netflix-busniess-case-study

April 26, 2024

#Netflix is an American subscription video on-demand over-the-top streaming service. The service primarily distributes original and acquired films and television shows from various genres, and it is available internationally in multiple languages "' source wikipedia "'

#Problem statement # Analysing the datset, get the insights and finally recommend the netflix to help its business to expand its growth across globe.

[ ]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
#loading the dataset
df = pd.read_csv("https://gist.github.com/singhsidhukuldeep/
 ↪564f271315abb6bc22647e81e6bf4762/raw/
 ↪66fb67a8bb014df6b7f924aad0a91aa662bc7fc2/netflix_titles.csv")
print(df.shape)
df.head()
```

```
(8807, 12)
```

[ ]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…	South Africa	September 24, 2021	2021	TV-MA	2 Seasons
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…	NaN				
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN				
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…	India				

```
2  September 24, 2021          2021  TV-MA   1 Season
3  September 24, 2021          2021  TV-MA   1 Season
4  September 24, 2021          2021  TV-MA  2 Seasons

                                                listed_in  \
0                                          Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act…
3                               Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV …

                                             description
0  As her father nears the end of his life, filmm…
1  After crossing paths at a party, a Cape Town t…
2  To protect his family from a powerful drug lor…
3  Feuds, flirtations and toilet talk go down amo…
4  In a city of coaching centers known to train I…
```

#Analyzing the basic metrics of given dataset.

[ ]:
```
#data types of all attributes
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
```

[ ]:
```
#Count of non-null values in each column:
df.count()
```

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

#When you call df.info() on a DataFrame df, it returns a summary that includes:

#The total number of entries (rows) is around 8807. #The number of columns is 12. #The data type of each column except release_year is object.

[ ]:
```
#summary of dataset since, year is the only numerical value we are getting␣
↪release_year
df.describe()
```

[ ]:
```
       release_year
count   8807.000000
mean    2014.180198
std        8.819312
min     1925.000000
25%     2013.000000
50%     2017.000000
75%     2019.000000
max     2021.000000
```

1 The describe indicate the netflix has been very active after 2017

[ ]:
```
#missing value detection and data cleaning and imputation
print('\n illustration columns with null values')
print(df.isnull().any())
print('Count of missing values in each column')
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)

print('Proportion of missing values in each column')
missing_percentage = (missing_values / len(df)) * 100
print("\nPercentage of missing values:\n", missing_percentage)
```

```
 illustration columns with null values
show_id         False
type            False
title           False
director         True
cast             True
country          True
date_added       True
release_year    False
rating           True
duration         True
listed_in       False
description     False
dtype: bool
Count of missing values in each column
Missing values:
 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
Proportion of missing values in each column

Percentage of missing values:
 show_id          0.000000
type             0.000000
title            0.000000
director        29.908028
cast             9.367549
country          9.435676
date_added       0.113546
release_year     0.000000
rating           0.045418
duration         0.034064
listed_in        0.000000
description      0.000000
dtype: float64
```

#From the above data it clearly indicated that below columns have null values with director column has maximum number with 2634 null values followed by cast with 825, country with 831,

date_added with 10, rating with 4 ,duration with 3 null values

The director column has maximum percentage of null values. Also, columns like has date_added,rating, duration has minimal percentage of null vales. hence removing the rows. Moreover, columns like director, country, cast are filling . finally the data will be out of null values as below.

[ ]:
```
#data imputation
'''
since the percentage of missing values for date added ,duration,rating rows␣
 ↪will be dropped.
for other columsn like director,cast, country dat will be imputed
'''
df.dropna(subset=['date_added'], inplace=True)
df.dropna(subset=['rating'], inplace=True)
df.dropna(subset=['duration'], inplace=True)
df.director.fillna("No_Director", inplace=True)
df.country.fillna("No_country", inplace=True)
df.cast.fillna("No_cast", inplace=True)

#checking the new dataset after cleaning and no more null
print(df.isnull().any())

print('Count of missing values in each column after data imputaion')
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)

print('Proportion of missing values in each column after imputation')
missing_percentage = (missing_values / len(df)) * 100
print("\nPercentage of missing values:\n", missing_percentage)
```

```
show_id         False
type            False
title           False
director        False
cast            False
country         False
date_added      False
release_year    False
rating          False
duration        False
listed_in       False
description     False
dtype: bool
Count of missing values in each column after data imputaion
Missing values:
 show_id         0
type            0
```

```
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
Proportion of missing values in each column after imputation

Percentage of missing values:
 show_id         0.0
type             0.0
title            0.0
director         0.0
cast             0.0
country          0.0
date_added       0.0
release_year     0.0
rating           0.0
duration         0.0
listed_in        0.0
description      0.0
dtype: float64
```

2 creating the year_added column to illustrate the content added in netflix

[ ]:
```
# Convert the "date_added" column to datetime format
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# Extract the year and create a new column
df['year_added'] = df['date_added'].dt.year  # Convert to integer

# Display the DataFrame with the new column
print(df)
```

```
     show_id     type                  title         director  \
0         s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1         s2  TV Show          Blood & Water      No_Director
2         s3  TV Show              Ganglands  Julien Leclercq
3         s4  TV Show  Jailbirds New Orleans      No_Director
4         s5  TV Show           Kota Factory      No_Director
…       …       …                     …              …
```

```
8802   s8803    Movie                 Zodiac    David Fincher
8803   s8804  TV Show            Zombie Dumb      No_Director
8804   s8805    Movie             Zombieland  Ruben Fleischer
8805   s8806    Movie                   Zoom     Peter Hewitt
8806   s8807    Movie                 Zubaan      Mozez Singh

                                                    cast        country  \
0                                                No_cast  United States
1     Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…   South Africa
2     Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…     No_country
3                                                No_cast     No_country
4     Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…           India
…                                                   …            …
8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J…  United States
8803                                             No_cast     No_country
8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, …  United States
8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma…  United States
8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…           India

     date_added  release_year rating   duration  \
0    2021-09-25          2020  PG-13     90 min
1    2021-09-24          2021  TV-MA  2 Seasons
2    2021-09-24          2021  TV-MA   1 Season
3    2021-09-24          2021  TV-MA   1 Season
4    2021-09-24          2021  TV-MA  2 Seasons
…           …           …    …         …
8802 2019-11-20          2007      R    158 min
8803 2019-07-01          2018  TV-Y7  2 Seasons
8804 2019-11-01          2009      R     88 min
8805 2020-01-11          2006     PG     88 min
8806 2019-03-02          2015  TV-14    111 min

                                          listed_in  \
0                                     Documentaries
1       International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act…
3                               Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV …
…                                                   …
8802                    Cult Movies, Dramas, Thrillers
8803            Kids' TV, Korean TV Shows, TV Comedies
8804                          Comedies, Horror Movies
8805                Children & Family Movies, Comedies
8806     Dramas, International Movies, Music & Musicals

                                        description  year  year_added
0     As her father nears the end of his life, filmm…  2021        2021
1     After crossing paths at a party, a Cape Town t…  2021        2021
```

```
2     To protect his family from a powerful drug lor…  2021        2021
3     Feuds, flirtations and toilet talk go down amo…  2021        2021
4     In a city of coaching centers known to train I…  2021        2021
…                                                   …   …         …
8802  A political cartoonist, a crime reporter and a…  2019        2019
8803  While living alone in a spooky town, a young g…  2019        2019
8804  Looking to survive in a world taken over by zo…  2019        2019
8805  Dragged from civilian life, a former superhero…  2020        2020
8806  A scrappy but poor boy worms his way into a ty…  2019        2019

[8790 rows x 14 columns]
```

3 checking the shape of the dataset after cleaning

4 Shape of the new data is (8790 , 13)

[ ]:
```
#checking the new dataset after adding new column.
print(df.isnull().any())
#information
print(df.info())
```

```
show_id         False
type            False
title           False
director        False
cast            False
country         False
date_added      False
release_year    False
rating          False
duration        False
listed_in       False
description     False
year            False
year_added      False
dtype: bool
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   cast          8790 non-null   object
 5   country       8790 non-null   object
```

```
 6   date_added    8790 non-null   datetime64[ns]
 7   release_year  8790 non-null   int64
 8   rating        8790 non-null   object
 9   duration      8790 non-null   object
 10  listed_in     8790 non-null   object
 11  description   8790 non-null   object
 12  year          8790 non-null   int32
 13  year_added    8790 non-null   int32
dtypes: datetime64[ns](1), int32(2), int64(1), object(10)
memory usage: 961.4+ KB
None
```

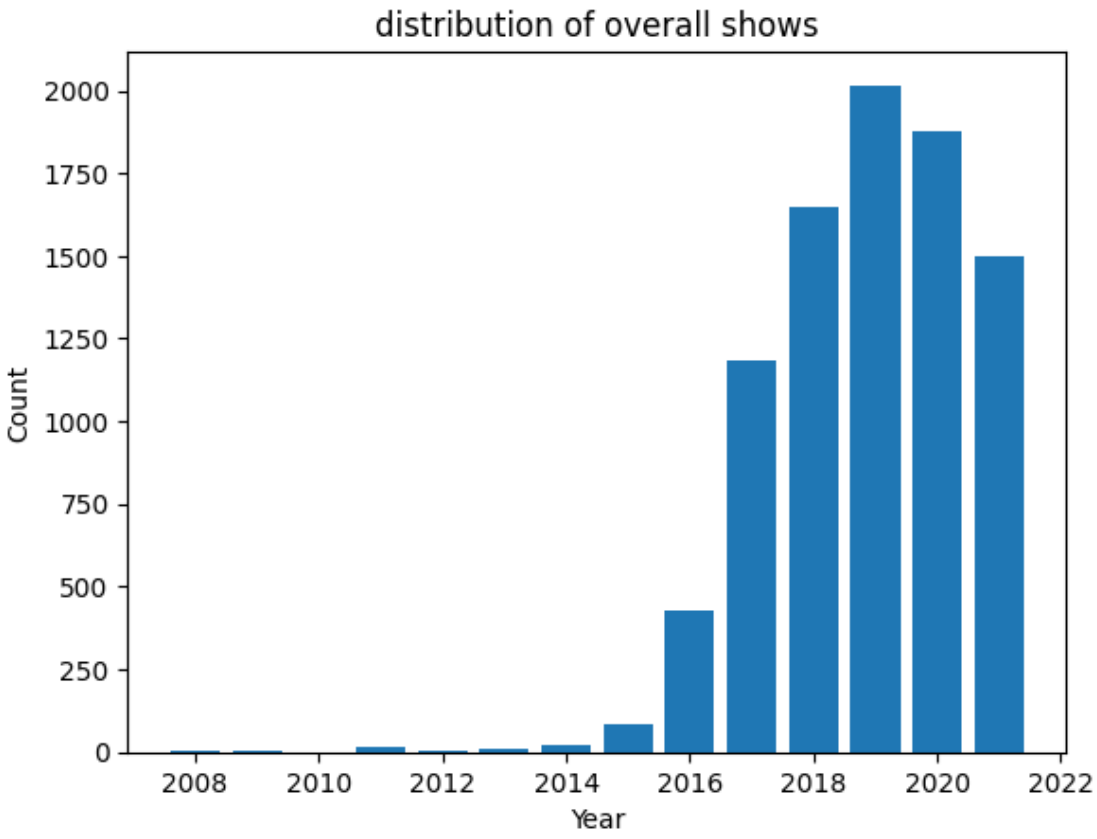
#Exploratory data analysis

#Trend of content added in netflix over the years

[ ]:
```
# Trend in release years
year_added_trend = df.groupby('year_added').size()
print("\nTrend in release years:\n", year_added_trend )
```

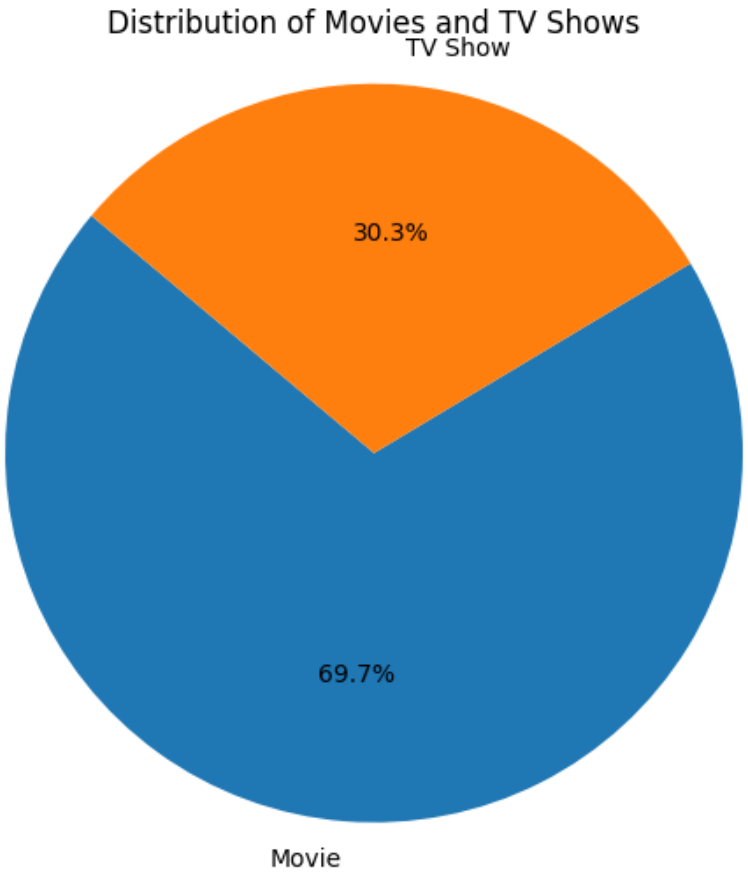
```
Trend in release years:
 year_added
2008       2
2009       2
2010       1
2011      13
2012       3
2013      11
2014      24
2015      82
2016     426
2017    1185
2018    1648
2019    2016
2020    1879
2021    1498
dtype: int64
```

[ ]:
```
#distribution of overall shows mvoesd
year_counts = df['year_added'].value_counts()
year_counts = year_counts.sort_index()

