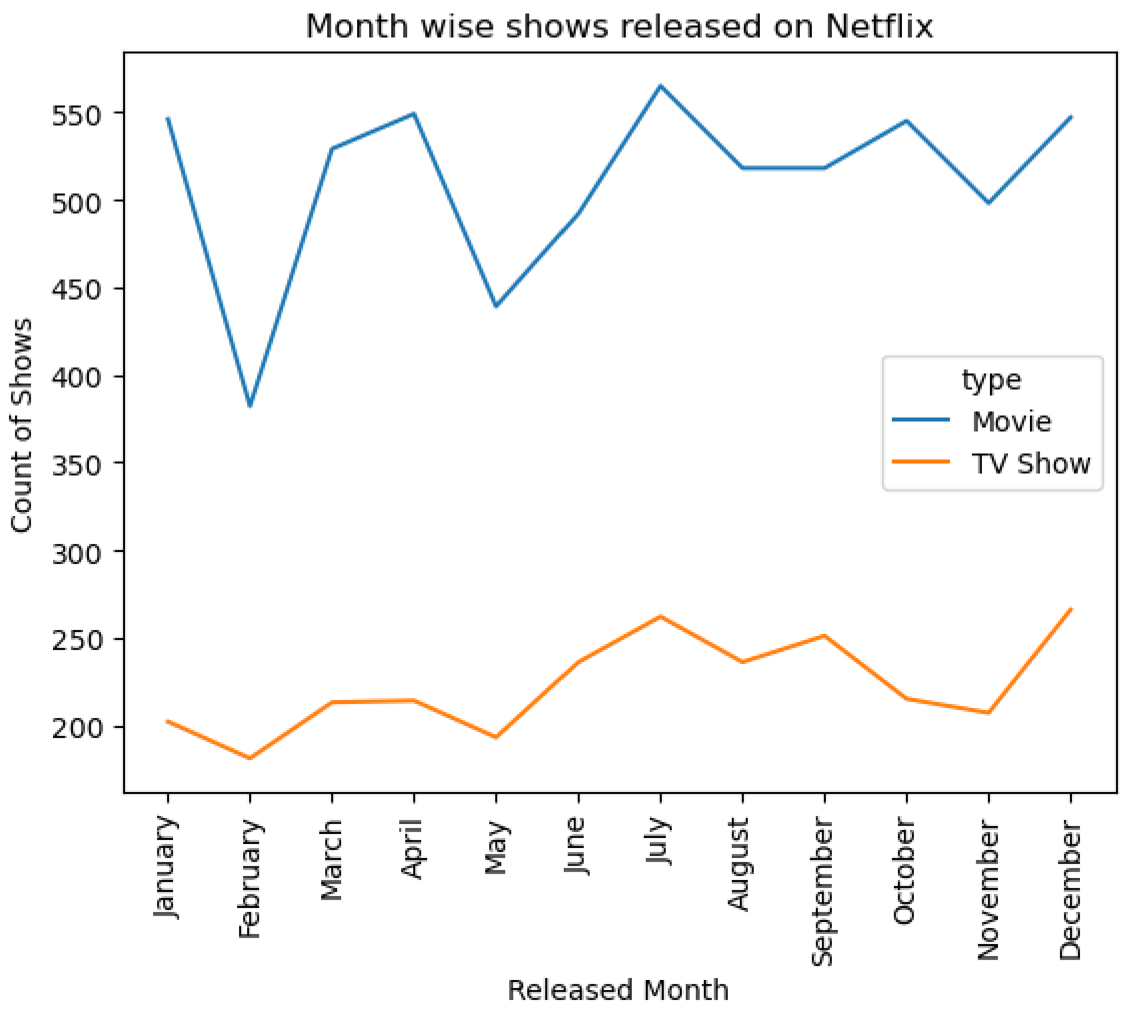
It can be seen that following the year 1990, the movies are tend to have an positive trend, while Tv shows started being popular after 1995, both reaching there saturation in year 2018 and 2020 correspondingly, That again due to pandemic situations.

It is seen that Movies / TV shows are added more and more on Netflix since 2016, since people started watching on online platforms, and had a considerable drop near 2021 end, as during pandemic production of the shows halted.

