🎥 📺 NETFLIX - *Business Case study* 🎥 🍿

`![nx.png](attachment:a143bc0c-26ae-4fc2-b7b6-3277d54c6ded.png)`

Introduction :

Netflix was founded in 1997 by `Reed Hastings` and `Marc Randolph`. The company started as a DVD-rental service, but it quickly transitioned to streaming in 2007. Netflix has grown rapidly over the past few decades, and it is now one of the most popular streaming services in the world. It is one of the most popular media and video streaming platforms. They have over 10000 movies or tv shows available on their platform, as of mid-2021, they have over 222M Subscribers globally.

A brief overview of Netflix's growth over the years:

- 1997: Netflix is founded.
- 2000: Netflix introduces a subscription model for DVD rentals.
- 2007: Netflix launches its streaming service.
- 2011: Netflix surpasses Blockbuster as the leading DVD-rental company in the United States.
- 2013: Netflix expands to international markets.
- 2016: Netflix releases its first original series, House of Cards.
- 2019: Netflix surpasses 150 million subscribers worldwide.
- 2022: Netflix experiences its first decline in subscribers in the North American market.

As of July 2023, Netflix has a market capitalization of `$196.44 billion dollar`. This makes it the world's 56th most valuable company by market cap.

Buisness Problem (EDA):

> `This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc. Analyze the data and generate insights that could help Netflix in deciding which type of shows/movies to produce and how they can grow the business in different countries.`

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Exploration of Data

In [2]:
```
df = pd.read_csv('netflix.csv')
```

In [3]:
```
df.head(3)
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dur
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	9
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Se
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Se

In [4]:
```
df.tail(4)
```

Out[4]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
8803	s8804	TV Show	Zombie Dumb	NaN	NaN	NaN	July 1, 2019	2018	TV-Y7
8804	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg, Woody Harrelson, Emma Stone, ...	United States	November 1, 2019	2009	R
8805	s8806	Movie	Zoom	Peter Hewitt	Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...	United States	January 11, 2020	2006	PG
8806	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	India	March 2, 2019	2015	TV-14

In [5]:
```
df.columns
```

Out[5]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

In [ ]:
```
df.shape
```

In [6]:
```
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
```

In [7]:
```
df.describe()
```

Out[7]:

	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

In [8]:
```
df.describe(include='object')
```

Out[8]:

	show_id	type	title	director	cast	country	date_added	rating	duration
count	8807	8807	8807	6173	7982	7976	8797	8803	8804
unique	8807	2	8807	4528	7692	748	1767	17	220
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	TV-MA	1 Season
freq	1	6131	1	19	19	2818	109	3207	1793

In [9]:
```
df.describe(include='all')
```

Out[9]:

	show_id	type	title	director	cast	country	date_added	release_year	ratin
count	8807	8807	8807	6173	7982	7976	8797	8807.000000	880
unique	8807	2	8807	4528	7692	748	1767	NaN	1
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	NaN	TV M
freq	1	6131	1	19	19	2818	109	NaN	320
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2014.180198	Na
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.819312	Na
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1925.000000	Na
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2013.000000	Na
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2017.000000	Na
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019.000000	Na
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021.000000	Na

In [10]:
```
df.sample()
```

Out[10]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
7687	s7688	Movie	P	Paul Spurrier	Suangporn Jaturaphut, Opal, Dor Yodrak, Pisama...	United Kingdom, Thailand	May 31, 2019	2006	TV-MA	105

In [11]:
```
df.dtypes
```

Out[11]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

In [12]:
```
df[df.duplicated()]
```

Out[12]:

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_i

🖌️ Data Wrangling

📚 Unnesting the columns (directors , casts , countrys , listed_in)

In [13]:
```
unnesting = ['director', 'cast', 'listed_in','country']
for column in unnesting:
    df[column] = df[column].str.split(', ')
    df = df.explode(column)
```

In [14]:
```
df.shape
```

Out[14]:
```
(201991, 12)
```

In [15]:
```
df.dtypes
```

Out[15]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

In [16]:
```
df.reset_index(drop=True,inplace=True)
```

In [17]:
```
df
```

Out[17]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13
1	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA
2	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA
3	s2	TV Show	Blood & Water	NaN	Ama Qamata	South Africa	September 24, 2021	2021	TV-MA
4	s2	TV Show	Blood & Water	NaN	Khosi Ngema	South Africa	September 24, 2021	2021	TV-MA
...	...	...	...	...	...	...	...	...	...
201986	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	March 2, 2019	2015	TV-14
201987	s8807	Movie	Zubaan	Mozez Singh	Anita Shabdish	India	March 2, 2019	2015	TV-14
201988	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14
201989	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14
201990	s8807	Movie	Zubaan	Mozez Singh	Chittaranjan Tripathy	India	March 2, 2019	2015	TV-14

201991 rows × 12 columns

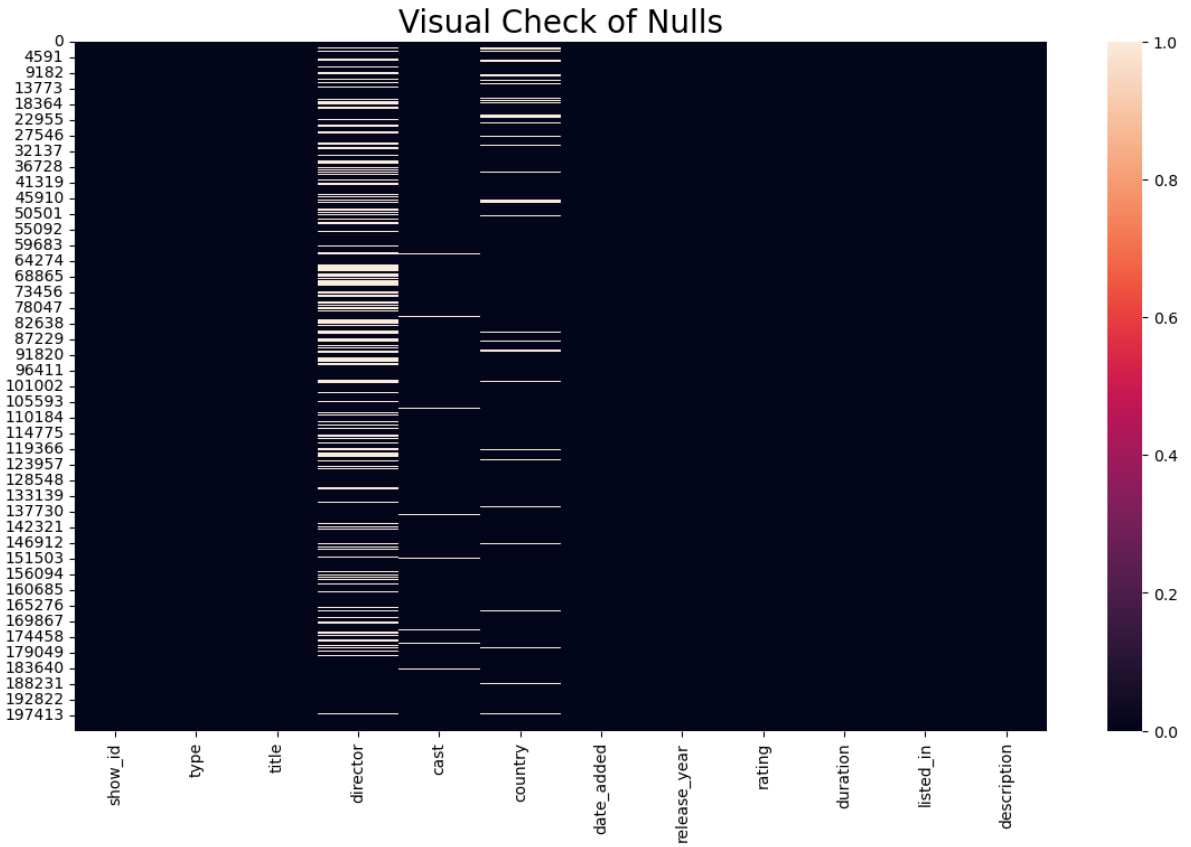
👉Insights :

Note after unnesting the no.of movies and TvShows rows are increased but still we use nunique() instead of counts to get the accurate data...

- Movie - 6131
- TV Show - 2676
- Total - 8807

Treating Nulls

In [18]:
```
plt.figure(figsize=(14,8))
sns.heatmap(df.isnull())
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```



In [19]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[19]:
```
director        50643
country         11897
cast             2146
date_added        158
rating             67
duration            3
show_id             0
type                0
title               0
release_year        0
listed_in           0
description         0
dtype: int64
```

In [20]:
```
for i in df.columns:
    null_pct = (df[i].isna().sum() / df.shape[0]) *100
    if null_pct > 0 :
        print(f'Null_pct of {i} is {round(null_pct,3)} %')
```

```
Null_pct of director is 25.072 %
Null_pct of cast is 1.062 %
Null_pct of country is 5.89 %
Null_pct of date_added is 0.078 %
Null_pct of rating is 0.033 %
Null_pct of duration is 0.001 %
```

In [21]:
```
df[df.date_added.isna()]
```

Out[21]:

	show_id	type	title	director	cast	country	date_added	release_year	rat
136893	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe	United Kingdom	NaN	2013	
136894	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe	United Kingdom	NaN	2013	
136895	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe	United Kingdom	NaN	2013	
136896	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Jon Hamm	United Kingdom	NaN	2013	
136897	s6067	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Jon Hamm	United Kingdom	NaN	2013	
...	...	...	...	...	...	...	...	...	
186891	s8183	TV Show	The Adventures of Figaro Pho	NaN	Charlotte Hamlyn	Australia	NaN	2015	TV
186892	s8183	TV Show	The Adventures of Figaro Pho	NaN	Stavroula Mountzouris	Australia	NaN	2015	TV
186893	s8183	TV Show	The Adventures of Figaro Pho	NaN	Stavroula Mountzouris	Australia	NaN	2015	TV
186894	s8183	TV Show	The Adventures of Figaro Pho	NaN	Aletheia Burney	Australia	NaN	2015	TV
186895	s8183	TV Show	The Adventures of Figaro Pho	NaN	Aletheia Burney	Australia	NaN	2015	TV

158 rows × 12 columns

In [22]:
```
df['date_added'] = pd.to_datetime(df['date_added'] ,format="%B %d, %Y" , errors='co
```

In [23]:
```
df['date_added'].fillna(df['date_added'].mode()[0], inplace=True)
```

In [24]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[24]:
```
director        50643
country         11897
cast             2146
rating             67
duration            3
show_id             0
type                0
title               0
date_added          0
release_year        0
listed_in           0
description         0
dtype: int64
```

In [25]:
```
df.dtypes
```

Out[25]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
dtype: object
```

In [26]:
```
df['year_added'] = df['date_added'].dt.year
```

In [27]:
```
df.sample()
```

Out[27]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	durati
53533	s2265	Movie	Aiyyaa	Sachin Kundalkar	Anita Date	India	2020-07-05	2012	TV-14	151 r

In [28]:
```
df.dtypes
```

Out[28]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
year_added               int32
dtype: object
```