# Plot the histogram
plt.bar(year_counts.index, year_counts.values)
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('distribution of overall shows')
plt.show()
```



#Insights from plot: #It can be clearly seen from the above graph that content adding in the netflix has taken a surge in 2017 and maintaining of adding more than 1500 content which inlcudes both movies and TV shows in next corresponding years .

[ ]:
```
#distribution of movies and tv shows in netflix.
type_counts = df['type'].value_counts()

# Create labels and sizes for the pie chart
labels = type_counts.index
sizes = type_counts.values

# Plotting the pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Distribution of Movies and TV Shows')
plt.show()
```



#Insights from pie chart: #So far, Movies has the more contribution to netflix compared TV Shows with more than 30 % range difference.

[ ]:
```
country_df = df[["title", "country"]]
country_df["unnested_country"] = country_df ["country"].apply(lambda x: str(x).
  ↪split(", "))
country_df = country_df.explode("unnested_country")
country_df.head(10)
```

```
<ipython-input-53-2552acf7a06d>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
  country_df["unnested_country"] = country_df ["country"].apply(lambda x:
str(x).split(", "))
```

[ ]:
```
                            title  \
0             Dick Johnson Is Dead
1                    Blood & Water
2                        Ganglands
3            Jailbirds New Orleans
4                     Kota Factory
5                    Midnight Mass
6  My Little Pony: A New Generation
7                          Sankofa
7                          Sankofa
7                          Sankofa

                                        country unnested_country
0                                 United States    United States
1                                  South Africa     South Africa
2                                    No_country       No_country
3                                    No_country       No_country
4                                         India            India
5                                    No_country       No_country
6                                    No_country       No_country
7  United States, Ghana, Burkina Faso, United Kin…    United States
7  United States, Ghana, Burkina Faso, United Kin…            Ghana
7  United States, Ghana, Burkina Faso, United Kin…     Burkina Faso
```

5 unnesting the country column to get the insights for individual country share to netflix content.

[ ]:
```
country_df.groupby('unnested_country')["title"].nunique()
```

[ ]:
```
unnested_country
                  2
Afghanistan       1
Albania           1
Algeria           3
Angola            1
                 ..
Vatican City      1
Venezuela         4
Vietnam           7
West Germany      5
Zimbabwe          3
Name: title, Length: 128, dtype: int64
```

[ ]:
```
print('Proportion of missing values in each column')
missing_percentage = (missing_values / len(df)) * 100
print("\nPercentage of missing values:\n", missing_percentage)
```

[ ]:
```
top_countries = country_df['unnested_country'].value_counts().head(10)
print(top_countries)
```

```
unnested_country
United States     3680
India             1046
No_country         829
United Kingdom     803
Canada             445
France             393
Japan              316
Spain              232
South Korea        231
Germany            226
Name: count, dtype: int64
```

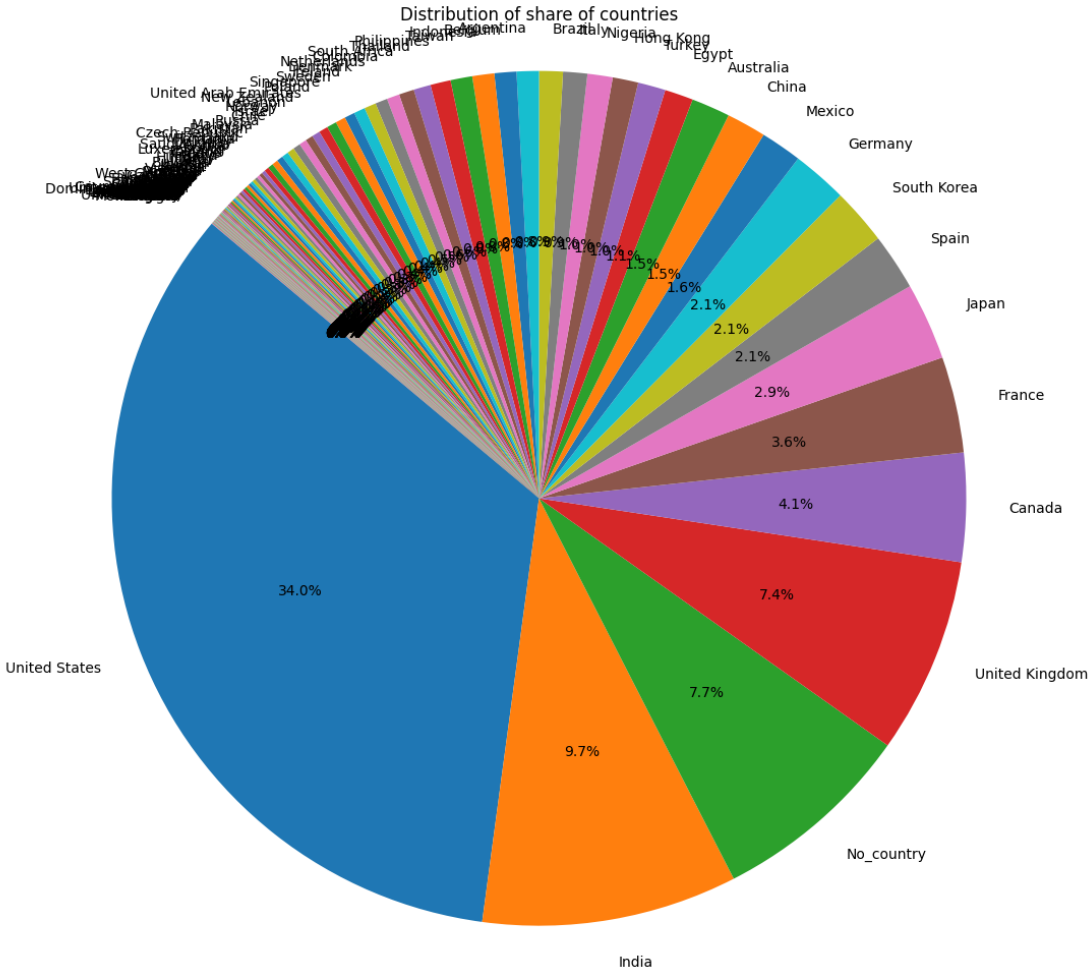
[ ]:
```
print('Proportion of country share in netflix content')
country_df_share = (top_countries / len(country_df)) * 100
print("\nPercentage of country share in netflix content:\n", country_df_share)
```

```
Proportion of country share in netflix content

Percentage of country share in netflix content:
 unnested_country
United States     33.985962
India              9.660140
No_country         7.656077
United Kingdom     7.415959
Canada             4.109716
France             3.629479
Japan              2.918360
Spain              2.142593
South Korea        2.133358
Germany            2.087181
Name: count, dtype: float64
```

6 Insight from above data indicated that united states tops the list with major share , followed by India , united kingdom.

[ ]:
```
#distribution of movies and tv shows in netflix.
country_counts =country_df['unnested_country'].value_counts()

# Create labels and sizes for the pie chart
labels = country_counts.index
sizes = country_counts.values
```

```
# Plotting the pie chart
plt.figure(figsize=(14, 12))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Distribution of share of countries')
plt.show()
```



7 Pie chart clearly showcase the distrubution of each country share.

8 unnesting the actor column for preprocessing

[ ]:
```
cast_df = df[["title", "cast"]]
cast_df["unnested_cast"] = cast_df ["cast"].apply(lambda x: str(x).split(", "))
cast_df = cast_df.explode("unnested_cast")
cast_df.head(10)
```

```
<ipython-input-97-add1bcc4348b>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
                title                                               cast  \
0  Dick Johnson Is Dead                                            No_cast
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

        unnested_cast
0             No_cast
1          Ama Qamata
1         Khosi Ngema
1       Gail Mabalane
1      Thabang Molaba
1    Dillon Windvogel
1     Natasha Thahane
1         Arno Greeff
1  Xolile Tshabalala
1     Getmore Sithole
```

[ ]:
```
filtered_df = cast_df[cast_df['unnested_cast'] != 'No_cast']
filtered_df.unnested_cast.value_counts().head()
print(filtered_df.unnested_cast.value_counts().head())
```

```
unnested_cast
Anupam Kher          43
Shah Rukh Khan       35
Julie Tejwani        33
Naseeruddin Shah     32
Takahiro Sakurai     32
Name: count, dtype: int64
```

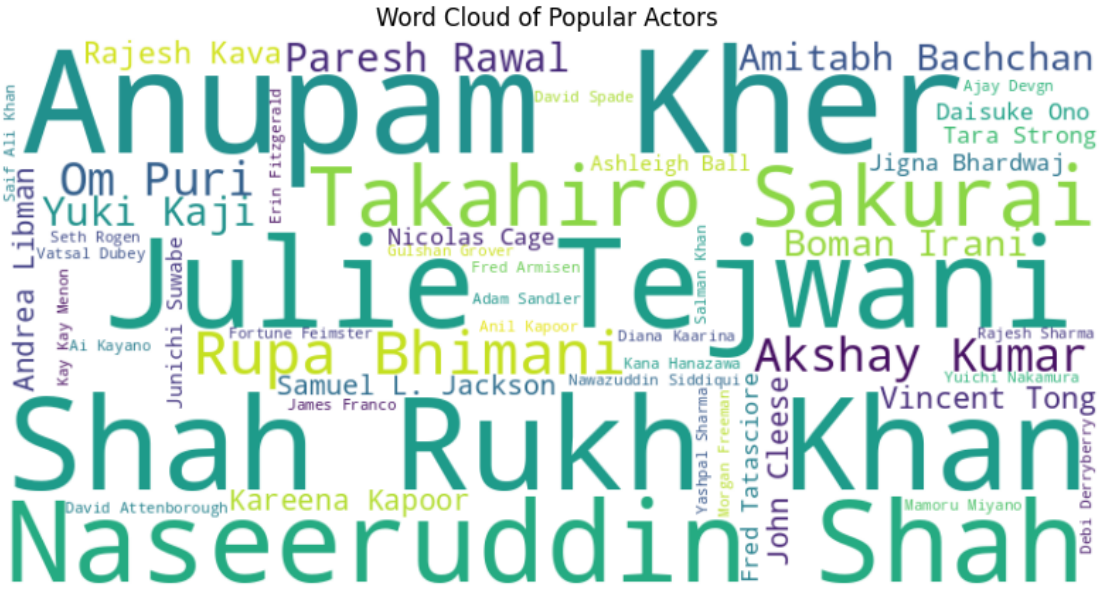
9 From the above analysis it is clearly states that anupam kher tops the list followed by Shah Rukh Khan , Julie Tejwani in most popular actoror as by netflix data.

[ ]:
```
print(filtered_df)
```

```
              title                                                cast  \
1     Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1     Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1     Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1     Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1     Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
…               …                                                  …
8806         Zubaan  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…
8806         Zubaan  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…
8806         Zubaan  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…
8806         Zubaan  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…
8806         Zubaan  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…

              unnested_cast
1               Ama Qamata
1              Khosi Ngema
1             Gail Mabalane
1            Thabang Molaba
1          Dillon Windvogel
…                       …
8806       Manish Chaudhary
8806           Meghna Malik
8806          Malkeet Rauni
8806         Anita Shabdish
8806  Chittaranjan Tripathy

[64016 rows x 3 columns]
```

[ ]:
```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
# Assuming you already have a DataFrame called cast_df
# Filter out rows where 'unnested_cast' is 'No_cast'
filtered_castdf = cast_df[cast_df['unnested_cast'] != 'No_cast']

# Get the top 10 most popular unnested cast members
top_cast_counts = filtered_castdf['unnested_cast'].value_counts().head(50)

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate_from_frequencies(top_cast_counts)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Popular Actors ')
plt.axis('off')
plt.show()
```



#unnesting the director column

[ ]:
```
#most popular directors
director_df = df[["title", "director"]]
director_df["unnested_director"] = director_df ["director"].apply(lambda x:␣
 ↪str(x).split(", "))
director_df= director_df.explode("unnested_director")
director_df.head(10)
```

```
<ipython-input-110-d1cce321843b>: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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

