netflix-case-study

March 4, 2024

NETFLIX Business Case study

Netlix, Inc. is an American technology and media services provider and production company headquartered in Los Gatos, California. Ne lix was founded in 1997 by Reed Has ngs and Marc Randolph in Sco s Valley, California. The company's primary business is its subscrip on-based streaming service, which offers online streaming of a library of films and television series, including those produced in-house.

Importing required Libraries

[27]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import gdown
from collections import Counter
from wordcloud import WordCloud
```

0.1 Basic Data Analyses

[36]:
```
df=pd.read_csv('netflix.csv')
df.head()
```

[36]:

	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				
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				

```
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…
```

[29]:
```
#finding out the no of rows and columns in the dataset
df.shape
```

[29]:
```
(8807, 12)
```

[30]:
```
#finding out the list of columns present in the dataset
df.columns
```

[30]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

[31]:
```
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
```

[13]:
```
df.describe()
```

[13]:
```
       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
```

[14]:
```
df['type'].value_counts()
```

[14]:
```
Movie      6131
TV Show    2676
Name: type, dtype: int64
```

[32]:
```
#Missing Valuses in each column
df.isna().sum()
```

[32]:
```
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
```

Replacing Null values as 'Unknown_column_name' in the given netflix dataset

[33]:
```
def replace_null_categorical(df):
    categorical_columns = df.select_dtypes(include=['object']).columns
    for column in categorical_columns:
        df[column] = df[column].fillna("unknown " + column)
    return df
```

[34]:
```
df1= replace_null_categorical(df)
df1.head()
```

[34]:
```
  show_id     type                  title          director  \
0      s1    Movie   Dick Johnson Is Dead   Kirsten Johnson
1      s2  TV Show          Blood & Water  unknown director
2      s3  TV Show              Ganglands   Julien Leclercq
3      s4  TV Show  Jailbirds New Orleans  unknown director
4      s5  TV Show           Kota Factory  unknown director

                                                cast          country  \
0                                        unknown cast    United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…     South Africa
2  Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…  unknown country
3                                        unknown cast  unknown country
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…
```

[19]:
```
#checking null values after tidying dataset
df1.isna().sum()
```

[19]:
```
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
```

Un-nest the columns those have cells with multiple comma separated values by creating multiple rows

[20]:
```
df2=df
df2.head(2)
```

[20]:
```
  show_id     type                 title          director  \
0      s1    Movie  Dick Johnson Is Dead   Kirsten Johnson
1      s2  TV Show         Blood & Water  unknown director

                                          cast        country  \
0                                  unknown cast  United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…   South Africa

           date_added  release_year rating   duration  \
0  September 25, 2021          2020  PG-13     90 min
1  September 24, 2021          2021  TV-MA  2 Seasons

                                    listed_in  \
0                                Documentaries
1  International TV Shows, TV Dramas, TV Mysteries

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

- For columns 'listed_in' and 'cast' we have multiple comma seperated values

[21]:
```
df2['cast'] = df2['cast'].str.split(',')
df2= df2.explode('cast')
df2['listed_in'] = df2['listed_in'].str.split(', ')
df2 = df2.explode('listed_in')
df2.head(5)
```

[21]:
```
  show_id     type                 title          director          cast  \
0      s1    Movie  Dick Johnson Is Dead   Kirsten Johnson  unknown cast
1      s2  TV Show         Blood & Water  unknown director    Ama Qamata
1      s2  TV Show         Blood & Water  unknown director    Ama Qamata
1      s2  TV Show         Blood & Water  unknown director    Ama Qamata
1      s2  TV Show         Blood & Water  unknown director   Khosi Ngema
```

```
          country          date_added  release_year rating   duration  \
0   United States  September 25, 2021          2020  PG-13     90 min
1    South Africa  September 24, 2021          2021  TV-MA  2 Seasons
1    South Africa  September 24, 2021          2021  TV-MA  2 Seasons
1    South Africa  September 24, 2021          2021  TV-MA  2 Seasons
1    South Africa  September 24, 2021          2021  TV-MA  2 Seasons

                 listed_in                                        description
0           Documentaries  As her father nears the end of his life, filmm…
1  International TV Shows  After crossing paths at a party, a Cape Town t…
1               TV Dramas  After crossing paths at a party, a Cape Town t…
1            TV Mysteries  After crossing paths at a party, a Cape Town t…
1  International TV Shows  After crossing paths at a party, a Cape Town t…
```

[22]:
```
df2.shape
```

[22]:
```
(149512, 12)
```

We can clearly see that the no of rows have been increased in the dataset after unnesting the columns 'cast' and 'listed_in'

0.2 1. Find the counts of each categorical variable both using graphical and nongraphical analysis

[23]:
```
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      8807 non-null   object
 4   cast          8807 non-null   object
 5   country       8807 non-null   object
 6   date_added    8807 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8807 non-null   object
 9   duration      8807 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
```

[40]:
```
count_columns = ['type','director','cast','country','rating','listed_in']
val_counts = {}
for i in count_columns:
  val_counts[i] = df[i].value_counts()
for col, counts in val_counts.items():
    print(f"{col} counts:")
    print(counts)
    print()
```

```
type counts:
Movie      6131
TV Show    2676
Name: type, dtype: int64

director counts:
Rajiv Chilaka                     19
Raúl Campos, Jan Suter            18
Marcus Raboy                      16
Suhas Kadav                       16
Jay Karas                         14
                                  ..
Raymie Muzquiz, Stu Livingston     1
Joe Menendez                       1
Eric Bross                         1
Will Eisenberg                     1
Mozez Singh                        1
Name: director, Length: 4528, dtype: int64

cast counts:
David Attenborough
19
Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam,
Swapnil
14
Samuel West
10
Jeff Dunham
7
David Spade, London Hughes, Fortune Feimster
6
                                 ..
Michael Peña, Diego Luna, Tenoch Huerta, Joaquin Cosio, José María Yazpik, Matt
Letscher, Alyssa Diaz
1
Nick Lachey, Vanessa Lachey
1
Takeru Sato, Kasumi Arimura, Haru, Kentaro Sakaguchi, Takayuki Yamada, Kendo
```

```
Kobayashi, Ken Yasuda, Arata Furuta, Suzuki Matsuo, Koichi Yamadera, Arata Iura,
Chikako Kaku, Kotaro Yoshida     1
Toyin Abraham, Sambasa Nzeribe, Chioma Chukwuka Akpotha, Chioma Omeruah,
Chiwetalu Agu, Dele Odule, Femi Adebayo, Bayray McNwizu, Biodun Stephen
1
Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanana, Manish Chaudhary, Meghna Malik,
Malkeet Rauni, Anita Shabdish, Chittaranjan Tripathy
1
Name: cast, Length: 7692, dtype: int64

country counts:
United States                                2818
India                                         972
United Kingdom                                419
Japan                                         245
South Korea                                   199
                                              …
Romania, Bulgaria, Hungary                      1
Uruguay, Guatemala                              1
France, Senegal, Belgium                        1
Mexico, United States, Spain, Colombia          1
United Arab Emirates, Jordan                    1
Name: country, Length: 748, dtype: int64

rating counts:
TV-MA       3207
TV-14       2160
TV-PG        863
R            799
PG-13        490
TV-Y7        334
TV-Y         307
PG           287
TV-G         220
NR            80
G             41
TV-Y7-FV       6
NC-17          3
UR             3
74 min         1
84 min         1
66 min         1
Name: rating, dtype: int64

listed_in counts:
Dramas, International Movies                          362
Documentaries                                         359
Stand-Up Comedy                                       334
```