In [364…

```
# What is the best time to launch a TV show?
import calendar
x = df_f.copy()
x['month_released_netflix'] = df_f['date_added'].dt.month.apply(lambda x: calendar.
x['month_num'] = df_f['date_added'].dt.month
plot = x.groupby(['month_released_netflix','type','month_num'])['title'].nunique().
plot.sort_values(by = 'month_num', inplace = True)

sns.lineplot(data = plot, x = 'month_released_netflix', y = 'count',hue = 'type')
plt.xticks(rotation = 90)
plt.xlabel('Released Month')
plt.ylabel('Count of Shows')
plt.title('Month wise shows released on Netflix')
plt.show()
```



It can be seen from the historical data that, the best time to release a TV show is between June to October including December, Also it can be seen that there is a major drop in movie released in Feb and May.

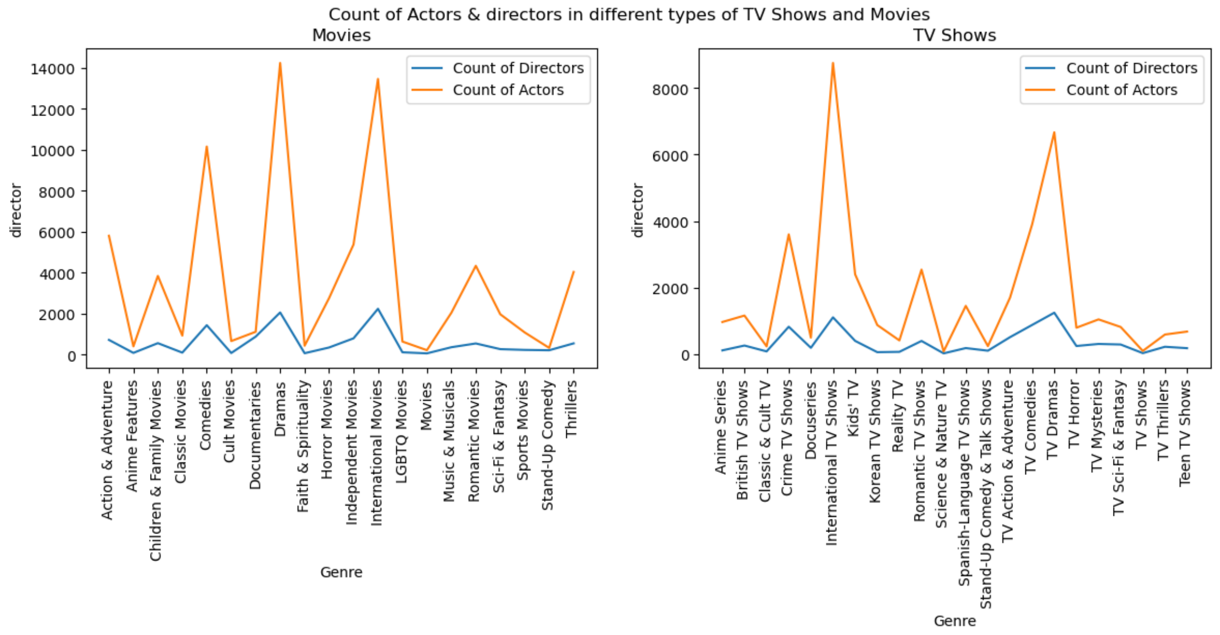
In [367…

```
# Analysis of actors/directors of different types of shows/movies.
plot_data = df_f.groupby(['listed_in','type']).agg({'director':'nunique','cast':'nu

fig, axs = plt.subplots(1, 2, figsize=(14, 4))
plt.suptitle('Count of Actors & directors in different types of TV Shows and Movies

plt.subplot(1,2,1)
plt.title('Movies')
sns.lineplot(data = plot_data[plot_data['type'] == 'Movie'], x = 'listed_in', y = '
sns.lineplot(data = plot_data[plot_data['type'] == 'Movie'], x = 'listed_in', y = '
plt.xticks(rotation = 90)
plt.xlabel('Genre')
plt.legend()

plt.subplot(1,2,2)
plt.title('TV Shows')
sns.lineplot(data = plot_data[plot_data['type'] == 'TV Show'], x = 'listed_in', y =
sns.lineplot(data = plot_data[plot_data['type'] == 'TV Show'], x = 'listed_in', y =
plt.xticks(rotation = 90)
plt.legend()
plt.xlabel('Genre')
plt.show()
```