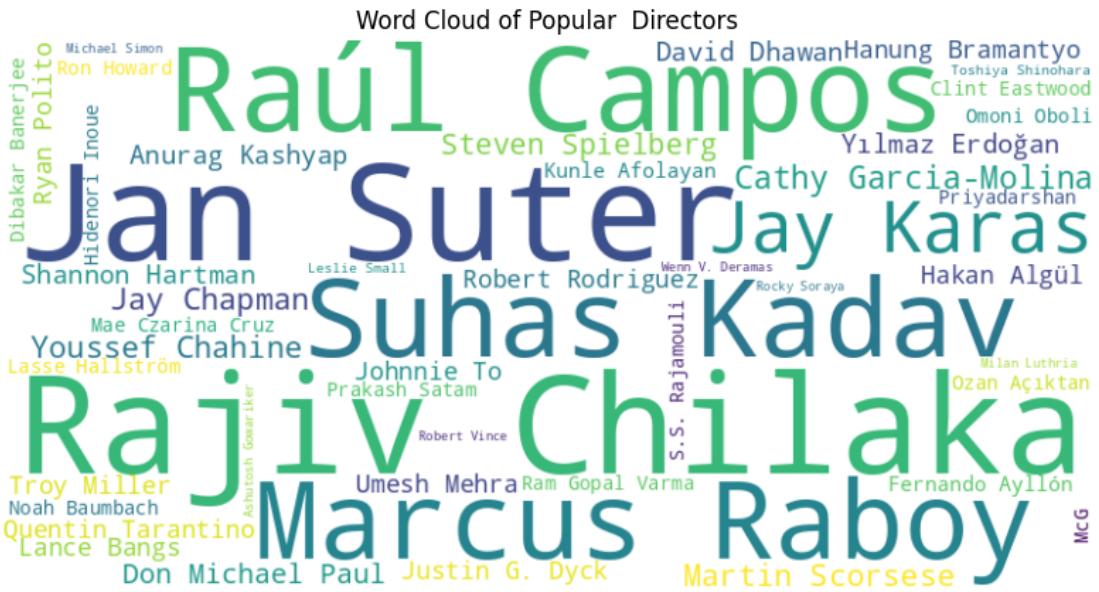
[ ]:
```
                              title                    director  \
0              Dick Johnson Is Dead                Kirsten Johnson
1                     Blood & Water                    No_Director
2                         Ganglands                Julien Leclercq
3             Jailbirds New Orleans                    No_Director
4                      Kota Factory                    No_Director
5                     Midnight Mass                   Mike Flanagan
6  My Little Pony: A New Generation  Robert Cullen, José Luis Ucha
6  My Little Pony: A New Generation  Robert Cullen, José Luis Ucha
7                            Sankofa                   Haile Gerima
8    The Great British Baking Show                Andy Devonshire

  unnested_director
0   Kirsten Johnson
1       No_Director
2   Julien Leclercq
3       No_Director
4       No_Director
5     Mike Flanagan
6     Robert Cullen
6    José Luis Ucha
7      Haile Gerima
8   Andy Devonshire
```

[ ]:
```
filtered_director_df = director_df[director_df['unnested_director'] !=␣
 ↪'No_Director']
filtered_director_df.unnested_director.value_counts().head()
print(filtered_director_df.unnested_director.value_counts().head())
```

```
unnested_director
Rajiv Chilaka    22
Jan Suter        21
Raúl Campos      19
Suhas Kadav      16
Marcus Raboy     16
Name: count, dtype: int64
```

#rahiv chilaka is the most popular actor combining movie and tv shows followed by jan sutur, raul campos.

[ ]:
```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Filter out rows where 'unnested_director' is 'No_Director'
filtered_director_df = director_df[director_df['unnested_director'] !=␣
 ↪'No_Director']

# Get the top 10 most popular unnested directors
top_director_counts = filtered_director_df['unnested_director'].value_counts().
 ↪head(50)

# Create a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate_from_frequencies(top_director_counts)

# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Popular  Directors')
plt.axis('off')
plt.show()
```



#unnesting the listed_in column to get unique genres

[ ]:
```
#most popular genere
unnested_listed_in_df = df[["title", "listed_in"]]
```

```
unnested_listed_in_df["unnested_listed_in"] = unnested_listed_in_df␣
↪["listed_in"].apply(lambda x: str(x).split(", "))
unnested_listed_in_df= unnested_listed_in_df.explode("unnested_listed_in")
unnested_listed_in_df.head(10)
```

```
<ipython-input-124-4c8c67c5e999>: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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
                   title                                          listed_in  \
0   Dick Johnson Is Dead                                       Documentaries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
3  Jailbirds New Orleans                               Docuseries, Reality TV
3  Jailbirds New Orleans                               Docuseries, Reality TV
4           Kota Factory  International TV Shows, Romantic TV Shows, TV …

       unnested_listed_in
0           Documentaries
1  International TV Shows
1               TV Dramas
1            TV Mysteries
2          Crime TV Shows
2  International TV Shows
2   TV Action & Adventure
3              Docuseries
3              Reality TV
4  International TV Shows
```

#most frequent geners

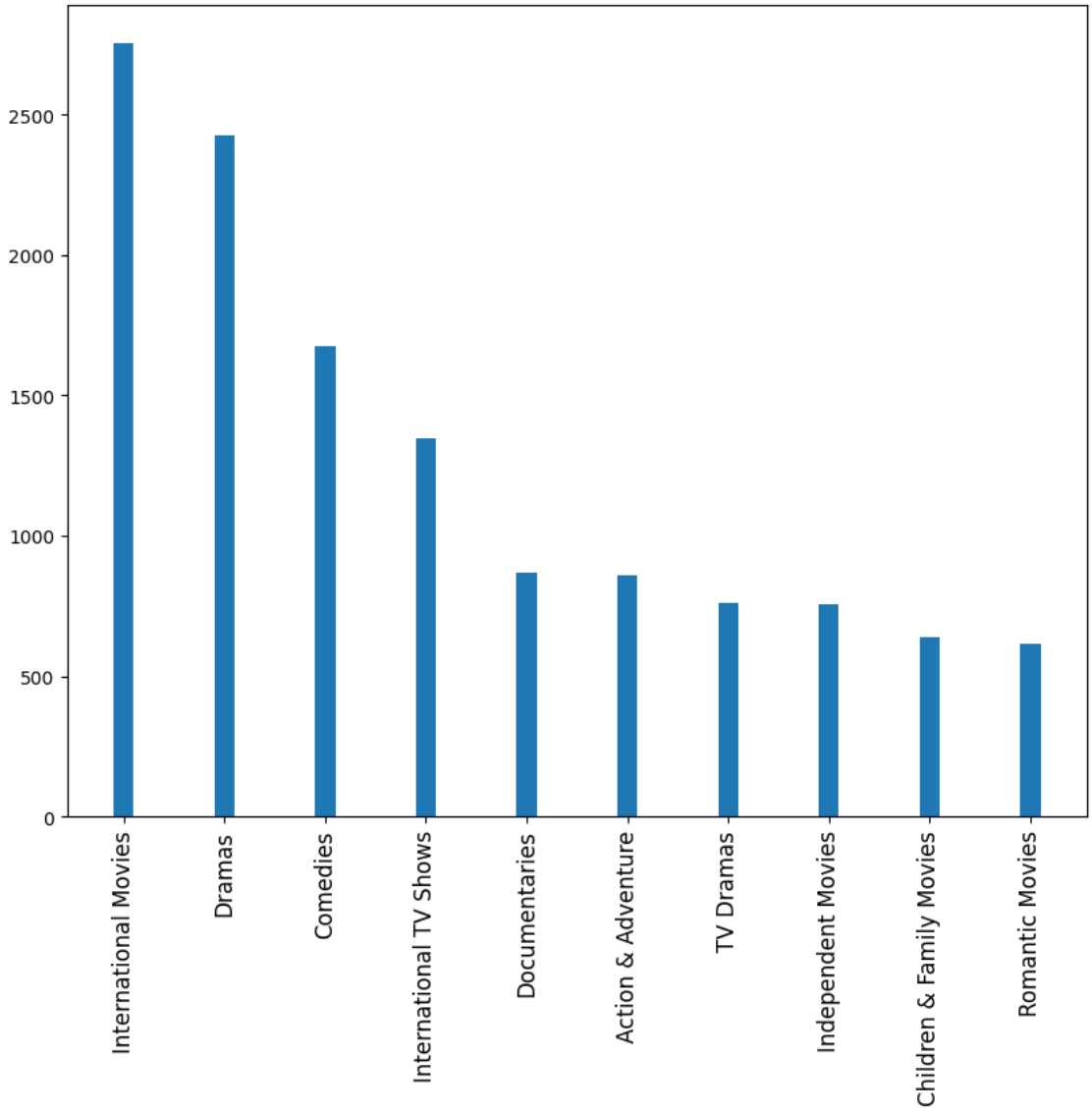
[ ]:
```
listed_in_count_head= unnested_listed_in_df.unnested_listed_in.value_counts().
↪head()
listed_in_count=unnested_listed_in_df.unnested_listed_in.value_counts().head(10)
print(listed_in_count_head)
```

```
unnested_listed_in
International Movies      2752
```

```
Dramas                    2426
Comedies                  1674
International TV Shows    1349
Documentaries              869
Name: count, dtype: int64
```

10 It can be clearly illustrated that genere international movies tops the list across the netflix content combining movies and tv shows.

[ ]:
```
x_bar=listed_in_count.index
y_bar=listed_in_count
#plt.bar(x_bar,y_bar)


plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2)
plt.xticks(rotation=90, fontsize=12)
plt.show()
```



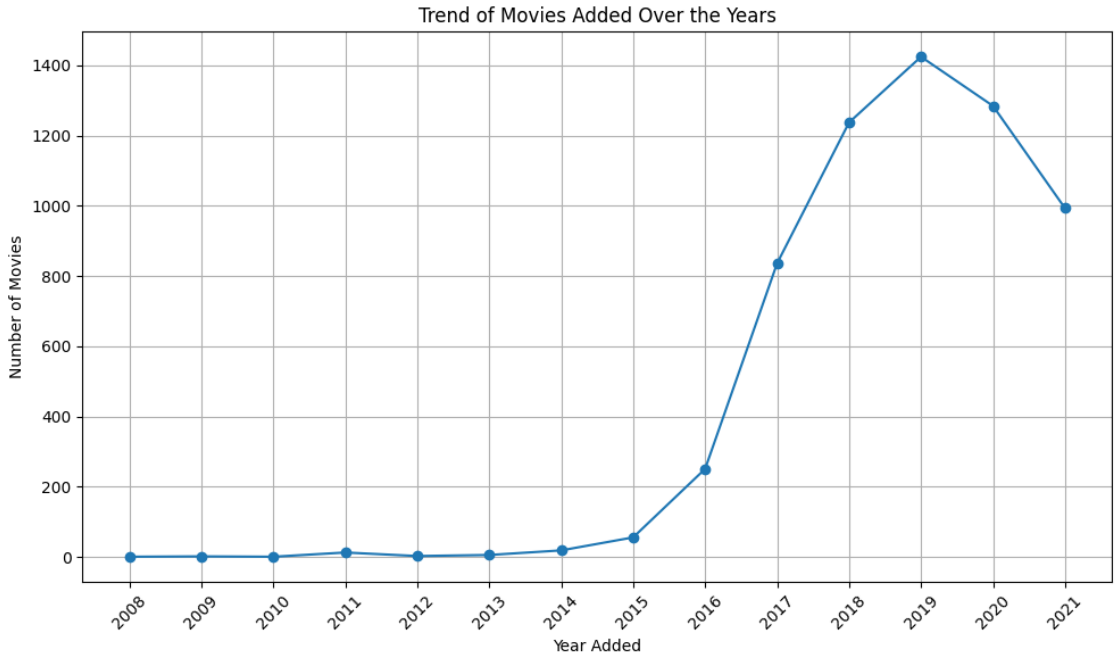
#Plot clearly showcase that international movies has more than 2500 content, followed by drama ,comedies ,International TV Shows which has more than 1000 content each.

#Bi variate annalysis

#Trend of Movies over the years

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame called df with 'year_added' and 'type' columns

# Filter the DataFrame to only include rows where the type is 'Movie'
movies_df = df[df['type'] == 'Movie']
```

```
# Group by 'year_added' and count the number of movies added each year
movies_count_by_year = movies_df.groupby('year_added').size()

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(movies_count_by_year.index, movies_count_by_year.values, marker='o',␣
 ↪linestyle='-')
plt.title('Trend of Movies Added Over the Years')
plt.xlabel('Year Added')
plt.ylabel('Number of Movies')
plt.grid(True)
plt.xticks(movies_count_by_year.index, rotation=45)  # Rotate x-axis labels for␣
 ↪better readability
plt.tight_layout()
plt.show()
```

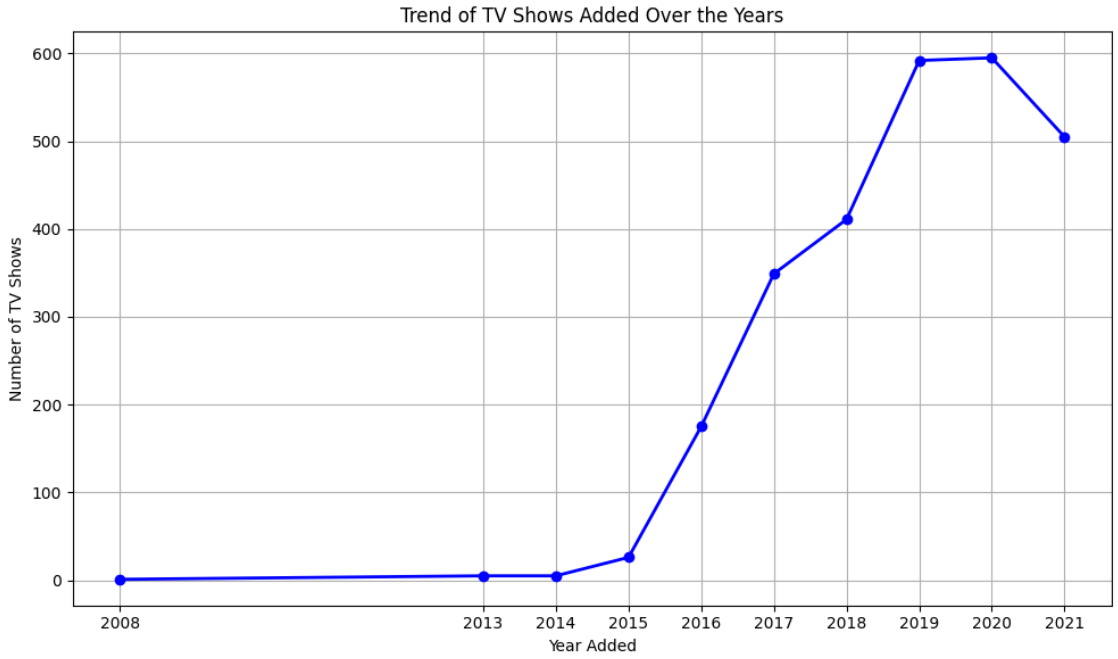


#The plot deplects that netflix has adding content significantly during 2019 and maintained its adding content of 1000 movies till 2021.

#Trend of TVShow over the years

[ ]:
```
# Filter the DataFrame to include only TV Shows
tv_shows_df = df[df['type'] == 'TV Show']

