In [81]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots

import random

import warnings
warnings.filterwarnings("ignore")

sns.set_style("darkgrid")
```

In [82]:

```
df = pd.read_csv("data exploration.csv")
```

Problem Statement

Which type of shows/movies to produce: Understanding the preferences and trends of viewers to create content that attracts more subscribers and retains existing ones.

Initial Data Exploration

In [83]:

```
df.shape
```

Out[83]:

```
(202010, 16)
```

In [84]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202010 entries, 0 to 202009
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Unnamed: 0    202010 non-null  int64  
 1   show_id       202010 non-null  object 
 2   type          202010 non-null  object 
 3   title         202010 non-null  object 
 4   release_year  202010 non-null  int64  
 5   rating        201943 non-null  object 
 6   duration      202007 non-null  float64
 7   description   202010 non-null  object 
 8   cast          199861 non-null  object 
 9   country       190007 non-null  object 
 10  listed_in     202010 non-null  object 
 11  director      151367 non-null  object 
 12  dayname       201852 non-null  object 
 13  day           201852 non-null  float64
 14  month         201852 non-null  object 
 15  year          201852 non-null  float64
dtypes: float64(3), int64(2), object(11)
memory usage: 24.7+ MB
```

In [85]:
```
df.head(10)
```

Out[85]:

	Unnamed: 0	show_id	type	title	release_year	rating	duration	description	cast	country	listed_in
0	0	s1	Movie	Dick Johnson Is Dead	2020	PG-13	90.0	As her father nears the end of his life, filmm...	NaN	United States	Documentaries
1	1	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Ama Qamata	South Africa	International TV Shows
2	2	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Ama Qamata	South Africa	TV Dramas
3	3	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Ama Qamata	South Africa	TV Mysteries
4	4	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Khosi Ngema	South Africa	International TV Shows
5	5	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Khosi Ngema	South Africa	TV Dramas
6	6	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Khosi Ngema	South Africa	TV Mysteries
7	7	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Gail Mabalane	South Africa	International TV Shows
8	8	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Gail Mabalane	South Africa	TV Dramas
9	9	s2	TV Show	Blood & Water	2021	TV-MA	2.0	After crossing paths at a party, a Cape Town t...	Gail Mabalane	South Africa	TV Mysteries

1. We can convert `date_added` to datetime, then extract yearly, monthly, weekly columns
2. Convert Duration into numerical column.
3. Need to unnest the cast, director, country and listed_in columns.

4. We can drop Description and Title column as they are unique columns.

Let's check how much missing data is present:

In [86]:
```
pd.concat([df.isna().sum(),(df.isna().sum()/len(df))*100], axis = 1)
```

Out[86]:

	0	1
Unnamed: 0	0	0.000000
show_id	0	0.000000
type	0	0.000000
title	0	0.000000
release_year	0	0.000000
rating	67	0.033167
duration	3	0.001485
description	0	0.000000
cast	2149	1.063809
country	12003	5.941785
listed_in	0	0.000000
director	50643	25.069551
dayname	158	0.078214
day	158	0.078214
month	158	0.078214
year	158	0.078214

We can see almost 30% of director data and approx 10% of both cast and country are missing, Except the above mentioned columns date_added, duration and rating has some missing values but they don't amount to much

Let's check if any row is duplicated?

In [87]:
```
df.duplicated().sum()
```

Out[87]:
```
0
```

Let's check some statistical data

In [88]:

```
df.describe()
```

Out[88]:

	Unnamed: 0	release_year	duration	day	year
count	202010.000000	202010.000000	202007.000000	201852.000000	201852.000000
mean	101004.500000	2013.448334	77.678877	12.181579	2018.965425
std	58315.408277	9.013446	51.486115	9.847270	1.551863
min	0.000000	1925.000000	1.000000	1.000000	2008.000000
25%	50502.250000	2012.000000	4.000000	1.000000	2018.000000
50%	101004.500000	2016.000000	95.000000	12.000000	2019.000000
75%	151506.750000	2019.000000	112.000000	20.000000	2020.000000
max	202009.000000	2021.000000	312.000000	31.000000	2021.000000

- Min value of `release_year` is 1925, so some TV Shows or Movies are present that are almost 95 years old
- Only 25% of records that are present in this dataset were released before 2013. So,we have a lot of data that were released in the past decade

In [89]:

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

Out[89]:

	show_id	type	title	rating	description	cast	country	listed_in	director	dayname	month
count	202010	202010	202010	201943	202010	199861	190007	202010	151367	201852	201852
unique	8807	2	8807	17	8775	36439	122	42	4993	7	12
top	s7165	Movie	Kahlil Gibran's The Prophet	TV-MA	A troubled young girl and her mother find sola...	Liam Neeson	United States	Dramas	Martin Scorsese	Friday	July
freq	700	145862	700	73867	700	161	59325	29787	419	57980	20302

- Rajiv Chilaka is has directed most Movies or TV Shows
- Most of the TV Shows or Movies were available in United States
- David Attenborough has worked in most Movies or TV Shows
- Even this particular `"Paranormal activity at a lush...."` description has been repeated four times in Movies/TV Shows. It can suspected that other descrptions are also be repeated
- One thing to Note as we have not yet **unnested the data** these above basic insights might not hold true

In [90]:
```
df.loc[df.duplicated('description',keep = False)].sort_values('description')
```

Out[90]:

	Unnamed: 0	show_id	type	title	release_year	rating	duration	description	cast	country
11423	11423	s471	Movie	Bridgerton - The Afterparty	2021	TV-14	39.0	"Bridgerton" cast members share behind-the-sce...	Fortune Feimster	NaN
11421	11421	s471	Movie	Bridgerton - The Afterparty	2021	TV-14	39.0	"Bridgerton" cast members share behind-the-sce...	David Spade	NaN
11422	11422	s471	Movie	Bridgerton - The Afterparty	2021	TV-14	39.0	"Bridgerton" cast members share behind-the-	London Hughes	NaN

Description column helped to find the repeated Movies/TV Shows or the Movies/TV Shows that were released in other languages

Unnesting the Columns

In [91]:
```
final_df = df.copy()
```

In [92]:
```
def remove_spaces(x):
    if x != x:
        return np.nan
    return x.strip()

def unnesting (new_df,col):

    dataframe =new_df.copy()
    dataframe[col] = dataframe[col].str.split(',')
    dataframe = dataframe.explode(col)
    dataframe[col] = dataframe[col].apply(remove_spaces)
    return dataframe
```

In [93]:
```
%%time
final_df = unnesting(df,'cast')
print('After splitting cast into muliple rows', final_df.shape)
final_df = unnesting(final_df,'country')
print('After splitting country into muliple rows', final_df.shape)
final_df = unnesting(final_df,'listed_in')
print('After splitting listed_in into muliple rows', final_df.shape)
final_df = unnesting(final_df,'director')
print('After splitting listed_in into muliple rows', final_df.shape)


final_df = final_df.reset_index(drop = True)
```

```
After splitting cast into muliple rows (202010, 16)
After splitting country into muliple rows (202010, 16)
After splitting listed_in into muliple rows (202010, 16)
After splitting listed_in into muliple rows (202010, 16)
CPU times: total: 1.52 s
Wall time: 1.72 s
```

Handling Missing Data

In [94]:

```
pd.concat([final_df.isna().sum(),(final_df.isna().sum()/len(final_df))*100], axis = 1)
```

Out[94]:

	0	1
Unnamed: 0	0	0.000000
show_id	0	0.000000
type	0	0.000000
title	0	0.000000
release_year	0	0.000000
rating	67	0.033167
duration	3	0.001485
description	0	0.000000
cast	2149	1.063809
country	12003	5.941785
listed_in	0	0.000000
director	50643	25.069551
dayname	158	0.078214
day	158	0.078214
month	158	0.078214
year	158	0.078214

In [95]:

```
#Smart Imputations is done here
# mode of country grouped by director is imputed for missing values in country
# director_country = (final_df.groupby('director')['country'].\
#                      agg(lambda x: x.mode()[0] if len(x.mode()) > 1 else x.mode())).to_dict()

# # final_df['country1'] = final_df.apply(lambda x: director_country.get(x['director']) if x['c
# final_df['country'] = final_df['country'].fillna(final_df['director'].map(director_country))
```

In [97]:

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202010 entries, 0 to 202009
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Unnamed: 0    202010 non-null  int64  
 1   show_id       202010 non-null  object 
 2   type          202010 non-null  object 
 3   title         202010 non-null  object 
 4   release_year  202010 non-null  int64  
 5   rating        202010 non-null  object 
 6   duration      202010 non-null  float64
 7   description   202010 non-null  object 
 8   cast          202010 non-null  object 
 9   country       202010 non-null  object 
 10  listed_in     202010 non-null  object 
 11  director      202010 non-null  object 
 12  dayname       201852 non-null  object 
 13  day           201852 non-null  float64
 14  month         201852 non-null  object 
 15  year          201852 non-null  float64
dtypes: float64(3), int64(2), object(11)
memory usage: 24.7+ MB
```

In [96]:

```
final_df['country']=final_df['country'].fillna('Unknown Country')
final_df['cast']=final_df['cast'].fillna('Unknown Actor')
final_df['director'] = final_df['director'].fillna('Unknown Director')
final_df['listed_in']  = final_df['listed_in'].fillna('Unknown Genre')
final_df['rating'] = final_df['rating'].fillna('Unknown Rating')
final_df['duration'] = final_df['duration'].fillna(0)
final_df['date_added'] = final_df['date_added'].fillna(final_df['date_added'].mode()[0])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\envs\new\Lib\site-packages\pandas\core\indexes\base.py:3791, in Index.get_lo
c(self, key)
   3790 try:
-> 3791     return self._engine.get_loc(casted_key)
   3792 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

KeyError: 'date_added'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[96], line 7
      5 final_df['rating'] = final_df['rating'].fillna('Unknown Rating')
      6 final_df['duration'] = final_df['duration'].fillna(0)
----> 7 final_df['date_added'] = final_df['date_added'].fillna(final_df['date_added'].mode()
[0])

File ~\anaconda3\envs\new\Lib\site-packages\pandas\core\frame.py:3893, in DataFrame.__getitem
__(self, key)
   3891 if self.columns.nlevels > 1:
   3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
   3894 if is_integer(indexer):
   3895     indexer = [indexer]

File ~\anaconda3\envs\new\Lib\site-packages\pandas\core\indexes\base.py:3798, in Index.get_lo
c(self, key)
   3793     if isinstance(casted_key, slice) or (
   3794         isinstance(casted_key, abc.Iterable)
   3795         and any(isinstance(x, slice) for x in casted_key)
   3796     ):
   3797         raise InvalidIndexError(key)
-> 3798     raise KeyError(key) from err
   3799 except TypeError:
   3800     # If we have a listlike key, _check_indexing_error will raise
   3801     #  InvalidIndexError. Otherwise we fall through and re-raise
   3802     #  the TypeError.
   3803     self._check_indexing_error(key)