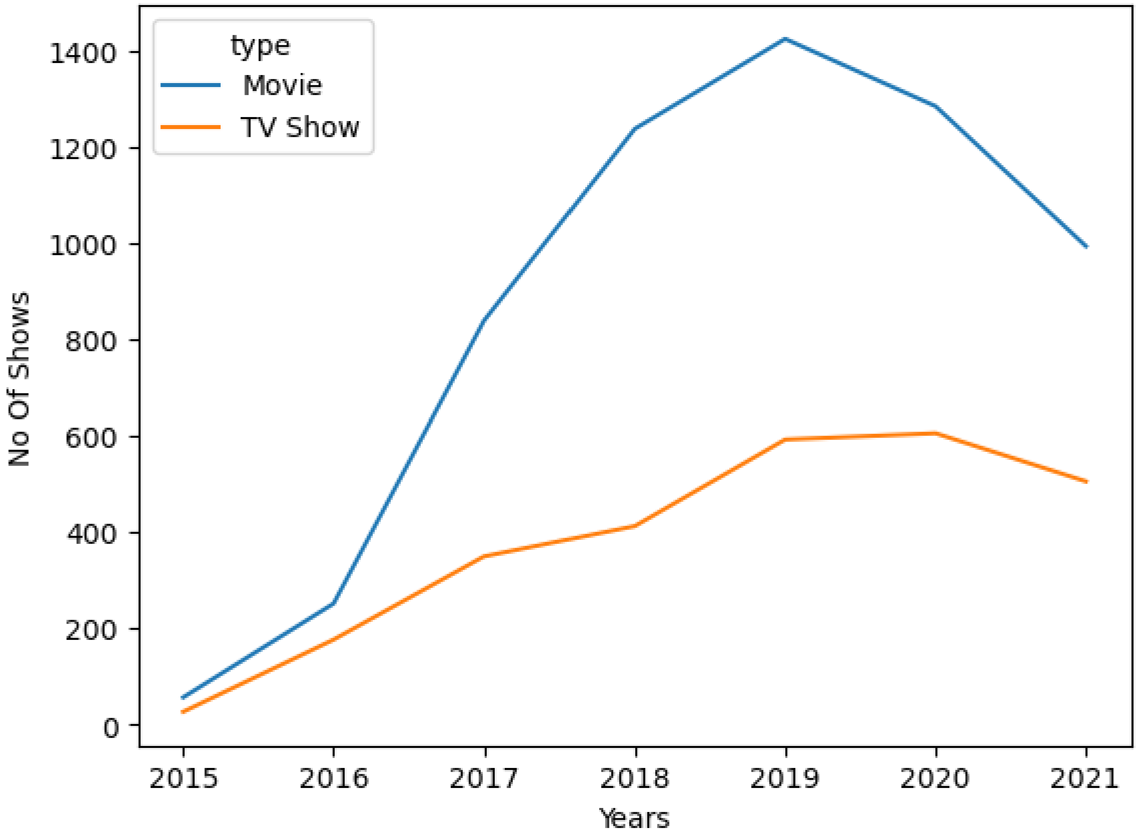


It is seen that in Movies, Most of the Actors and directors are working in following genre's : 'Comedies' / 'Dramas' / 'International Movies'

It is seen that in TV Shows, Most of the Actors and directors are working in following genre's : 'Crime TV Shows' / 'TV Dramas' / 'International TV Shows' / 'TV Comedies'