# Group by 'year_added' and count the occurrences
```

```
tv_show_counts = tv_shows_df.groupby('year_added').size()

# Plot the trend
plt.figure(figsize=(10, 6))
tv_show_counts.plot(kind='line', marker='o', color='blue', linewidth=2)
plt.title('Trend of TV Shows Added Over the Years')
plt.xlabel('Year Added')
plt.ylabel('Number of TV Shows')
plt.grid(True)
plt.xticks(tv_show_counts.index)  # Setting x-ticks to be the years
plt.tight_layout()
plt.show()
```

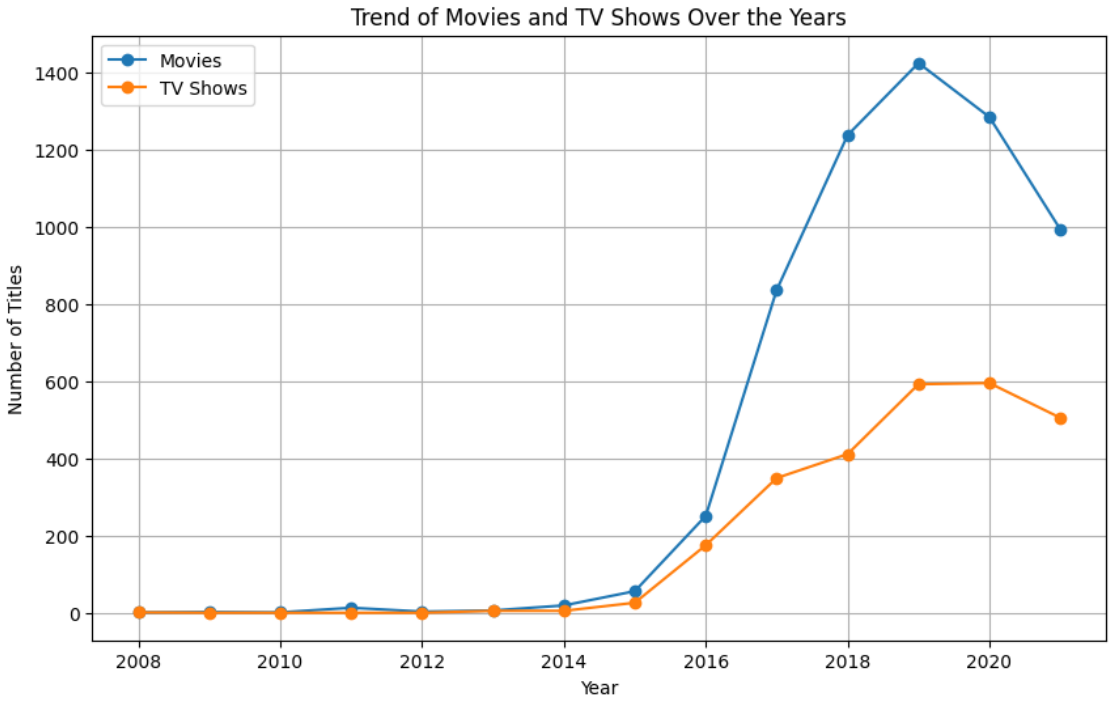


#TV shows in netflix has taken surge from 2016 to 2017 around 200%. ALso it repeated its magic again from 2018 to 2019 by adding more thna 200 % of TV Show content.

#comparasion of Movie and TV Show over the year.

[ ]:
```
# Group the data by year and type, and count the occurrences
grouped_data = df.groupby(['year_added', 'type']).size().unstack(fill_value=0)

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

# Plotting lines for TV shows and movies
plt.plot(grouped_data.index, grouped_data['Movie'], marker='o', label='Movies')
```

```
plt.plot(grouped_data.index, grouped_data['TV Show'], marker='o', label='TV␣
↪Shows')

# Adding labels and title
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.title('Trend of Movies and TV Shows Over the Years')
plt.legend()

# Display the plot
plt.grid(True)
plt.show()
```



11 By seeing the plot. It is clearly indicating that content of movies added in netflix is far more than TV shows over the years.

[ ]:
```
country_df
```

[ ]:

	title	country	unnested_country
0	Dick Johnson Is Dead	United States	United States
1	Blood & Water	South Africa	South Africa

```
2                 Ganglands     No_country       No_country
3     Jailbirds New Orleans     No_country       No_country
4              Kota Factory          India            India
…                       …           …              …
8802                Zodiac  United States    United States
8803            Zombie Dumb     No_country       No_country
8804             Zombieland  United States    United States
8805                   Zoom  United States    United States
8806                Zubaan          India            India

[10828 rows x 3 columns]
```

[ ]:
```
# Merge the unnested data back to the original DataFrame
df_processed_movies = df.drop(['country'], axis=1).join([country_df])
print(df_processed_movies)
print(df_processed_movies.info())
```

```
     show_id     type                 title_x         director  \
0         s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1         s2  TV Show           Blood & Water      No_Director
2         s3  TV Show               Ganglands  Julien Leclercq
3         s4  TV Show  Jailbirds New Orleans      No_Director
4         s5  TV Show            Kota Factory      No_Director
…      …       …                        …              …
8802   s8803    Movie                 Zodiac    David Fincher
8803   s8804  TV Show            Zombie Dumb      No_Director
8804   s8805    Movie             Zombieland  Ruben Fleischer
8805   s8806    Movie                   Zoom     Peter Hewitt
8806   s8807    Movie                 Zubaan      Mozez Singh

                                                  cast date_added  \
0                                                No_cast 2021-09-25
1     Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban… 2021-09-24
2     Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi… 2021-09-24
3                                                No_cast 2021-09-24
4     Mayur More, Jitendra Kumar, Ranjan Raj, Alam K… 2021-09-24
…                                                   …        …
8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J… 2019-11-20
8803                                             No_cast 2019-07-01
8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, … 2019-11-01
8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma… 2020-01-11
8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan… 2019-03-02

      release_year rating   duration  \
0             2020  PG-13     90 min
1             2021  TV-MA  2 Seasons
2             2021  TV-MA   1 Season
```

```
3             2021  TV-MA   1 Season
4             2021  TV-MA  2 Seasons
…              …     …         …
8802          2007      R    158 min
8803          2018  TV-Y7  2 Seasons
8804          2009      R     88 min
8805          2006     PG     88 min
8806          2015  TV-14    111 min

                                          listed_in  \
0                                       Documentaries
1         International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act…
3                               Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV …
…                                                   …
8802                      Cult Movies, Dramas, Thrillers
8803             Kids' TV, Korean TV Shows, TV Comedies
8804                           Comedies, Horror Movies
8805                 Children & Family Movies, Comedies
8806      Dramas, International Movies, Music & Musicals

                                           description  year  year_added  \
0     As her father nears the end of his life, filmm…  2021        2021
1     After crossing paths at a party, a Cape Town t…  2021        2021
2     To protect his family from a powerful drug lor…  2021        2021
3     Feuds, flirtations and toilet talk go down amo…  2021        2021
4     In a city of coaching centers known to train I…  2021        2021
…                                                    …   …         …
8802  A political cartoonist, a crime reporter and a…  2019        2019
8803  While living alone in a spooky town, a young g…  2019        2019
8804  Looking to survive in a world taken over by zo…  2019        2019
8805  Dragged from civilian life, a former superhero…  2020        2020
8806  A scrappy but poor boy worms his way into a ty…  2019        2019

                   title_y        country unnested_country
0      Dick Johnson Is Dead  United States    United States
1             Blood & Water   South Africa     South Africa
2                 Ganglands     No_country       No_country
3     Jailbirds New Orleans     No_country       No_country
4              Kota Factory          India            India
…                       …            …               …
8802                Zodiac  United States    United States
8803            Zombie Dumb     No_country       No_country
8804             Zombieland  United States    United States
8805                   Zoom  United States    United States
8806                Zubaan          India            India
```

```
[10828 rows x 16 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 10828 entries, 0 to 8806
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   show_id           10828 non-null  object
 1   type              10828 non-null  object
 2   title_x           10828 non-null  object
 3   director          10828 non-null  object
 4   cast              10828 non-null  object
 5   date_added        10828 non-null  datetime64[ns]
 6   release_year      10828 non-null  int64
 7   rating            10828 non-null  object
 8   duration          10828 non-null  object
 9   listed_in         10828 non-null  object
 10  description       10828 non-null  object
 11  year              10828 non-null  int32
 12  year_added        10828 non-null  int32
 13  title_y           10828 non-null  object
 14  country           10828 non-null  object
 15  unnested_country  10828 non-null  object
dtypes: datetime64[ns](1), int32(2), int64(1), object(12)
memory usage: 1.3+ MB
None
```

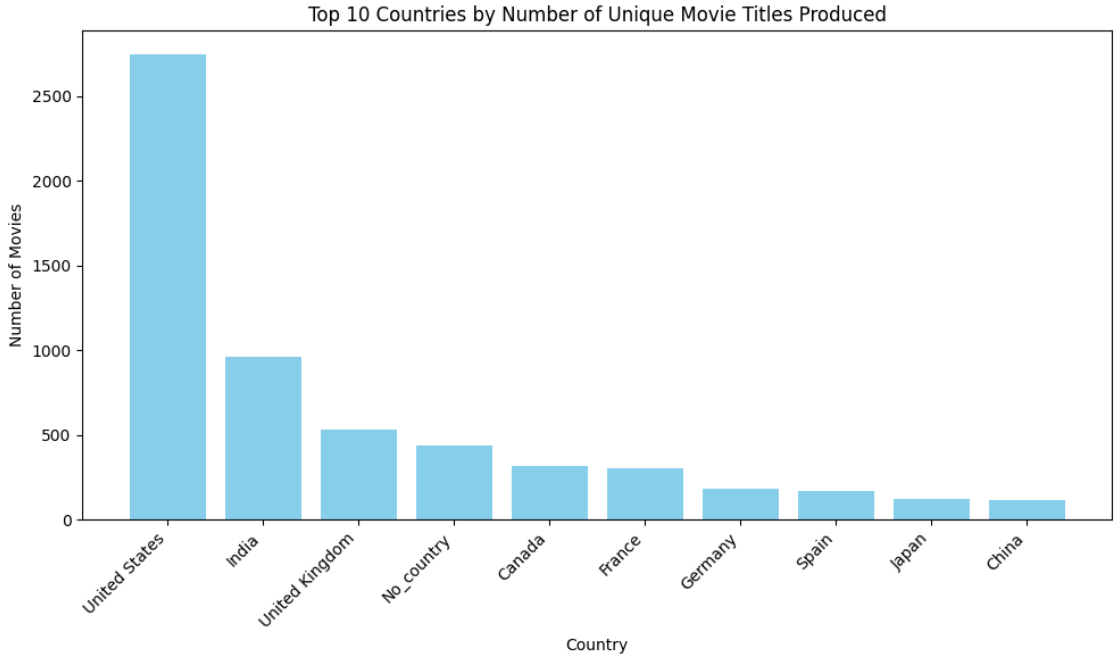
#Movie Content with individual country contribution

[ ]:
```
# Filter rows where the type is "Movie"
movies_df = df_processed_movies[df_processed_movies['type'] == 'Movie']

# Grouping the filtered DataFrame by country and counting the unique titles
movies_per_country = movies_df.groupby('unnested_country')['title_x'].nunique().
 ↪reset_index()

# Renaming the columns for better readability
movies_per_country.columns = ['Country', 'Number of Movies']

# Sorting the DataFrame by the number of movies in descending order
movies_per_country = movies_per_country.sort_values(by='Number of Movies',␣
 ↪ascending=False)

# top 10 countries
top_10_countries = movies_per_country.head(10)

print(top_10_countries)
```

```
          Country  Number of Movies
```

115	United States	2748
43	India	962
113	United Kingdom	532
76	No_country	439
20	Canada	319
34	France	303
36	Germany	182
101	Spain	171
51	Japan	119
23	China	114

[ ]:
```
# Plotting the data
plt.figure(figsize=(10, 6))
plt.bar(top_10_countries['Country'], top_10_countries['Number of Movies'],␣
↪color='skyblue')
plt.xlabel('Country')
plt.ylabel('Number of Movies')
plt.title('Top 10 Countries by Number of Unique Movie Titles Produced')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



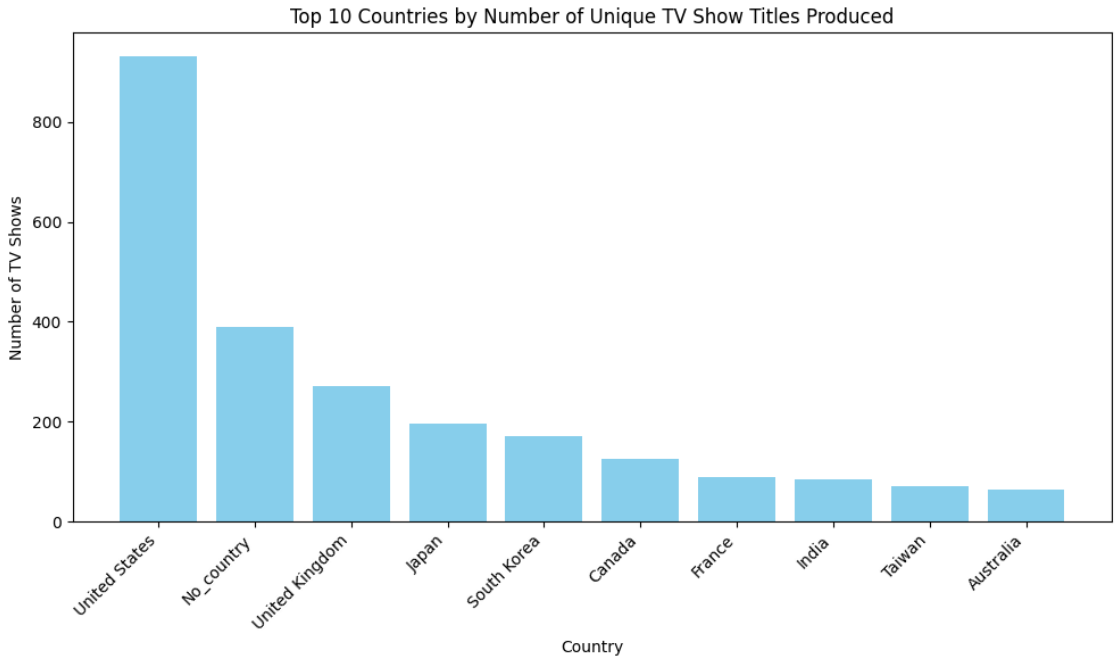
12 Individual countries contribution in movies showcase that United states tops the list with 2748 count , followed by indian with 962.