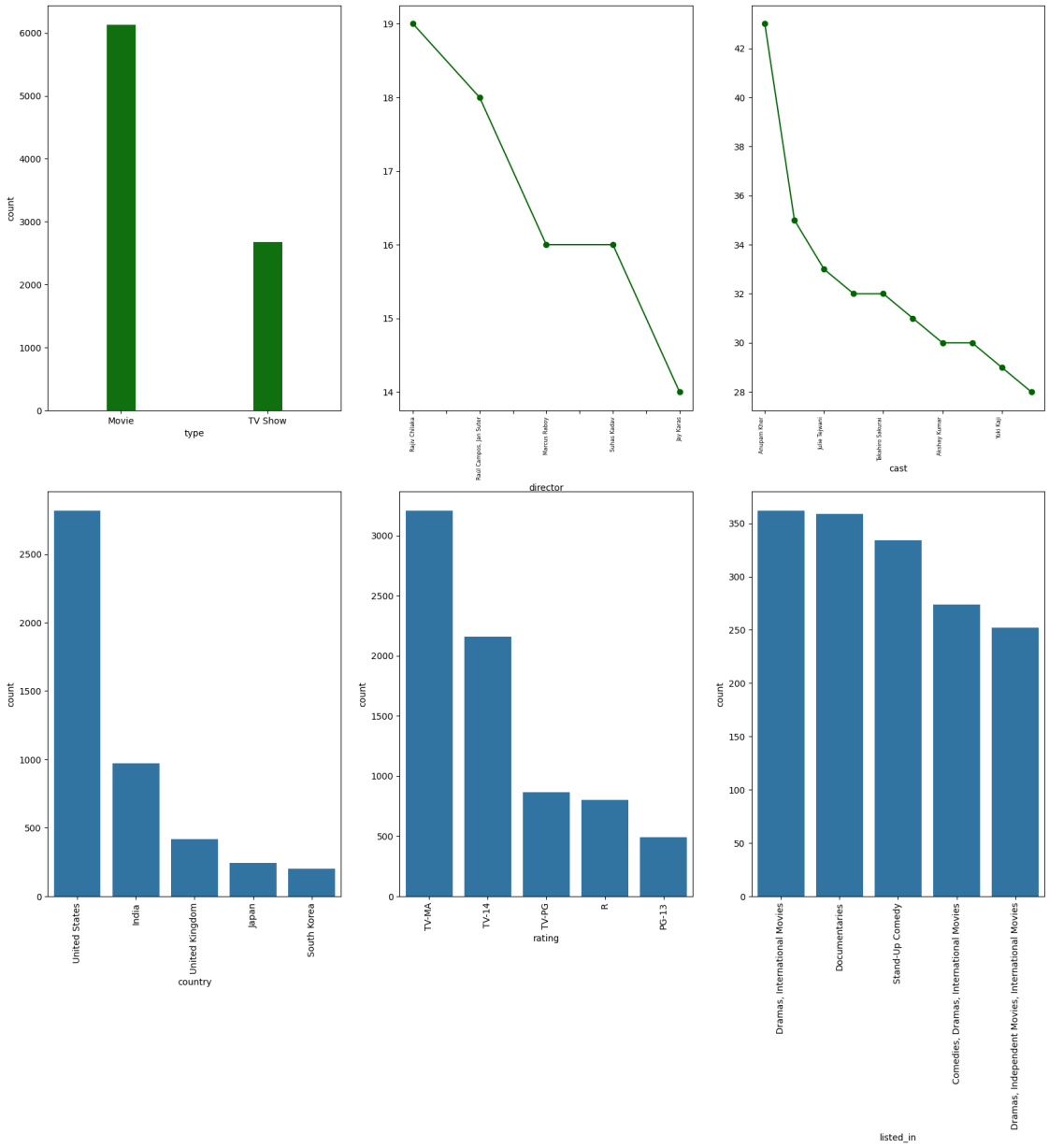
```
Comedies, Dramas, International Movies                274
Dramas, Independent Movies, International Movies      252
                                                      ...
Kids' TV, TV Action & Adventure, TV Dramas              1
TV Comedies, TV Dramas, TV Horror                       1
Children & Family Movies, Comedies, LGBTQ Movies        1
Kids' TV, Spanish-Language TV Shows, Teen TV Shows      1
Cult Movies, Dramas, Thrillers                          1
Name: listed_in, Length: 514, dtype: int64
```

Data Visualization

[39]:
```
type_df = df['type'].value_counts()
dir_df = df['director'].value_counts().head()
cast_df = df['cast'].str.split(', ', expand=True).stack().reset_index(level=1,␣
↪drop=True).rename('cast')
cast_counts = cast_df.value_counts().sort_values(ascending=False).head(10)
country_df = df['country'].value_counts().head(5)
rating_df = df['rating'].value_counts().head()
listed_in_df =df['listed_in'].value_counts().head()

plt.figure(figsize=(20,18))
plt.subplot(2,3,1)
sns.countplot(data=df,x='type',order=type_df.index,width=0.2,color='green')
plt.xlabel('type')
plt.subplot(2,3,2)
dir_df.plot(kind='line', marker='o', color='darkgreen')
#sns.countplot(data=df,x='director',order=dir_df.index)
plt.xlabel('director')
plt.xticks(rotation=90,fontsize=6)
plt.subplot(2,3,3)
cast_counts.plot(kind='line', marker='o', color='darkgreen')
#sns.countplot(data=df,x='cast',order=cast_counts.index)
plt.xlabel('cast')
plt.xticks(rotation=90,fontsize=6,ha='right')
plt.subplot(2,3,4)
sns.countplot(data=df,x='country',order=country_df.index)
plt.xlabel('country')
plt.xticks(rotation=90)
plt.subplot(2,3,5)
sns.countplot(data=df,x='rating',order=rating_df.index)
plt.xlabel('rating')
plt.xticks(rotation=90)
plt.subplot(2,3,6)
#listed_in_df.plot(kind='line', marker='o', color='skyblue')
sns.countplot(data=df,x='listed_in',order=listed_in_df.index)
plt.xlabel('listed_in')
```

```
plt.xticks(rotation=90)
```

[39]:
```
([0, 1, 2, 3, 4],
 [Text(0, 0, 'Dramas, International Movies'),
  Text(1, 0, 'Documentaries'),
  Text(2, 0, 'Stand-Up Comedy'),
  Text(3, 0, 'Comedies, Dramas, International Movies'),
  Text(4, 0, 'Dramas, Independent Movies, International Movies')])
```



0.3 2. Comparison of tv shows vs. movies.

#a. Find the number of movies produced in each country and pick the top 10 countries.

[41]:
```
#a. Find the number of movies produced in each country and pick the top 10␣
 ↪countries.
movies=df.loc[df['type']=='Movie']
movies['Total_Movies']= movies.groupby('country')['type'].apply('count')
movies_count= movies.groupby('country').size().
 ↪reset_index(name='Movies_Produced')
Top10_countries = movies_count.
 ↪sort_values(by='Movies_Produced',ascending=False).head(10)
Top10_countries
```

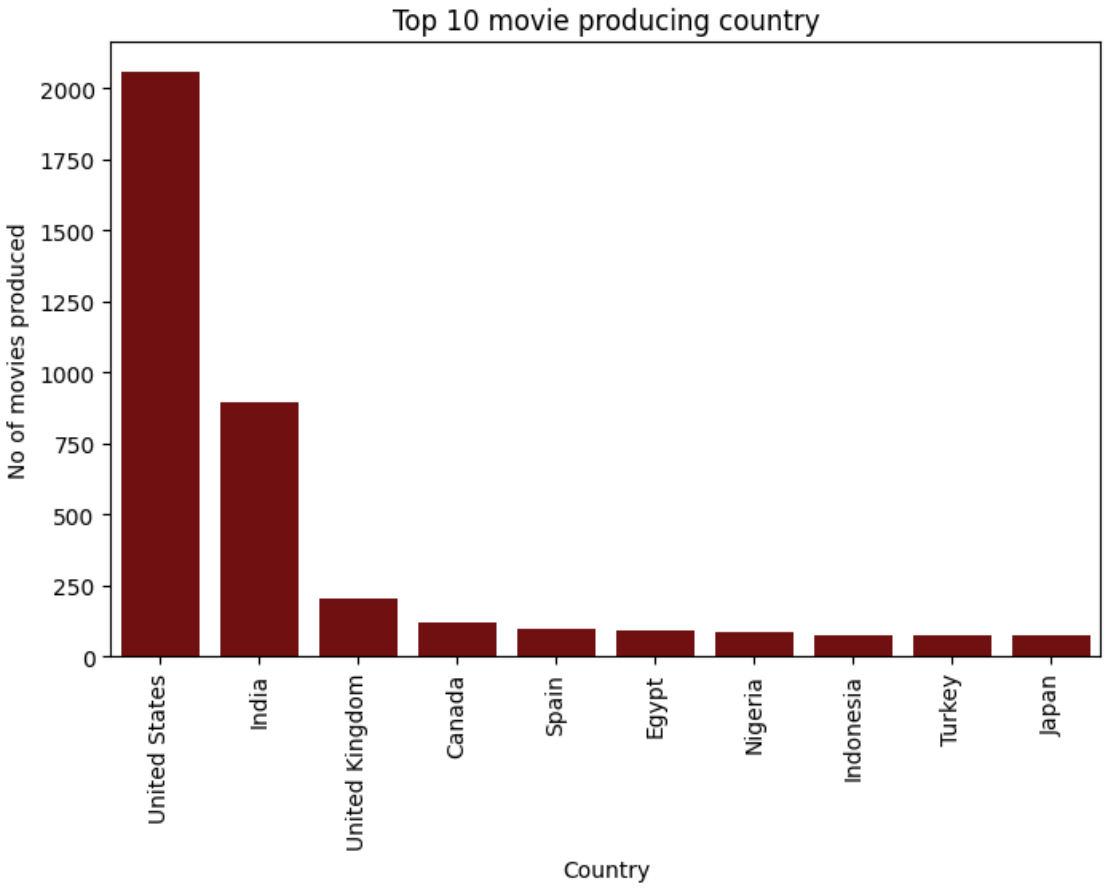
```
<ipython-input-41-f0960a1432e8>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies['Total_Movies']= movies.groupby('country')['type'].apply('count')
```