In [371…

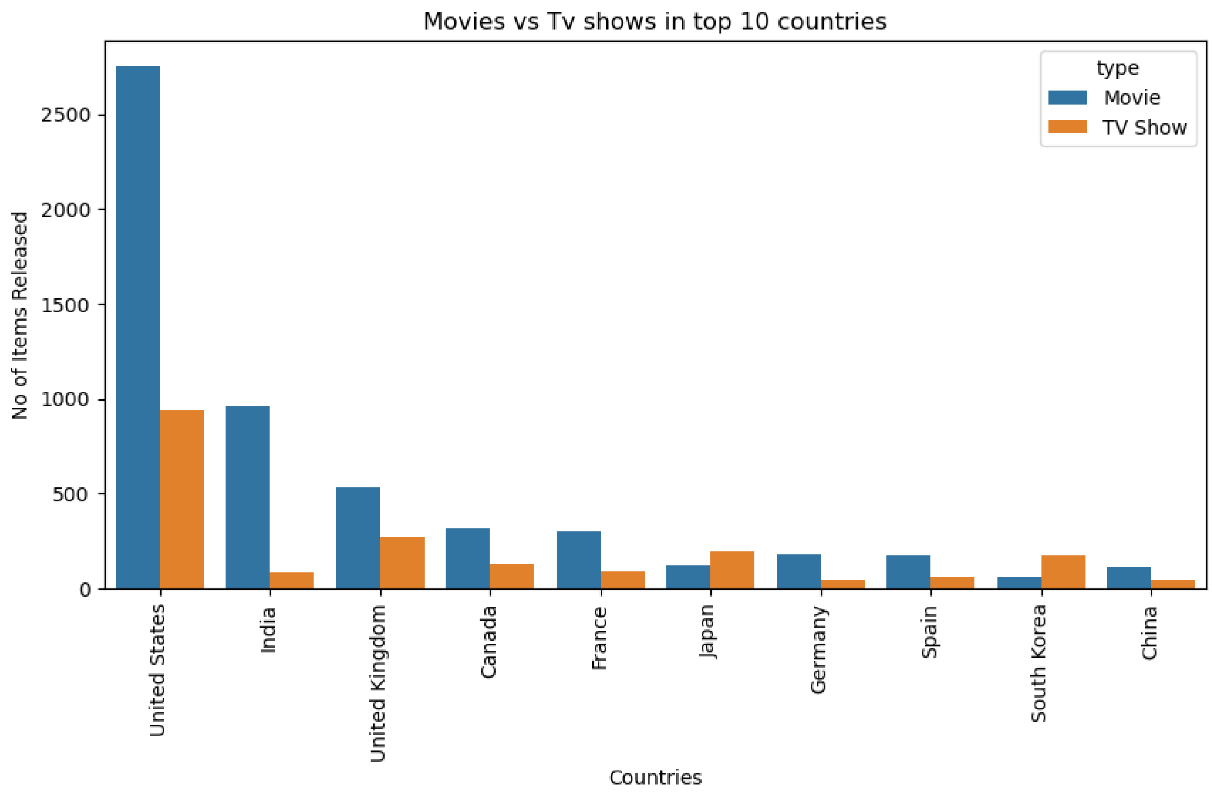
```
# Does Netflix has more focus on TV Shows than movies in recent years
x = df_f[df_f['date_added'].dt.year >= 2015].copy()
x['year_released_netflix'] = df_f['date_added'].dt.year
a = x.groupby(['year_released_netflix','type'])['title'].nunique().rename('counts')
sns.lineplot(data = a, x = 'year_released_netflix', y = 'counts',hue = 'type')
plt.xlabel('Years')
plt.ylabel('No Of Shows')
plt.show()
```



It can be seen that Netflix is focusing on the movies than TV shows, in recent years.

