1. Let's First install the required libraries for EDA.

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

2. Downloading the given dataset using gdown command

```
!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, 24.0MB/s]
```

3. Downloaded file is ready to read. Lets read it using the pandas

```
data = pd.read_csv('/content/netflix.csv')
data.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rat
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	TV-
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel	NaN	September 24, 2021	2021	TV-

Next steps: Generate code with data | View recommended plots

4. following command will help us to know the shape of the dataset. The Given Dataset has 8807 Records/Rows and 12 columns

```
data.shape
```

```
(8807, 12)
```

5. Lets check the name of the columns

```
data.columns
```

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

```
# Lets randomly try to explore the dataset

data.sample(10)
```

	show_id	type	title	director	cast	country	date_added	release_y
7423	s7424	Movie	Max Rose	Daniel Noah	Jerry Lewis, Kerry Bishé, Illeana Douglas, Ran...	United States	January 15, 2017	2
2559	s2560	Movie	Becoming	Nadia Hallgren	Michelle Obama	United States	May 6, 2020	2
1611	s1612	TV Show	Are You The One	NaN	Ryan Devlin	United States	December 1, 2020	2
2458	s2459	Movie	Kenny Sebastian: The Most Interesting Person i...	Angshuman Ghosh	Kenny Sebastian	India	May 29, 2020	2
8779	s8780	Movie	Yes or No 2.5	Kirati Nakintanon	Supanart Jittaleela, Pimpakan Bangchawong, Cha...	Thailand	November 8, 2018	2
7716	s7717	Movie	Paul Blart: Mall Cop	Steve Carr	Kevin James, Keir O'Donnell, Jayma Mays, Raini...	United States	November 1, 2020	2
5858	s5859	Movie	He Never Died	Jason Krawczyk	Henry Rollins, Booboo Stewart, Kate Greenhouse...	Canada	March 18, 2016	2

Zhu

6. As we can see above columns (directors, cast, country, and listed_in) has comma (',') separated values. We first need to arrange the data in this coulmns properly.

```
# Spliting the directors column

df_director_r = pd.DataFrame(data['director'].apply(lambda x: str(x).split(',')).to_list(), index = data['title'])

# To transpose from rows to columns we have used '.stack()'
df_director_r = df_director_r.stack().reset_index()

df_director_r.drop('level_1', axis = 1, inplace = True)

df_director_r.rename(columns = {0:'director'}, inplace = True)

df_director_r.sample(10)
```

	title	director
9073	The Gospel of Mark	David Batty
166	Kid-E-Cats	nan
6189	A Mighty Team	Thomas Sorriaux
5378	Bullet Head	Paul Solet
9067	The Golem	Yoav Paz
973	Mine	nan
3205	Night on Earth: Shot in the Dark	nan
4510	Memory Love	nan
8246	Night Moves	Kelly Reichardt
1168	My Love: Six Stories of True Love	Chico Pereira

```
# Spliting the cast column

df_cast_r = pd.DataFrame(data['cast'].apply(lambda x: str(x).split(',')).tolist(), index = data['title'])

df_cast_r = df_cast_r.stack().reset_index()

df_cast_r.drop('level_1', axis = 1, inplace = True)

df_cast_r.rename(columns = {0 : 'cast'}, inplace = True)

df_cast_r.sample(10)
```

	title	cast
34951	Anjaan	Samantha Ruth Prabhu
1410	Poseidon	Emmy Rossum
55430	Movie 43	Emma Stone
33771	Beyblade Burst Evolution	Marina Inoue
63838	Under The Skin	Andrew Gorman
49129	Don't Look Down	Richard Branson
6973	Collateral Beauty	Helen Mirren
4203	A Land Imagined	Ishtiaque Zico
62535	The Returned	Jérôme Kircher
54397	Maggie & Bianca: Fashion Friends	Sergio Ruggeri

```
# Spliting the country column

df_country_r = pd.DataFrame(data['country'].apply(lambda x: str(x).split(',')).tolist(), index = data['title'])

df_country_r = df_country_r.stack().reset_index()

df_country_r.drop('level_1', axis = 1, inplace = True)

df_country_r.rename(columns = {0:'country'}, inplace = True)

df_country_r.sample(10)
```

	title	country
1723	Night Stalker: The Hunt for a Serial Killer	United States
3531	Cuddle Weather	Philippines
4384	Léa & I	nan
7823	Colonia	Luxembourg
9264	Océans	United States
5191	Line Walker	Hong Kong
1207	Oloibiri	Nigeria
8205	Frozen Planet	United States
3650	Nailed It! Germany	Germany
6213	Judah Friedlander: America Is the Greatest Cou...	United States

```
# Spliting the listed_in column

df_listed_r = pd.DataFrame(data['listed_in'].apply(lambda x: str(x).split(',')).tolist(), index = data['title'])

df_listed_r = df_listed_r.stack().reset_index()

df_listed_r.drop('level_1', axis = 1, inplace = True)

df_listed_r.rename(columns = {0 : 'listed_in'}, inplace = True)

df_listed_r.sample(10)
```

	title	listed_in
17267	Rembat	Action & Adventure
4606	Meet the In-Laws	International Movies
18938	Trixie Mattel: Moving Parts	Documentaries
15974	Kung Fu Panda: Holiday	Comedies
10086	Illang: The Wolf Brigade	Sci-Fi & Fantasy
17062	Planet Earth: The Complete Collection	British TV Shows
10021	Welcome to Sajjanpur	Dramas
4628	Takki	International TV Shows
17566	Sex and the City: The Movie	Romantic Movies
17102	Power Rangers Operation Overdrive	Kids' TV