KeyError: 'date_added'
```

Feature Engineering

Converted Date added to DateTime column and extracted dayname, day, month, year and week of the year

In [ ]:

```
final_df['date_added'] = pd.to_datetime(final_df['date_added'].apply(lambda x: str(x).strip()))
final_df['dayname'] = final_df['date_added'].dt.day_name()
final_df['day'] = final_df['date_added'].dt.day
final_df['month'] = final_df['date_added'].dt.month_name()
final_df['year'] = final_df['date_added'].dt.year
final_df['week'] = final_df['date_added'].dt.isocalendar().week
final_df['year_diff'] = final_df['year'] - final_df['release_year']
final_df.drop(columns=['date_added'],inplace = True)
```

In [ ]:

```
final_df.columns
```

In [ ]:

```
def release_year_bins(x):
    if x <= 1960:
        return '<1960'
    elif x>1960 and x <= 1970:
        return '60s'
    elif x>1970 and x <= 1980:
        return '70s'
    elif x>1980 and x <= 1990:
        return '80s'
    else:
        return x

def days_bins(x):
    if x>=1 and x<=7:
        return '1st week'
    elif x>7 and x<=14:
        return '2nd week'
    elif x>14 and x<= 21:
        return '3rd week'
    else:
        return '4th week'
```

In [ ]:

```
final_df['release_year_bins'] = final_df['release_year'].apply(release_year_bins)
final_df['days_bins'] = final_df['day'].apply(days_bins)
```

Converted Duration column from object to numerical column

In [ ]:

```
#converting the duration from object type to float
final_df['duration'] = final_df['duration'].str.split(' ',expand = True)[0].astype('float')
```

Statistical Summary in unnested data:

In [ ]:

```
final_df.describe()
```

In [ ]:

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

Here we cannot derive much inferences as due to nesting many records are

Non-Graphical Analysis: Value counts and unique attributes

In [ ]:

```
# this function is to bold python output
def bold_text(text):
    bold_start = '\033[1m'
    bold_end = '\033[0m'
    return bold_start + text + bold_end
```

In [ ]:

```
cols_list = ['type','director','cast','country','release_year','rating','duration','listed_in']
```

Value counts and unique attributes in original data

In [ ]:

```
for i in cols_list:
    print(bold_text(i.upper()+':'))
    print(f'Number of unique elements in {i} is:\n {df[i].nunique()}\n')
    print(f'Unique elements present in {i} column is:\n {df[i].unique()}\n')
    print(f'Value Counts of {i} columns is:\n{df[i].value_counts()}\n\n\n')
```

Value counts and unique attributes in unnested data

In [ ]:

```
cols_list = ['type','rating','director','cast','country','listed_in','release_year_bins','year'
```

In [ ]:

```
for i in cols_list:
    print(bold_text(i.upper()+':'))
    print(f'Number of unique elements in {i} is:\n {final_df[i].nunique()}\n')
    print(f'Unique elements present in {i} column is:\n {final_df[i].unique()}\n')
    print(f'Value Counts of {i} columns is:\n{final_df[i].value_counts()}\n\n\n')
```

Replacing values in Listed in

In [ ]:

```
values = {
    'Dramas':'Drama','Comedies':'Comedy','TV Dramas':'Drama','TV Comedies':'Comedy',
    'Romantic Movies':'Romantic','Romantic TV Shows':'Romantic',
    'Crime TV Shows':'Crime','Horror Movies':'Horror',"Kids' TV":'Kids','Children & Family Movi
    'International Movies':'International','International TV Shows':'International',
    'Independent Movies':'Movies',
    'Music & Musicals':'Music','Anime Series':'Anime','TV Action & Adventure':'Action & Adventu
    'Spanish-Language TV Shows':'Spanish','British TV Shows':'British','Sports Movies':'Sports'
    'TV Mysteries':'Mystery','Korean TV Shows':'Korean','Cult Movies':'Cult','TV Sci-Fi & Fanta
    'Anime Features':'Anime','TV Horror':'Horror','Docuseries':'Documentaries','TV Thrillers':'
    'Reality TV':'Reality','Stand-Up Comedy':'Comedy','Stand-Up Comedy & Talk Shows':'Comedy',

}
final_df['listed_in'] = final_df['listed_in'].replace(values)
```

In [ ]:

```
final_df['listed_in'].nunique()
```

Dividing the dataset into two categories Movies and Shows

```
In [ ]: movies = final_df[final_df['type'] =='Movie']
        shows = final_df[final_df['type'] == 'TV Show']
```

```
In [ ]: cols_list = ['type','rating','director','cast','country','listed_in','release_year_bins','year'
```

```
In [ ]: for i in cols_list:
            print(bold_text(i.upper()+':'))
            print(f'Number of unique elements in {i} is:\n {movies[i].nunique()}\n')
            print(f'Unique elements present in {i} column is:\n {movies[i].unique()}\n')
            print(f'Value Counts of {i} columns is:\n{movies[i].value_counts()}\n\n\n')
```

```
In [ ]: for i in cols_list:
            print(bold_text(i.upper()+':'))
            print(f'Number of unique elements in {i} is:\n {shows[i].nunique()}\n')
            print(f'Unique elements present in {i} column is:\n {shows[i].unique()}\n')
            print(f'Value Counts of {i} columns is:\n{shows[i].value_counts()}\n\n\n')
```