[41]:

	country	Movies_Produced
525	United States	2058
218	India	893
440	United Kingdom	206
50	Canada	122
384	Spain	97
128	Egypt	92
319	Nigeria	86
238	Indonesia	77
428	Turkey	76
278	Japan	76

Data visualization

[42]:
```
plt.figure(figsize=(8,5))
sns.barplot(data=Top10_countries,x='country',y='Movies_Produced',color='Maroon')
plt.xlabel('Country')
plt.ylabel('No of movies produced')
plt.title('Top 10 movie producing country')
plt.xticks(rotation=90)
plt.show()
```



b. Find the number of Tv-Shows produced in each country and pick the top 10 countries.

[43]:
```
Shows=df.loc[df['type']=='TV Show']
Shows['Total_Shows']= Shows.groupby('country')['type'].apply('count')
Shows_count= Shows.groupby('country').size().reset_index(name='Shows_Produced')
Top10_TVShow_countries = Shows_count.
  ↪sort_values(by='Shows_Produced',ascending=False).head(10)
Top10_TVShow_countries
```

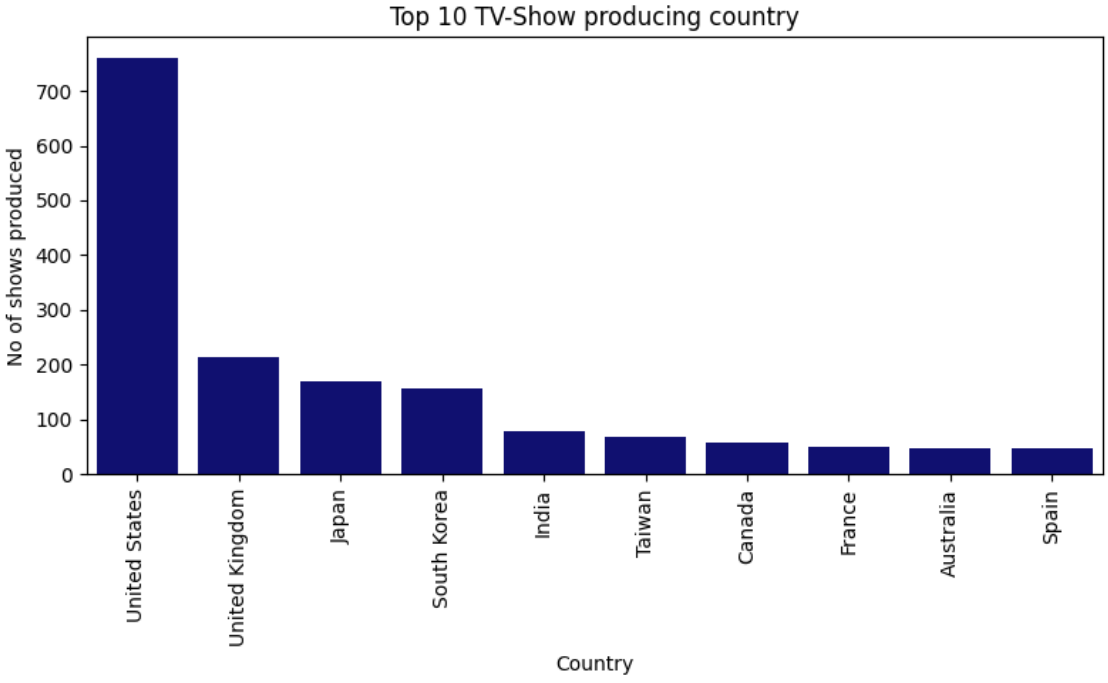
```
<ipython-input-43-0f7cb01e1f82>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  Shows['Total_Shows']= Shows.groupby('country')['type'].apply('count')
```

[43]:

	country	Shows_Produced
160	United States	760
140	United Kingdom	213
83	Japan	169
120	South Korea	158
66	India	79
132	Taiwan	68
17	Canada	59
47	France	49
4	Australia	48
125	Spain	48

Data visualization

[44]:
```
plt.figure(figsize=(8,5))
sns.
 ↪barplot(data=Top10_TVShow_countries,x='country',y='Shows_Produced',color='Navy')
plt.xlabel('Country')
plt.ylabel('No of shows produced')
plt.title('Top 10 TV-Show producing country')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



0.4 3. What is the best time to launch a TV show?

a. Find which is the best week to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies

[45]:
```
df_movie=df[df['type']=='Movie']
df_movie['date_added']= pd.to_datetime(df_movie['date_added'])
df_movie['week_number'] = df_movie['date_added'].dt.isocalendar().week
movie_group =df_movie.groupby('week_number').size().
  ↪reset_index(name='Total_movies')
movie_group.sort_values(by='Total_movies',ascending=False).head(10)
```

```
<ipython-input-45-41207b5dc422>: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
  df_movie['date_added']= pd.to_datetime(df_movie['date_added'])
<ipython-input-45-41207b5dc422>: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
  df_movie['week_number'] = df_movie['date_added'].dt.isocalendar().week
```

[45]:

	week_number	Total_movies
0	1	316
43	44	243
39	40	215
8	9	207
25	26	195
34	35	189
30	31	185
12	13	174
17	18	173
26	27	154

[46]:
```
df_shows=df[df['type']=='TV Show']
df_shows['date_added']= pd.to_datetime(df_shows['date_added'])
df_shows['week_number'] = df_shows['date_added'].dt.isocalendar().week
shows_group =df_shows.groupby('week_number').size().
  ↪reset_index(name='Total_shows')
shows_group.sort_values(by='Total_shows',ascending=False).head()
```

```
<ipython-input-46-9dbeb9995587>: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
```

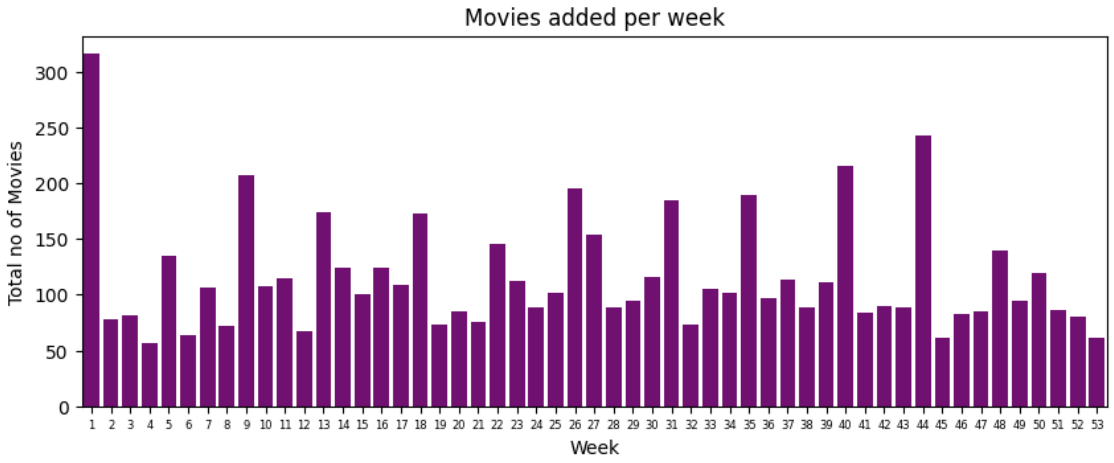
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_shows['date_added']= pd.to_datetime(df_shows['date_added'])
<ipython-input-46-9dbeb9995587>:3: SettingWithCopyWarning:
A value is trying to be set on a 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
  df_shows['week_number'] = df_shows['date_added'].dt.isocalendar().week
```