#Contribution of individual countries in TV SHows .

[ ]:
```
# Filter for TV shows only
tv_shows_df = df_processed_movies[df_processed_movies['type'] == 'TV Show']

# Group by country and count the unique titles
tv_shows_by_country = tv_shows_df.groupby('unnested_country')['title_x'].
 ↪nunique().reset_index()

# Sort by the count of unique titles in descending order
tv_shows_by_country_sorted = tv_shows_by_country.sort_values(by='title_x',␣
 ↪ascending=False)

# top 10 countries
top_10_countries_tvshows = tv_shows_by_country_sorted.head(10)

print(top_10_countries_tvshows)
```

```
   unnested_country  title_x
64    United States      932
42       No_country      390
63   United Kingdom      271
30            Japan      197
53      South Korea      170
8            Canada      126
19           France       90
25            India       84
58           Taiwan       70
2          Australia      64
```

[ ]:
```
import matplotlib.pyplot as plt

# Plotting the data
plt.figure(figsize=(10, 6))
plt.bar(top_10_countries_tvshows['unnested_country'],␣
 ↪top_10_countries_tvshows['title_x'], color='skyblue')
plt.xlabel('Country')
plt.ylabel('Number of TV Shows')
plt.title('Top 10 Countries by Number of Unique TV Show Titles Produced')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



#In TV Shows content united states once tops the list once gain followed by united kingdom , japan and korean has almost similar share.

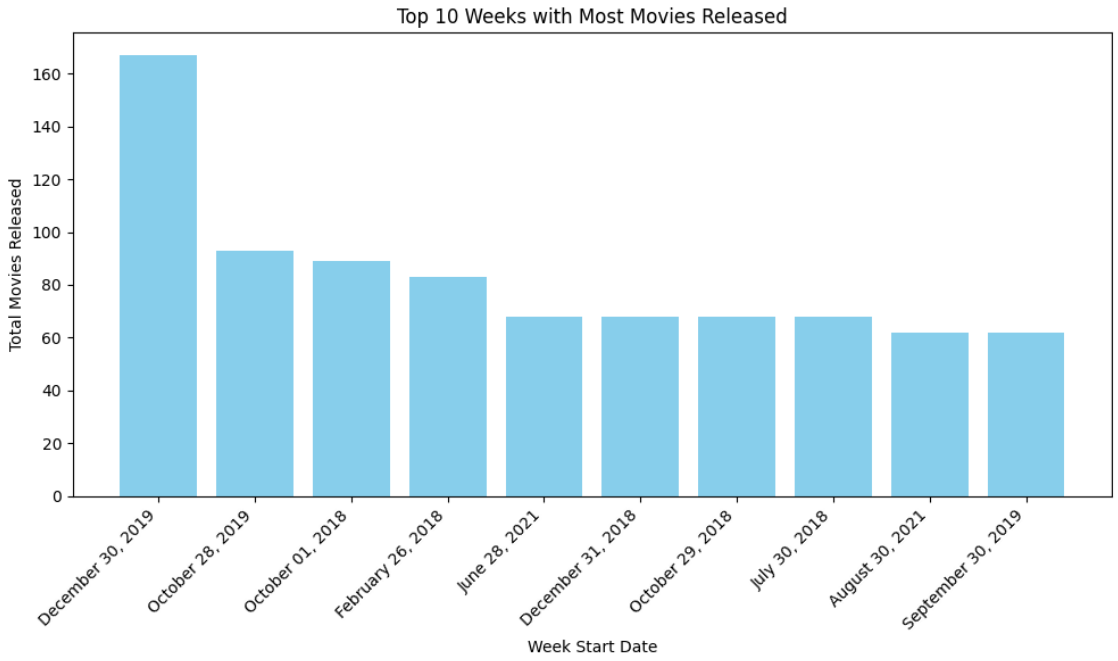
#Best time to launch a movie

[ ]:
```
import pandas as pd

# Convert 'date_added' column to datetime
df['date_added'] = pd.to_datetime(df['date_added'])

# Extract week number and week start date
df['week_number'] = df['date_added'].dt.isocalendar().week
df['week_start_date'] = df['date_added'] - pd.to_timedelta(df['date_added'].dt.
 ↪weekday, unit='d')

# Group by week number and count the total number of movies
movies_per_week = df[df['type'] == 'Movie'].groupby(['week_number',␣
 ↪'week_start_date']).size().reset_index(name='total_movies')

# Sort by total_movies in descending order and get top 5 weeks
top_weeks = movies_per_week.sort_values(by='total_movies', ascending=False).
 ↪head(10)

print("Top 5 weeks with most movies released:")
print("Week\tStart Date\t\tTotal Movies Released")
for index, row in top_weeks.iterrows():
```

```
    print(f"{row['week_number']}\t{row['week_start_date'].strftime('%B %d,␣
 ↪%Y')}\t{row['total_movies']}")
```

```
Top 5 weeks with most movies released:
Week    Start Date              Total Movies Released
1       December 30, 2019       167
44      October 28, 2019        93
40      October 01, 2018        89
9       February 26, 2018       83
26      June 28, 2021   68
1       December 31, 2018       68
44      October 29, 2018        68
31      July 30, 2018   68
35      August 30, 2021 62
40      September 30, 2019      62
```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Plot the top 10 weeks with most movies released
plt.figure(figsize=(10, 6))
plt.bar(top_weeks['week_start_date'].dt.strftime('%B %d, %Y'),␣
 ↪top_weeks['total_movies'], color='skyblue')
plt.title('Top 10 Weeks with Most Movies Released')
plt.xlabel('Week Start Date')
plt.ylabel('Total Movies Released')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



#plot indicates that last week of december has maximum number of movies released. Also, the october month has maximum realeases in 2019 and 2018 as well.

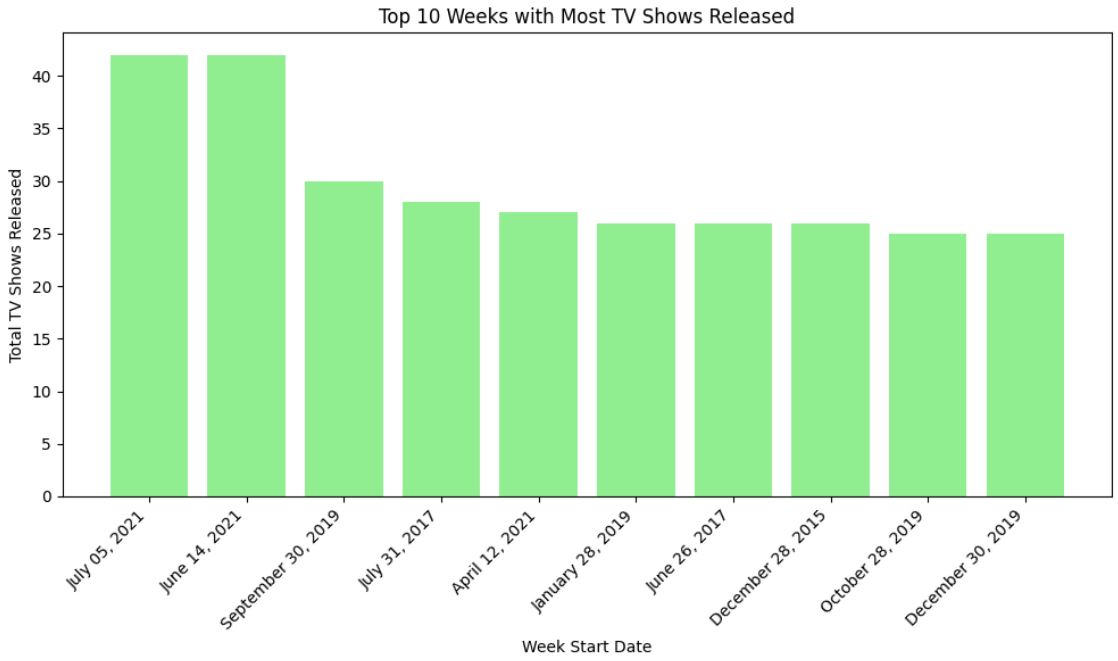
#Best time to launch a TV SHow

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with 'type' and 'date_added' columns

# Convert 'date_added' column to datetime
df['date_added'] = pd.to_datetime(df['date_added'])

# Extract week number and week start date
df['week_number'] = df['date_added'].dt.isocalendar().week
df['week_start_date'] = df['date_added'] - pd.to_timedelta(df['date_added'].dt.
 ↪weekday, unit='d')

# Group by week number and count the total number of TV shows
tv_shows_per_week = df[df['type'] == 'TV Show'].groupby(['week_number',␣
 ↪'week_start_date']).size().reset_index(name='total_tv_shows')

# Sort by total_tv_shows in descending order and get top 10 weeks
top_weeks = tv_shows_per_week.sort_values(by='total_tv_shows', ascending=False).
 ↪head(10)
```

```
# Print top 10 weeks with TV shows
print("Top 10 weeks with TV shows:")
print("Week\tStart Date\t\tTotal TV Shows Released")
for index, row in tv_shows_per_week.head(10).iterrows():
    print(f"{row['week_number']}\t{row['week_start_date'].strftime('%B %d,␣
 ↪%Y')}\t{row['total_tv_shows']}")
```

```
Top 10 weeks with TV shows:
Week    Start Date              Total TV Shows Released
1       January 01, 2018        10
1       December 31, 2018       12
1       December 30, 2019       25
1       January 04, 2021        9
2       January 11, 2016        1
2       January 09, 2017        2
2       January 08, 2018        3
2       January 07, 2019        9
2       January 06, 2020        8
2       January 11, 2021        7
```

[ ]:
```
# Plot the top 10 weeks with most TV shows released
plt.figure(figsize=(10, 6))
plt.bar(top_weeks['week_start_date'].dt.strftime('%B %d, %Y'),␣
 ↪top_weeks['total_tv_shows'], color='lightgreen')
plt.title('Top 10 Weeks with Most TV Shows Released')
plt.xlabel('Week Start Date')
plt.ylabel('Total TV Shows Released')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



13 It is clearly indicates that in 2021, june ,july has more TV shows added .Also, IN 2019, decmber and january has surge of adding TV shows in netflix.

#Most popular genre::

[ ]:
```
merge_df = pd.merge(
    left=cast_df,
    right=country_df,
    on="title"
)
print(merge_df.head())
```

```
                 title                                               cast  \
0  Dick Johnson Is Dead                                            No_cast
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

    unnested_cast        country unnested_country
0         No_cast  United States    United States
1      Ama Qamata   South Africa     South Africa
2     Khosi Ngema   South Africa     South Africa
3   Gail Mabalane   South Africa     South Africa
```

```
4  Thabang Molaba   South Africa     South Africa
```

[ ]:
```
merge_country_df = pd.merge(
    left=merge_df,
    right=country_df,
    on="title"
)
print(merge_country_df.head())
```

```
                 title                                                cast  \
0  Dick Johnson Is Dead                                            No_cast
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

    unnested_cast      country_x unnested_country_x      country_y  \
0         No_cast  United States      United States  United States
1      Ama Qamata   South Africa       South Africa   South Africa
2     Khosi Ngema   South Africa       South Africa   South Africa
3   Gail Mabalane   South Africa       South Africa   South Africa
4  Thabang Molaba   South Africa       South Africa   South Africa

  unnested_country_y
0      United States
1       South Africa
2       South Africa
3       South Africa
4       South Africa
```

#un nesting of listed_in column

[ ]:
```
#most popular genere
unnest_listedin_df = df[["title", "listed_in"]]
unnest_listedin_df["unnested_listed_in"] = unnest_listedin_df["listed_in"].
 ↪apply(lambda x: str(x).split(", "))
unnest_listedin_df= unnest_listedin_df.explode("unnested_listed_in")
unnest_listedin_df.head(10)
```

```
<ipython-input-198-31cfa377c4e1>: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/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
                  title                                          listed_in  \
0   Dick Johnson Is Dead                                       Documentaries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
1          Blood & Water    International TV Shows, TV Dramas, TV Mysteries
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
2              Ganglands  Crime TV Shows, International TV Shows, TV Act…
3  Jailbirds New Orleans                               Docuseries, Reality TV
3  Jailbirds New Orleans                               Docuseries, Reality TV
4           Kota Factory  International TV Shows, Romantic TV Shows, TV …

       unnested_listed_in
0           Documentaries
1   International TV Shows
1                TV Dramas
1             TV Mysteries
2           Crime TV Shows
2   International TV Shows
2    TV Action & Adventure
3              Docuseries
3              Reality TV
4   International TV Shows
```

[ ]:
```
#unnested_listed_in_df

merge_listedin_df = pd.merge(
    left=merge_country_df,
    right=unnest_listedin_df,
    on="title"
)
print(merge_listedin_df.head())
```