```
#Lets combine all this separated data to create the final dataset file to work on

df_new_1 = df_director_r.merge(df_cast_r, how = 'inner', on = 'title')

df_new_1 = df_new_1.merge(df_country_r, how = 'inner', on = 'title')

df_new_1 = df_new_1.merge(df_listed_r, how = 'inner', on = 'title')

df_final = df_new_1.merge(data[['show_id', 'type', 'title', 'date_added',
       'release_year', 'rating', 'duration', 'description']], how = 'inner', on = 'title')

df_final.sample(10)
```

	title	director	cast	country	listed_in	show_id	type	date_adde
73385	Jimmy Neutron: Boy Genius	John A. Davis	Crystal Scales	United States	Sci-Fi & Fantasy	s3063	Movie	January 1 202
33050	Pablo Escobar, el patrón del mal	nan	Angie Cepeda	Colombia	Crime TV Shows	s1350	TV Show	February 3 202
172262	My Only Mother	Wenn V. Deramas	Marvin Agustin	nan	Dramas	s7544	Movie	March 4 201
94100	Delhi Crime	nan	Mridul Sharma	India	Crime TV Shows	s3982	TV Show	March 22 201
193793	The Pelican Brief	Alan J. Pakula	Julia Roberts	United States	Dramas	s8450	Movie	Novembe 1, 201
37758	Giving Voice	Fernando Villena	Aaron Guy	United States	Documentaries	s1559	Movie	Decembe 11, 202
	Don Quixote: The	Will	Lorena	United				January 5

Our Final non comma separated file is ready and we are gonna use this for all our analysis further.

7. Most Common thing in EDA is to deal with blank/Null/NaN/nan Values. Lets see it one by one.

7.1 In 'duration' column we have null values lets update them with '0 min'

```
# Update the code to use fillna() and replace missing values
df_final['duration'].fillna('0 min', inplace=True)

# Update the 'duration' for specific titles where 'duration' is '0 min'
df_final.loc[(df_final['title'] == 'Louis C.K. 2017') & (df_final['duration'] == '0 min'), 'duration'] = '74 min'
df_final.loc[(df_final['title'] == 'Louis C.K.: Live at the Comedy Store') & (df_final['duration'] == '0 min'), 'duration'] = '66 min'

# Print the updated DataFrame
df_final.sample(10)
```

	title	director	cast	country	listed_in	show_id	type	date_added	release_year	rating	duration	description
143233	Blood Father	Jean-François Richet	William H. Macy	France	International Movies	s6337	Movie	March 26, 2020	2016	R	88 min	An ex-convict and estranged father finds the c...
3299	El patrón, radiografía de un crimen	Sebastián Schindel	Joaquín Furriel	Venezuela	Dramas	s141	Movie	September 1, 2021	2014	TV-MA	100 min	A lawyer defends an illiterate man whose explo...
3528	I Got the Hook Up	Michael Martin	Joe Estevez	United States	Comedies	s150	Movie	September 1, 2021	1998	R	93 min	After getting their hands on a misdirected shi...
60208	White Lines	nan	Juan Diego Botto	United Kingdom	International TV Shows	s2531	TV Show	May 15, 2020	2020	TV-MA	1 Season	Zoe Walker leaves her quiet life behind to inv...
180238	Sadma	Balu Mahendra	Sridevi	India	International Movies	s7918	Movie	December 31, 2019	1983	TV-14	137 min	After a traumatic injury leaves her with amnes...
119278	The Method	nan	Paulina Andreeva	Russia	TV Dramas	s5171	TV Show	November 15, 2017	2015	TV-MA	1 Season	An ambitious young law enforcement graduate is...
65990	Ozark	nan	Lisa Emery	United States	TV Dramas	s2768	TV Show	March 27, 2020	2020	TV-MA	3 Seasons	A financial adviser drags his family

8. The 'rating' column has specious entries (74, 84, 66 min respectively) which needs to removed.

```
df_final['rating'].value_counts()
```

```
rating
TV-MA       73915
TV-14       43957
R           25860
PG-13       16246
TV-PG       14926
PG          10919
TV-Y7        6304
TV-Y         3665
TV-G         2779
NR           1573
G            1530
NC-17         149
TV-Y7-FV       86
UR             86
74 min          1
84 min          1
66 min          1
Name: count, dtype: int64
```

8.1 Updating the values which has came in rating to proper column 'duration'.

```
# Update the 'duration' for specific titles where 'duration' is '0 min'
df_final.loc[(df_final['title'] == 'Louis C.K. 2017') & (df_final['rating'] == '74 min'), 'rating'] = 'TV-MA'
df_final.loc[(df_final['title'] == 'Louis C.K.: Live at the Comedy Store') & (df_final['rating'] == '66 min'), 'rating'] = 'TV-14'
df_final.loc[(df_final['rating'] == '84 min'), 'rating'] = 'G'
# Print the updated DataFrame
df_final.sample(10)
```

	title	director	cast	country	listed_in	show_id	type	date_added	release_year	rating	duration	description
133743	Jen Kirkman: I'm Gonna Die Alone (And I Feel F...	Lance Bangs	Jen Kirkman	United States	Stand-Up Comedy	s5912	Movie	May 22, 2015	2015	TV-MA	78 min	Jen Kirkman delivers some sharp, hilarious tru...
196552	The Young Vagabond	Sze Yu Lau	Jason Pai Piao	Hong Kong	Action & Adventure	s8570	Movie	August 16, 2018	1985	TV-14	85 min	To avenge his master, a wine-loving young man ...
156263	Grease	Randal Kleiser	Sid Caesar	United States	Music & Musicals	s6894	Movie	November 1, 2019	1978	PG	110 min	John Travolta and Olivia Newton-John star in t...
200878	Wrong Side Raju	Mikhil Musale	Kimberley Louisa McBeath	India	Dramas	s8763	Movie	October 1, 2017	2016	TV-MA	140 min	The life of a chauffeur and part-time bootlegg...
149554	Di Renjie zhi Sidatianwang	Hark Tsui	Mark Chao	Hong Kong	Sci-Fi & Fantasy	s6606	Movie	February 6, 2019	2018	TV-14	132 min	Framed by an empress who plans to steal a drag...
2337	Angamaly Diaries	Lijo Jose Pellissery	Sreekanth Dasan	India	Action & Adventure	s106	Movie	September 5, 2021	2017	TV-14	128 min	After growing up amidst the gang wars of his h...
143141	Blaze	Ethan Hawke	Richard Linklater	United States	Dramas	s6332	Movie	August 30, 2020	2018	R	129 min	An influential, if unsung country songwriter r...

7.2 Lets remove the duplicated values if there is any.