In [374…

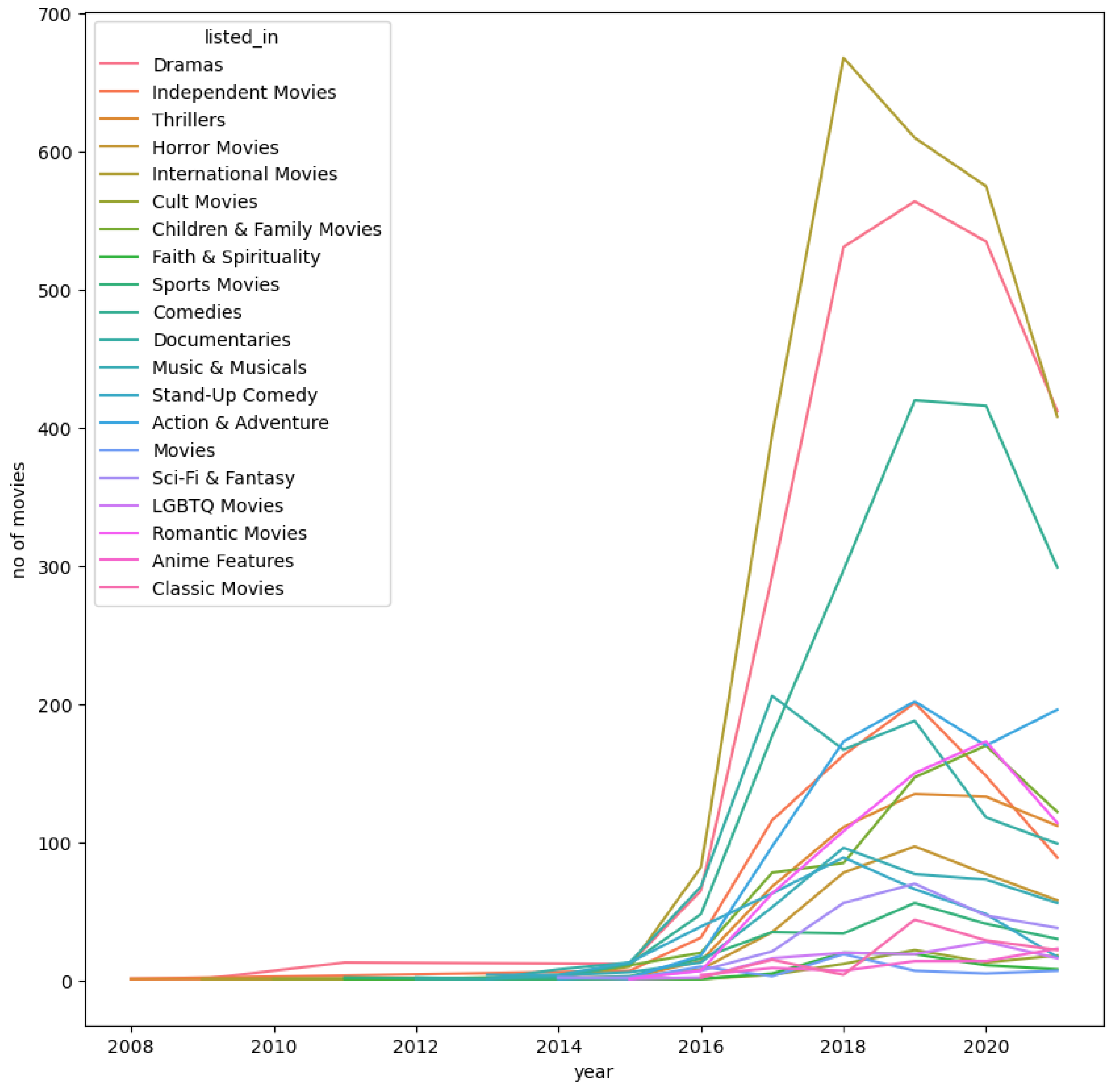
```
# Understanding what content is available in top 10 different countries:
# Movies vs Tv shows in top 10 countries:
a = df_f.groupby(['country','type'])['title'].nunique().rename('counts').reset_inde
plot = a[ (a['country'].isin(a['country'].unique()[:11])) & (a['country'] != 'Unkno
plt.figure(figsize = (10,5))
sns.barplot(data =  plot, x = 'country',y = 'counts', hue = 'type')
plt.xticks(rotation = 90)
plt.xlabel('Countries')
plt.ylabel('No of Items Released')
plt.title('Movies vs Tv shows in top 10 countries')
plt.show()
```



It is seen that the data set contains highest amount of Tv shows and movies from United States followed by India, Also in most of the countries people are more tend to watch a Movie, as compare to a Tv show, except for 'Japan' & 'South Korea'.