In [29]:
```
df.shape
```

Out[29]:
```
(201991, 13)
```

In [30]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201991 entries, 0 to 201990
Data columns (total 13 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   show_id       201991 non-null  object        
 1   type          201991 non-null  object        
 2   title         201991 non-null  object        
 3   director      151348 non-null  object        
 4   cast          199845 non-null  object        
 5   country       190094 non-null  object        
 6   date_added    201991 non-null  datetime64[ns]
 7   release_year  201991 non-null  int64         
 8   rating        201924 non-null  object        
 9   duration      201988 non-null  object        
 10  listed_in     201991 non-null  object        
 11  description   201991 non-null  object        
 12  year_added    201991 non-null  int32         
dtypes: datetime64[ns](1), int32(1), int64(1), object(10)
memory usage: 19.3+ MB
```

In [31]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[31]:
```
director        50643
country         11897
cast             2146
rating             67
duration            3
show_id             0
type                0
title               0
date_added          0
release_year        0
listed_in           0
description         0
year_added          0
dtype: int64
```

In [32]:
```
df[df.rating.isna() | df.duration.isna()]
```

Out[32]:

	show_id	type	title	director	cast	country	date_added	release_year
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015
135125	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Oprah Winfrey	NaN	2017-01-26	2017
135126	s5990	Movie	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Ava DuVernay	NaN	2017-01-26	2017
...	...	...	...	...	...	...	...	...
171942	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Francesco Migliore	Italy	2017-03-01	2015
171943	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Albrecht Weimer	Italy	2017-03-01	2015
171944	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Giulia Dichiaro	Italy	2017-03-01	2015
171945	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Alessandra Oriti Niosi	Italy	2017-03-01	2015
171946	s7538	Movie	My Honor Was Loyalty	Alessandro Pepe	Andreas Segeritz	Italy	2017-03-01	2015

70 rows × 13 columns

In [33]:
```
df["country"].fillna("Unknown",inplace=True)
df["cast"].fillna("Unknown actors",inplace=True)
df["director"].fillna("Unknown director",inplace=True)
df["rating"].fillna("Unknown",inplace=True)
```

In [34]:
```
df.isna().sum()
```

Out[34]:
```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        3
listed_in       0
description     0
year_added      0
dtype: int64
```

In [35]:
```
df[df.duration.isna()]
```

Out[35]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	74 min	
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	84 min	
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	66 min	

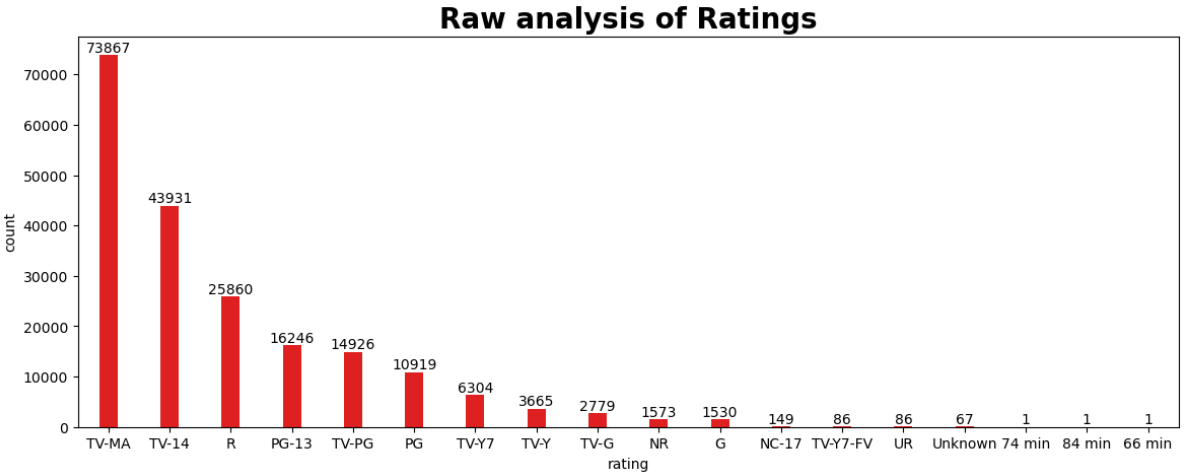
In [36]:
```
df.rating.value_counts()
```

Out[36]:
```
rating
TV-MA       73867
TV-14       43931
R           25860
PG-13       16246
TV-PG       14926
PG          10919
TV-Y7        6304
TV-Y         3665
TV-G         2779
NR           1573
G            1530
NC-17         149
TV-Y7-FV       86
UR             86
Unknown        67
74 min          1
84 min          1
66 min          1
Name: count, dtype: int64
```

In [37]:
```
rvc = df.rating.value_counts(dropna=False).reset_index()
```

In [38]:
```
plt.figure(figsize=(14,5))
a = sns.barplot(rvc , x='rating' , y='count' , color='red' , width=0.3)
plt.title('Raw analysis of Ratings',fontsize=20,fontweight='bold')
```

```
a.bar_label(a.containers[0], label_type='edge')
plt.show()
```



In [39]:
```
df[df.director=='Louis C.K.']
```

Out[39]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	74 min	
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	84 min	
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	66 min	

In [40]:
```
# df['duration'] = df.loc[df.director=='Louis C.K.'].replace(np.nan , df.rating) di
# df['duration'].fillna(df['rating'], inplace=True) # this works .. easy method ..
df.loc[df['director']=='Louis C.K.', 'duration']=df.loc[df['director']=='Louis C.K.
# b4 line end => .fillna(df.loc[df['director'] == 'Louis C.K.', 'rating'])
```

In [41]:
```
df[df.director=='Louis C.K.']
```

Out[41]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	74 min	74
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	84 min	84
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	66 min	66

In [42]:
```
df.loc[df['director'] == 'Louis C.K.', 'rating'] = 'Unknown'
```

In [43]:
```
df[df.director=='Louis C.K.']
```

Out[43]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	d
126537	s5542	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	Unknown	
131603	s5795	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	Unknown	
131737	s5814	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	Unknown	

In [44]:
```
df.shape
```

Out[44]:
```
(201991, 13)
```

In [45]:
```
df.dtypes
```

Out[45]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
year_added               int32
dtype: object
```

In [46]:
```
df.isna().sum()
```

Out[46]:
```
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
year_added      0
dtype: int64
```

📂 we will segregate the data into movie data and tv-shows data and fill the duration appropriately...

In [47]:
```
df.type.value_counts()
```

Out[47]:
```
type
Movie      145843
TV Show     56148
Name: count, dtype: int64
```

In [48]:
```
movies_data = df[df.type=='Movie']
```

In [49]:
```
movies_data.shape
```

Out[49]:
```
(145843, 13)
```

In [50]:
```
tvshows_data = df[df.type=='TV Show']
```

In [51]:
```
tvshows_data.shape
```

Out[51]:
```
(56148, 13)
```

In [52]:
```
movies_data.sample()
```

Out[52]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	du
12243	s497	Movie	Brick Mansions	Camille Delamarre	Carlo Rota	Canada	2021-07-07	2014	PG-13	

In [53]:
```
movies_data.isna().sum()
```

Out[53]:
```
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
year_added      0
dtype: int64
```

In [54]:
```
tvshows_data.sample()
```

Out[54]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duratic
87921	s3687	TV Show	The Last Czars	Unknown director	Robert Jack	United States	2019-07-03	2019	TV-MA	1 Seas

In [55]:
```
tvshows_data.isna().sum()
```

Out[55]:
```
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
year_added      0
dtype: int64
```

In [56]:
```
movies_data['runtime_in_mins'] = movies_data['duration'].str.split(' ').str[0]
tvshows_data['no_of_seasons'] = tvshows_data['duration'].str.split(' ').str[0]
```

In [57]:
```
movies_data.sample()
```

Out[57]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	dura
164345	s7209	Movie	King's Ransom	Jeffrey W. Byrd	Jay Mohr	Canada	2019-11-01	2005	PG-13	98

In [58]:
```
movies_data.dtypes
```

Out[58]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
year_added               int32
runtime_in_mins         object
dtype: object
```

In [59]:
```
movies_data.runtime_in_mins = movies_data.runtime_in_mins.astype(int)
```

In [60]:
```
movies_data.dtypes
```

Out[60]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
year_added               int32
runtime_in_mins          int32
dtype: object
```

In [61]:
```
movies_data = movies_data.drop(columns=['description','duration']).reset_index(drop
```

In [62]:
```
movies_data.shape
```

Out[62]:
```
(145843, 12)
```

In [ ]:
```
# movies data is done
```

In [ ]:
```
# tvshows....
```

In [63]:
```
tvshows_data.tail()
```

Out[63]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
201864	s8801	TV Show	Zindagi Gulzar Hai	Unknown director	Hina Khawaja Bayat	Pakistan	2016-12-15	2012	TV-PG
201865	s8801	TV Show	Zindagi Gulzar Hai	Unknown director	Hina Khawaja Bayat	Pakistan	2016-12-15	2012	TV-PG
201932	s8804	TV Show	Zombie Dumb	Unknown director	Unknown actors	Unknown	2019-07-01	2018	TV-Y7
201933	s8804	TV Show	Zombie Dumb	Unknown director	Unknown actors	Unknown	2019-07-01	2018	TV-Y7
201934	s8804	TV Show	Zombie Dumb	Unknown director	Unknown actors	Unknown	2019-07-01	2018	TV-Y7

In [64]:
```
tvshows_data.no_of_seasons.value_counts()
```

Out[64]:
```
no_of_seasons
1     35035
2      9559
3      5084
4      2134
5      1698
7       843
6       633
8       286
9       257
10      220
13      132
12      111
15       96
17       30
11       30
Name: count, dtype: int64
```

In [65]:
```
tvshows_data.dtypes
```

Out[65]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
description             object
year_added               int32
no_of_seasons           object
dtype: object
```

In [66]:
```
tvshows_data.no_of_seasons = tvshows_data.no_of_seasons.astype(int)
```

In [67]:
```
tvshows_data.no_of_seasons.dtypes
```

Out[67]:
```
dtype('int32')
```

In [68]:
```
tvshows_data = tvshows_data.drop(columns=['description','duration']).reset_index(dr
```

In [69]:
```
tvshows_data.sample(3)
```

Out[69]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
30414	s3675	TV Show	PILI Fantasy: War of Dragons	Unknown director	Zhan Yi-chun	Unknown	2019-07-12	2019	TV-14
48241	s6506	TV Show	Club Friday The Series 7	Unknown director	Oil Thana Suttikamul	Unknown	2018-06-18	2016	TV-MA
16224	s2146	TV Show	Terrace House: Tokyo 2019-2020	Unknown director	You	Japan	2020-08-10	2019	TV-MA

In [70]:
```
df = df.drop(columns=['description']).reset_index(drop=True)
```

In [71]:
```
print(f'Cleaned Netflix data has {df.shape[0]} Rows and {df.shape[1]} Columns')
print(f'Netflix Movies data has {movies_data.shape[0]} Rows and {movies_data.shape[
print(f'Netflix TV shows data has {tvshows_data.shape[0]} Rows and {tvshows_data.sh
```