```
# Handling the Duplicated records

df_final.drop_duplicates(keep= 'first', inplace= True, ignore_index=True)
```

7.3 Lets dill with 'nan' values

```
df_final.isna().sum()
```

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

```
#Handeling null values in Date_added column
mode_value=df_final['date_added'].mode()[0]
df_final['date_added'].fillna(mode_value, inplace=True)

#Handeling null values in rating column
mode_value=df_final["rating"].mode()[0]
df_final["rating"].fillna(mode_value, inplace=True)
```

7.4 We do have few 'na' values which need to be replaced with the common entry.

```
df_final['duration'].replace('nan', 'unknown director', inplace = True)
df_final['country'].replace('nan', 'unknown country', inplace = True)
df_final['director'].replace('nan', 'unknown director', inplace = True)
df_final['country'].replace('', 'unknown country', inplace = True)
df_final['cast'].replace('nan', 'unknown cast', inplace = True)
```

```
df_final.isna().sum()
```

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

As we have remove the Nulls, Dulicates and also the blanks in the given dataset. Lets focus on Problem statement

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

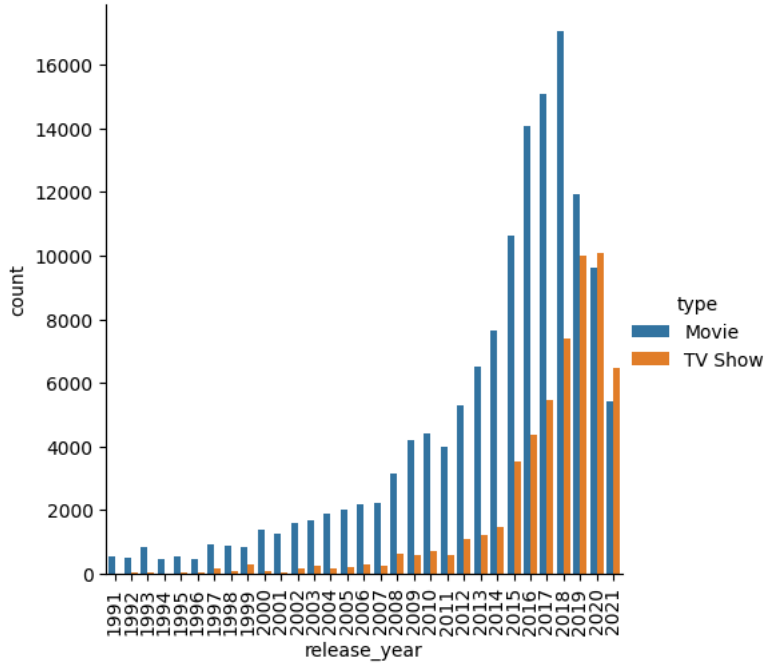
```
df_final['type'].value_counts().reset_index()
```

	type	count
0	Movie	145910
1	TV Show	56148

Business Insights :- We have most number of 'Movies' compared to 'TV Show'

```
# 1.    The number of movies released per year changed over the last 20-30 years?

df_ye = df_final.groupby(['type','release_year']).size().reset_index()

df_ye = df_ye[df_ye['release_year'] > 1990]

df_ye.rename(columns={0:'count'}, inplace = True)

plt.figure(figsize = (10,6))

sns.catplot(data=df_ye, x="release_year", y="count", hue="type", kind="bar")

plt.xticks(rotation = 90)

plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



Business Insights:- In Above graph, we can clearly see the rise in number of movies over past 20 - 30 years. But interenstingly the 'TV Shows' Has grown significantly over the past 5 - 6 years. Which is equivalent to Movies as well.

This clearly indicates the popularity of 'TV shows' Has increased over past 5 years, and it also surpases the Movies for last two years

```
# What type of content is available in different countries?

# Group by country, type, and genre and count occurrences
df_content = df_final.groupby(['country', 'type', 'listed_in']).size().reset_index(name='count')

df_content.sample(10)
```