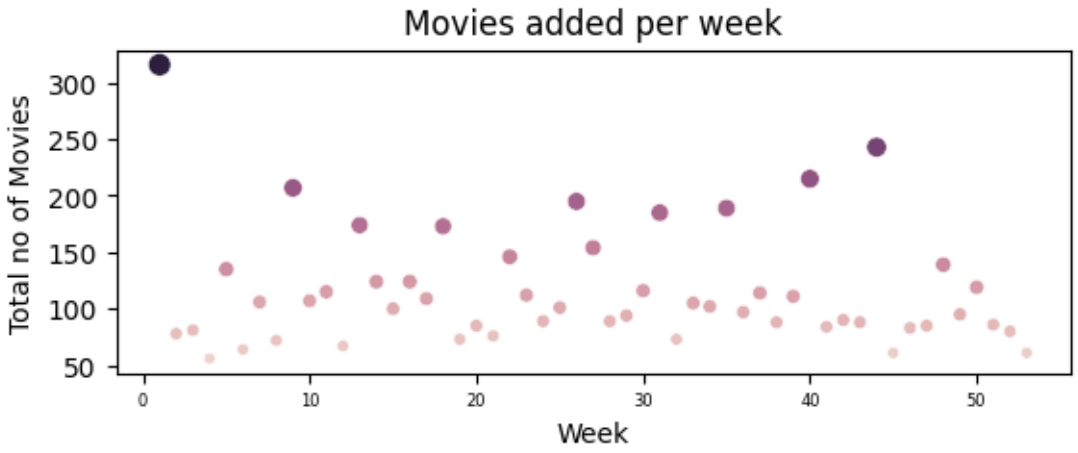
[46]:

	week_number	Total_shows
26	27	86
30	31	83
12	13	76
43	44	75
23	24	75

Data Visualization

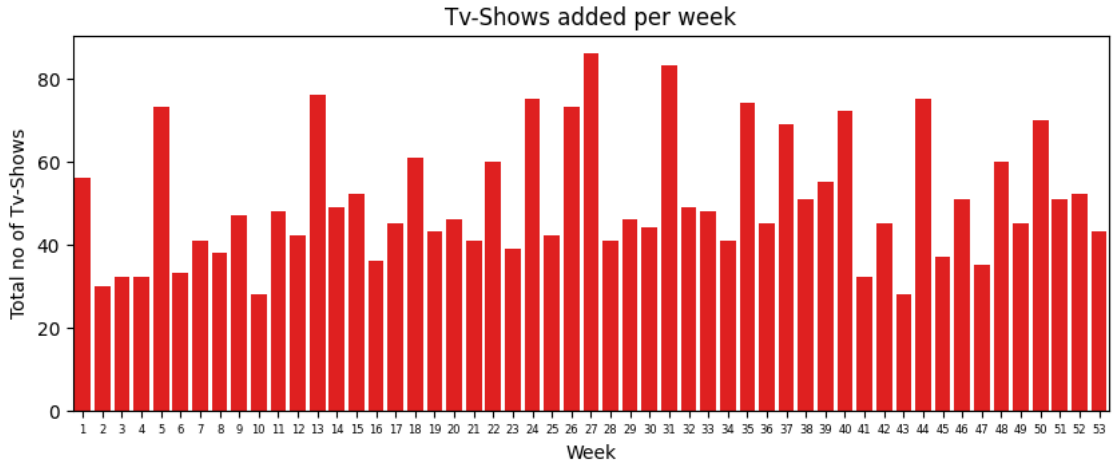
[87]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
sns.barplot(data=movie_group,x='week_number',y='Total_movies',width=0.
 ↪8,color='purple')
#sns.lineplot(data=movie_group,x='week_number',y='Total_movies',color='purple')
plt.title('Movies added per week')
plt.xlabel('Week')
plt.ylabel('Total no of Movies')
plt.xticks(fontsize=6)
plt.show()
plt.subplot(2,1,2)
sns.
 ↪scatterplot(data=movie_group,x='week_number',y='Total_movies',size='Total_movies',legend=Fa
plt.title('Movies added per week')
plt.xlabel('Week')
plt.ylabel('Total no of Movies')
plt.xticks(fontsize=6)
plt.show()
```

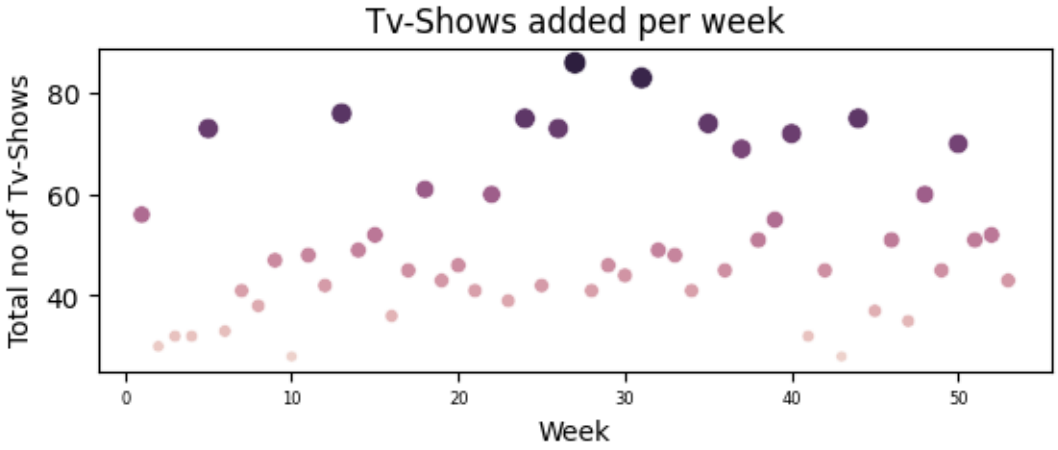




[90]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
sns.barplot(data=shows_group,x='week_number',y='Total_shows',width=0.
 ↪8,color='red')
plt.title('Tv-Shows added per week')
plt.xlabel('Week')
plt.ylabel('Total no of Tv-Shows')
plt.xticks(fontsize=6)
plt.show()
plt.subplot(2,1,2)
sns.
 ↪scatterplot(data=shows_group,x='week_number',y='Total_shows',size='Total_shows',legend=Fals
plt.title('Tv-Shows added per week')
plt.xlabel('Week')
```

```
plt.ylabel('Total no of Tv-Shows')
plt.xticks(fontsize=6)
plt.show()
```





Movies: 1. Most of the movies are released in the first week of the year which could coincide with the post-holiday season where people are spending more time indoors, leading to a higher demand for new content. 2. Netflix strategy of adding more and more movies leading upto the holiday season can also be observed 3. We can also observe continuous adding of content into the platform which shows very good Platform Engagement

TV Shows: 1. Most of the TV shows released are around 27th and 31st week of the year which indicates the higher viewership or demand for new content due to the summer vacation periods or more likely people spending more time indoors due to the weather conditions 2. Netflix is religiously adding new content to the platform every week, which shows good competetiveness with rival platforms

b. Find which is the best month to release the Tv-show or the movie. Do the analysis separately for Tv-shows and Movies text*

[47]:
```
movie_type=df[df['type']=='Movie']
movie_type['date_added'] = pd.to_datetime(movie_type['date_added'])
movie_type['Month'] = movie_type['date_added'].dt.strftime('%B')
movie_count = movie_type.groupby('Month').size().
 ↪reset_index(name='Total_Movies')
movie_count.sort_values(by='Total_Movies',ascending=False)
```

```
<ipython-input-47-1153ad4e51f6>: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
  movie_type['date_added'] = pd.to_datetime(movie_type['date_added'])
<ipython-input-47-1153ad4e51f6>: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
  movie_type['Month'] = movie_type['date_added'].dt.strftime('%B')
```