```
                  title                                                cast  \
0  Dick Johnson Is Dead                                             No_cast
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

  unnested_cast      country_x unnested_country_x      country_y  \
0       No_cast  United States      United States  United States
1    Ama Qamata   South Africa       South Africa   South Africa
2    Ama Qamata   South Africa       South Africa   South Africa
3    Ama Qamata   South Africa       South Africa   South Africa
4   Khosi Ngema   South Africa       South Africa   South Africa
```

```
  unnested_country_y                                        listed_in  \
0      United States                                       Documentaries
1       South Africa  International TV Shows, TV Dramas, TV Mysteries
2       South Africa  International TV Shows, TV Dramas, TV Mysteries
3       South Africa  International TV Shows, TV Dramas, TV Mysteries
4       South Africa  International TV Shows, TV Dramas, TV Mysteries

       unnested_listed_in
0           Documentaries
1  International TV Shows
2              TV Dramas
3           TV Mysteries
4  International TV Shows
```

[ ]:
```
merge_director_df = pd.merge(
    left=merge_listedin_df,
    right=director_df,
    on="title"
)
print(merge_director_df.head())
```

```
                 title                                                cast  \
0  Dick Johnson Is Dead                                             No_cast
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

  unnested_cast      country_x unnested_country_x      country_y  \
0       No_cast  United States      United States  United States
1    Ama Qamata   South Africa       South Africa   South Africa
2    Ama Qamata   South Africa       South Africa   South Africa
3    Ama Qamata   South Africa       South Africa   South Africa
4   Khosi Ngema   South Africa       South Africa   South Africa

  unnested_country_y                                        listed_in  \
0      United States                                       Documentaries
1       South Africa  International TV Shows, TV Dramas, TV Mysteries
2       South Africa  International TV Shows, TV Dramas, TV Mysteries
3       South Africa  International TV Shows, TV Dramas, TV Mysteries
4       South Africa  International TV Shows, TV Dramas, TV Mysteries

       unnested_listed_in         director unnested_director
0           Documentaries  Kirsten Johnson   Kirsten Johnson
1  International TV Shows      No_Director       No_Director
2              TV Dramas      No_Director       No_Director
3           TV Mysteries      No_Director       No_Director
4  International TV Shows      No_Director       No_Director
```

[ ]:
```
print(merge_director_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326773 entries, 0 to 326772
Data columns (total 11 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   title               326773 non-null  object
 1   cast                326773 non-null  object
 2   unnested_cast       326773 non-null  object
 3   country_x           326773 non-null  object
 4   unnested_country_x  326773 non-null  object
 5   country_y           326773 non-null  object
 6   unnested_country_y  326773 non-null  object
 7   listed_in           326773 non-null  object
 8   unnested_listed_in  326773 non-null  object
 9   director            326773 non-null  object
 10  unnested_director   326773 non-null  object
dtypes: object(11)
memory usage: 27.4+ MB
None
```

[ ]:
```
print(merge_director_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326773 entries, 0 to 326772
Data columns (total 6 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   title               326773 non-null  object
 1   unnested_cast       326773 non-null  object
 2   unnested_country_y  326773 non-null  object
 3   unnested_listed_in  326773 non-null  object
 4   director            326773 non-null  object
 5   unnested_director   326773 non-null  object
dtypes: object(6)
memory usage: 15.0+ MB
None
```

[ ]:
```
# Group by country and genre, then count unique titles
country_genre_counts = merge_director_df.groupby(['unnested_country_y',␣
 ↪'unnested_listed_in'])['title'].nunique()

# Get the top 10 countries
top_10_countries = country_genre_counts.groupby('unnested_country_y').sum().
 ↪nlargest(10).index
```

```
# Filter the counts for the top 10 countries
top_10_country_genre_counts = country_genre_counts.loc[top_10_countries]

# Find the most popular genre for each top 10 country
most_popular_genre_by_country = top_10_country_genre_counts.
 ↪groupby('unnested_country_y').idxmax().reset_index()


print(most_popular_genre_by_country)
```

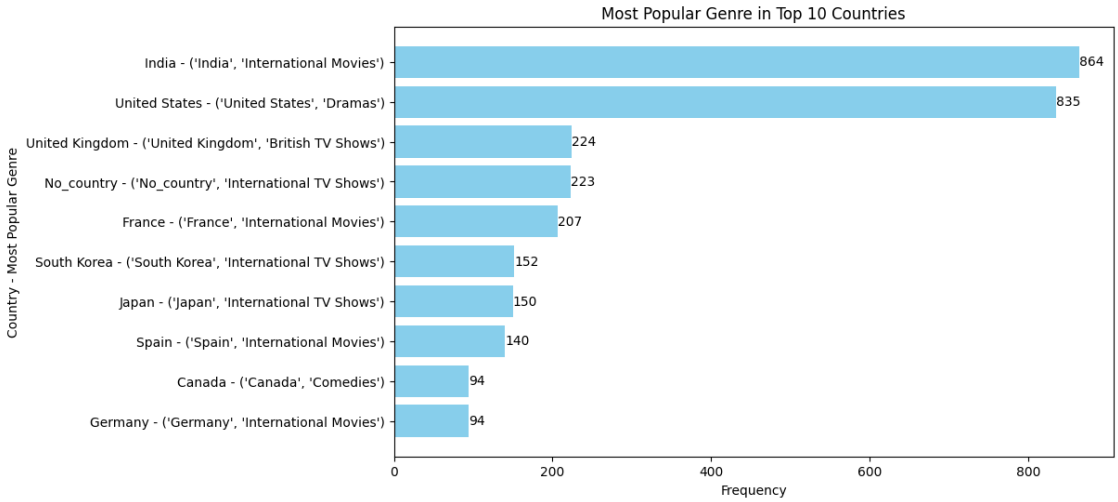
```
  unnested_country_y                                  title
0             Canada                    (Canada, Comedies)
1             France      (France, International Movies)
2            Germany     (Germany, International Movies)
3              India       (India, International Movies)
4              Japan     (Japan, International TV Shows)
5         No_country  (No_country, International TV Shows)
6        South Korea  (South Korea, International TV Shows)
7              Spain       (Spain, International Movies)
8     United Kingdom  (United Kingdom, British TV Shows)
9      United States             (United States, Dramas)
```

[ ]:
```
import pandas as pd


# Group by country and genre, then count unique titles
country_genre_counts = merge_director_df.groupby(['unnested_country_y',␣
 ↪'unnested_listed_in'])['title'].nunique()

# Get the top 10 countries
top_10_countries = country_genre_counts.groupby('unnested_country_y').sum().
 ↪nlargest(10).index

# Filter the counts for the top 10 countries
top_10_country_genre_counts = country_genre_counts.loc[top_10_countries]

# Find the most popular genre for each top 10 country
most_popular_genre_by_country = top_10_country_genre_counts.
 ↪groupby('unnested_country_y').idxmax().reset_index()

# Rename the column
most_popular_genre_by_country.rename(columns={'unnested_listed_in': 'Most␣
 ↪Popular Genre'}, inplace=True)

# Sort by frequency of the most popular genre in descending order
most_popular_genre_by_country['Frequency'] = top_10_country_genre_counts.
 ↪groupby('unnested_country_y').max().reset_index()['title']
```

```
most_popular_genre_by_country.sort_values(by='Frequency', ascending=False,␣
 ↪inplace=True)

# Reset index
most_popular_genre_by_country.reset_index(drop=True, inplace=True)

# Print the results
print(most_popular_genre_by_country)
```

	unnested_country_y	title	Frequency
0	India	(India, International Movies)	864
1	United States	(United States, Dramas)	835
2	United Kingdom	(United Kingdom, British TV Shows)	224
3	No_country	(No_country, International TV Shows)	223
4	France	(France, International Movies)	207
5	South Korea	(South Korea, International TV Shows)	152
6	Japan	(Japan, International TV Shows)	150
7	Spain	(Spain, International Movies)	140
8	Canada	(Canada, Comedies)	94
9	Germany	(Germany, International Movies)	94

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming most_popular_genre_by_country DataFrame contains the results from␣
 ↪the previous code

# Plotting
plt.figure(figsize=(10, 6))
for i, (country, genre, freq) in most_popular_genre_by_country.iterrows():
    plt.barh(f'{country} - {genre}', freq, color='skyblue')
    plt.text(freq, i, str(freq), ha='left', va='center')  # Add frequency value␣
 ↪next to each bar

plt.xlabel('Frequency')
plt.ylabel('Country - Most Popular Genre')
plt.title('Most Popular Genre in Top 10 Countries')
plt.gca().invert_yaxis()  # Invert y-axis to have the highest frequency at the␣
 ↪top
plt.show()
```



14 Graph showcase that India most popular content is International movies. On the similar lines france, south korea,japan, spain, Germany inclined to Internaltional movies only.

15 Most popular actor in respective country.

[ ]:
```
import pandas as pd



# Filter out rows where unnested_cast is 'No_cast'
filtered_df = merge_director_df[merge_director_df['unnested_cast'] != 'No_cast']

# Group by country and count the occurrences of each actor
actor_counts = filtered_df.groupby('unnested_country_y')['unnested_cast'].
 ↪value_counts()

# Get the top 10 countries based on total count of movies/shows
top_countries = filtered_df['unnested_country_y'].value_counts().head(10).index

most_popular_actors = {}

for country in top_countries:

    country_actor_counts = actor_counts[country]

    most_popular_actor = country_actor_counts.groupby('unnested_cast').sum().
 ↪idxmax()
```

```
    most_popular_actors[country] = most_popular_actor


for country, actor in most_popular_actors.items():
    print(f"The most popular actor in {country} is: {actor}")
```

```
The most popular actor in United States is: Alfred Molina
The most popular actor in United Kingdom is: David Attenborough
The most popular actor in India is: Anupam Kher
The most popular actor in France is: Liam Neeson
The most popular actor in Canada is: Liam Neeson
The most popular actor in No_country is: Julie Tejwani
The most popular actor in Germany is: Ben Whishaw
The most popular actor in Japan is: Luci Christian
The most popular actor in Spain is: David Attenborough
The most popular actor in China is: Donnie Yen
```

#The most popular actor in United States is: Alfred Molina #The most popular actor in United Kingdom is: David Attenborough #The most popular actor in India is: Anupam Kher #The most popular actor in France is: Liam Neeson #The most popular actor in Canada is: Liam Neeson #The most popular actor in No_country is: Julie Tejwani #The most popular actor in Germany is: Ben Whishaw #The most popular actor in Japan is: Luci Christian #The most popular actor in Spain is: David Attenborough #The most popular actor in China is: Donnie Yen

#Most popular director

[ ]:
```
import pandas as pd

merge_director_df = merge_director_df[merge_director_df['unnested_director'] !=␣
 ↪'No_Director']

director_counts = merge_director_df.
 ↪groupby('unnested_country_y')['unnested_director'].nunique()


top_10_countries = director_counts.sort_values(ascending=False).head(10)


most_popular_directors = pd.DataFrame(columns=['Country', 'Most Popular␣
 ↪Director'])

for country in top_10_countries.index:

    country_data = merge_director_df[merge_director_df['unnested_country_y'] ==␣
 ↪country]

    director_counts = country_data['unnested_director'].value_counts()
```

```
    most_popular_director = director_counts.idxmax()

    most_popular_directors = pd.concat([most_popular_directors, pd.
  ↪DataFrame({'Country': [country], 'Most Popular Director':␣
  ↪[most_popular_director]})])

print(most_popular_directors)
```

```
          Country Most Popular Director
0   United States       Martin Scorsese
0           India          David Dhawan
0  United Kingdom        Lars von Trier
0      No_country        Hidenori Inoue
0          France        Lars von Trier
0          Canada          Raja Gosnell
0         Germany        Lars von Trier
0           Spain       Federico Veiroj
0           Japan     Toshiya Shinohara
0           China           Wilson Yip
```

#United States most popular director isMartin Scorsese #India has David Dhawan #United Kingdom has Lars von Trier in most popular director list

[ ]:
```
#loading the dataset
df_type = pd.read_csv("https://gist.github.com/singhsidhukuldeep/
  ↪564f271315abb6bc22647e81e6bf4762/raw/
  ↪66fb67a8bb014df6b7f924aad0a91aa662bc7fc2/netflix_titles.csv")
print(df_type.shape)
df_type.head()
```

```
(8807, 12)
```

[ ]:
```
  show_id     type                  title         director  \
0      s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1      s2  TV Show          Blood & Water              NaN
2      s3  TV Show              Ganglands  Julien Leclercq
3      s4  TV Show  Jailbirds New Orleans              NaN
4      s5  TV Show           Kota Factory              NaN

                                                cast        country  \
0                                                NaN  United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…   South Africa
2  Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…            NaN
3                                                NaN            NaN
4  Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…          India

           date_added  release_year rating   duration  \
```

```
0  September 25, 2021          2020  PG-13     90 min
1  September 24, 2021          2021  TV-MA  2 Seasons
2  September 24, 2021          2021  TV-MA   1 Season
3  September 24, 2021          2021  TV-MA   1 Season
4  September 24, 2021          2021  TV-MA  2 Seasons

                                          listed_in  \
0                                     Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act…
3                            Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV …

                                        description
0  As her father nears the end of his life, filmm…
1  After crossing paths at a party, a Cape Town t…
2  To protect his family from a powerful drug lor…
3  Feuds, flirtations and toilet talk go down amo…
4  In a city of coaching centers known to train I…
```