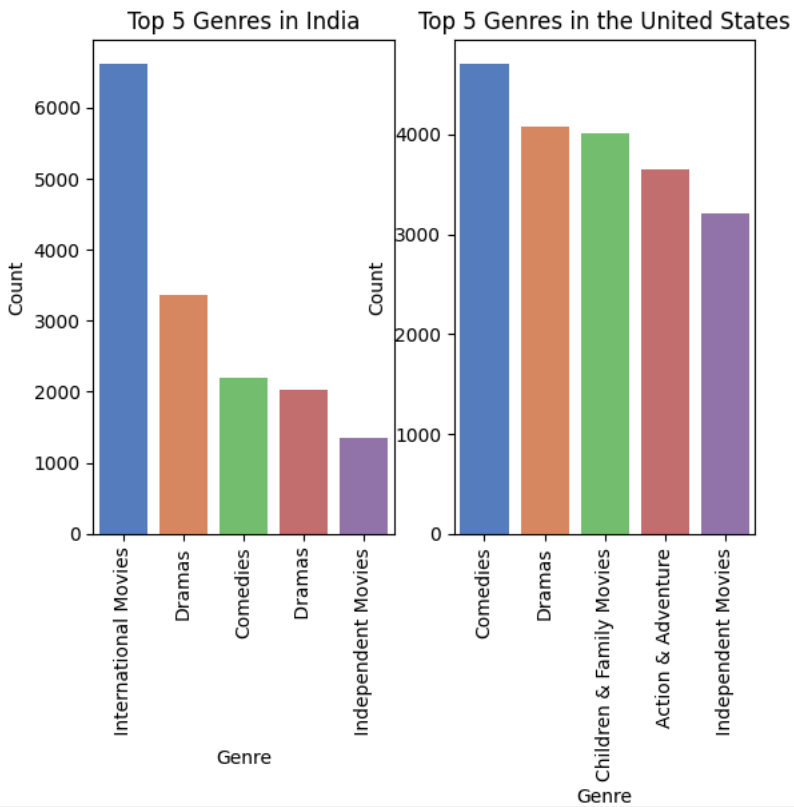
	country	type	listed_in	count
1882	Norway	Movie	Action & Adventure	24
1819	Netherlands	Movie	LGBTQ Movies	12
2112	Spain	Movie	LGBTQ Movies	20
66	Belgium	Movie	Dramas	103
1148	Austria	Movie	International Movies	10
1534	India	Movie	Dramas	2035
1701	Japan	TV Show	TV Horror	50
657	New Zealand	Movie	Sci-Fi & Fantasy	46
2318	United Kingdom	TV Show	Stand-Up Comedy & Talk Shows	1
548	Kazakhstan	Movie	Children & Family Movies	1

Above code shows the different genre of content available in countries w.r.t type of show

```
# Filter data for India and USA
df_filtered = df_final[df_final['country'].isin(['India', 'United States'])]

# What type of content is available in different countries?

# Group by country, type, and genre and count occurrences
df_content = df_filtered.groupby(['country', 'type', 'listed_in']).size().reset_index(name='count')

# Pivot the data to create a stacked bar chart
df_pivot = df_content.pivot_table(index='country', columns=['type', 'listed_in'], values='count', fill_value=0)

# Group by country and genre, count occurrences, and find top 5 genres for each country
```

```
top_genres = df_filtered.groupby(['country', 'listed_in']).size().reset_index(name='count')
top_genres = top_genres.sort_values(by=['country', 'count'], ascending=[True, False]).groupby('country').head(5)

# Bar chart for India
plt.subplot(1, 2, 1)
sns.barplot(data=top_genres[top_genres['country'] == 'India'], x='listed_in', y='count', palette='muted')
plt.title('Top 5 Genres in India')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=90)

# Bar chart for the United States
plt.subplot(1, 2, 2)
sns.barplot(data=top_genres[top_genres['country'] == 'United States'], x='listed_in', y='count', palette='muted')
plt.title('Top 5 Genres in the United States')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-23-d07c8cb9fdcb>:18: 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

  sns.barplot(data=top_genres[top_genres['country'] == 'India'], x='listed_in', y='count', palette='muted')
<ipython-input-23-d07c8cb9fdcb>:26: 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

  sns.barplot(data=top_genres[top_genres['country'] == 'United States'], x='listed_in', y='count', palette='muted')
```



Business Insights :- Above plot shows the top 5 genres of Tv_Shows/Movies in India and USA.

In India 'International Movies is most popular genre while in USA 'comedies' is leading.

```
#Top 10 country where most movies is release
temp_country=df_final[["show_id","type","country"]].drop_duplicates(keep="first").reset_index(drop=True)
temp_country[temp_country["type"]=="Movie"]["country"].value_counts()[:10]
```

```
country
United States      2364
India               927
unknown country     446
 United States      388
```