```
In [ ]: print("Number of directors that directed both movies and shows are:",\
        len(set(movies['director'].unique()).intersection(shows['director'].unique())) )
```

```
In [ ]: print("Number of cast members that worked in both movies and shows are:",\
              len(set(movies['cast'].unique()).intersection(shows['cast'].unique())) )
```

Insights from Non Graphical Analysis:

Type:

There are Only Two types of Show -> Movies and TV Shows

b. Out of 8807 shows 6131 shows are Movies and 2676 shows are TV Shows

Rating:

a. There were a total of 17 ratings present for movies. Only 9 of which are ratings used in TV Shows

Director:

a. There were a total of 4528 directors in original dataset

b. There are a total of 4993 directors in the unnested dataset. Out of which 4777 directors worked in movies and only 299 directors worked in TV shows. Also, 84 directors directed both in Movies and TV Shows.

Cast:

a. There were a total of 7692 actors in original dataset

b. There are a total of 36439 casted actors/actress present in the unnested dataset. Out of which 25951 worked in movies and 14863 worked in TV Shows. Only 4376 worked both in Movies and TV Shows.

Country:

a. There were a total of 748 different values of clubbe country in original dataset

b. There are a total of 123 countries where these shows were available. Movies were accessible in 118 different countries and only 66 countries for TV Shows

Genre/Listed_in:

a. There are a total of 28 genres values of present in the dataset. Out of which 18 belong to Movies and 21 belong the TV shows

b. There are a total of 123 countries where these shows were available

c. Drama and International Genres have the highest number of movies and TV Shows.

Years:

a. These movies/TV Shows were released in 74 different years starting from 1925. First TV Shows that was realeased in the dataset was in year 1925 and Movie was in year 1942.

b. 75% of movies were released in the last decade and 75% of Shows were released in last 7 years.

c. Only from 2008 these tv shows/movies were added in the company. Most of the tv shows/movies were added in July following by December

Most of the tv shows/movies were released in Friday followed by Thursday

Visual Analysis - Univariate, Bivariate after pre-processing of the data

In [ ]:

```
plt.figure(figsize =(15,5))

plt.subplot(1,2,1)
movies[['show_id','rating']].drop_duplicates(keep = 'first')['rating'].value_counts().plot(kinc
plt.title('Frequency of Rating in movies')
plt.grid()


plt.subplot(1,2,2)
shows[['show_id','rating']].drop_duplicates(keep = 'first')['rating'].value_counts().plot(kind
plt.title('Frequency of Rating in TV Shows')
plt.grid()

plt.show()
```

In [ ]:

```
mrating_others = ['NR', 'G', 'TV-Y7-FV', 'NC-17', 'UR', 'Unknown Rating',
       '74 min', '84 min', '66 min']
srating_others = ['NR', 'R','TV-G',
       'Unknown Rating', 'TV-Y7-FV']

movies['rating_new'] = movies.rating.apply(lambda x: 'others' if x in mrating_others else x)
shows['rating_new'] = shows.rating.apply(lambda x: 'others' if x in srating_others else x)
```

In [ ]:

```
shows[['show_id','rating_new']].drop_duplicates(keep = 'first')['rating_new'].value_counts().ir
```

In [ ]:

```
plt.figure(figsize =(20,10))

plt.subplot(1,2,1)
mpie = movies[['show_id','rating_new']].drop_duplicates(keep = 'first')['rating_new'].value_cou
plt.pie(mpie, labels= mpie.index, autopct='%.0f%%')
plt.title('Frequency of Rating in movies')


plt.subplot(1,2,2)
tpie = shows[['show_id','rating_new']].drop_duplicates(keep = 'first')['rating_new'].value_cour
plt.pie(tpie, labels= tpie.index, autopct='%.0f%%')
plt.title('Frequency of Rating in TV Shows')

plt.show()
```

Inferences from Rating:

a. Netlix caters to a lot of Mature audience, 34% of movies and 48% of tv shows that are avaiable content is for mature

b. 23% and 27% movies and tv shows rated respectively as TV-14 i.e. children under age of 14 are not suitable to watch, target audience been mid and late teens

c. There are around 13% R Rated movies.

d.There are only 4% movies and 14% of TV Shows available for kids(TV-Y and TV-Y7)

In [ ]:

```
label = ['less than 1hr', 'between 1hr and 2hr','between 2hr and 3hr','greater than 3hr']
movies_duration = movies.drop_duplicates(subset=['show_id','duration'], keep='first')['duratior
(pd.cut(movies.drop_duplicates(subset=['show_id','duration'], keep='first')['duration'],
               bins=[1,60,120,180,1000],
               labels = label
).value_counts()/len(movies_duration))*100
```

In [ ]:

```
shows_duration = shows[['show_id','duration']].drop_duplicates(keep = 'first')['duration']
shows_duration.value_counts()#/len(shows_duration)*100
```

In [ ]:

```
#binning duration of movies
label = ['less than 1hr', 'between 1hr and 2hr','between 2hr and 3hr','greater than 3hr']
movies_duration = movies.drop_duplicates(subset=['show_id','duration'], keep='first')['duratior
(pd.cut(movies.drop_duplicates(subset=['show_id','duration'], keep='first')['duration'],
               bins=[1,60,120,180,1000],
               labels = label
).value_counts()/len(movies_duration))*100

plt.figure(figsize =(10,5))
plt.subplot(1,2,1)
label = ['less than 1hr', 'between 1hr and 2hr','between 2hr and 3hr','greater than 3hr']
plt.title('Frequency of Duration of movies')
pd.cut(movies.drop_duplicates(subset=['show_id','duration'], keep='first')['duration'],
               bins=[1,60,120,180,1000],
               labels = label
).value_counts(ascending = True).plot(kind = 'barh')
plt.grid()

plt.subplot(1,2,2)
shows[['show_id','duration']].drop_duplicates(keep = 'first')['duration'].value_counts(ascendir
plt.title('Frequency of Seasons of TV Shows')
plt.grid()

plt.show()
```

Inferences for Duration:

a. 4499(~73%) movies are between 1hr and 2hr. 1095 Movies are between 2hr and 3hr.

b. 487 movies are less than 1hr. Only 47 movies are greater than 3hr.

c. TV Shows are mostly of only one season around 65%. There's one such TV Show which has 17 seasons.

d. There are only 26 such TV shows which have more than 8 seasons

In [ ]:

```
plt.figure(figsize = (20,7))

plt.subplot(1,2,1)
mask = movies['director'] == 'Unknown Director'
movies_director= movies.loc[~mask,['show_id','director']].drop_duplicates(keep = 'first')['dire
sns.barplot(x = movies_director, y = movies_director.index )
plt.title('Directors that produce highest number of movies')
plt.ylabel('')
plt.xlabel('')



plt.subplot(1,2,2)
mask = shows['director'] == 'Unknown Director'
shows_director= shows.loc[~mask,['show_id','director']].drop_duplicates(keep = 'first')['direct
sns.barplot(x = shows_director, y = shows_director.index )
plt.title('Directors that produce highest number of shows')
plt.ylabel('')
plt.xlabel('')

plt.show()
```

Inferences for Directors:

a. Rajiv Chilaka directed highest number of movies.
b. Alaistar Fothergill directed highest number of TV Shows.

In [ ]:

```
plt.figure(figsize =(20,8))

plt.subplot(1,2,1)
mask = movies['cast'] == 'Unknown Actor'
casts = movies.loc[~mask,['show_id','cast']].drop_duplicates(keep = 'first')['cast'].value_cour
sns.barplot(x=casts,y = casts.index)
plt.title('Actors who have worked in most movies')
plt.ylabel('')
plt.xlabel('')

plt.subplot(1,2,2)
mask = shows['cast'] == 'Unknown Actor'
casts = shows.loc[~mask,['show_id','cast']].drop_duplicates(keep = 'first')['cast'].value_count
sns.barplot(x=casts,y = casts.index)
plt.title('Actors who have worked in most TV shows')
plt.ylabel('')
plt.xlabel('')

plt.show()
```

Inferences from Cast:

a. Anupam Kher has appeared in most of movies.
b. Takahiko Sakurai has apperead in most of TV Shows.

In [ ]:

```
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
mask = movies['country'] == 'Unknown Country'
movies.loc[~mask,['show_id','country']].drop_duplicates(keep = 'first')['country'].value_counts
plt.title('Highest Number of movies released')


plt.subplot(1,2,2)
mask = shows['country'] == 'Unknown Country'
shows.loc[~mask,['show_id','country']].drop_duplicates(keep = 'first')['country'].value_counts(
plt.title('Highest Number of shows released')

plt.show()
```

Inferences from Country:

a. Highest number of movies were released in United States Followed by India and Uk.

b. Highest number of TV Shows were released in United States followed by UK and Japan.

In [ ]:

```
plt.figure(figsize =(15,5))
plt.subplot(1,2,1)
movies[['show_id','listed_in']].drop_duplicates(keep = 'first')['listed_in'].value_counts().hea
plt.title('Highest Number of movies released per Genre')
plt.ylabel('')
plt.xlabel('')


plt.subplot(1,2,2)
shows[['show_id','listed_in']].drop_duplicates(keep = 'first')['listed_in'].value_counts().head
plt.title('Highest Number of shows released per Genre')
plt.ylabel('')
plt.xlabel('')

plt.show()
```

Observations from Genres:

a. Highest Number of Movies/TV Shows are from International Movies, Dramas and Comedy Shows.

In [ ]:

```
plt.figure(figsize =(10,5))

plt.subplot(1,2,1)
day_name = movies[['show_id','dayname']].drop_duplicates(keep = 'first')['dayname'].value_count
plt.pie(day_name, labels= day_name.index, autopct='%.0f%%')
plt.title('Shows released frequencies across the week')

plt.subplot(1,2,2)
day_name = shows[['show_id','dayname']].drop_duplicates(keep = 'first')['dayname'].value_counts
plt.pie(day_name, labels= day_name.index, autopct='%.0f%%')
plt.title('Shows released frequencines across the week')
plt.show()
```

```
In [ ]: plt.figure(figsize =(10,5))

        plt.subplot(1,2,1)
        month_name = shows[['show_id','month']].drop_duplicates(keep = 'first')['month'].value_counts(a
        plt.pie(month_name, labels= month_name.index, autopct='%.0f%%')
        plt.title('Shows released frequencies across the month of Year')

        plt.subplot(1,2,2)
        month_name = shows[['show_id','month']].drop_duplicates(keep = 'first')['month'].value_counts(a
        plt.pie(month_name, labels= month_name.index, autopct='%.0f%%')
        plt.title('Shows released frequencies across the month of Year')

        plt.show()
```

Observations:

a. Most of the TV Shows/Movies are added in December or July