In [377…

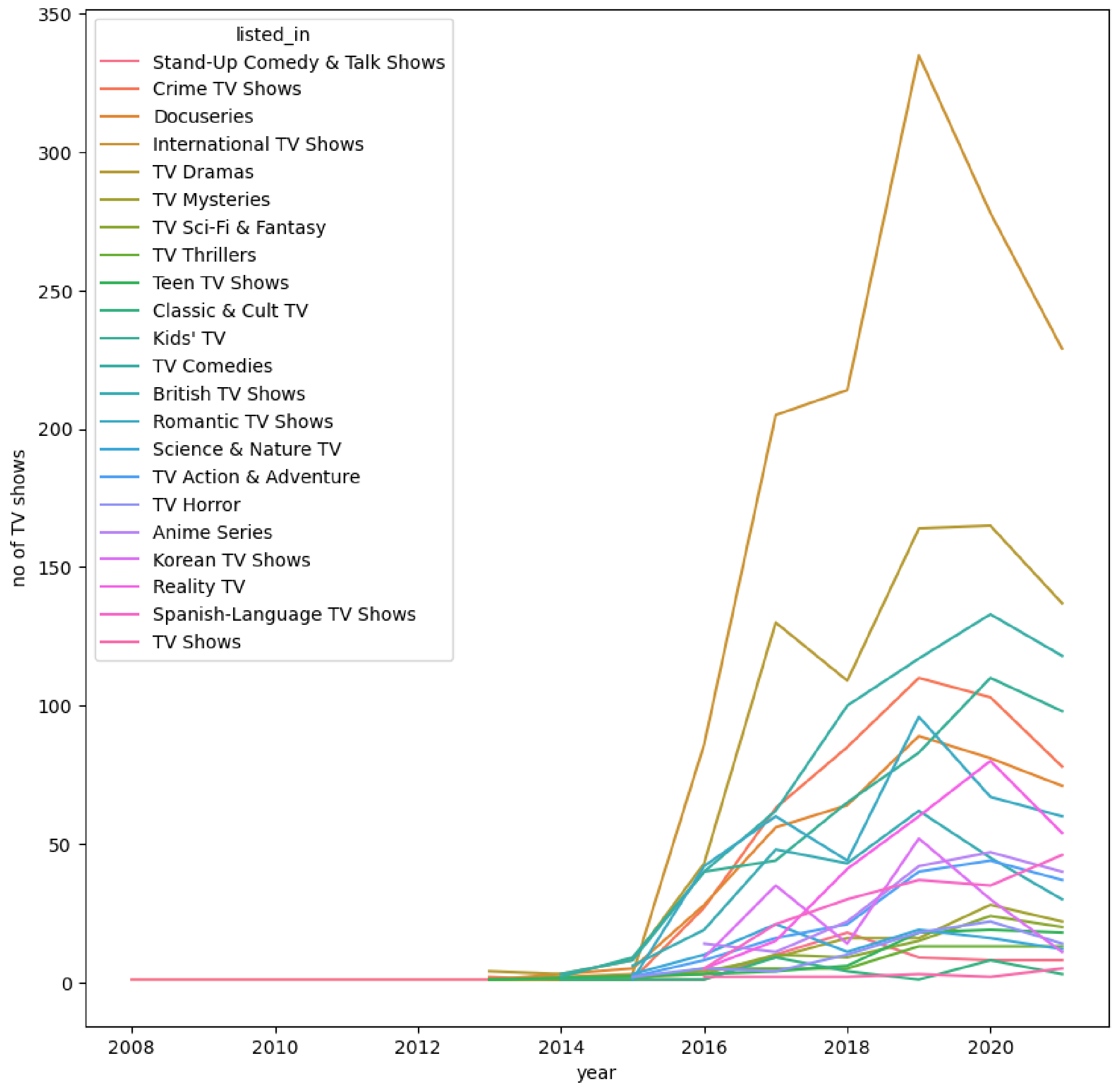
```
# Movie Genres throughout the year:
x = df_f.copy()
x['year'] = x['date_added'].dt.year
x = x[x['type'] == 'Movie'].groupby(['year','listed_in'])['title'].nunique().reset_
plt.figure(figsize = (10,10))
sns.lineplot(data = x, x = 'year', y = 'title',hue = 'listed_in',markers = True)
plt.ylabel('no of movies')
plt.show()
```



International Movies top the charts in the Genres past 2016, before that classic movies were the top