```
Cleaned Netflix data has 201991 Rows and 12 Columns
Netflix Movies data has 145843 Rows and 12 Columns
Netflix TV shows data has 56148 Rows and 12 Columns
```

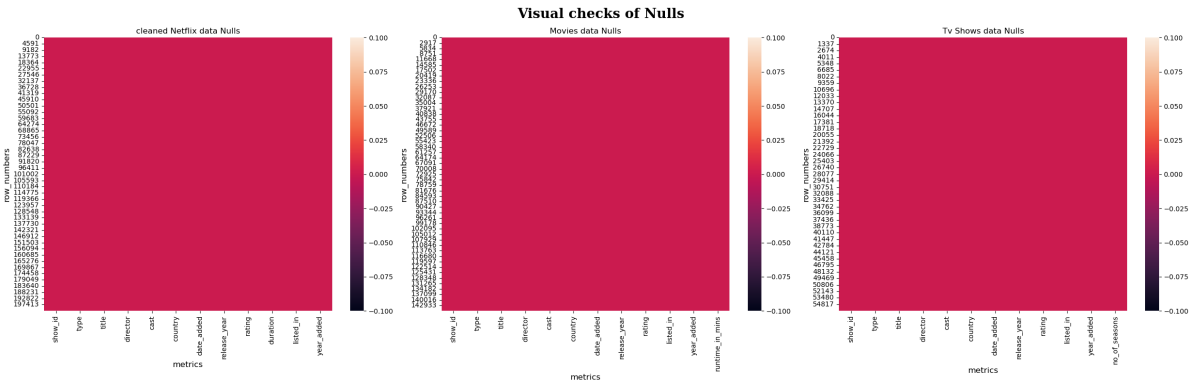
In [72]:
```
df.shape
```

Out[72]:
```
(201991, 12)
```

In [73]:
```
df.type.value_counts()
```

Out[73]:
```
type
Movie      145843
TV Show     56148
Name: count, dtype: int64
```

In [74]:
```
plt.figure(figsize=(25,8), layout='tight').suptitle('Visual checks of Nulls',fontsi


plt.subplot(1,3,1)
sns.heatmap(df.isnull())
plt.title('cleaned Netflix data Nulls',fontsize=12)
plt.xlabel('metrics',fontsize=12)
plt.ylabel('row_numbers',fontsize=12)

plt.subplot(1,3,2)
sns.heatmap(movies_data.isnull())
plt.title('Movies data Nulls',fontsize=12)
plt.xlabel('metrics',fontsize=12)
plt.ylabel('row_numbers',fontsize=12)


plt.subplot(1,3,3)
sns.heatmap(tvshows_data.isnull())
plt.title('Tv Shows data Nulls',fontsize=12)
plt.xlabel('metrics',fontsize=12)
plt.ylabel('row_numbers',fontsize=12)

plt.show()
```



👉Insights:

- 👆🏽 Red color indicates data has 0 % nulls in all columns

In [ ]:
```
# saving the files for further analysis:

df.to_csv('netflix_cleaned_data.csv',sep=',',index=False)
movies_data.to_csv('cleaned_movies_data.csv',sep=',',index=False)
tvshows_data.to_csv('cleaned_tvshows_data.csv',sep=',',index=False)
```

Exploratory Data Analysis (EDA):

In [2]:
```
nx = pd.read_csv('netflix_cleaned_data.csv')
md = pd.read_csv('cleaned_movies_data.csv')
tvd = pd.read_csv('cleaned_tvshows_data.csv')
```

📌Q. How are contents distributed in Netflix Platform ?

In [5]:
```
pg = nx.groupby('type')['show_id'].nunique()
pg
```

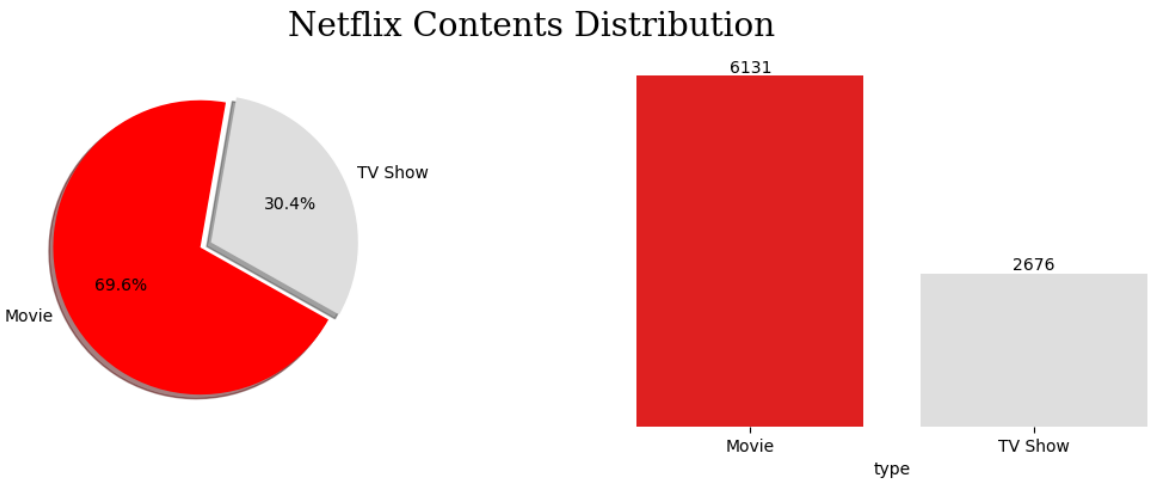
Out[5]:
```
type
Movie      6131
TV Show    2676
Name: show_id, dtype: int64
```

In [6]:
```
pgdf = pg.reset_index()
pgdf
```

Out[6]:

	type	show_id
0	Movie	6131
1	TV Show	2676

In [11]:
```
plt.figure(figsize=(13.5,4))
font = {'weight':'bold',
        'family':'serif'}
plt.suptitle("Netflix Contents Distribution",fontdict=font,fontsize=20)

plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
plt.pie(pg,
        labels=pg.index,
        startangle=80, explode=(0.08,0), colors=['red','#dedede'],
        shadow=True, autopct='%1.1f%%',textprops={'color':"k"})

plt.subplot(1,2,2)
a = sns.barplot(y=pgdf.show_id , data=pgdf , x=pgdf.type , palette=['red','#dedede'
a.bar_label(a.containers[0], label_type='edge')
sns.despine(left=True,bottom=True)
plt.yticks([])
plt.ylabel('')


plt.show()
```

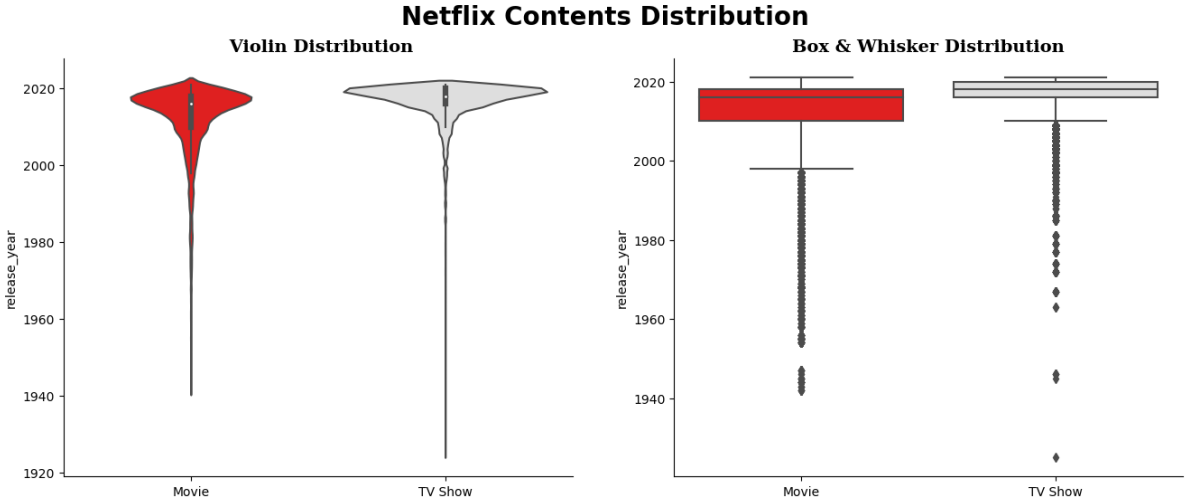


👉Insights :

- we can clearly interpret that nearly *70%* contents are `Movies` whereas *30%* are `Tvshows` contents in **Netflix** content library.

📌 Q. Outliers check:

In [32]:
```
plt.figure(figsize=(16,6))
font = {'weight':'bold',
        'family':'serif'}
plt.suptitle("Netflix Contents Distribution",fontweight='bold',fontsize=20)

plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
sns.violinplot(nx,x='type',y='release_year',palette=['red','#dedede'])
sns.despine()
plt.xlabel('')
plt.title(" Violin Distribution",fontdict=font,fontsize=14)

plt.subplot(1,2,2)
sns.boxplot(nx,x='type',y='release_year',palette=['red','#dedede'])
sns.despine()
plt.xlabel('')
plt.title("Box & Whisker Distribution",fontdict=font,fontsize=14)