[ ]:

```
casttype_df = df_type[["title", "cast"]]
casttype_df["unnested_cast"] = casttype_df ["cast"].apply(lambda x: str(x).
↪split(", "))
casttype_df = casttype_df.explode("unnested_cast")
casttype_df.head(10)
```

```
<ipython-input-8-cfcd3369f0ae>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
  casttype_df["unnested_cast"] = casttype_df ["cast"].apply(lambda x:
str(x).split(", "))
```

[ ]:

```
                  title                                               cast  \
0  Dick Johnson Is Dead                                                NaN
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
```

```
       unnested_cast
0                nan
1          Ama Qamata
1         Khosi Ngema
1       Gail Mabalane
1      Thabang Molaba
1    Dillon Windvogel
1     Natasha Thahane
1         Arno Greeff
1   Xolile Tshabalala
1     Getmore Sithole
```

[ ]:
```
typecast_df = df_type[["title", "type"]]
typecast_df["unnested_type"] = typecast_df ["type"].apply(lambda x: str(x).
 ↪split(", "))
typecast_df = typecast_df.explode("unnested_type")
typecast_df.head(10)
```

```
<ipython-input-9-fb0b89a6a2f6>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
  typecast_df["unnested_type"] = typecast_df ["type"].apply(lambda x:
str(x).split(", "))
```

[ ]:
```
                            title     type unnested_type
0             Dick Johnson Is Dead    Movie         Movie
1                    Blood & Water  TV Show       TV Show
2                        Ganglands  TV Show       TV Show
3            Jailbirds New Orleans  TV Show       TV Show
4                     Kota Factory  TV Show       TV Show
5                    Midnight Mass  TV Show       TV Show
6  My Little Pony: A New Generation    Movie         Movie
7                           Sankofa    Movie         Movie
8    The Great British Baking Show  TV Show       TV Show
9                     The Starling    Movie         Movie
```

[ ]:
```
merge_typecast_df = pd.merge(
    left=casttype_df,
    right=typecast_df,
    on="title"
)
print(merge_typecast_df.head())
```

```
                 title                                               cast  \
```

```
0  Dick Johnson Is Dead                                                  NaN
1         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4         Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

    unnested_cast     type unnested_type
0             nan    Movie         Movie
1      Ama Qamata  TV Show       TV Show
2     Khosi Ngema  TV Show       TV Show
3   Gail Mabalane  TV Show       TV Show
4  Thabang Molaba  TV Show       TV Show
```

[ ]:
```
merge_typecast_df = merge_typecast_df[merge_typecast_df['unnested_cast'] !=␣
↪'nan']

movies_df = merge_typecast_df[merge_typecast_df['unnested_type'] == 'Movie']

cast_movie_counts = movies_df.groupby('unnested_cast')['title'].nunique()

cast_movie_counts_sorted = cast_movie_counts.sort_values(ascending=False)


top_10_Actors= cast_movie_counts_sorted.head(10)


print("Top 10 Actors who have appeared in the most movies:")
print(top_10_Actors)
```

```
Top 10 Actors who have appeared in the most movies:
unnested_cast
Anupam Kher          42
Shah Rukh Khan       35
Naseeruddin Shah     32
Akshay Kumar         30
Om Puri              30
Paresh Rawal         28
Amitabh Bachchan     28
Julie Tejwani        28
Boman Irani          27
Rupa Bhimani         27
Name: title, dtype: int64
```

#Top three actors across globe are Anupam Kher , Shah Rukh Khan, Naseeruddin Shah.

#Actors appeared in most TV shows.

[ ]:
```
merge_typecast_df = merge_typecast_df[merge_typecast_df['unnested_cast'] !=␣
↪'No_cast']


movies_df = merge_typecast_df[merge_typecast_df['unnested_type'] == 'TV Show']

 #Group by director and count unique titles
cast_movie_counts = movies_df.groupby('unnested_cast')['title'].nunique()


cast_movie_counts_sorted = cast_movie_counts.sort_values(ascending=False)

# top 10 directors
top_10_Actors= cast_movie_counts_sorted.head(10)

print("Top 10 Actors who have appeared in the most TV shows:")
print(top_10_Actors)
```

```
Top 10 Actors who have appeared in the most TV shows:
unnested_cast
Takahiro Sakurai      25
Yuki Kaji             19
Junichi Suwabe        17
Daisuke Ono           17
Ai Kayano             17
Yuichi Nakamura       16
Yoshimasa Hosoya      15
Jun Fukuyama          15
David Attenborough    14
Kana Hanazawa         13
Name: title, dtype: int64
```

#Top three popular actors in TV Shows across globe are Takahiro, Yuki, junichi

[ ]:
```
countrynew_df = df_type[["title", "country"]]
countrynew_df["unnested_countrynew"] = countrynew_df ["country"].apply(lambda x:
↪ str(x).split(", "))
countrynew_df = countrynew_df.explode("unnested_countrynew")
countrynew_df.head(10)
```

```
<ipython-input-17-4979b66dbfc8>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
  countrynew_df["unnested_countrynew"] = countrynew_df ["country"].apply(lambda
x: str(x).split(", "))
```

[ ]:
```
                           title  \
0             Dick Johnson Is Dead
1                    Blood & Water
2                        Ganglands
3            Jailbirds New Orleans
4                     Kota Factory
5                    Midnight Mass
6  My Little Pony: A New Generation
7                          Sankofa
7                          Sankofa
7                          Sankofa

                                        country unnested_countrynew
0                                 United States       United States
1                                  South Africa        South Africa
2                                           NaN                 nan
3                                           NaN                 nan
4                                         India               India
5                                           NaN                 nan
6                                           NaN                 nan
7  United States, Ghana, Burkina Faso, United Kin…       United States
7  United States, Ghana, Burkina Faso, United Kin…               Ghana
7  United States, Ghana, Burkina Faso, United Kin…        Burkina Faso
```

[ ]:
```
merge_typecastcountry_df = pd.merge(
    left=merge_typecast_df,
    right=countrynew_df,
    on="title"
)
print(merge_typecastcountry_df.head())
```

```
          title                                                 cast  \
0  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

      unnested_cast     type unnested_type       country unnested_countrynew
0        Ama Qamata  TV Show       TV Show  South Africa        South Africa
1       Khosi Ngema  TV Show       TV Show  South Africa        South Africa
2     Gail Mabalane  TV Show       TV Show  South Africa        South Africa
3    Thabang Molaba  TV Show       TV Show  South Africa        South Africa
4  Dillon Windvogel  TV Show       TV Show  South Africa        South Africa
```

[ ]:
```
#most popular directors
directornew_df = df_type[["title", "director"]]
```

```
directornew_df["unnested_director"] = directornew_df ["director"].apply(lambda
 ↪x: str(x).split(", "))
directornew_df= directornew_df.explode("unnested_director")
directornew_df.head(10)
```

```
<ipython-input-20-65db7ea081f6>: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/user_guide/indexing.html#returning-a-view-versus-a-copy
  directornew_df["unnested_director"] = directornew_df ["director"].apply(lambda
x: str(x).split(", "))
```

[ ]:

	title	director	\
0	Dick Johnson Is Dead	Kirsten Johnson	
1	Blood & Water	NaN	
2	Ganglands	Julien Leclercq	
3	Jailbirds New Orleans	NaN	
4	Kota Factory	NaN	
5	Midnight Mass	Mike Flanagan	
6	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	
6	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha	
7	Sankofa	Haile Gerima	
8	The Great British Baking Show	Andy Devonshire	

	unnested_director
0	Kirsten Johnson
1	nan
2	Julien Leclercq
3	nan
4	nan
5	Mike Flanagan
6	Robert Cullen
6	José Luis Ucha
7	Haile Gerima
8	Andy Devonshire

[ ]:
```
merge_typecastcountrydirector_df = pd.merge(
    left=merge_typecastcountry_df,
    right=directornew_df,
    on="title"
)
print(merge_typecastcountrydirector_df .head())
```

```
           title                                                cast  \
0  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
```

```
1  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

      unnested_cast     type unnested_type       country unnested_countrynew  \
0        Ama Qamata  TV Show       TV Show  South Africa        South Africa
1       Khosi Ngema  TV Show       TV Show  South Africa        South Africa
2     Gail Mabalane  TV Show       TV Show  South Africa        South Africa
3    Thabang Molaba  TV Show       TV Show  South Africa        South Africa
4  Dillon Windvogel  TV Show       TV Show  South Africa        South Africa

  director unnested_director
0      NaN               nan
1      NaN               nan
2      NaN               nan
3      NaN               nan
4      NaN               nan
```

#Most popular actor in movies in Top 10 countries

[ ]:
```
# Filter the DataFrame for rows where unnested_type is "Movie"
movie_df =␣
 ↪merge_typecastcountrydirector_df[merge_typecastcountrydirector_df['unnested_type']␣
 ↪== 'Movie']

# Get the top 10 countries by frequency
top_10_countries = movie_df['unnested_countrynew'].value_counts().head(10).
 ↪index.tolist()

# Filter the DataFrame for only the top 10 countries
top_10_df = movie_df[movie_df['unnested_countrynew'].isin(top_10_countries)]

# Group by country and find the most frequent actor for each
most_popular_actors = top_10_df.groupby('unnested_countrynew')['unnested_cast'].
 ↪agg(lambda x: x.value_counts().index[0])
print("most popular actor in movie in every country")
print(most_popular_actors)
```

```
most popular actor in movie in every country
unnested_countrynew
Canada            John Paul Tremblay
China                    Donnie Yen
France                   Liam Neeson
Germany                 Hugo Weaving
India                    Anupam Kher
Japan                      Yuki Kaji
Spain                    Mario Casas
```

```
United Kingdom           John Cleese
United States           James Franco
nan                    Julie Tejwani
Name: unnested_cast, dtype: object
```

#most popular actor in movie in every country are as follows

#Canada most popular actor in movies John Paul Tremblay, #China most popular actor in movies Donnie Yen #France most popular actor in movies Liam Neeson #Germany most popular actor in movies Hugo Weaving, #India most popular actor in movies Anupam Kher, #Japan most popular actor in movies Yuki Kaji, #Spain most popular actor in movies Mario Casas, #United Kingdom most popular actor in movies John Cleese, #United States has James Franco.

#Most popular actor in TV Shows in in top 10 Countries.

[ ]:
```
# Filter the DataFrame for rows where unnested_type is "Movie"
movie_df =␣
 ↪merge_typecastcountrydirector_df[merge_typecastcountrydirector_df['unnested_type']␣
 ↪== 'TV Show']

# Get the top 10 countries by frequency
top_10_countries = movie_df['unnested_countrynew'].value_counts().head(10).
 ↪index.tolist()

# Filter the DataFrame for only the top 10 countries
top_10_df = movie_df[movie_df['unnested_countrynew'].isin(top_10_countries)]

# Group by country and find the most frequent actor for each
most_popular_actors = top_10_df.groupby('unnested_countrynew')['unnested_cast'].
 ↪agg(lambda x: x.value_counts().index[0])
print("most popular actor in TV show in every country")
print(most_popular_actors)
```

```
most popular actor in TV show in every country
unnested_countrynew
Canada            John Paul Tremblay
France                  Tahar Rahim
India                   Prakash Raj
Japan                Takahiro Sakurai
Mexico               Sebastián Rulli
South Korea               Bae Doona
Spain                  Carlos Cuevas
United Kingdom    David Attenborough
United States           Grey Griffin
nan                    Frederick Lee
Name: unnested_cast, dtype: object
```

#Canada most popular actor in TV Show John Paul Tremblay #France most popular actor in TV Show Tahar Rahim #India most popular actor in TV Show Prakash Raj #Japan most popular actor in TV Show Takahiro Sakurai #Mexico most popular actor in TV Show Sebastián Rulli

#South Korea most popular actor in TV Show Bae Doona #Spain most popular actor in TV Show Carlos Cuevas #United Kingdom most popular actor in TV Show David Attenborough #United States most popular actor in TV Show Grey Griffin

[ ]:
```
listed_in_df = df_type[["title", "listed_in"]]
listed_in_df["unlisted_in"] = listed_in_df ["listed_in"].apply(lambda x: str(x).
 ↪split(", "))
listed_in_df = listed_in_df.explode("unlisted_in")
listed_in_df.head(10)
```

```
<ipython-input-43-b5ee66156239>:2: 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/user_guide/indexing.html#returning-a-view-versus-a-copy
  listed_in_df["unlisted_in"] = listed_in_df ["listed_in"].apply(lambda x:
str(x).split(", "))
```

[ ]:

	title	listed_in	unlisted_in
0	Dick Johnson Is Dead	Documentaries	Documentaries
1	Blood & Water	International TV Shows, TV Dramas, TV Mysteries	International TV Shows
1	Blood & Water	International TV Shows, TV Dramas, TV Mysteries	TV Dramas
1	Blood & Water	International TV Shows, TV Dramas, TV Mysteries	TV Mysteries
2	Ganglands	Crime TV Shows, International TV Shows, TV Act…	Crime TV Shows
2	Ganglands	Crime TV Shows, International TV Shows, TV Act…	International TV Shows
2	Ganglands	Crime TV Shows, International TV Shows, TV Act…	TV Action & Adventure
3	Jailbirds New Orleans	Docuseries, Reality TV	Docuseries
3	Jailbirds New Orleans	Docuseries, Reality TV	Reality TV
4	Kota Factory	International TV Shows, Romantic TV Shows, TV …	International TV Shows