In [379…

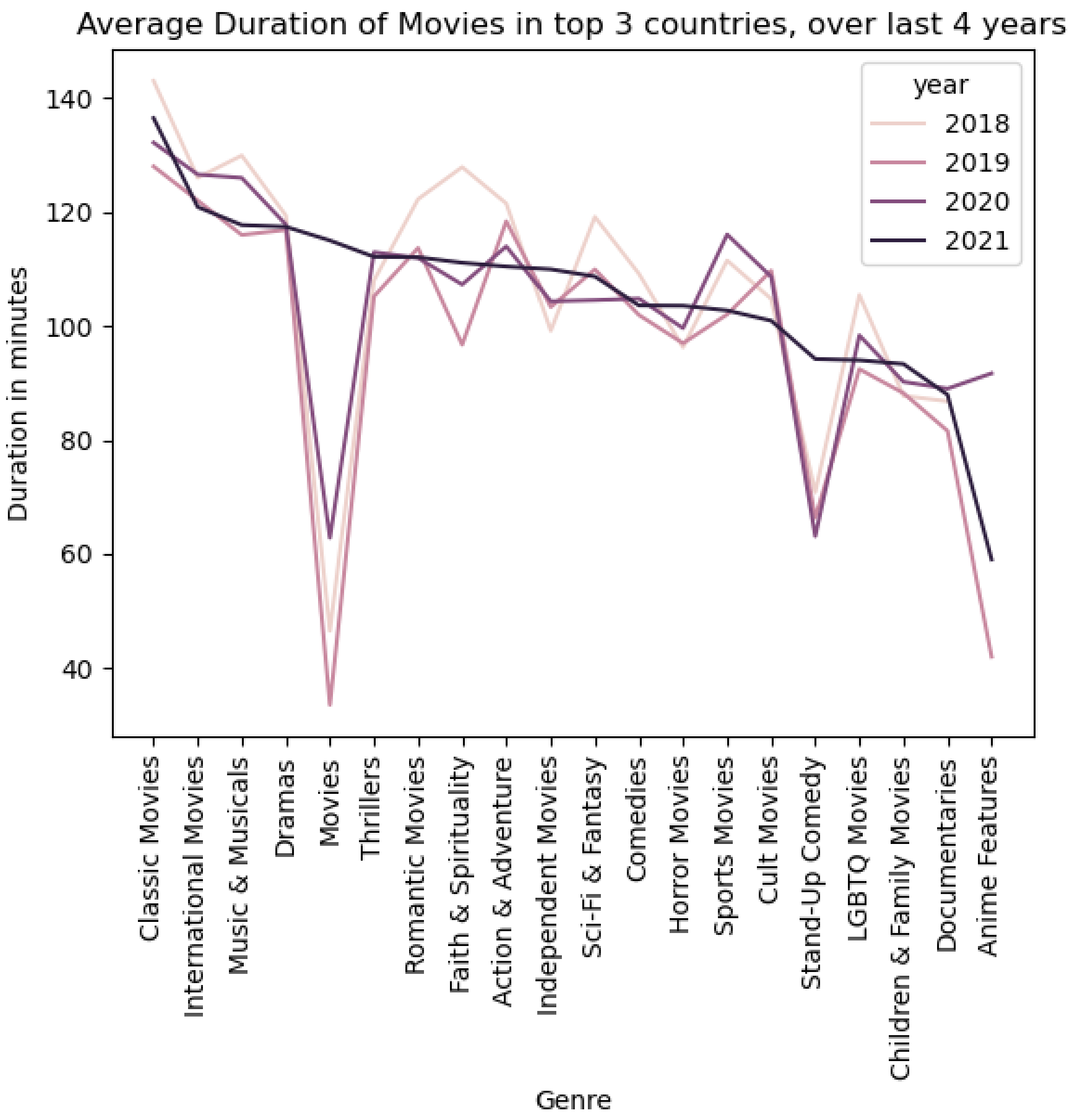
```
# TV Shows Genres throughout the year:
x = df_f.copy()
x['year'] = x['date_added'].dt.year
x = x[x['type'] == 'TV Show'].groupby(['year','listed_in'])['title'].nunique().rese
plt.figure(figsize = (10,10))
sns.lineplot(data = x, x = 'year', y = 'title',hue = 'listed_in',markers = True)
plt.ylabel('no of TV shows')
plt.show()
```



International TV shows top the charts, with considerable amount.