plt.show()
```



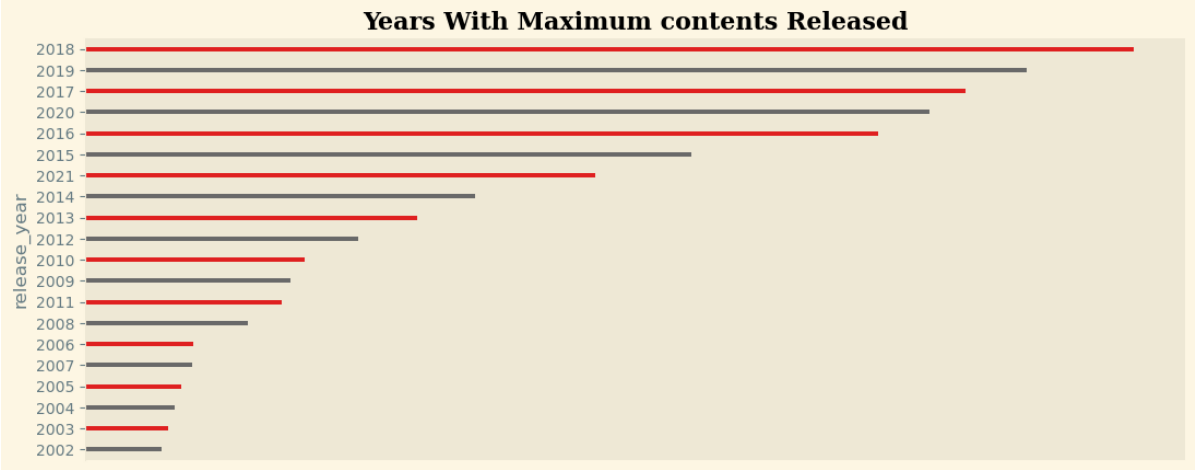
👉 Insights :

- Here we can easily identify that the contents released before 1960 are the outliers.

📌 Q. In which year maximum contents got released ?

In [95]:
```
ryvc = nx.release_year.value_counts()[:20]
```

In [97]:
```
plt.figure(figsize=(13,5))
plt.style.use('seaborn-v0_8-bright')
sns.countplot(nx , y='release_year' , order = ryvc.index , palette=['red','dimgrey'
sns.despine(bottom=True)
plt.xticks([])
plt.xlabel('')
plt.title('Years With Maximum contents Released',fontsize=16,fontweight='bold',font
plt.show()
```



👉Insights :

- Netflix began enlarging their contents library from 2000 and acquired maximum contents so far in `2018` followed by `2019` & `2017`.

📌Q. What are the top 15 countries consumption of movies and tvshows ?

In [34]:
```
# countrywise content count with movies_data
cm = md.groupby('country')[['show_id']].nunique().sort_values(by='show_id',ascendir
cm = cm[:15]
cwm = cm[cm.index!=('Unknown')]
cwm
```

Out[34]:

country	show_id
United States	2751
India	962
United Kingdom	532
Canada	319
France	303
Germany	182
Spain	171
Japan	119
China	114
Mexico	111
Egypt	102
Hong Kong	100
Nigeria	94
Australia	94

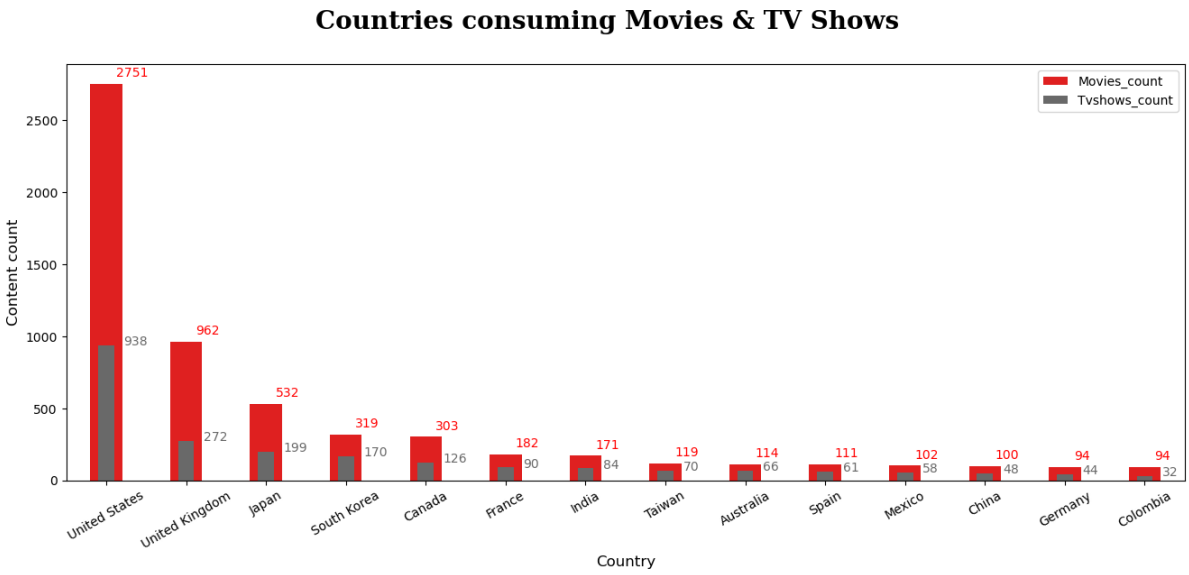
In [82]:
```
# countrywise content count with tvshows_data
ctv = tvd.groupby('country')[['show_id']].nunique().sort_values(by='show_id',ascend
cwtv = ctv[:15]
cwtv = cwtv[cwtv.index!=('Unknown')]
cwtv
```

Out[82]:

country	show_id
United States	938
United Kingdom	272
Japan	199
South Korea	170
Canada	126
France	90
India	84
Taiwan	70
Australia	66
Spain	61
Mexico	58
China	48
Germany	44
Colombia	32

In [83]:
```
# Graphical Analysis
plt.figure(figsize=(16,6))
plt.suptitle('Countries consuming Movies & TV Shows',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('seaborn-v0_8-bright')

c1 = sns.barplot(cwm, x=cwm.index , y=cwm.show_id,
                 color='red' , width=0.4 , label='Movies_count')
#c1.bar_label(c1.containers[0], label_type='edge',color='r')
plt.xlabel('Country',fontsize=12)
plt.ylabel('Content count',fontsize=12)
plt.legend(loc='upper right')
plt.xticks(rotation=30)

c2 = sns.barplot(cwtv, x=cwtv.index , y=cwtv.show_id,
                 color='dimgray' , width=0.2 , label='Tvshows_count')
plt.xlabel('Country',fontsize=12)
plt.ylabel('Content count',fontsize=12)
plt.legend(loc='upper right')
plt.xticks(rotation=30)

top_n = 14
for i in range(top_n):
    c1.annotate(cwm.show_id[i], (i+0.12, cwm.show_id[i]+50),
                ha='left', va='baseline',color='red')

for i in range(top_n):
    c2.annotate(cwtv.show_id[i], (i+0.22, cwtv.show_id[i]),
```

```
                ha='left', va='baseline', color='dimgrey')

plt.show()
```



👉Insights :

- The top 5 countries with the highest count of Movies and TV Shows are aiding in recognizing the key players in TV show production and Movies production.
- We can infer that `US` , `India` , `UK` , `France` , `Canada` , `Japan` are the top entertainment consumers while other countries significantly contribute to the OTT content library.

📌Q. How much contents are added every year in netflix ?

In [12]:
```
yc = nx.groupby(['year_added','type'])[['show_id']].nunique().reset_index()
yc.sort_values(by='show_id',ascending=False)
```

Out[12]:

	year_added	type	show_id
18	2019	Movie	1424
20	2020	Movie	1284
16	2018	Movie	1237
22	2021	Movie	993
14	2017	Movie	839
21	2020	TV Show	692
19	2019	TV Show	575
23	2021	TV Show	505
17	2018	TV Show	388
15	2017	TV Show	325
12	2016	Movie	253
13	2016	TV Show	165
10	2015	Movie	56
8	2014	Movie	19
11	2015	TV Show	17
4	2011	Movie	13
6	2013	Movie	6
9	2014	TV Show	4
7	2013	TV Show	4
5	2012	Movie	3
2	2009	Movie	2
1	2008	TV Show	1
3	2010	Movie	1
0	2008	Movie	1

In [85]:
```
yc['show_id'].sum()
```

Out[85]:
```
8807
```

In [13]:
```
ycm = md.groupby(['year_added','type'])[['show_id']].nunique().reset_index()
ycm
```

Out[13]:

	year_added	type	show_id
0	2008	Movie	1
1	2009	Movie	2
2	2010	Movie	1
3	2011	Movie	13
4	2012	Movie	3
5	2013	Movie	6
6	2014	Movie	19
7	2015	Movie	56
8	2016	Movie	253
9	2017	Movie	839
10	2018	Movie	1237
11	2019	Movie	1424
12	2020	Movie	1284
13	2021	Movie	993

In [87]:
```
ycm['show_id'].sum()
```

Out[87]:
```
6131
```

In [49]:
```
yctv = tvd.groupby(['year_added','type'])[['show_id']].nunique().reset_index()
yctv
```

Out[49]:

	year_added	type	show_id
0	2008	TV Show	1
1	2013	TV Show	4
2	2014	TV Show	4
3	2015	TV Show	17
4	2016	TV Show	165
5	2017	TV Show	325
6	2018	TV Show	388
7	2019	TV Show	575
8	2020	TV Show	692
9	2021	TV Show	505

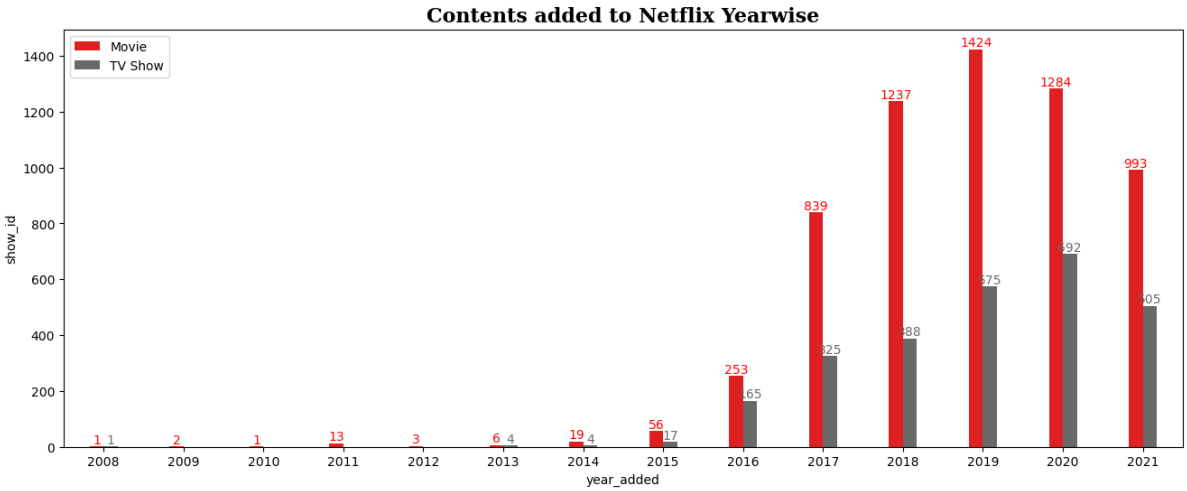
In [89]:
```
yctv['show_id'].sum()
```

Out[89]:
```
2676
```

MULTIVARIATE ANALISYS: Type by year_added

In [15]:
```
plt.figure(figsize=(16,6))
plt.style.use('default')
```

```
plt.style.use('seaborn-v0_8-bright')
c = sns.barplot(data = yc, x = 'year_added' , y = 'show_id' ,
                hue = 'type', palette=['red','dimgrey'] , width=0.35)
plt.title('Contents added to Netflix Yearwise',
          fontsize=16,fontweight="bold",fontfamily='serif')
c.bar_label(c.containers[0], label_type='edge',color='red')
c.bar_label(c.containers[1], label_type='edge',color='dimgray')
plt.legend(loc='upper left')
plt.show()
```



👉Insights :

Netflix's content acquisition strategy over time.

- The bar plot shows the distribution of content added to Netflix across different years.
- There appears to be an increasing trend in the total number of content items added to Netflix over the years.
- The bars tend to get taller as you move from left to right, suggesting that Netflix has been continuously expanding its content library.
- The variation in bar heights from year to year highlights how Netflix's content strategy has evolved.
- Some years show significant spikes, while others have lower counts, indicating variations in content acquisition.

📌Q. How much contents gets released every year ?

In [295…