[ ]:
```
merge_typecastcountrydirectorlisted_df = pd.merge(
    left=merge_typecastcountry_df,
    right=listed_in_df,
    on="title"
)
```

```
print(merge_typecastcountrydirectorlisted_df .head())
```

```
           title                                                cast  \
0  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
1  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
2  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
3  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…
4  Blood & Water  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…

  unnested_cast     type unnested_type       country unnested_countrynew  \
0    Ama Qamata  TV Show       TV Show  South Africa        South Africa
1    Ama Qamata  TV Show       TV Show  South Africa        South Africa
2    Ama Qamata  TV Show       TV Show  South Africa        South Africa
3   Khosi Ngema  TV Show       TV Show  South Africa        South Africa
4   Khosi Ngema  TV Show       TV Show  South Africa        South Africa

                                      listed_in             unlisted_in
0  International TV Shows, TV Dramas, TV Mysteries  International TV Shows
1  International TV Shows, TV Dramas, TV Mysteries                TV Dramas
2  International TV Shows, TV Dramas, TV Mysteries            TV Mysteries
3  International TV Shows, TV Dramas, TV Mysteries  International TV Shows
4  International TV Shows, TV Dramas, TV Mysteries                TV Dramas
```

[ ]:
```
df_type.head()


df_type['date_added'] = pd.to_datetime(df_type['date_added'], errors='coerce')

# Extract the year and create a new column
df_type['year_added'] = df_type['date_added'].dt.year  # Convert to integer

print(df_type)
```

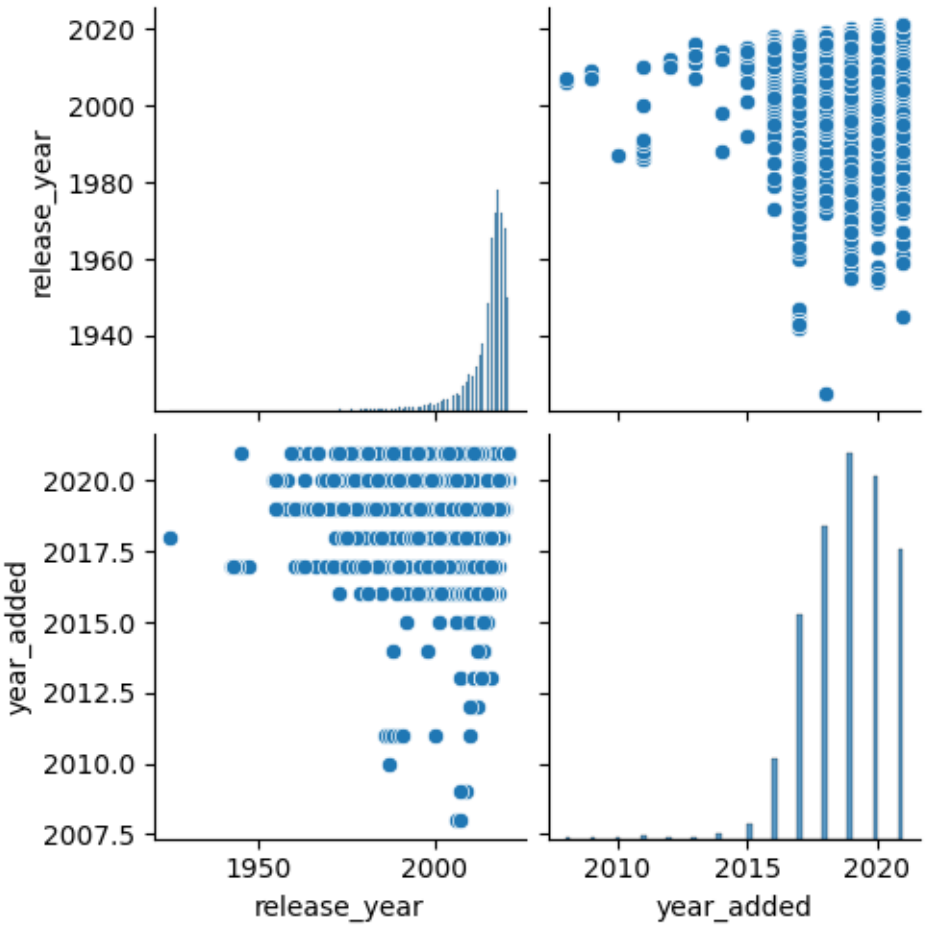
```
     show_id     type                  title         director  \
0         s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1         s2  TV Show          Blood & Water              NaN
2         s3  TV Show              Ganglands  Julien Leclercq
3         s4  TV Show  Jailbirds New Orleans              NaN
4         s5  TV Show           Kota Factory              NaN
…       …       …                      …              …
8802   s8803    Movie                Zodiac    David Fincher
8803   s8804  TV Show            Zombie Dumb              NaN
8804   s8805    Movie             Zombieland  Ruben Fleischer
8805   s8806    Movie                  Zoom     Peter Hewitt
8806   s8807    Movie                Zubaan      Mozez Singh

                                                 cast        country  \
0                                                 NaN  United States
```

```
1     Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…   South Africa
2     Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…            NaN
3                                                  NaN            NaN
4     Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…          India
…                                                    …            …
8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J…  United States
8803                                               NaN            NaN
8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, …  United States
8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma…  United States
8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan…          India

     date_added  release_year rating   duration  \
0    2021-09-25          2020  PG-13     90 min
1    2021-09-24          2021  TV-MA  2 Seasons
2    2021-09-24          2021  TV-MA   1 Season
3    2021-09-24          2021  TV-MA   1 Season
4    2021-09-24          2021  TV-MA  2 Seasons
…           …           …    …         …
8802 2019-11-20          2007      R    158 min
8803 2019-07-01          2018  TV-Y7  2 Seasons
8804 2019-11-01          2009      R     88 min
8805 2020-01-11          2006     PG     88 min
8806 2019-03-02          2015  TV-14    111 min

                                          listed_in  \
0                                     Documentaries
1       International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act…
3                               Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV …
…                                                   …
8802                    Cult Movies, Dramas, Thrillers
8803            Kids' TV, Korean TV Shows, TV Comedies
8804                           Comedies, Horror Movies
8805                Children & Family Movies, Comedies
8806    Dramas, International Movies, Music & Musicals

                                           description  year_added
0     As her father nears the end of his life, filmm…      2021.0
1     After crossing paths at a party, a Cape Town t…      2021.0
2     To protect his family from a powerful drug lor…      2021.0
3     Feuds, flirtations and toilet talk go down amo…      2021.0
4     In a city of coaching centers known to train I…      2021.0
…                                                   …           …
8802  A political cartoonist, a crime reporter and a…      2019.0
8803  While living alone in a spooky town, a young g…      2019.0
8804  Looking to survive in a world taken over by zo…      2019.0
8805  Dragged from civilian life, a former superhero…      2020.0
```

```
8806  A scrappy but poor boy worms his way into a ty…      2019.0

[8807 rows x 13 columns]
```

#pair plot of release year and year added in netflix.

[ ]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named df with columns 'release_year' and␣
 ↪'year_added'

# Create pair plot
sns.pairplot(df_type, vars=['release_year', 'year_added'])

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



#From the pair plot we can illustrate that year added and release year takes positive skewness. it is clearly indicated that majority of the content added to netflix the same release year.

#**Missing values and outlier treatement** #->In the current dataset, we can see that there are 8807 rows for each column. #->However, 7 columns have missing values. #->out of which director has around 30% missing value percentage. #->cast has around 9% missing value percentage. #->country has around 9.435676% missing value percentage. #->date_added has around 0.113546% missing value percentage. #->rating has around 0.045418% missing value percentage. #->duration has around 0.034064% missing value percentage.

#->finally we have removed the rows with less than one percentage which include date_Added, release_year,rating,duration.

#->We have used data imputation with specific names in colunns like director,cast,country.

#-> While analysis we have make sure considering the popula actor , director or country confined to the data available only, rather considering the imputed values.

#->However, While doing the analysis , we can impute the missing values with relevant names.

#->Since the data is corruptiong more, we have opted for using the accurate data available.

#->However , the insights might get slightly differ while doing the analysis with the incomplete data. We are considering the top frequency count with the avvailable data only.

#**Insights based on Non-Graphical and Visual Analysis**

#Comments on the range of attributes

#->There are different attributes in the given data set which include show_id,type ,title ,director,cast,date_added ,release_year ,rating,duration,listed_in,description

#->Out of above only the release year is numerical data , date added contains date and time as month date year. rest of the columns are categorical variables.

#In the range of attributes netflix has maximum share in last one decade.

#Comments on the distribution of the variables and relationship between them

16 ->Variables are distributed with respect to unique titles and correspondence director actor and country as well as their release year in theatre and date and time added in netlix.

#->Considering most Significant column has country , we try to find out the most popular actor, director , generes with respect to each country and try to get best out of it.

#Comments for each univariate and bivariate plot

#->As part of data analysis of netflix data set, we have wide range of plots.

#->Each Plot has lot of insights updated below

#->Trend of content added in netflix over the years showcase the netflix has more content from 2016 to 2021. #->Insights from pie chart shows the distribution of content of movie is more compared to TV Show . #->distrubution of country share chart illustrate the individual country share with

U.S, India, UK tops the list. #->most popular actor across the globe are anupham kher, sharuk khan, julie tejwani #->most popular director across the globe are rajiv chilaka, jan suter, raul campos. #->Most popular genre in netflix content are International movies followed by dramas.

#->Comparing different variables

#->Trend of movies over the years has surged in 2019 reaching its count by 1500. #->trend of tv shows has surged in 2020 with count of 500. #->comparison of movie and TV show over the year showcase the movie content has dominate the TV Show in netflix. #->movie content with indidvidial contribution of country has u.s,india, U.K in top 3

#->TV Show content with individual contribution of country are U.S, U.K, japan.

#->best time to launch a movie is december month as it has maximum number of content added.

#->best time to launch a TV Show showcases july with maximum releases.

17 ->most popular genre in every country.india and maximum countries has Internationa Movies . Where as U.S as Drama genre is popular and finally in U.K british Tv shows are popular .

#->most popular actors in respective countries #The most popular actor in United States is Alfred Molina #The most popular actor in United Kingdom is David Attenborough #The most popular actor in India is Anupam Kher #The most popular actor in France is Liam Neeson #The most popular actor in Canada is Liam Neeson #The most popular actor in Germany is Ben Whishaw #The most popular actor in Japan is Luci Christian #The most popular actor in Spain is David Attenborough #The most popular actor in China is Donnie Yen

#->most popular directors in respective countries-> #India has David Dhawan #United Kingdom has Lars von Trier #France has Lars von Trier #Canada has Raja Gosnel #Germany has Lars von Trier #Spain has Federico Veiroj #Japan has Toshiya Shinohara #China has Wilson Yip

#->Top 3 Actors who have appeared in the most TV shows are Takahiro Sakurai, Yuki Kaji, Junichi Suwabe

18 -> Most popular actor in tv shows in respective countries.

#Canada most popular actor John Paul Tremblay #France most popular actor Tahar Rahim #India most popular actor Prakash Raj #Japan most popular actor Takahiro Sakurai #Mexico most popular actor Sebastián Rulli #South Korea most popular actor Bae Doona #Spain most popular actor Carlos Cuevas #United Kingdom most popular actor David Attenborough #United States most popular actor Grey Griffin

#Business Insights

#-> Quantity In netflix content clearly indicates that movies dominate the TV shows in content wise over the years consistantly.

#-> Netflix has a surge of adding the content from 2016 and reached its peak in 2019 .

#Also,regarding the movie content, it maintained its consistency of adding more than more than 1200 movie content from 2018 to 2020. Same time maintained a minimum of 1000 movies since 2017.

#Also, when it comes to tv show content it has linear growth since 2018 to 2019 and maintained its growth till 2020 of around 600 content per year.

#-> Regarding content added in netflix to the realease year we see a majority content added to netflix in the same year when the movie released in theatre.

#->Most popular genre across the globe is international movies. At the same time every country has a specific popular genre with respect to movie and TV show content .

#-> Also, it is clearly visible that most popular actors in movies and tv shoows are different.

#-> Every country has unique popular actor when it comes to movies and TV shows.

#->Moreover, Majority of movie content added in the month of december, followed by october.

#Furthermore, The addition of tv show content showcase a july month followed by june in recent years.

#**Recommendations**

#->Majority Content type available in netflix is movies. Hence, netflix has to improve the quantity of TV Shows to engage the audience in terms of streaming minutes.

19 -> Need to improve the content creation in second most popular country which is india as per the data interms of content contribution.

#->Inaddition, since the huge difference in share of U.S and India in terms of content. making sure difference has to minimize and produce more content with interest of indian audiences followed by U.K and next popular countries.

#->Furthermore, need to find the top 3 popular actors with respect to the country and Movie content. Finally making sure to doing more movies and enagage the actor with their popular actor of country.

#->On the similar lines, by finding the popular actors with respect to country and TV show content. making sure to increase the episodes or web series of TV Shows to engage the audience every year and maintain the subscription consistantly.

#->Also, produce the more content interms of second and third most popular genre, inadditon to popular genre of every country. So that every customer will have wide range of content experience in different genres.

#->Mainly, native TV shows and movies will be having a international appeal. Hence it is recommend to make different generes like action, dramas and many other with repsect to native country engaging native actors and directors.

#-> finding the top 5 popular directors of country interms of movie and TV shows. produce more content to engage subscriber to excperience the individual cinematic experience.

#-> Having a good combination of actor and directors of specific country can attract the audience to subscribe for any new content.

#->Mainly, Making sure the target audience of specific movie or TV Show.Then the best release time can be found out.

#-> For example, if the target audience is from specific country, then release the content in netflix in any long festival or holiday season which garner the large pool of subscribers.