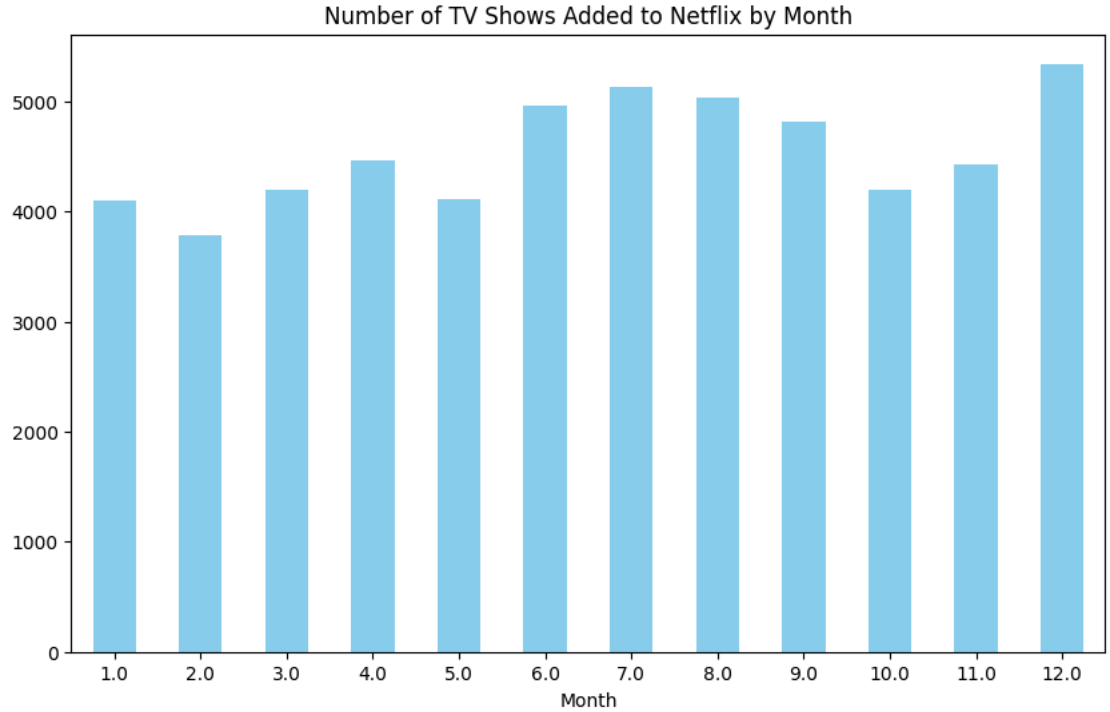
```
United Kingdom      382
Canada              187
 France             155
 United Kingdom     152
France              148
 Canada             132
Name: count, dtype: int64
```

```
#Top 10 country where most TV Shows is release

temp_country[temp_country['type'] == "TV Show"]['country'].value_counts()[:10]
```

```
country
United States      847
unknown country    392
United Kingdom     246
Japan              174
South Korea        164
 United States      91
Canada              84
India               81
Taiwan              70
France              64
Name: count, dtype: int64
```

```
df_final['date_added'] = pd.to_datetime(df_final['date_added'], errors='coerce', infer_datetime_format=True)

df_final['month_added'] = df_final['date_added'].dt.month

tv_shows = df_final[df_final['type'] == 'TV Show']

tv_shows_by_month = tv_shows['month_added'].value_counts().sort_index()

# Plot the results
plt.figure(figsize=(10, 6))

# Bar plot of TV shows added by month
tv_shows_by_month.plot(kind='bar', color='skyblue')

# Set plot labels and title
plt.xlabel('Month')
plt.ylabel('Number of TV Shows Added')
plt.title('Number of TV Shows Added to Netflix by Month')
plt.xticks(rotation=0)  # Keep month labels horizontal for better readability

# Show plot
plt.show()

# Print the month with the highest number of TV show launches
best_month = tv_shows_by_month.idxmax()
best_month_count = tv_shows_by_month.max()

print(f"The best month to launch a TV show is {best_month} with {best_month_count} shows added")
```

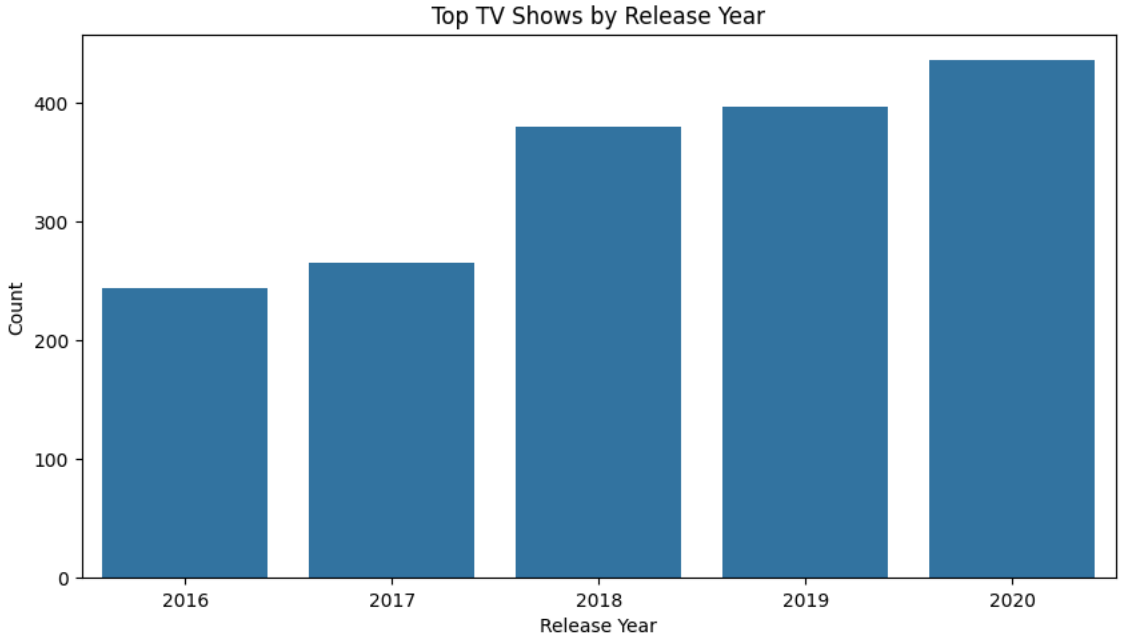
```
/thon-input-24-27af0cd1033d>:1: UserWarning: The argument 'infer_datetime_format' is deprecated and will be removed in a future version.
f_final['date_added'] = pd.to_datetime(df_final['date_added'], errors='coerce', infer_datetime_format=True)
```