```
mr = md.groupby('release_year')[['title']].nunique()
mr = mr.reset_index()
mr
```

Out[295]:

	release_year	title
0	1942	2
1	1943	3
2	1944	3
3	1945	3
4	1946	1
...	...	...
68	2017	767
69	2018	767
70	2019	633
71	2020	517
72	2021	277

73 rows × 2 columns

In [92]:
```
mr.title.sum()
```

Out[92]:
```
6131
```

In [296…
```
tvr = tvd.groupby('release_year')[['title']].nunique()
tvr = tvr.reset_index()
tvr
```

Out[296]:

	release_year	title
0	1925	1
1	1945	1
2	1946	1
3	1963	1
4	1967	1
5	1972	1
6	1974	1
7	1977	1
8	1979	1
9	1981	1
10	1985	1
11	1986	2
12	1988	2
13	1989	1
14	1990	3
15	1991	1
16	1992	3
17	1993	4
18	1994	2
19	1995	2
20	1996	3
21	1997	4
22	1998	4
23	1999	7
24	2000	4
25	2001	5
26	2002	7
27	2003	10
28	2004	9
29	2005	13
30	2006	14
31	2007	14
32	2008	23
33	2009	34
34	2010	40
35	2011	40

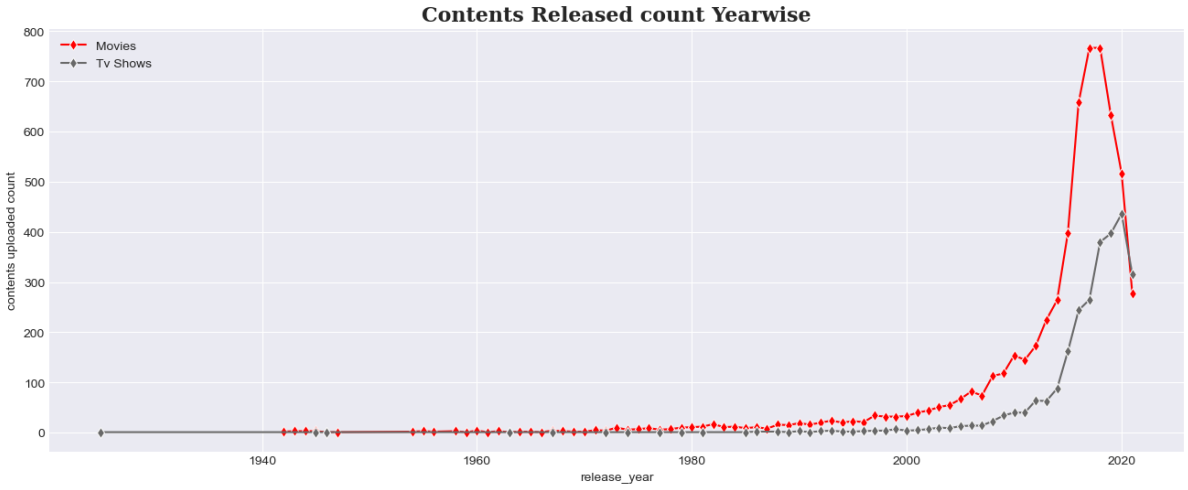
	release_year	title
36	2012	64
37	2013	63
38	2014	88
39	2015	162
40	2016	244
41	2017	265
42	2018	380
43	2019	397
44	2020	436
45	2021	315

In [94]:
```
tvr.title.sum()
```

Out[94]: 2676

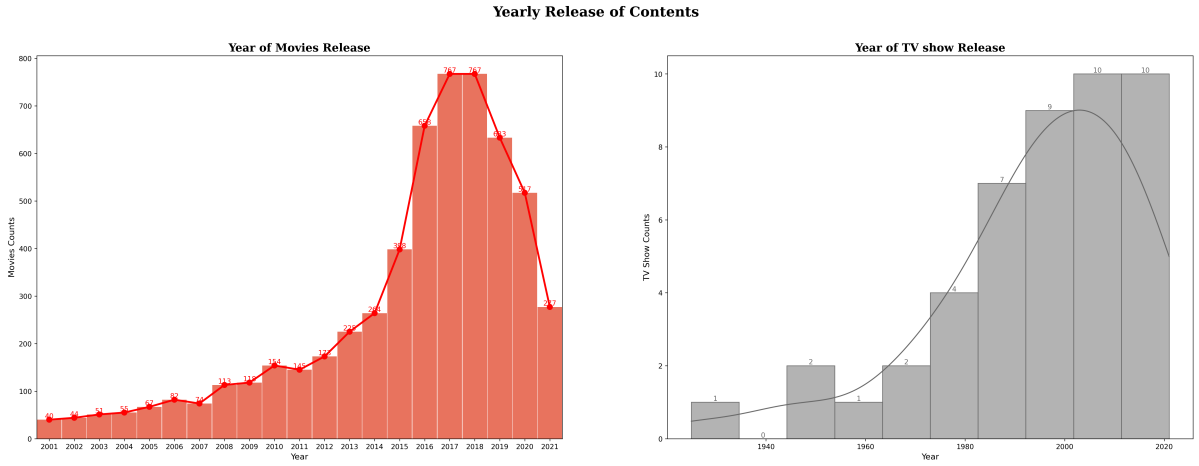
UNIVARIATE ANALISYS

In [106…
```
plt.figure(figsize=(16,6))
plt.style.use('seaborn-v0_8-darkgrid')
sns.lineplot(data=mr , x='release_year' , y='title' , color='r' ,
             label = 'Movies', marker='d')
sns.lineplot(data=tvr , x='release_year' , y='title' , color='dimgrey',
             label='Tv Shows' , marker='d')
plt.title('Contents Released count Yearwise',fontsize=16,
                      fontweight="bold",fontfamily='serif')
plt.ylabel('contents uploaded count')
plt.legend(loc='upper left')
plt.show()
```



In [98]:
```
plt.figure(figsize=(30,10) , dpi=250)
plt.suptitle('Yearly Release of Contents',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

mr= mr[mr.release_year>2000]
plt.subplot(1,2,1)
c = sns.barplot(mr , x = 'release_year' , y='title', color='tomato',width=0.98)
```

```
c.bar_label(c.containers[0], label_type='edge',color='r')
sns.pointplot(mr , x='release_year' , y='title' , color='r')
plt.xlabel("Year",fontsize=12)
plt.ylabel("Movies Counts", fontsize=12)
plt.title("Year of Movies Release", fontsize=16,fontweight="bold",fontfamily='serif

plt.subplot(1,2,2)
d = sns.histplot(x = tvr.release_year, bins = 10, kde = True,
             color='dimgrey' , edgecolor ='dimgrey')
d.bar_label(d.containers[0], label_type='edge',color='dimgrey')
plt.xlabel('Year',fontsize=12)
plt.ylabel("TV Show Counts", fontsize=12)
plt.title("Year of TV show Release", fontsize=16,fontweight="bold",fontfamily='seri

plt.show()
```



In [292…

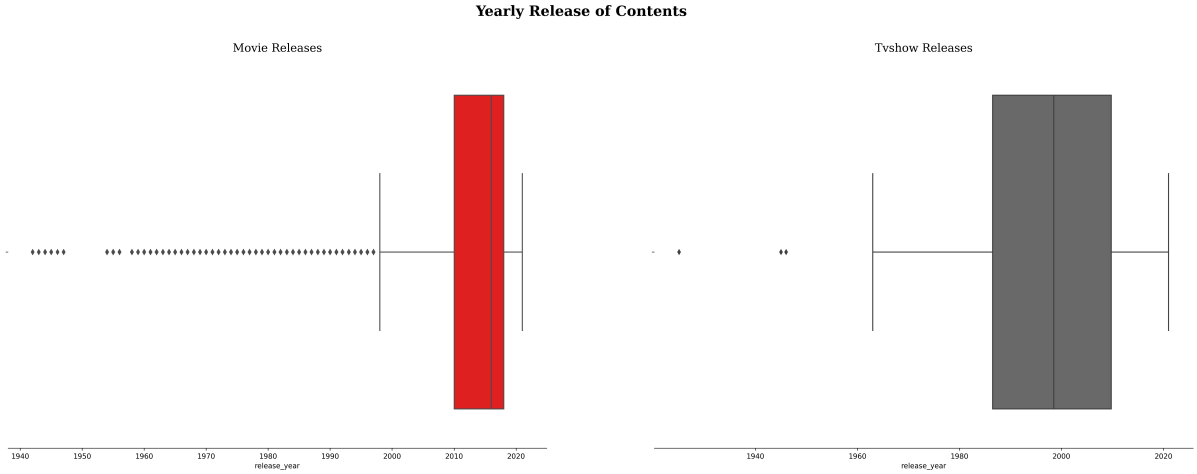
```
mr.dtypes
```

Out[292]:

```
release_year    int64
title           int64
dtype: object
```

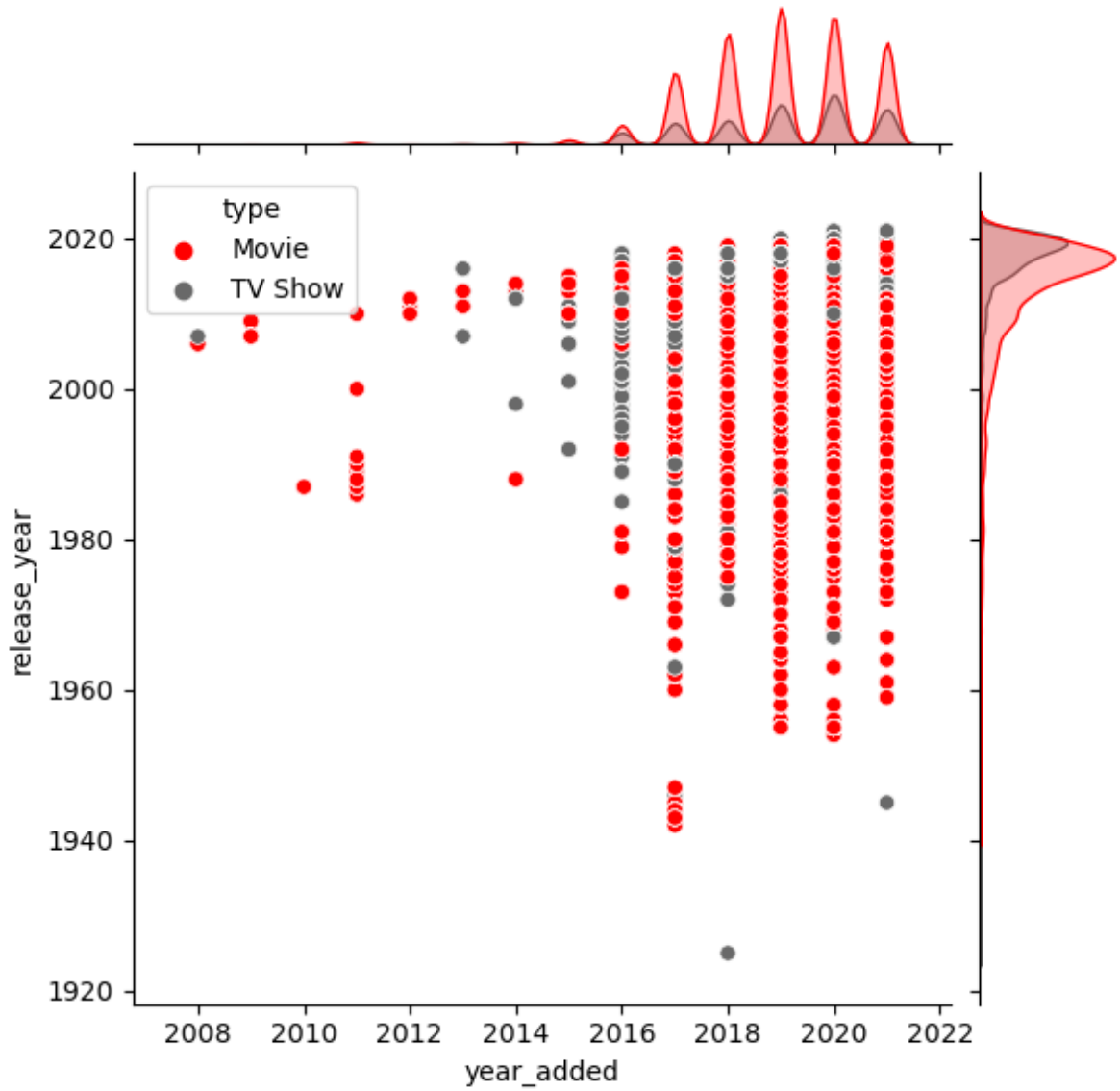
In [307…