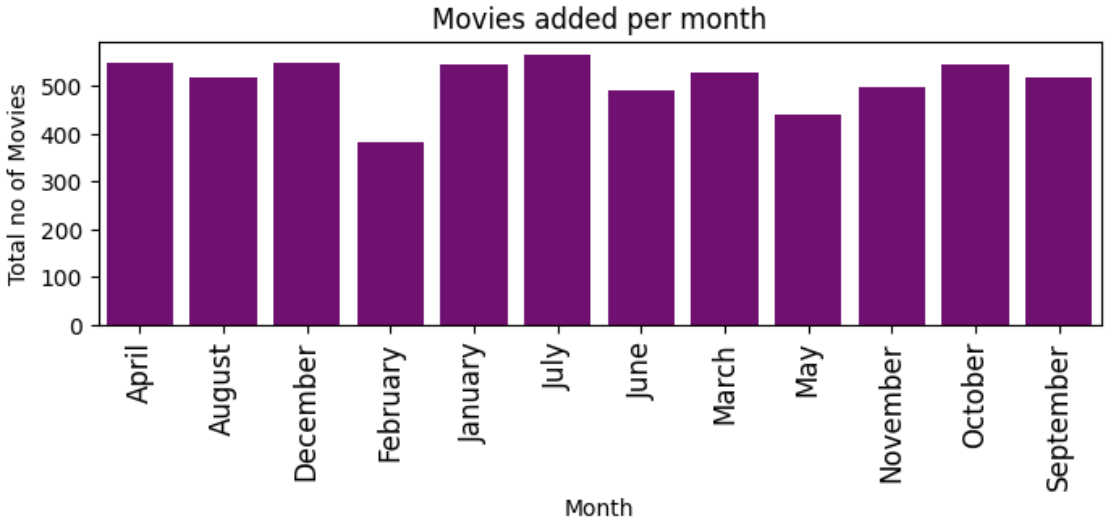
[47]:

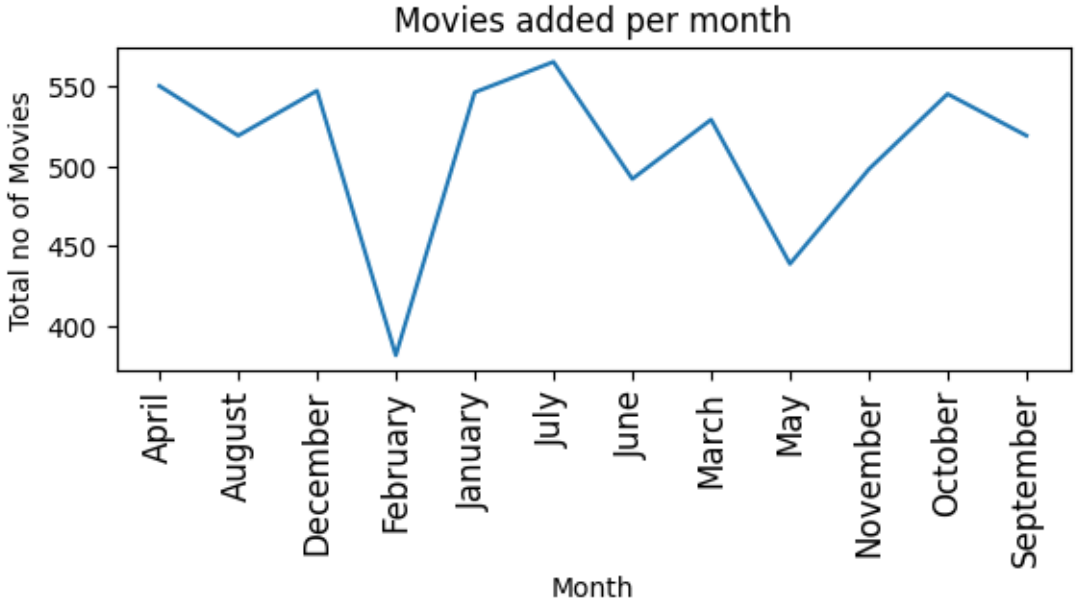
	Month	Total_Movies
5	July	565
0	April	550
2	December	547
4	January	546
10	October	545
7	March	529
1	August	519
11	September	519
9	November	498
6	June	492
8	May	439
3	February	382

Data Visualization

[103]:
```
plt.figure(figsize=(8,5))
plt.subplot(2,1,1)
sns.barplot(data=movie_count,x='Month',y='Total_Movies',width=0.
 ↪8,color='purple')
plt.title('Movies added per month')
plt.xlabel('Month')
plt.ylabel('Total no of Movies')
```

```
plt.xticks(rotation=90,fontsize=12)
plt.show()
plt.subplot(2,1,2)
sns.lineplot(data=movie_count,x='Month',y='Total_Movies')
plt.title('Movies added per month')
plt.xlabel('Month')
plt.ylabel('Total no of Movies')
plt.xticks(rotation=90,fontsize=12)
plt.show()
```





[49]:
```
show_type=df[df['type']=='TV Show']
show_type['date_added'] = pd.to_datetime(show_type['date_added'])
show_type['Month'] = show_type['date_added'].dt.strftime('%B')
show_count = show_type.groupby('Month').size().reset_index(name='Total_Shows')
show_count.sort_values(by='Total_Shows',ascending=False)
```

```
<ipython-input-49-eca8783924f6>: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
  show_type['date_added'] = pd.to_datetime(show_type['date_added'])
<ipython-input-49-eca8783924f6>: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
  show_type['Month'] = show_type['date_added'].dt.strftime('%B')
```

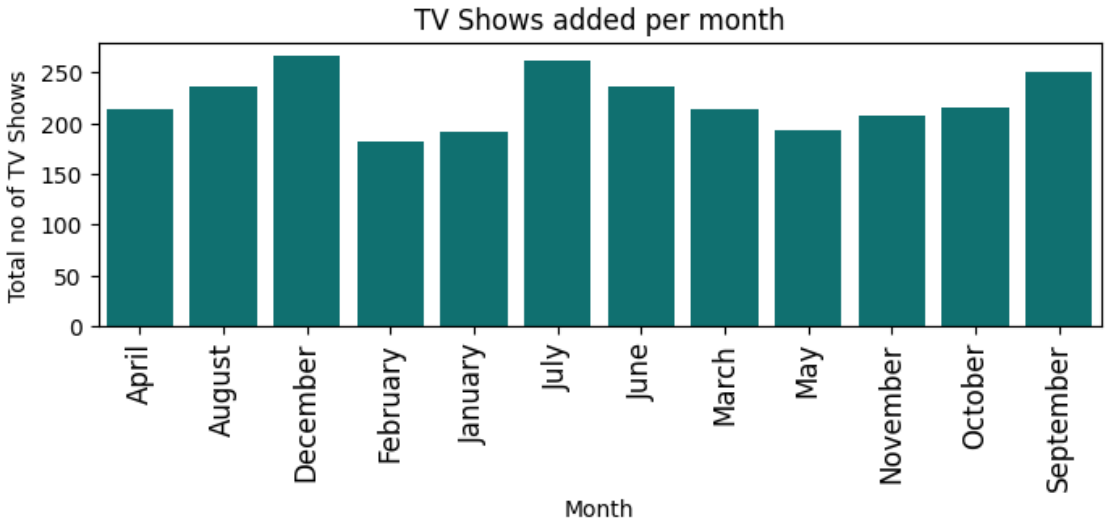
[49]:

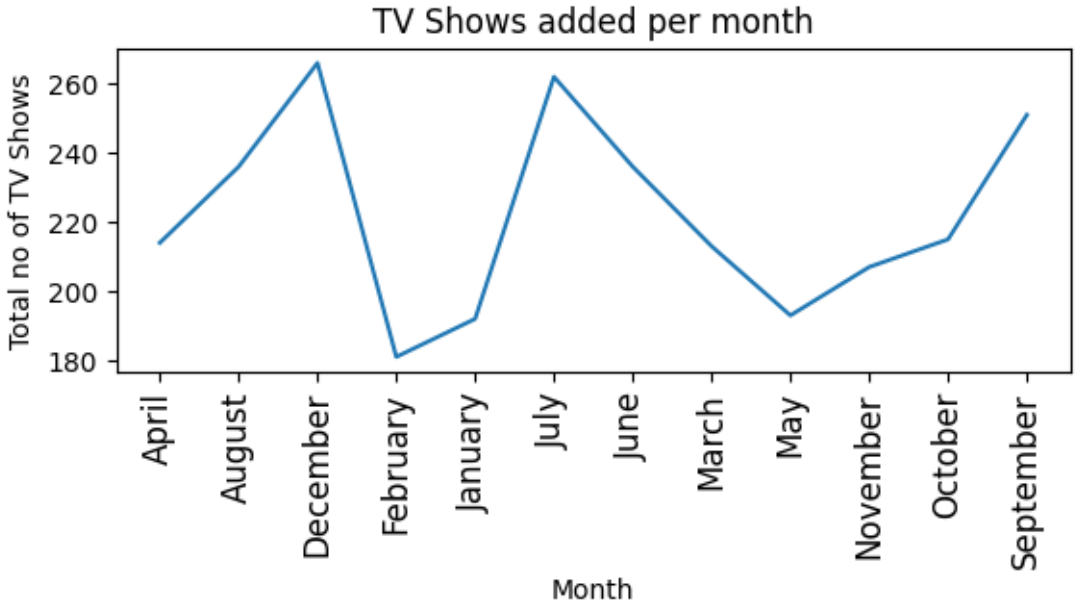
	Month	Total_Shows
2	December	266
5	July	262
11	September	251
1	August	236
6	June	236
10	October	215
0	April	214
7	March	213
9	November	207
8	May	193
4	January	192
3	February	181