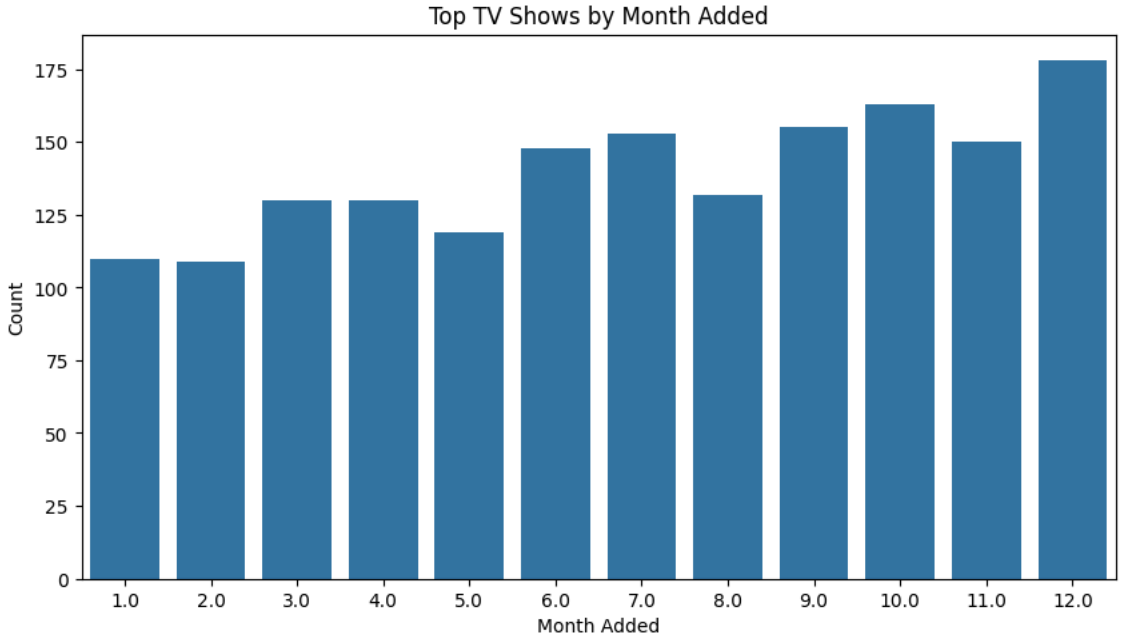


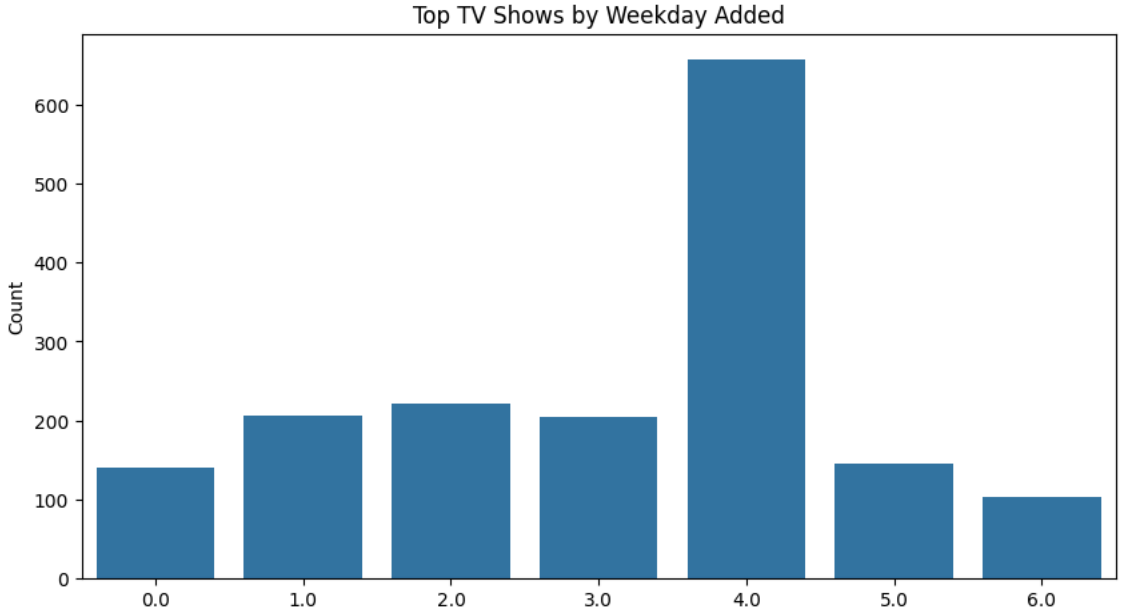

```
best month to launch a TV show is 12.0 with 5341 shows added
```

```
df_final['day_added'] = df_final['date_added'].dt.day

df_final['week_day_added'] = df_final['date_added'].dt.weekday

df_an = df_final[['show_id','type','date_added', 'release_year', 'month_added', 'day_added', 'week_day_added']]

df_an = df_an.drop_duplicates(keep = "first")


top_year = df_an['release_year'].value_counts().index[:5]

top_tv_show = df_an[df_an['release_year'].isin(top_year) & (df_an['type'] == "TV Show")][['show_id','type','date_added', 'release_year', 'mo

top_tv_show.sample(10)

year = top_tv_show['release_year'].value_counts().reset_index()

month = top_tv_show['month_added'].value_counts().reset_index()

week_day = top_tv_show['week_day_added'].value_counts().reset_index()

# Create subplots
fig, axes = plt.subplots(3, 1, figsize=(10, 18))

# Yearly bar plot
sns.barplot(x='release_year', y='count', data=year, ax=axes[0])
axes[0].set_title('Top TV Shows by Release Year')
axes[0].set_xlabel('Release Year')
axes[0].set_ylabel('Count')

# Monthly bar plot
sns.barplot(x='month_added', y='count', data=month, ax=axes[1])
axes[1].set_title('Top TV Shows by Month Added')
axes[1].set_xlabel('Month Added')
axes[1].set_ylabel('Count')

# Weekday bar plot
sns.barplot(x='week_day_added', y='count', data=week_day, ax=axes[2])
axes[2].set_title('Top TV Shows by Weekday Added')
axes[2].set_xlabel('Weekday Added')
axes[2].set_ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```







Weekday Added

Business Insights :- As we can interpret from above plots. Most of the Tv shows were aired in the month of December and on the 4th Day of the week i.e. Thuresday

Best Time to Release a TV show is on Thuresday and in 12th Month of the year

4. Analysis of actors and directors of different types of shows/movies