```
plt.figure(figsize=(30,10) , dpi=250)
plt.suptitle('Yearly Release of Contents',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
sns.boxplot(md , x= 'release_year', color='red')
sns.despine()
plt.title('Movie Releases',fontsize=16,fontfamily='serif')

plt.subplot(1,2,2)
sns.boxplot(tvr , x= 'release_year', color='dimgrey')
sns.despine(left=True)
plt.title('Tvshow Releases',fontsize=16,fontfamily='serif')

plt.show()
```



In [367…

```
sns.jointplot(nx , x='year_added' , y='release_year' , hue='type' ,
                        palette=['red','dimgrey'])
plt.show()
```



👉Insights :

- The plot uses two distinct lines in different colors (red for movies and dimgrey for TV shows) to enable a side-by-side comparison.
- The plot reveals that Movies has been more dominant in terms of release counts in any given year and in the recent past audience focus shifts on watching web series.

- The dominance of movies and TV shows over the years can indicate changing audience preferences, industry trends and Netflix's strategic decisions.
- Netflix has been continuously expanding its content library, offering more choices to its subscribers.

📌 Q. How the contents genre segregated ?

In [96]:
```
mg = md.groupby(['listed_in'])[['title']].nunique().sort_values(by='title',ascendir
mg = mg.reset_index()
mg
```

Out[96]:

	listed_in	title
0	International Movies	2752
1	Dramas	2427
2	Comedies	1674
3	Documentaries	869
4	Action & Adventure	859
5	Independent Movies	756
6	Children & Family Movies	641
7	Romantic Movies	616
8	Thrillers	577
9	Music & Musicals	375
10	Horror Movies	357
11	Stand-Up Comedy	343
12	Sci-Fi & Fantasy	243
13	Sports Movies	219
14	Classic Movies	116
15	LGBTQ Movies	102
16	Cult Movies	71
17	Anime Features	71
18	Faith & Spirituality	65
19	Movies	57

In [97]:
```
tvg = tvd.groupby(['listed_in'])[['title']].nunique().sort_values(by='title',ascend
tvg = tvg.reset_index()
tvg
```

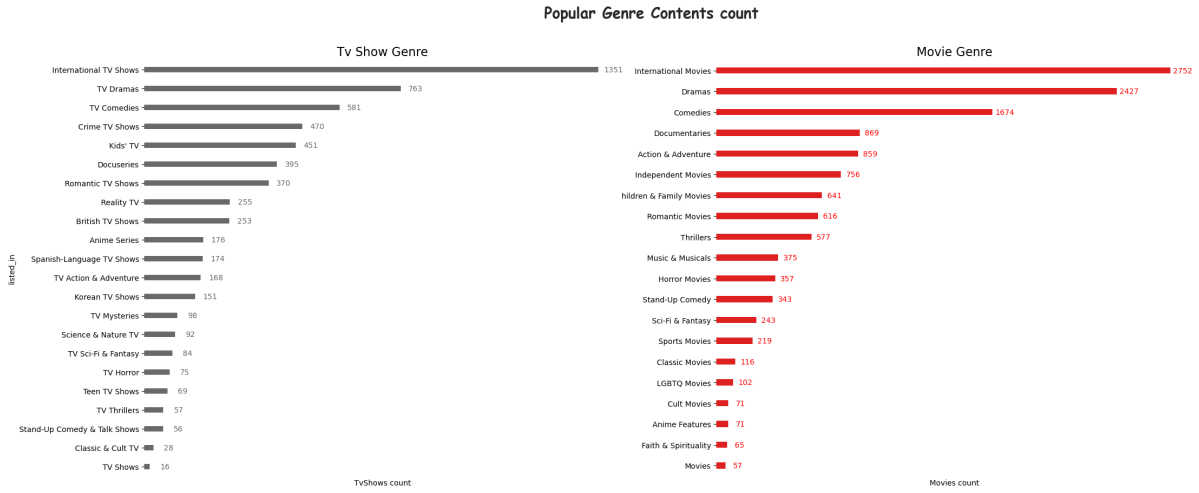
Out[97]:

	listed_in	title
0	International TV Shows	1351
1	TV Dramas	763
2	TV Comedies	581
3	Crime TV Shows	470
4	Kids' TV	451
5	Docuseries	395
6	Romantic TV Shows	370
7	Reality TV	255
8	British TV Shows	253
9	Anime Series	176
10	Spanish-Language TV Shows	174
11	TV Action & Adventure	168
12	Korean TV Shows	151
13	TV Mysteries	98
14	Science & Nature TV	92
15	TV Sci-Fi & Fantasy	84
16	TV Horror	75
17	Teen TV Shows	69
18	TV Thrillers	57
19	Stand-Up Comedy & Talk Shows	56
20	Classic & Cult TV	28
21	TV Shows	16

In [98]:

```
plt.figure(figsize=(25,10))
plt.suptitle('Popular Genre Contents count',fontsize=20,
             fontweight="bold",fontfamily='cursive')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,2)
sns.barplot(mg , x='title' , y='listed_in' , color='red' , width=0.3)
sns.despine(left=True,bottom=True,trim=True)
plt.title('Movie Genre',fontsize=16)
plt.xlabel('Movies count')
plt.xticks([])
n=20
for i in range(n):
    plt.annotate(mg.title[i], (mg.title[i]+75,i+0.2),
                 ha='center' , va='bottom' , color='r')

plt.subplot(1,2,1)
sns.barplot(tvg , x='title' , y='listed_in' , color='dimgrey' , width=0.3)
sns.despine(left=True,bottom=True,trim=True)
plt.title('Tv Show Genre',fontsize=16)
plt.xlabel('TvShows count')
plt.xticks([])
```

```
nn=22
for i in range(nn):
    plt.annotate(tvg.title[i], (tvg.title[i]+45,i+0.2),
                 ha='center' , va='bottom' , color='dimgrey')

plt.show()
```



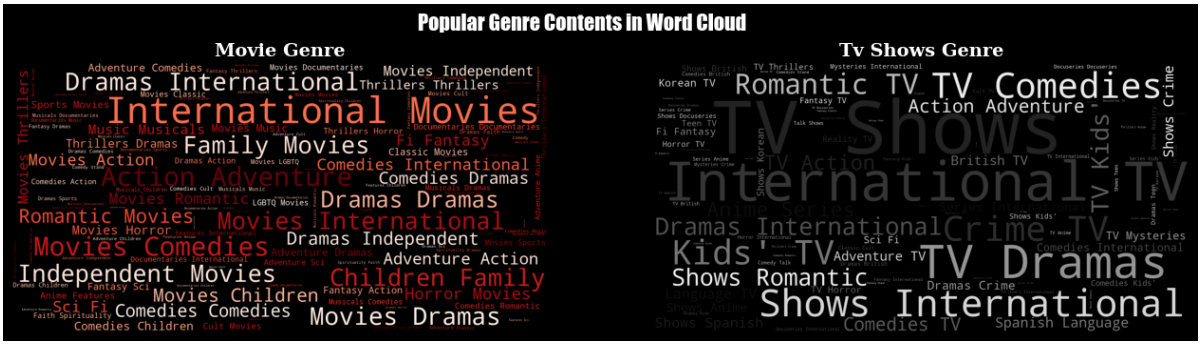
Genre-WordCloud

In [309…

```
from wordcloud import WordCloud
```

In [317…

```
plt.figure(figsize=(16,4))
plt.suptitle('Popular Genre Contents in Word Cloud',
             fontsize=16,fontweight="bold",fontfamily='fantasy')
plt.style.use('default')
plt.style.use('dark_background')

plt.subplot(1,2,1)
mgwc = WordCloud(width=1600, height=800, background_color='black',
                 colormap='Reds').generate(md.listed_in.to_string())
plt.imshow(mgwc)
plt.axis('off')
plt.title("Movie Genre",fontsize=14,fontweight='bold',fontfamily='serif')

plt.subplot(1,2,2)
tvgwc = WordCloud(width=1600, height=800, background_color='black',
                  colormap='Greys').generate(tvd.listed_in.to_string())
plt.imshow(tvgwc)
plt.axis('off')
plt.title("Tv Shows Genre",fontsize=14,fontweight='bold',fontfamily='serif')

plt.show()
```



👉Insights :

- The plot illustrates the popularity of various genres in Movies and TV Shows on the platform.
- Here, we can see that `Hollywood contents` , `Dramas` , `Comedies` are the Top and Evergreen genres
- The plot provides insights into audience preferences, indicating which genres are more prevalent in Movies and TV Shows.

📌 Q. what genre's are more preferred by directors ?

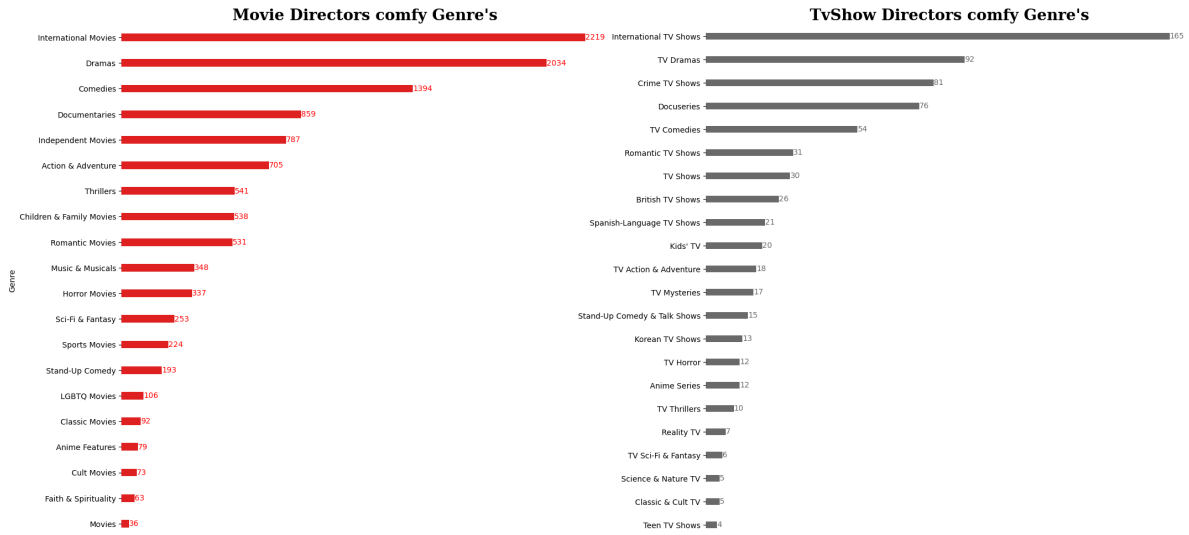
In [281…
```
mdgc = md.groupby('listed_in')['director'].nunique().sort_values(ascending=False)
mdgc
```

Out[281]:
```
listed_in
International Movies        2219
Dramas                      2034
Comedies                    1394
Documentaries                859
Independent Movies           787
Action & Adventure           705
Thrillers                    541
Children & Family Movies     538
Romantic Movies              531
Music & Musicals             348
Horror Movies                337
Sci-Fi & Fantasy             253
Sports Movies                224
Stand-Up Comedy              193
LGBTQ Movies                 106
Classic Movies                92
Anime Features                79
Cult Movies                   73
Faith & Spirituality          63
Movies                        36
Name: director, dtype: int64
```

In [282…
```
tvdgc = tvd.groupby('listed_in')['director'].nunique().sort_values(ascending=False)
tvdgc
```

Out[282]:
```
listed_in
International TV Shows          165
TV Dramas                        92
Crime TV Shows                   81
Docuseries                       76
TV Comedies                      54
Romantic TV Shows                31
TV Shows                         30
British TV Shows                 26
Spanish-Language TV Shows        21
Kids' TV                         20
TV Action & Adventure            18
TV Mysteries                     17
Stand-Up Comedy & Talk Shows     15
Korean TV Shows                  13
TV Horror                        12
Anime Series                     12
TV Thrillers                     10
Reality TV                        7
TV Sci-Fi & Fantasy               6
Science & Nature TV               5
Classic & Cult TV                 5
Teen TV Shows                     4
Name: director, dtype: int64
```

In [384…
```
plt.figure(figsize=(25, 12))
plt.suptitle('Directors popular Genre Contents',
                fontsize=20,fontweight="bold",fontfamily='cursive')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
a = sns.barplot(y=mdgc.index, x=mdgc.values, color='r',width=0.3)
a.bar_label(a.containers[0], label_type='edge',color='r')
plt.title('Movie Directors comfy Genre\'s',fontsize=20,
                  fontweight="bold",fontfamily='serif')
sns.despine(left=True,bottom=True,trim=True)
plt.ylabel('Genre')
plt.xticks([])