```
In [ ]: plt.figure(figsize =(10,5))

        plt.subplot(1,2,1)
        days = movies[['show_id','day']].drop_duplicates(keep = 'first')['day']
        sns.histplot(days,bins = 8)
        plt.title('Movie Frequencies Across the Days of the Month')
        plt.ylabel('')

        plt.subplot(1,2,2)
        days = shows[['show_id','day']].drop_duplicates(keep = 'first')['day']
        sns.histplot(days,bins = 8)
        plt.title('Movie Releases Across the Day of the Month')
        plt.ylabel('')
        plt.show()
```

```
In [ ]: plt.figure(figsize =(10,5))

        plt.subplot(1,2,1)
        days = movies[['show_id','days_bins']].drop_duplicates(keep = 'first')['days_bins']
        sns.histplot(days,bins = 8)
        plt.title('Movie Frequencies Across the Days of the Month')
        plt.ylabel('')

        plt.subplot(1,2,2)
        days = shows[['show_id','days_bins']].drop_duplicates(keep = 'first')['days_bins']
        sns.histplot(days,bins = 8)
        plt.title('Movie Releases Across the Day of the Month')
        plt.ylabel('')
        plt.show()
```

Observations:

a. Most of the TV Shows/Movies are added in the first week

```
In [ ]: movies[['listed_in','director']].drop_duplicates(keep = 'first').groupby('listed_in').agg(lambd
In [ ]: mon_list = np.array(['December','July'])
        mon_movies = movies.loc[movies['month'].isin(mon_list),['show_id','day','month']].drop_duplicat
        plt.figure(figsize = (15,5))
        sns.countplot(data = mon_movies,x = 'day',hue = 'month')
        plt.legend(loc='center')
        plt.show()
```

```
In [ ]: plt.figure(figsize =(10,5))

        mon_list = np.array(['December','July'])
        mon_movies = movies.loc[movies['month'].isin(mon_list),['show_id','listed_in','month']].drop_du
        sns.countplot(data = mon_movies,y = 'listed_in',hue = 'month')
        plt.xticks(rotation  = 90)
        plt.show()
```

```
In [ ]: plt.figure(figsize =(10,3))

        plt.subplot(1,2,1)
        days = movies[['show_id','year']].drop_duplicates(keep = 'first')['year']
        sns.histplot(days,bins = 30)
        plt.title('Movie Added Frequencies Across the Years')
        plt.ylabel('')

        plt.subplot(1,2,2)
        days = shows[['show_id','year']].drop_duplicates(keep = 'first')['year']
        sns.histplot(days,bins = 30)
        plt.title('TV Shows Added Frequencies Across the Years')
        plt.ylabel('')
        plt.show()
```

Inferences from Date Added :

a. Most of the TV Shows/Movies are added in December or July

b. Most of the TV Shows/Movies are added in the first week

c. Most of the movies are added in Month of December or July in the first week or last week

d. Most of the movies are added in Month of December or July have genres Dramas International Movies and Comedies

e. Most of the TV Shows are added in Month of December or July in the first week or last week

f. Most of the TV Shows are added in Month of December or July have genres Dramas International Movies and Comedies

g. Range of Year Added in 13 years

```
In [ ]: plt.figure(figsize = (20,7))
        plt.subplot(2,1,1)
        sns.boxplot(data = movies,x= 'release_year')
        plt.title('Release Year Distribution in Movies')
        plt.xlabel('')

        plt.subplot(2,1,2)
        sns.boxplot(data = shows,x = 'release_year')
        plt.title('Release Year Distribution in Shows')
        plt.xlabel('')

        plt.show()
```

```
In [ ]: df[df['type'] == 'Movie'].describe()
```

```
In [ ]: df[df['type'] == 'TV Show'].describe()
```

Inferences from Release Year:

a. Very few movies were released before 2000 that are present in this dataset
b. Very few TV Shows were released before 2010 that are present in this dataset
c. Most of the movies were released between 2012 to 2018 that are present in this dataset
d. Very few TV Shows were released between 2016 to 2020 that are present in this dataset
e. Range of Release Year for Movies is equal to 79 years, for TV Shows it is equal to 96 years

```
In [ ]: plt.figure(figsize = (20,5))
        box = final_df[['show_id','type','year_diff']].drop_duplicates()
        sns.boxplot(data = box,x='year_diff',y  = 'type')
        plt.show()
```

```
In [ ]: plt.figure(figsize = (7,3))
        box = final_df[['show_id','type','year_diff']].drop_duplicates()
        sns.kdeplot(data = box,x='year_diff',hue= 'type')
        plt.show()
```

```
In [ ]: box[box['type'] == 'Movie'].max()
```

```
In [ ]: box[box['type'] == 'TV Show'].max()
```

Inferences from difference between year added and year released:

a. Most of the movies/tv shows were added in the same year as it was released
b. Highest year difference between when it was released and when it was added is 75 and 93 for movies and TV Shows respectively

```
In [ ]: plt.figure(figsize = (7,3))

        movies_released_per_year = df.loc[df['type']=='Movie','release_year'].value_counts().sort_index
        sns.lineplot(x = movies_released_per_year.index,y = movies_released_per_year,label = 'Movies')

        shows_released_per_year = df.loc[df['type']=='TV Show','release_year'].value_counts().sort_inde
        sns.lineplot(x =shows_released_per_year.index,y = shows_released_per_year,label = 'TV Shows')

        plt.xlabel('Release Year')
        plt.ylabel('')
        plt.title('Comparison of Number of Movies and TV Shows released over the years')
        plt.legend(loc = 'center')

        plt.show()
```

In [ ]:

```
plt.figure(figsize = (7,3))

movies_added_per_year = movies.groupby('year')['show_id'].nunique()
sns.lineplot(x = movies_added_per_year.index,y = movies_added_per_year,label = 'Movies')

shows_added_per_year = shows.groupby('year')['show_id'].nunique()
sns.lineplot(x =shows_added_per_year.index,y = shows_added_per_year,label = 'TV Shows')

plt.xlabel('Added Year')
plt.ylabel('')
plt.title('Comparison of Number of Movies and TV Shows added over the years')
plt.legend(loc = 'center')

plt.show()
```

Number of Shows Released Across the Years :

a. In the recent years we can there has been a drop in release as well as drop in addition of Movies and Tv Shows. This maybe due to lack of data. As we donot have data we cannot conclude the above statement as true

In [ ]:

```
plt.figure(figsize = (7,3))

movies_added_per_year = movies.groupby('year')['show_id'].nunique()
sns.lineplot(x = movies_added_per_year.index,y = movies_added_per_year,label = 'Movies')

shows_added_per_year = shows.groupby('year')['show_id'].nunique()
sns.lineplot(x =shows_added_per_year.index,y = shows_added_per_year,label = 'TV Shows')

plt.xlabel('Added Year')
plt.ylabel('')
plt.title('Comparison of Number of Movies and TV Shows added over the years')
plt.legend(loc = 'center')
plt.xlim(2008,2015)
plt.ylim(0,60)

plt.show()
```

Number of Shows Added across the years:

a. There has been spike in addtion of Movies and spike in addtion of TV Shows from 2013 and 2014 respectively.

In [ ]:

```
sns.pairplot(data = movies)
plt.show()
```

In [ ]:

```
sns.pairplot(data = shows)
plt.show()
```

In [ ]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.heatmap(movies[['release_year','duration','day','year','week','year_diff']].corr(),annot =



plt.subplot(1,2,2)
sns.heatmap(shows[['release_year','duration','day','year','week','year_diff']].corr(),annot =
plt.show()
```

In [ ]:

```
plt.figure(figsize=(10,3))

plt.subplot(1,2,1)
corr_mov_data = movies[['release_year','duration','year']].drop_duplicates()
sns.heatmap(corr_mov_data.corr(),annot = True)



plt.subplot(1,2,2)
corr_shows_data = shows[['release_year','duration','year']].drop_duplicates()
sns.heatmap(corr_shows_data.corr(),annot = True)
plt.show()
```

Observations:

a. Except for release_year and year_diff, any clear correlation between any other columns cannot been seen.

In [ ]:

```
mask = movies['country'] == 'Unknown Country'
mov_country_list = movies.loc[~mask,['show_id','country']].drop_duplicates(keep = 'first')['cou

mask = shows['country'] == 'Unknown Country'
show_country_list = shows.loc[~mask,['show_id','country']].drop_duplicates(keep = 'first')['cou


mov_cg = movies[movies['country'].isin(mov_country_list)]
show_cg = shows[shows['country'].isin(show_country_list)]

mov_order = movies[['show_id','listed_in']].drop_duplicates(keep = 'first')['listed_in'].value_
show_order = shows[['show_id','listed_in']].drop_duplicates(keep = 'first')['listed_in'].value_

plt.figure(figsize = (15,20))

plt.subplot(2,1,1)
sns.countplot(data = mov_cg,x = 'listed_in',hue = 'country',order = mov_order,hue_order=mov_cou
plt.ylabel('Genres')
plt.xlabel('')
plt.xticks(rotation = 90)

plt.subplot(2,1,2)
sns.countplot(data = show_cg,x = 'listed_in',hue = 'country',order = show_order,hue_order=show_
plt.ylabel('Genres')
plt.xlabel('')
plt.xticks(rotation = 90)
plt.show()
```

Inferneces from Top 5 Countries and Genres:

a. Most TV shows in United States are of Dramas, Comedy and Kids Genre.

b. Most TV Shows in United Kingdom are of British TV shows, International Shows and Dramas.

c. Most TV shows in Japan are of International Shows and Anime Series.

d. Most TV Shows in South Korea are of International Shows, Korean and Romantic TV Shows.

e. Most Movies in United States are of Dramas and Comedy.

f. Most Movies in United Kingdom are of International Movies, Dramas and Comedy Genre.

g. Most Movies in India are of International Movies, Dramas and Comedy Genre.

h. Most Movies in France are of International Movies and Dramas.

Buisness Insights

Type:

a. There are Only Two types of Show -> Movies and TV Shows

b. Out of 8807 shows 6131 shows are Movies and 2676 shows are TV Shows

Rating:

a. There were a total of 17 ratings present for movies. Only 9 of which are ratings used in TV Shows

b. Netlix caters to a lot of Mature audience, 34% of movies and 48% of tv shows that are avaiable content is for mature

c. 23% and 27% movies and tv shows rated respectively as TV-14 i.e. children under age of 14 are not suitable to watch, target audience been mid and late teens

d. There are around 13% R Rated movies.

e.There are only 4% movies and 14% of TV Shows available for kids(TV-Y and TV-Y7)

Duration:

a. 4499(~73%) movies are between 1hr and 2hr. 1095 Movies are between 2hr and 3hr.

b. 487 movies are less than 1hr. Only 47 movies are greater than 3hr.

c. TV Shows are mostly of only one season around 65%. There's one such TV Show which has 17 seasons.

d. There are only 26 such TV shows which have more than 8 seasons

Director:

a. There were a total of 4528 directors in original dataset

b. There are a total of 4993 directors in the unnested dataset. Out of which 4777 directors worked in movies and only 299 directors worked in TV shows. Only 84 directors worked both in Movies and TV Shows

c. Rajiv Chilaka directed highest number of movies.

d. Alaistar Fothergill directed highest number of TV Shows.