Data Visualization

[95]:
```
plt.figure(figsize=(8,5))
plt.subplot(2,1,1)
sns.barplot(data=show_count,x='Month',y='Total_Shows',width=0.8,color='Teal')
plt.title('TV Shows added per month')
plt.xlabel('Month')
plt.ylabel('Total no of TV Shows')
plt.xticks(rotation=90,fontsize=12)
plt.show()
plt.subplot(2,1,2)
```

```
sns.lineplot(data=show_count,x='Month',y='Total_Shows')
plt.title('TV Shows added per month')
plt.xlabel('Month')
plt.ylabel('Total no of TV Shows')
plt.xticks(rotation=90,fontsize=12)
plt.show()
```





Movies: 1. Most of the Movies released are around july which indicates the higher viewership

or demand for new content due to the summer vacation periods or more likely people spending more time indoors due to the weather conditions 2. Netflix is religiously adding new content to the platform every week, which shows good competetiveness with rival platforms

TV Shows: 1. Most of the TV Shows are released in December which could coincide with the post-holiday season where people are spending more time indoors, leading to a higher demand for new content. 2. Netflix strategy of adding more and more movies leading upto the holiday season can also be observed 3. We can also observe continuous adding of content into the platform which shows very good Platform Engagement

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

a. Identify the top 10 directors who have appeared in most movies or TV shows.

[37]:
```
df_c =df
df_filtered = df_c.dropna(subset=['cast'])
df_filtered['cast'] = df_filtered['cast'].str.split(', ')
actors_df = df_filtered.explode('cast')
actor_appearances = actors_df.groupby('cast').size().
 ↪reset_index(name='appearances')
actor_appearances.sort_values(by='appearances', ascending=False).head(10)
```

```
<ipython-input-37-2a8cedddf6e2>: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
  df_filtered['cast'] = df_filtered['cast'].str.split(', ')
```

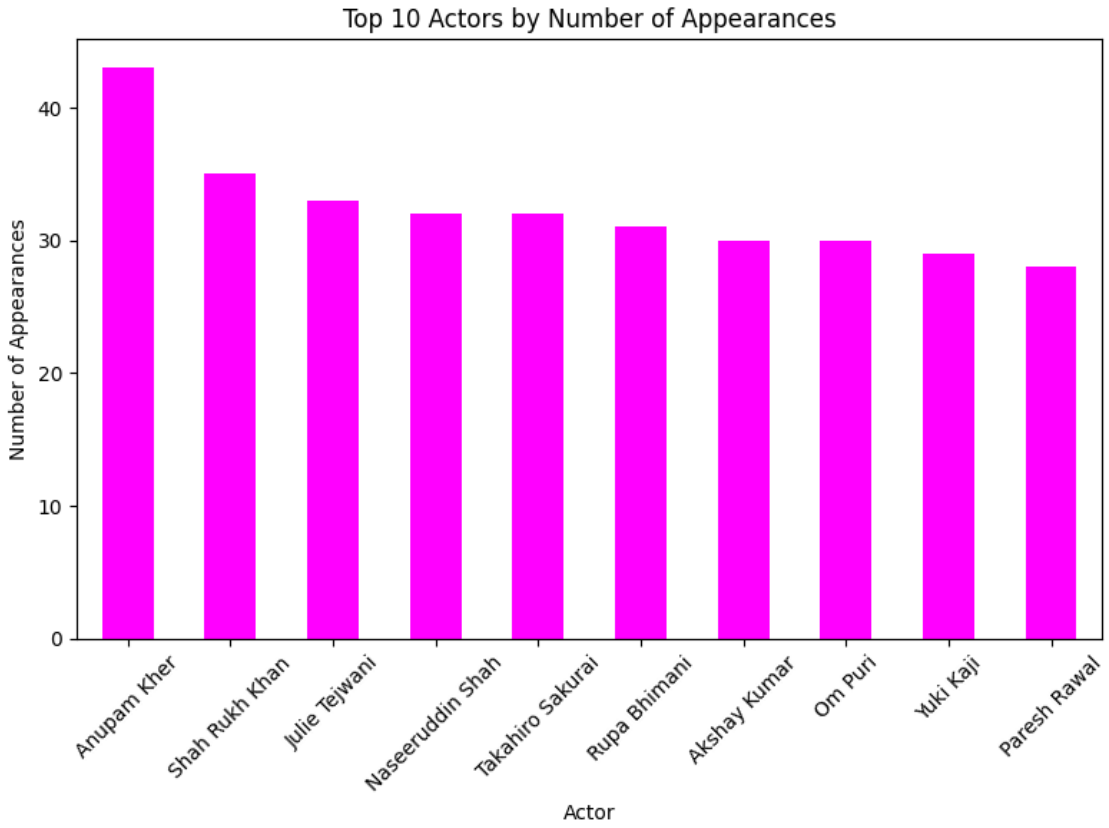
[37]:

	cast	appearances
2833	Anupam Kher	43
30489	Shah Rukh Khan	35
16697	Julie Tejwani	33
24215	Naseeruddin Shah	32
32591	Takahiro Sakurai	32
28974	Rupa Bhimani	31
846	Akshay Kumar	30
25424	Om Puri	30
35880	Yuki Kaji	29
1774	Amitabh Bachchan	28