plt.subplot(1,2,2)
a = sns.barplot(y=tvdgc.index, x=tvdgc.values, color='dimgrey',width=0.3)
a.bar_label(a.containers[0], label_type='edge',color='dimgrey')
plt.title('TvShow Directors comfy Genre\'s',fontsize=20,
                  fontweight="bold",fontfamily='serif')
sns.despine(left=True,bottom=True,trim=True)
plt.ylabel('')
plt.xticks([])
plt.show()
```



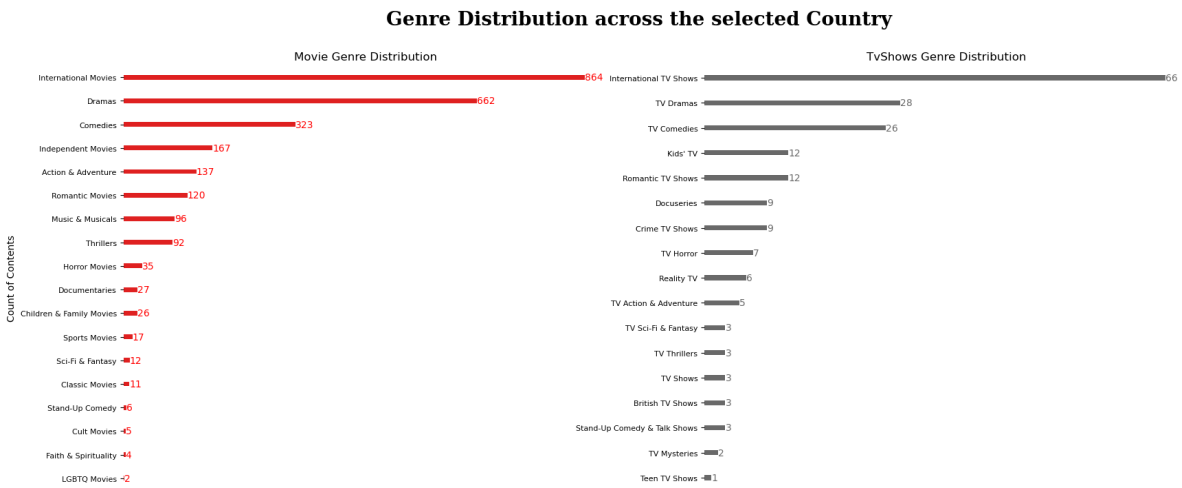
👉Insights:

- The diversity of genre that the directors are more comfortable indicates that Netflix has content on all genres in its library.
- The top genres with the most directors are International Movies, Dramas, Comedies, Documentaries, Independent Movies, and Action & Adventure.

📌Q. What are genres more preferred in each country ?

In [99]:
```
plt.figure(figsize=(20, 8))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')


given_country = input("Enter your preferred Choice of Country : ")

mcountry = md[md["country"] == given_country]
tvcountry = tvd[tvd["country"] == given_country]

mc_data = mcountry.groupby(['listed_in','type'])[['show_id']].nunique()
mc_data = mc_data.sort_values(by=["show_id"],ascending = False).reset_index()

mtv_data = tvcountry.groupby(['listed_in','type'])[['show_id']].nunique()
mtv_data = mtv_data .sort_values(by=["show_id"],ascending = False).reset_index()


plt.suptitle('Genre Distribution across the selected Country'
             ,fontsize=20,fontweight="bold",fontfamily='serif')

plt.subplot(1,2,1)
a = sns.barplot(mc_data , y='listed_in', x='show_id',color='red',width = 0.2)
a.bar_label(a.containers[0], label_type='edge',color='r')
plt.title('Movie Genre Distribution')
plt.xlabel('')
plt.xticks([])
plt.ylabel('Count of Contents')
plt.xticks(rotation=90, ha = 'center',fontsize = 8)
plt.yticks(fontsize =8)

plt.subplot(1,2,2)
b = sns.barplot(mtv_data , y='listed_in', x='show_id',color='dimgrey',width = 0.2)
b.bar_label(b.containers[0], label_type='edge',color='dimgrey')
```

```
sns.despine(bottom=True,left=True)
plt.title('TvShows Genre Distribution')
plt.xlabel('')
plt.xticks([])
plt.ylabel('')
plt.xticks(rotation=90, ha = 'center',fontsize = 8)
plt.yticks(fontsize =8)

plt.show()
```



📌 Q. How are contents distributed based on Runtime & Seasons ?

In [34]:
```
tvd.groupby(['no_of_seasons'])[['title']].nunique().sum()
```

Out[34]:
```
title    2676
dtype: int64
```

In [103…
```
md.groupby(['runtime_in_mins'])[['title']].nunique().sum()
```

Out[103]:
```
title    6131
dtype: int64
```

In [91]:
```
mrt = md.groupby(['runtime_in_mins'])[['title']].nunique().sort_values(by='title',a
mrt = mrt.reset_index()
mrt
```

Out[91]:

	runtime_in_mins	title
0	90	152
1	97	146
2	93	146
3	94	146
4	91	144
...	...	...
200	189	1
201	191	1
202	193	1
203	194	1
204	312	1

205 rows × 2 columns

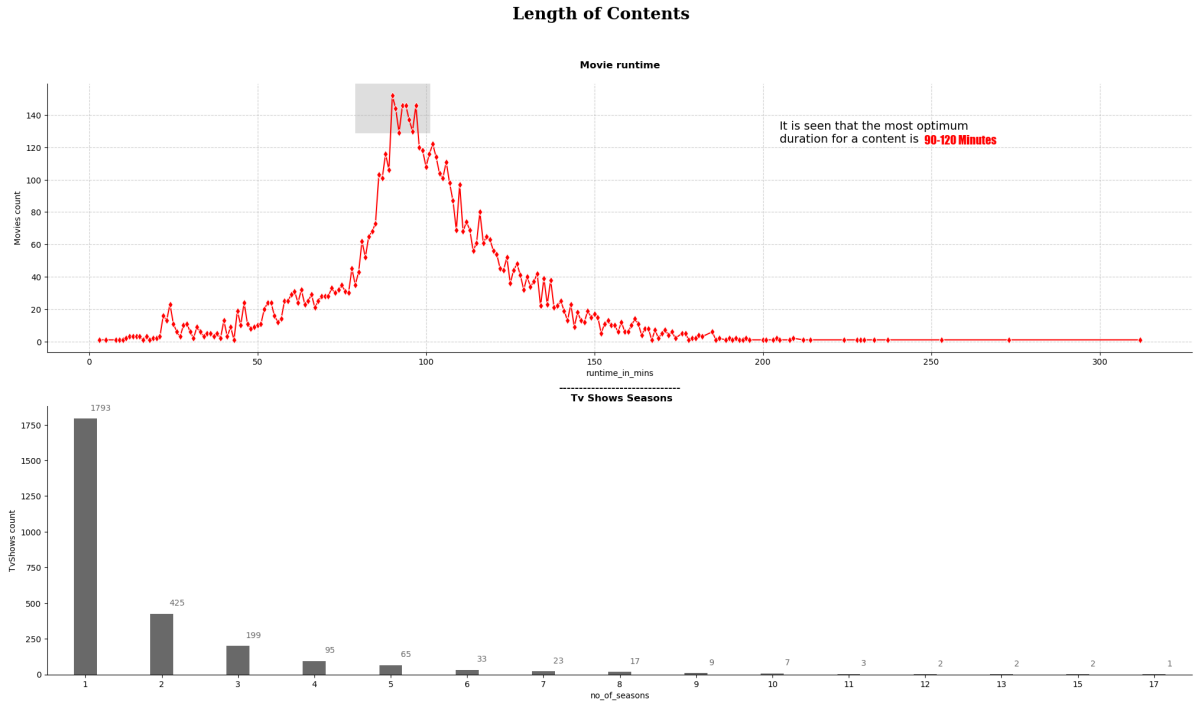
In [92]:
```
tvs = tvd.groupby(['no_of_seasons'])[['title']].nunique().sort_values(by='title',as
tvs = tvs.reset_index()
tvs
```

Out[92]:

	no_of_seasons	title
0	1	1793
1	2	425
2	3	199
3	4	95
4	5	65
5	6	33
6	7	23
7	8	17
8	9	9
9	10	7
10	13	3
11	11	2
12	12	2
13	15	2
14	17	1

In [142…
```
plt.figure(figsize=(25,13))
plt.suptitle('Length of Contents',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
```

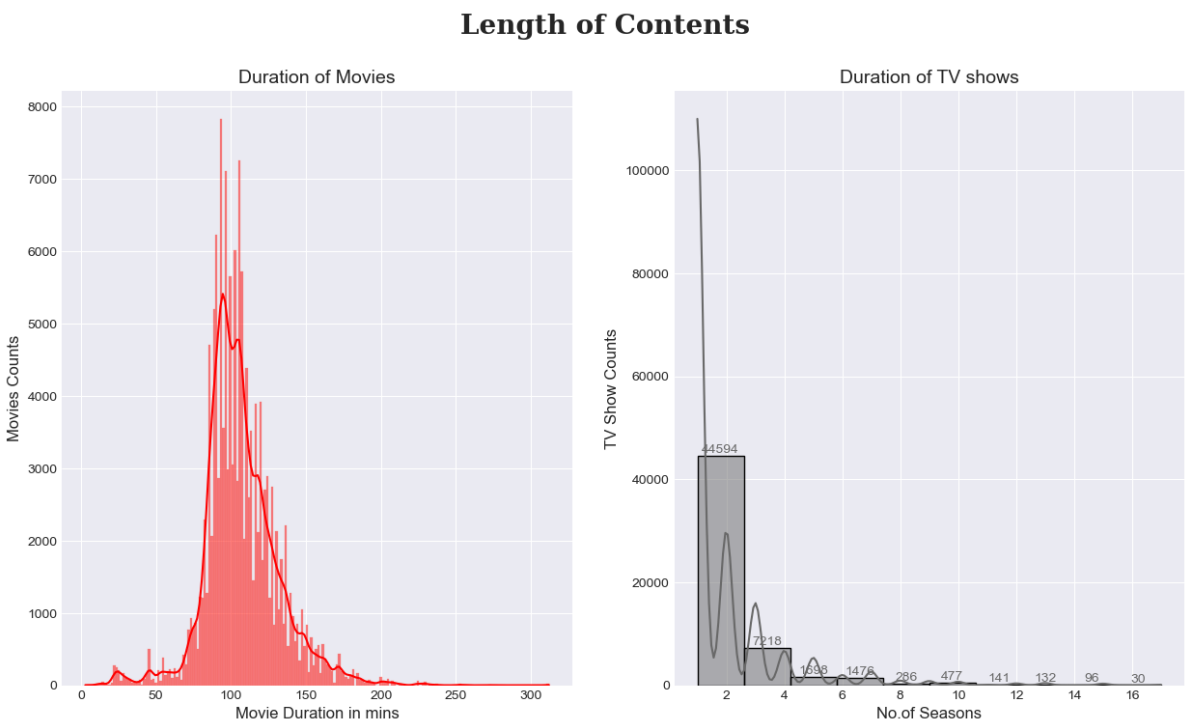
```
plt.subplot(2,1,1)
sns.lineplot(mrt , y='title' , x='runtime_in_mins' , color='red' , marker='d')
sns.despine()
plt.grid(True, linestyle='--', alpha=0.6)
plt.title('Movie runtime\n' ,fontsize=12,fontweight="bold")
plt.ylabel('Movies count')
plt.text(205,123,'It is seen that the most optimum\nduration for a content is',
         fontsize=14,fontfamily='sans-serif')
plt.text(248,122,'90-120 Minutes',color='r',
         fontsize=14,fontfamily='fantasy',fontweight='bold')
max_value = mrt.title.max()
max_x = mrt[mrt.title == max_value]['runtime_in_mins']
sns.scatterplot(x=max_x, y=max_value, color='#dedede', marker='s', s=10000)