In [383…

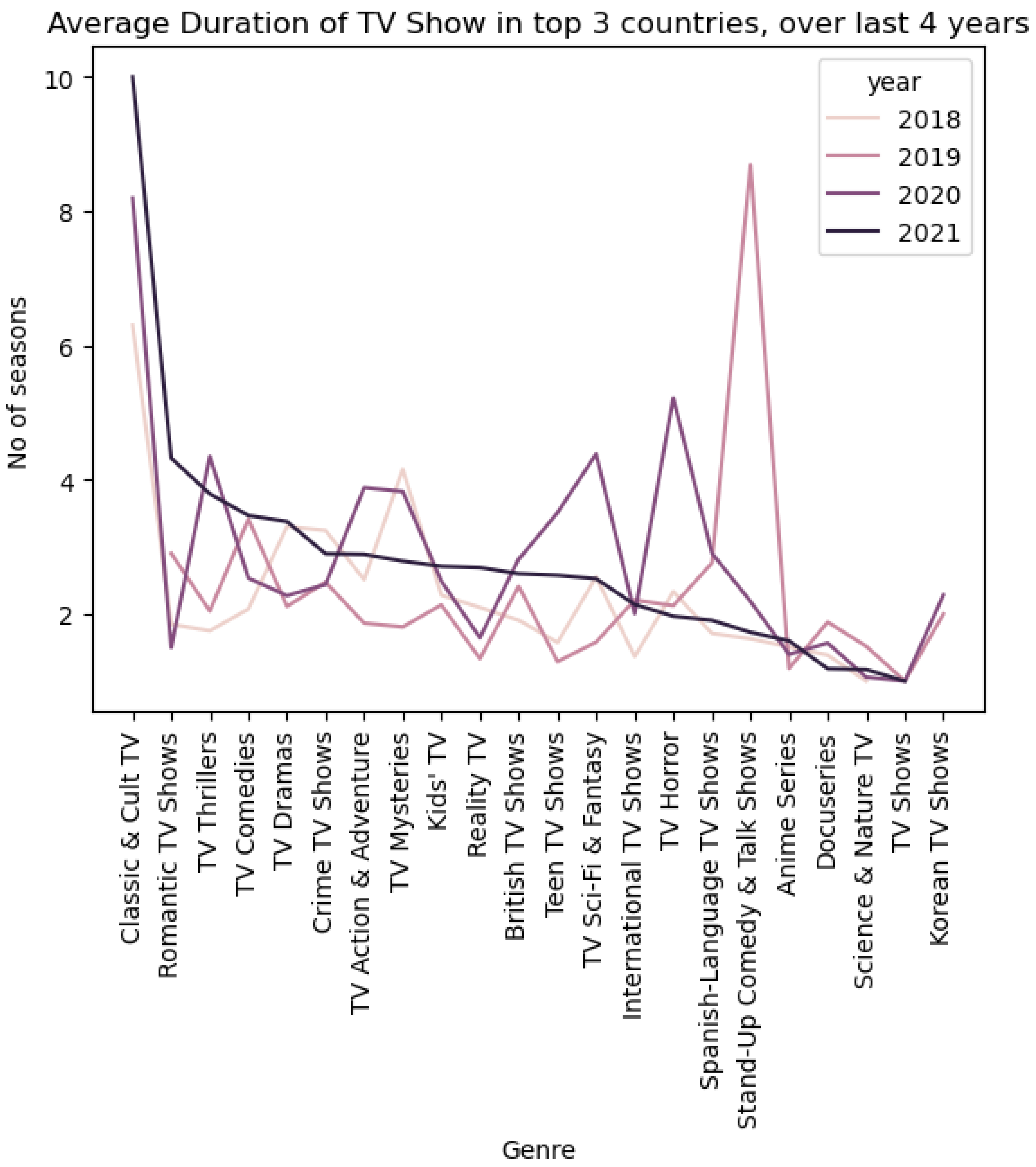
```
# Average Duration of Movies in top 3 countries, over last 4 years:
x = df_f.copy()
x['year'] = df_f['date_added'].dt.year
z = x['country'].value_counts().index[:3]
plot = x[(x['type'] == 'Movie') & (x['country'].isin(z)) & (x['year']>2017) ].group
sns.lineplot(data = plot, x = 'listed_in', y = 'duration', hue = 'year')
plt.xlabel('Genre')
plt.ylabel('Duration in minutes')
plt.xticks(rotation = 90)
plt.title('Average Duration of Movies in top 3 countries, over last 4 years')
plt.show()
```



It can be seen that in 2018, most of the genre of movies had a comparatively higher duration, Also the min duration goes to genre 'Movies' & 'Stand-up- comedy' & highest to the classic movies

In [386…

```
# Average Duration of TV Shows in top 3 countries, over last 4 years:
x = df_f.copy()
x['year'] = df_f['date_added'].dt.year
z = x['country'].value_counts().index[:3]
plot = x[(x['type'] == 'TV Show') & (x['country'].isin(z)) & (x['year']>2017) ].gro
sns.lineplot(data = plot, x = 'listed_in', y = 'duration', hue = 'year')
plt.xlabel('Genre')
plt.ylabel('No of seasons')
plt.xticks(rotation = 90)
plt.title('Average Duration of TV Show in top 3 countries, over last 4 years')
plt.show()
```



It is seen that in Classic & Cult TV has comparatively higher seasons in recent years, while in 2019, big hike was seen in the 'Stand-up comedy and Talk shows'.