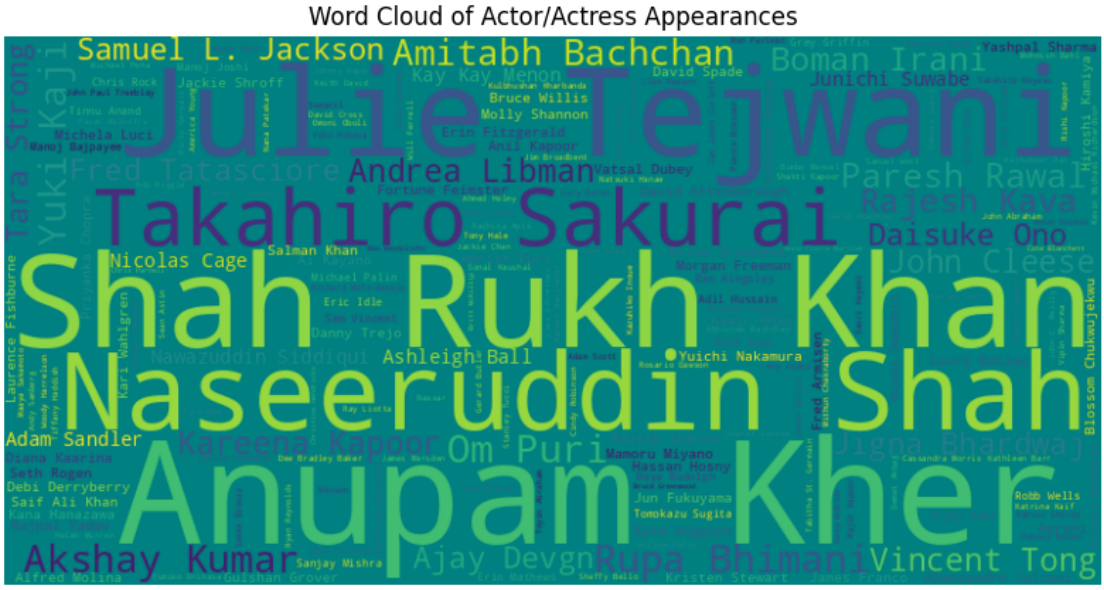
Data Visualization

[38]:
```
actor_counts = actors_df['cast'].value_counts().head(10)
plt.figure(figsize=(8,6))
actor_counts.plot(kind='bar',color='Magenta')
plt.title('Top 10 Actors by Number of Appearances')
plt.xlabel('Actor')
```

```
plt.ylabel('Number of Appearances')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:
```
Total_appearances = dict(zip(actor_appearances['cast'],␣
 ↪actor_appearances['appearances']))
wordcloud = WordCloud(width=800, height=400, background_color='Teal').
 ↪generate_from_frequencies(Total_appearances)
plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Actor/Actress Appearances')
plt.show()
```



b. Identify the top 10 directors who have appeared in most movies or TV shows.

[ ]:
```
df_filtered_director = df_c.dropna(subset=['director'])
directors_df = df_filtered_director.explode('director')
director_appearances = directors_df.groupby('director').size().
 ↪reset_index(name='films_directed')
director_appearances.sort_values(by='films_directed', ascending=False).head(10)
```

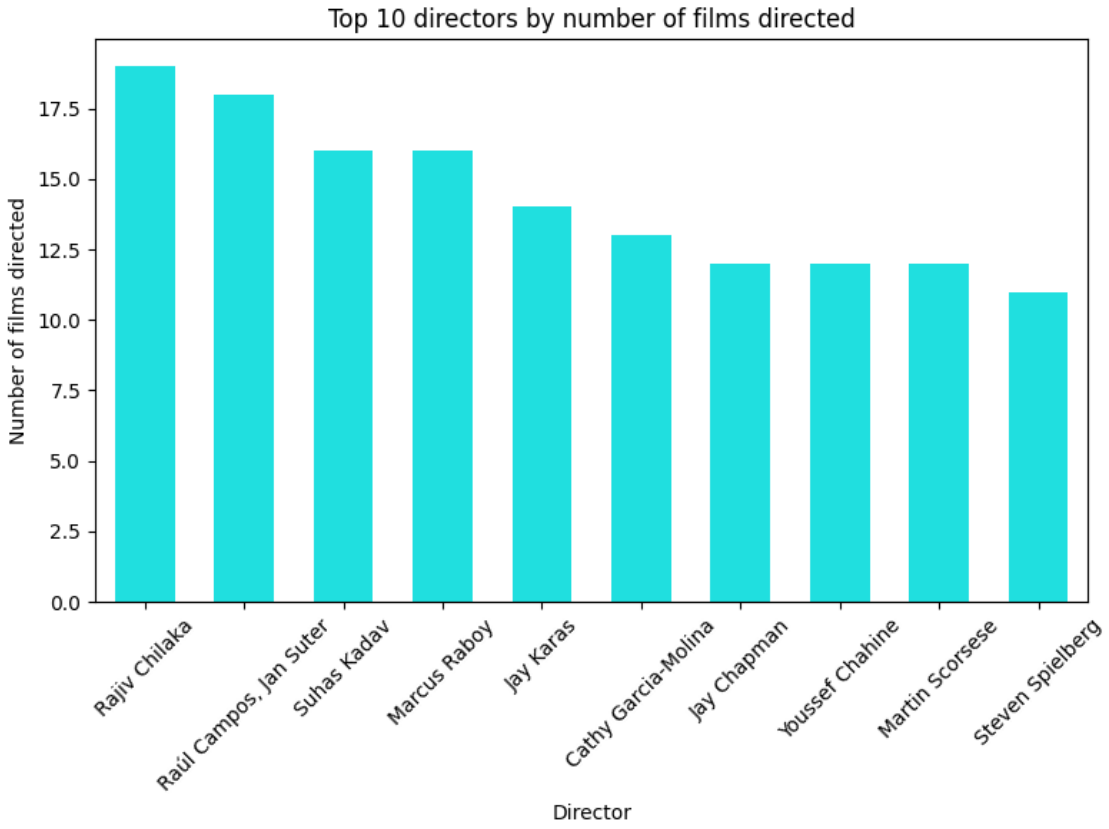
[ ]:

	director	films_directed
3392	Rajiv Chilaka	19
3443	Raúl Campos, Jan Suter	18
4046	Suhas Kadav	16
2598	Marcus Raboy	16
1790	Jay Karas	14
685	Cathy Garcia-Molina	13
1787	Jay Chapman	12
4480	Youssef Chahine	12
2671	Martin Scorsese	12
4020	Steven Spielberg	11

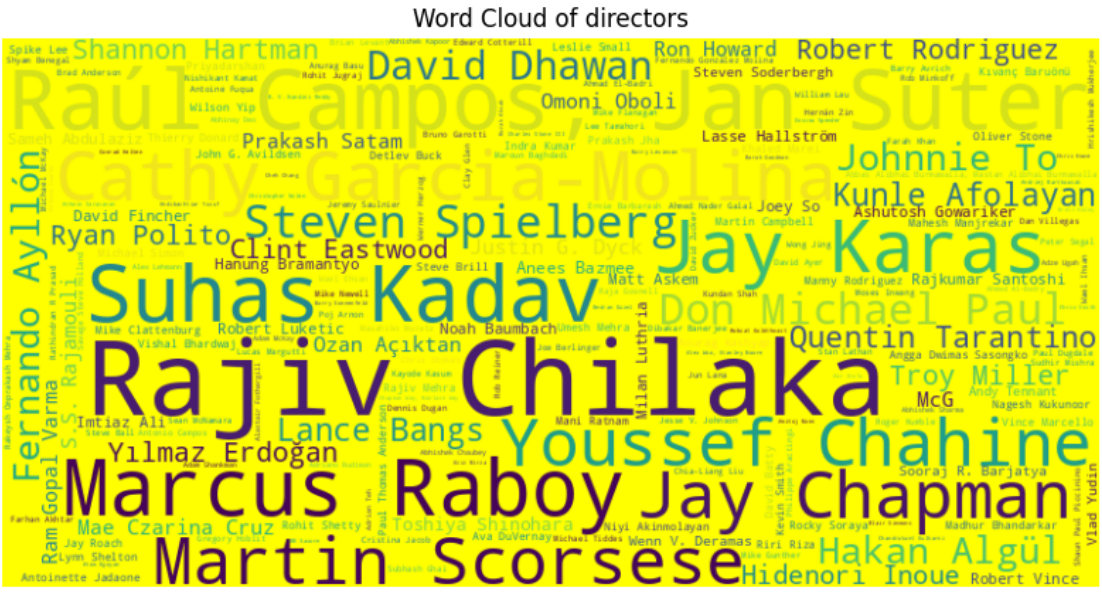
Data Visualization

[ ]:
```
#df_filtered_director = df_c.dropna(subset=['director'])
#directors_df = df_filtered_director.explode('director')
#director_appearances = directors_df.groupby('director').size().
 ↪reset_index(name='films_directed')
top_directors = director_appearances.sort_values(by='films_directed',␣
 ↪ascending=False).head(10)
```

```
plt.figure(figsize=(8,6))
sns.
 ↪barplot(data=top_directors,x='director',y='films_directed',color='Cyan',width=0.
 ↪6)
plt.title('Top 10 directors by number of films directed')
plt.xlabel('Director')
plt.ylabel('Number of films directed')
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to prevent cropping
plt.show()
```



[ ]:
```
Total_appearances_director = dict(zip(director_appearances['director'],␣
 ↪director_appearances['films_directed']))
wordcloud = WordCloud(width=800, height=400, background_color='Yellow').
 ↪generate_from_frequencies(Total_appearances_director)
plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of directors')
plt.show()
```



1. Rajiv Chilaka has directed most of the films that are added in netflix which also indicates that netflix is proritizing adding kids content in the platform

0.6 5. Which genre movies are more popular or produced more

[ ]:
```
genre= df['listed_in']
genre=genre.str.lower()
genre_count= Counter([genre.strip() for sublist in genre.str.split(",") for␣
 ↪genre in sublist])
genre_count_data= pd.DataFrame(list(genre_count.items()), columns=['genre',␣
 ↪'count'])
genre_count_data.sort_values(by='count',ascending=False).head(10)
```

[ ]:

	genre	count
14	international movies	2752
12	dramas	2427
16	comedies	1674
1	international tv shows	1351
0	documentaries	869
25	action & adventure	859
2	tv dramas	763
13	independent movies	756
11	children & family movies	641
19	romantic movies	616

Data Visualization