```
# 4.1   Analysis of directors of different types of shows/movies.

movies = df_final[df_final['type'] == 'Movie']

tv_shows = df_final[df_final['type'] == 'TV Show']

movie_directors = movies['director'].value_counts().reset_index()

movie_directors.columns = ['director', 'movie_count']

tv_directors = tv_shows['director'].value_counts().reset_index()

tv_directors.columns = ['director', 'tv_shows_count']

director_analysis = pd.merge(movie_directors, tv_directors, on = 'director', how = 'outer').fillna(0)

director_analysis['movie_count'] = director_analysis['movie_count'].astype(int)

director_analysis['tv_shows_count'] = director_analysis['tv_shows_count'].astype(int)

director_analysis.sort_values(by = ['movie_count', 'tv_shows_count'], ascending = False)[1:11]
```

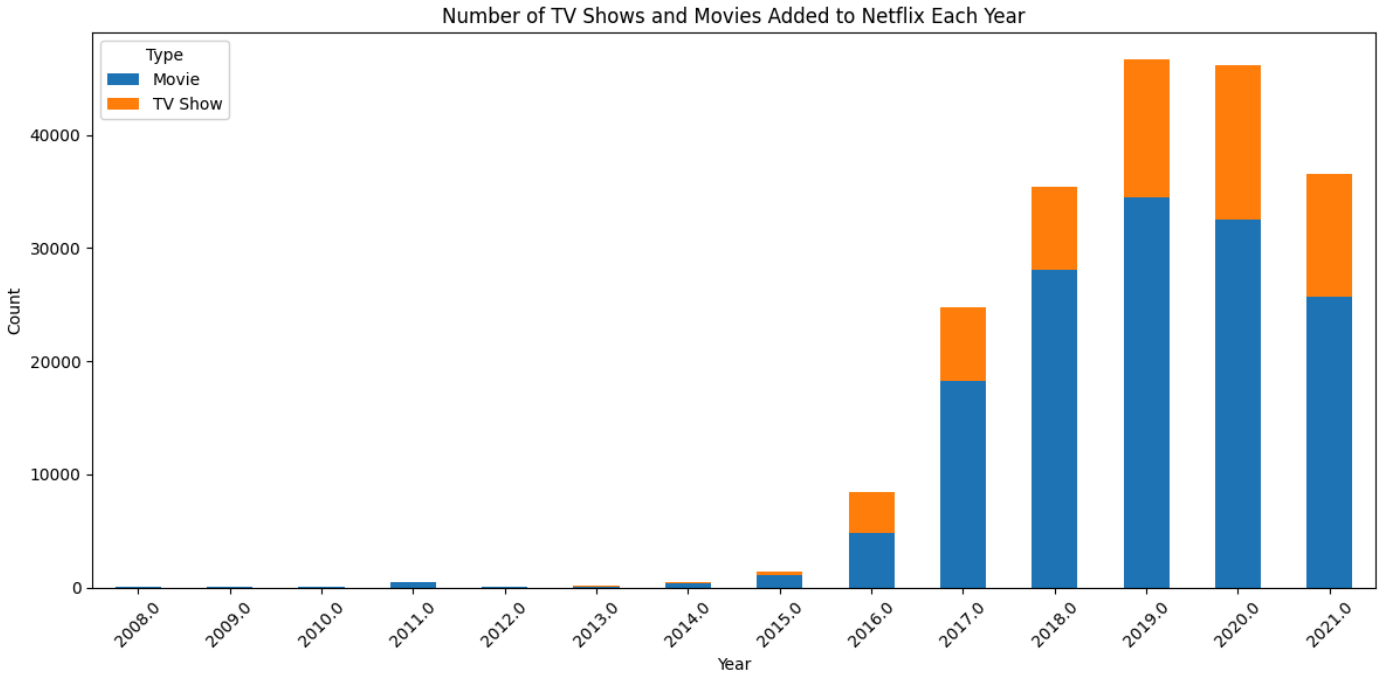
	director	movie_count	tv_shows_count
1	Martin Scorsese	419	0
2	Youssef Chahine	409	0
3	Cathy Garcia-Molina	356	0
4	Steven Spielberg	355	0
5	Lars von Trier	336	0
6	Raja Gosnell	308	0
7	Tom Hooper	306	0
8	McG	293	0
9	David Dhawan	270	0
10	Wilson Yip	260	0

```
# 4.2   Analysis of directors of different types of shows/movies.

movie_actors = movies['cast'].value_counts().reset_index()

movie_actors.columns = ['cast', 'movie_count']

tv_show_actors = tv_shows['cast'].value_counts().reset_index()

tv_show_actors.columns = ['cast', 'tv_shows_count']

actor_analysis = pd.merge(movie_actors, tv_show_actors, on = 'cast', how = 'outer').fillna(0)

actor_analysis['movie_count'] = actor_analysis['movie_count'].astype(int)

actor_analysis['tv_shows_count'] = actor_analysis['tv_shows_count'].astype(int)