Cast:

a. There were a total of 7692 actors in original dataset

b. There are a total of 36439 casted actors/actress present in the unnested dataset. Out of which 25951 worked in movies and 14863 worked in TV Shows. Only 4376 worked both in Movies and TV Shows

c. Anupam Kher has appeared in most of movies.

d. Takahiko Sakurai has apperead in most of TV Shows.

Country:

a. There were a total of 748 different values of clubbed country in original dataset

b. There are a total of 123 countries where these shows were available. Movies were accessible in 118 different countries and 66 countries for TV Shows

c. Highest number of movies were released in United States Followed by India and UK.

d. Highest number of TV Shows were released in United States followed by UK and Japan.

Genre/Listed_in:

a. There are a total of 28 genres values of present in the dataset. Out of which 18 belong to Movies and 21 belong the TV shows

b. There are a total of 123 countries where these shows were available

c. Drama and International Genres have the highest number of movies and TV Shows.

Years:

a. These movies/TV Shows were released in 74 different years starting from 1925. First TV Shows that was realeased in the dataset was in year 1925 and Movie was in year 1942.

b. 75% of movies were released in the last decade and 75% of Shows were released in last 7 years.

c. Only from 2008 these tv shows/movies were added. Most of the tv shows/movies were added in July following by December

d. Most of the tv shows/movies were released in Friday followed by Thursday

e. Most of the TV Shows/Movies are added in December or July

f. Most of the TV Shows/Movies are added in the first week

g. Most of the movies are added in Month of December or July in the first week or last week

h. Most of the movies are added in Month of December or July have genres Dramas International Movies and Comedies

i. Most of the TV Shows are added in Month of December or July in the first week or last week

j. Most of the TV Shows are added in Month of December or July have genres Dramas International Movies and Comedies

k. Range of Year Added in 13 years

l. Very few movies were released before 2000 that are present in this dataset

m. Very few TV Shows were released before 2010 that are present in this dataset

n. Most of the movies were released between 2012 to 2018 that are present in this dataset

o. Very few TV Shows were released between 2016 to 2020 that are present in this dataset

p. Range of Release Year for Movies is equal to 79 years, for TV Shows it is equal to 96 years

a. Most TV shows in United States are of Dramas, Comedy and Kids Genre.

b. Most TV Shows in United Kingdom are of British TV shows, International Shows and Dramas.

c. Most TV shows in Japan are of International Shows and Anime Series.

d. Most TV Shows in South Korea are of International Shows, Korean TV shows and Romantic TV Shows.

e. Most Movies in United States are of Dramas, Comedy and Children & Family Genre.

f. Most Movies in United Kingdom are of International Movies, Dramas and Comedy Genre.

g. Most Movies in India are of International Movies, Dramas and Comedy Genre.

h. Most Movies in France are of International Movies and Dramas.

Inferences from Top 5 Countries and Genres:

a. Most TV shows in United States are of Dramas, Comedy and Kids Genre.

b. Most TV Shows in United Kingdom are of British TV shows, International Shows and Dramas.

c. Most TV shows in Japan are of International Shows and Anime Series.

d. Most TV Shows in South Korea are of International Shows, Korean TV shows and Romantic TV Shows.

e. Most Movies in United States are of Dramas, Comedy and Children & Family Genre.

f. Most Movies in United Kingdom are of International Movies, Dramas and Comedy Genre.

g. Most Movies in India are of International Movies, Dramas and Comedy Genre.

h. Most Movies in France are of International Movies and Dramas.

Other Inferences:

a. Most of the movies/tv shows were added in the same year as it was released

b. Highest year difference between when it was released and when it was added is 75 and 93 for movies and TV Shows respectively

c. In the recent years we can there has been a drop in release as well as drop in addition of Movies and Tv Shows.

d. There has been spike in addtion of Movies and spike in addtion of TV Shows from 2013 and 2014 respectively

Recommendations

1. Most of the shows are catered to mature audiences. Diversifying content genres is also important to attract a broader range of viewers. A mix of genres, including drama, comedy, action, romance, and documentary, to cater to varied tastes.
2. Given the popularity of TV-14 rated content, more shows and movies should be tailored for the late teens demographic.
3. Can Experiment with other genres like Sci-Fi, Fantasy, Thriller, and Documentaries.
4. Due to kids less attention span, shows of length 15-20 mins should be available more. Side by Side it is also very important to implement a robust parental control and ensure that the content is suitable for this age group
5. Focus on producing movies that fall within the popular 1-hour to 2-hour duration range.
6. A strategic approach is to develop TV shows spanning 3-5 seasons, with each season having a compelling cliffhanger. This will captivate viewers interest and anticipation, making them to eagerly await for the next season.
7. Additionally we can create brief glimpses of behind the screens or share entertaining bloopers, providing a relatable and authentic connection to our audience.
8. Some of the most old movies that are not present can be added, that were released before 2010, which will help to cater the elderly audience, creating a feeling of nostalgia. It will work especially well in a country like Japan due its higher older demographic.
9. The trend of adding most TV shows and movies in Friday and Thursday in the first and last week of Decemeber and July can be leveraged. The release of highly anticipated original content can be done during these months to attract maximum viewership.

```
In [ ]:
```