```
[ ]: genre_count_data = dict(zip(genre_count_data['genre'],␣
      ↪genre_count_data['count']))
     wordcloud = WordCloud(width=800, height=400, background_color='white').
      ↪generate_from_frequencies(genre_count_data)
     plt.figure(figsize=(10,8))
     plt.imshow(wordcloud, interpolation='bilinear')
     plt.axis('off')
     plt.show()
```

1. As per the given dataset the genre of 'international Movies' are more popular or more produced succeded by 'Drama' genre.
2. It shows that netflix is having a higher global reach as the international movies are resonating with the viewers beyond their country
3. this may also be due to the covid pandamic effect where people started to view more diversifying content beyond their country of origin
4. it also indicates the possibble market expansion in different countries by providing content at a global level by acquiring content from the local
5. it also indicates the surge in the exploration of viewers in the different cultures

0.7 6. Find After how many days the movie will be added to Netflix after the release of the movie (you can consider the recent past data)

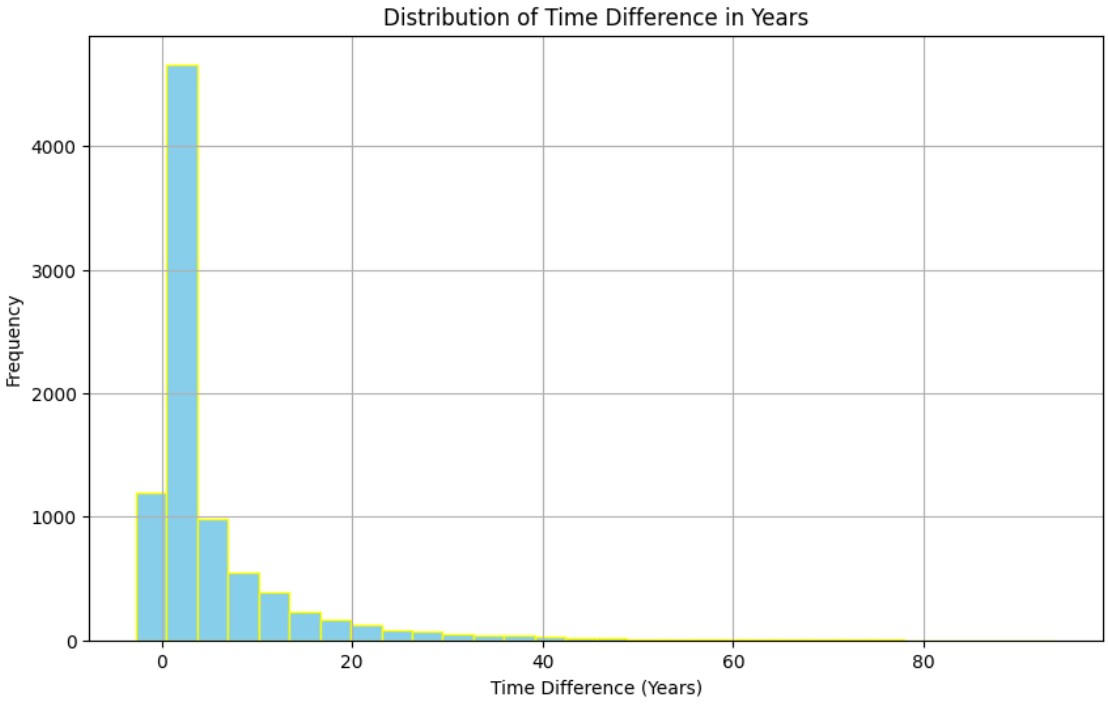
```
[ ]: df_r=df
     df_r['date_added'] = df_r['date_added'].replace('unknown date_added', pd.NaT)
     df_r['date_added'] = pd.to_datetime(df_r['date_added'], format='%B %d, %Y',␣
      ↪errors='coerce')
     df_r['release_year'] = df_r['release_year'].replace('unknown release_year', pd.
      ↪NaT)
```

```
df_r['release_year'] = pd.to_datetime(df_r['release_year'], format='%Y',␣
 ↪errors='coerce')
df_r['Released_Time'] = (df_r['date_added'].dt.year - df_r['release_year'].dt.
 ↪year)
df_r['time_difference_years'] = (df_r['date_added'] - df_r['release_year'])/pd.
 ↪Timedelta(days=365)
df_count= df_r.groupby('time_difference_years').size()
df_count.sort_values(ascending = False)
```

[ ]:
```
time_difference_years
0.915068     37
1.750685     35
1.495890     35
0.997260     29
0.665753     28
             ..
6.602740      1
6.583562      1
6.553425      1
6.545205      1
94.057534     1
Length: 2699, dtype: int64
```

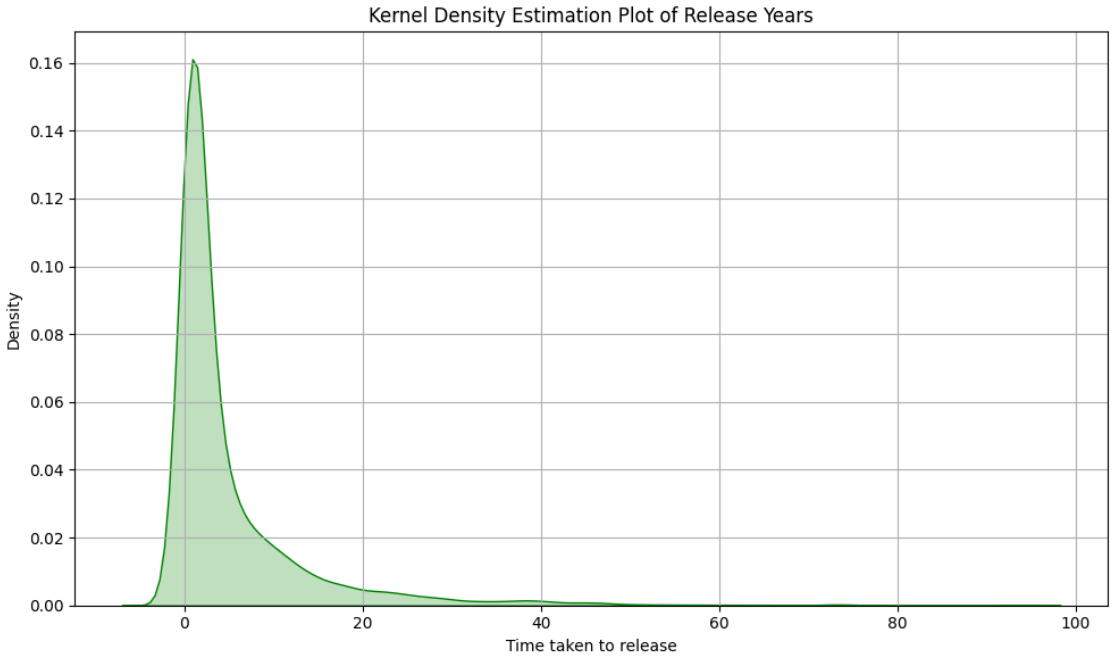
Data Visualization

[ ]:
```
#df_r=df
#df_r['date_added'] = df_r['date_added'].replace('unknown date_added', pd.NaT)
#df_r['date_added'] = pd.to_datetime(df_r['date_added'], format='%B %d, %Y',␣
 ↪errors='coerce')
#df_r['release_year'] = df_r['release_year'].replace('unknown release_year', pd.
 ↪NaT)
#df_r['release_year'] = pd.to_datetime(df_r['release_year'], format='%Y',␣
 ↪errors='coerce')
#df_r['time_difference_years'] = (df_r['date_added'] - df_r['release_year'])/pd.
 ↪Timedelta(days=365)
#df_diff=df_r['time_difference_years']
plt.figure(figsize=(10, 6))
plt.hist(df_r['time_difference_years'], bins=30, color='skyblue',␣
 ↪edgecolor='Yellow')
plt.title('Distribution of Time Difference in Years')
plt.xlabel('Time Difference (Years)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



[ ]:
```
plt.figure(figsize=(10, 6))
sns.kdeplot(df_r['time_difference_years'], color='green', shade=True)
plt.title('Kernel Density Estimation Plot of Release Years')
plt.xlabel('Time taken to release')
plt.ylabel('Density')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
<ipython-input-33-cdcd7a74dd0f>:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df_r['time_difference_years'], color='green', shade=True)
```



1. Most of the movies/shows in Netflix were added within 2 years to 6 months of its release date it suggests that netflix is prioritize adding recently relased films into the platform
2. The quick addition of films to Netflix within 2 years of release could indicate that the platform aims to make popular and recent content available to its subscribers in a timely manner, potentially capitalizing on the hype surrounding newly released films
3. Offering a large selection of recently released films could serve as a competitive advantage for Netflix compared to other streaming platforms. It could attract subscribers who are looking for the latest movies and shows.