actor_analysis.sort_values(by = ['movie_count', 'tv_shows_count'], ascending = False)[1:11]
```

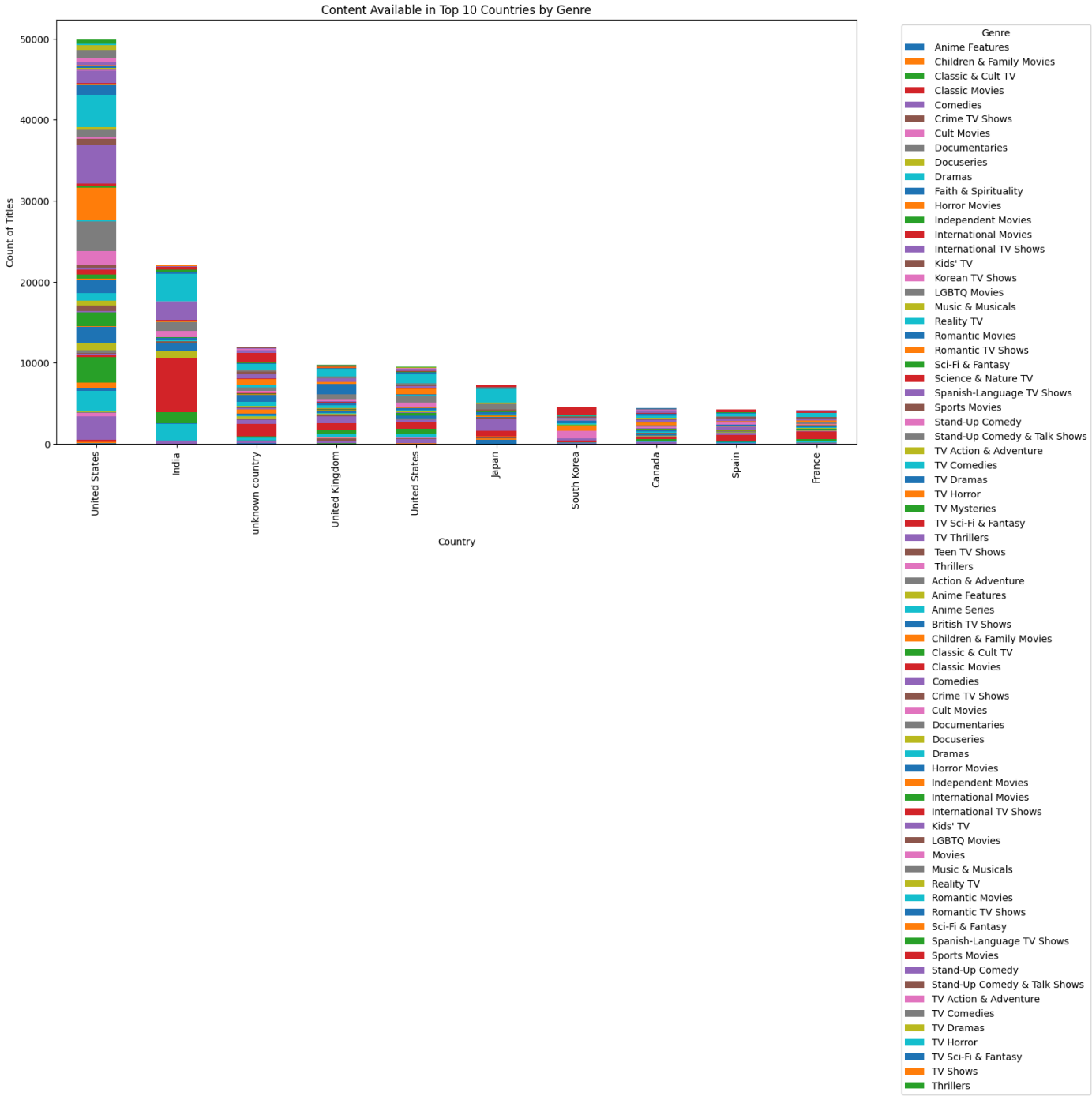
	cast	movie_count	tv_shows_count
1	Alfred Molina	157	3
2	Salma Hayek	130	0
3	Frank Langella	128	0
5	John Krasinski	120	1
4	Liam Neeson	120	0
6	John Rhys-Davies	116	9
7	Anupam Kher	107	9
8	Quvenzhané Wallis	100	0
9	Jim Broadbent	86	6
10	Ben Whishaw	86	0

```
# Convert 'date_added' to datetime
df_final['date_added'] = pd.to_datetime(df_final['date_added'], errors='coerce', infer_datetime_format=True)

# Extract year from 'date_added'
df_final['year_added'] = df_final['date_added'].dt.year

# Group by year and type
yearly_focus = df_final.groupby(['year_added', 'type']).size().unstack(fill_value=0)

# Plot the trend over years
plt.figure(figsize=(12, 6))
yearly_focus.plot(kind='bar', stacked=True, ax=plt.gca())
plt.title('Number of TV Shows and Movies Added to Netflix Each Year')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Type')
plt.tight_layout()
plt.show()
```

```
<ipython-input-59-6729e2ca1a05>:2: UserWarning: The argument 'infer_datetime_format' is deprecated and will be removed in a future versi
  df_final['date_added'] = pd.to_datetime(df_final['date_added'], errors='coerce', infer_datetime_format=True)
```



```
# Group by country and genre (listed_in) and count the occurrences
country_genre = df_final.groupby(['country', 'listed_in']).size().unstack(fill_value=0)

# Display the top 10 countries with the most content
top_countries = country_genre.sum(axis=1).sort_values(ascending=False).head(10).index
top_country_genre = country_genre.loc[top_countries]

# Plotting the data
plt.figure(figsize=(15, 8))
top_country_genre.plot(kind='bar', stacked=True, ax=plt.gca())
plt.title('Content Available in Top 10 Countries by Genre')
plt.xlabel('Country')
plt.ylabel('Count of Titles')
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```




```
df_final.columns
```

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

```
df_final['year_added'] = pd.to_datetime(df_final['year_added'])


tv_shows = df_final[df_final['type'] == 'TV Show']
movie = df_final[df_final['type'] == 'Movie']

# Extract the year from date_added as integer
tv_shows['year_added'] = tv_shows['date_added'].dt.year.fillna(0).astype(int)
```