plt.subplot(2,1,2)
sns.barplot(tvs , y='title' , x='no_of_seasons' , color='dimgrey' , width=0.3)
sns.despine()
plt.title('------------------------------\n Tv Shows Seasons',
          fontsize=12,fontweight="bold")
plt.ylabel('TvShows count')
n=15
for i in range(n):
     plt.annotate(tvs.title[i], (i+0.2,tvs.title[i]+45),
                  ha='center' , va='bottom' , color='dimgrey')

plt.show()
```



In [159…

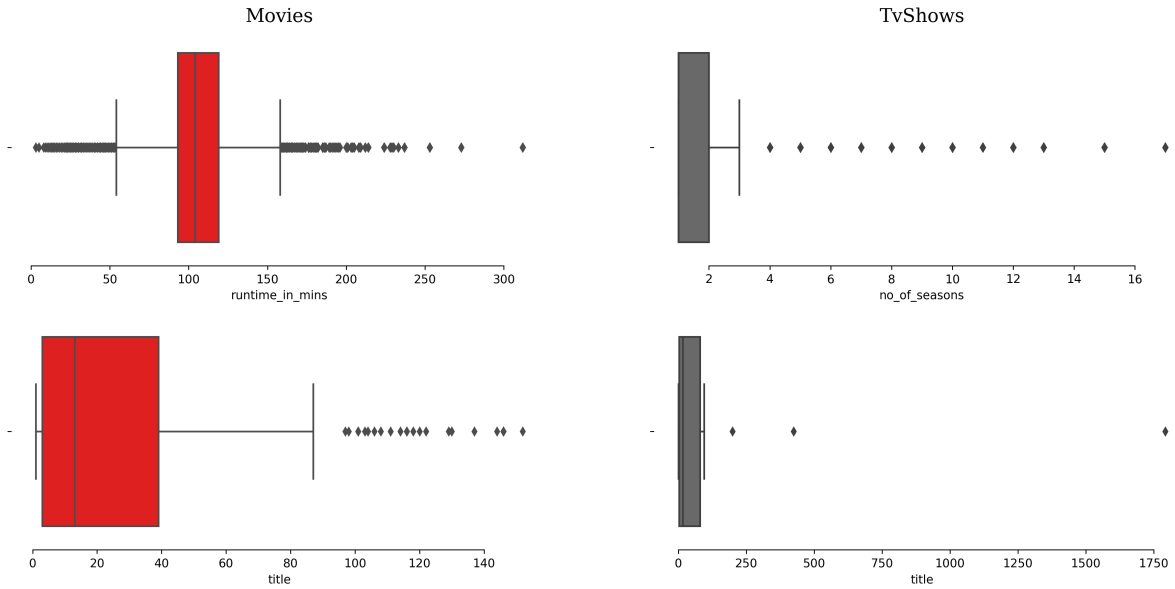
```
plt.figure(figsize=(15,8))
plt.suptitle('Length of Contents',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('seaborn-v0_8-darkgrid')

plt.subplot(1,2,1)
sns.histplot(x = md.runtime_in_mins, bins = 200, color='red',
             kde = True, edgecolor = 'salmon')
plt.xlabel("Movie Duration in mins",fontsize=12)
plt.ylabel("Movies Counts", fontsize=12)
plt.title("Duration of Movies", fontsize=14)

plt.subplot(1,2,2)
```

```
b = sns.histplot(x = tvd.no_of_seasons, bins = 10, kde = True,
             color='dimgrey' , edgecolor ='k')
b.bar_label(b.containers[0], label_type='edge',color='dimgrey')
plt.xlabel('No.of Seasons',fontsize=12)
plt.ylabel("TV Show Counts", fontsize=12)
plt.title("Duration of TV shows", fontsize=14)

plt.show()
```



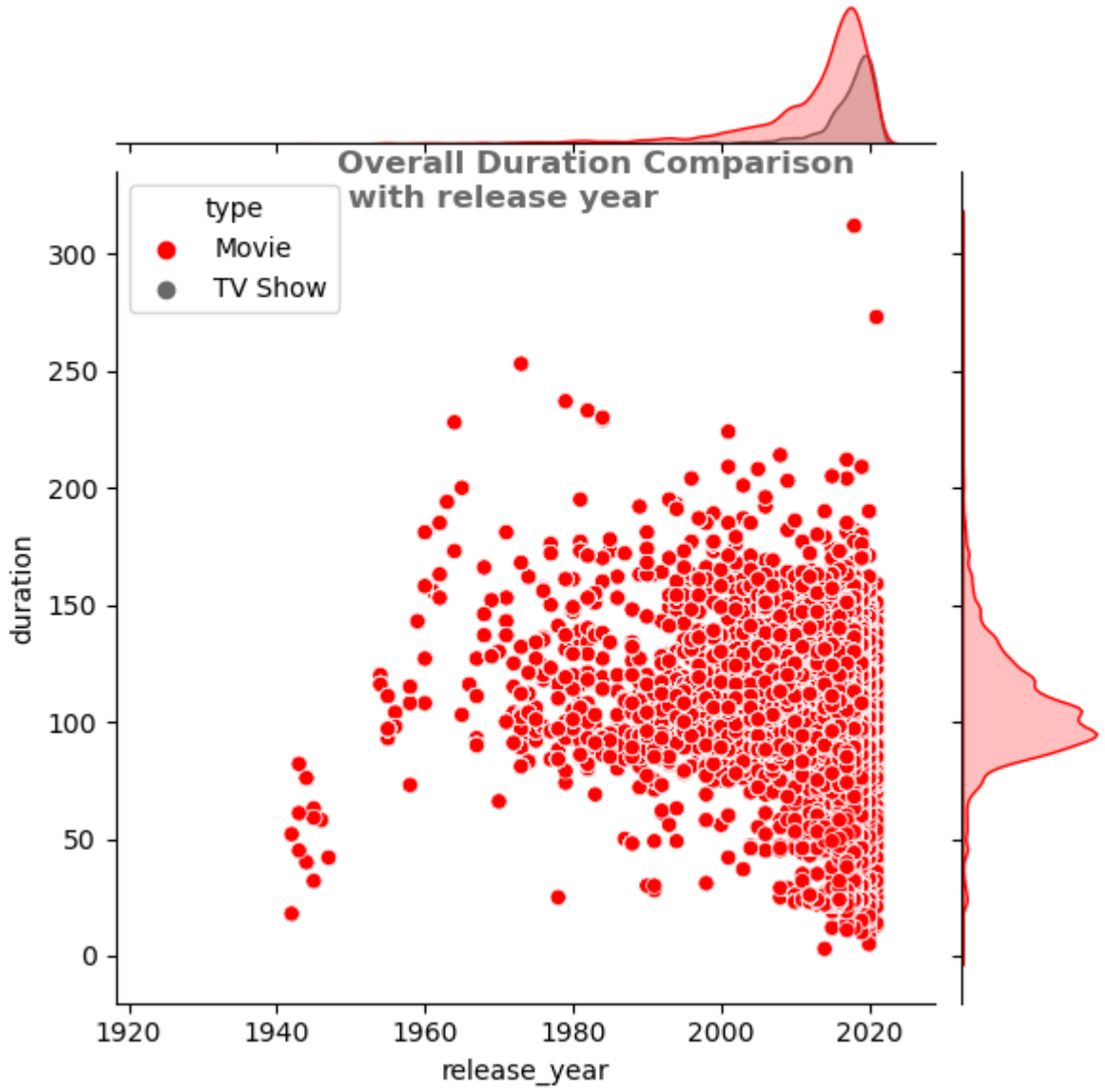
In [433…

```
plt.figure(figsize=(18,8) , dpi=250)
plt.suptitle('Contents Duration',
             fontsize=20,fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(2,2,1)
sns.boxplot(md , x ='runtime_in_mins', color='red',showfliers=True)
plt.title('Movies',fontsize=16,fontfamily='serif')

plt.subplot(2,2,3)
sns.boxplot(mrt , x='title', color='red')

plt.subplot(2,2,2)
sns.boxplot(tvd , x= 'no_of_seasons', color='dimgrey')
plt.title('TvShows',fontsize=16,fontfamily='serif')

plt.subplot(2,2,4)
sns.boxplot(tvs , x='title', color='dimgrey')
sns.despine(left=True,trim=True)

plt.show()
```



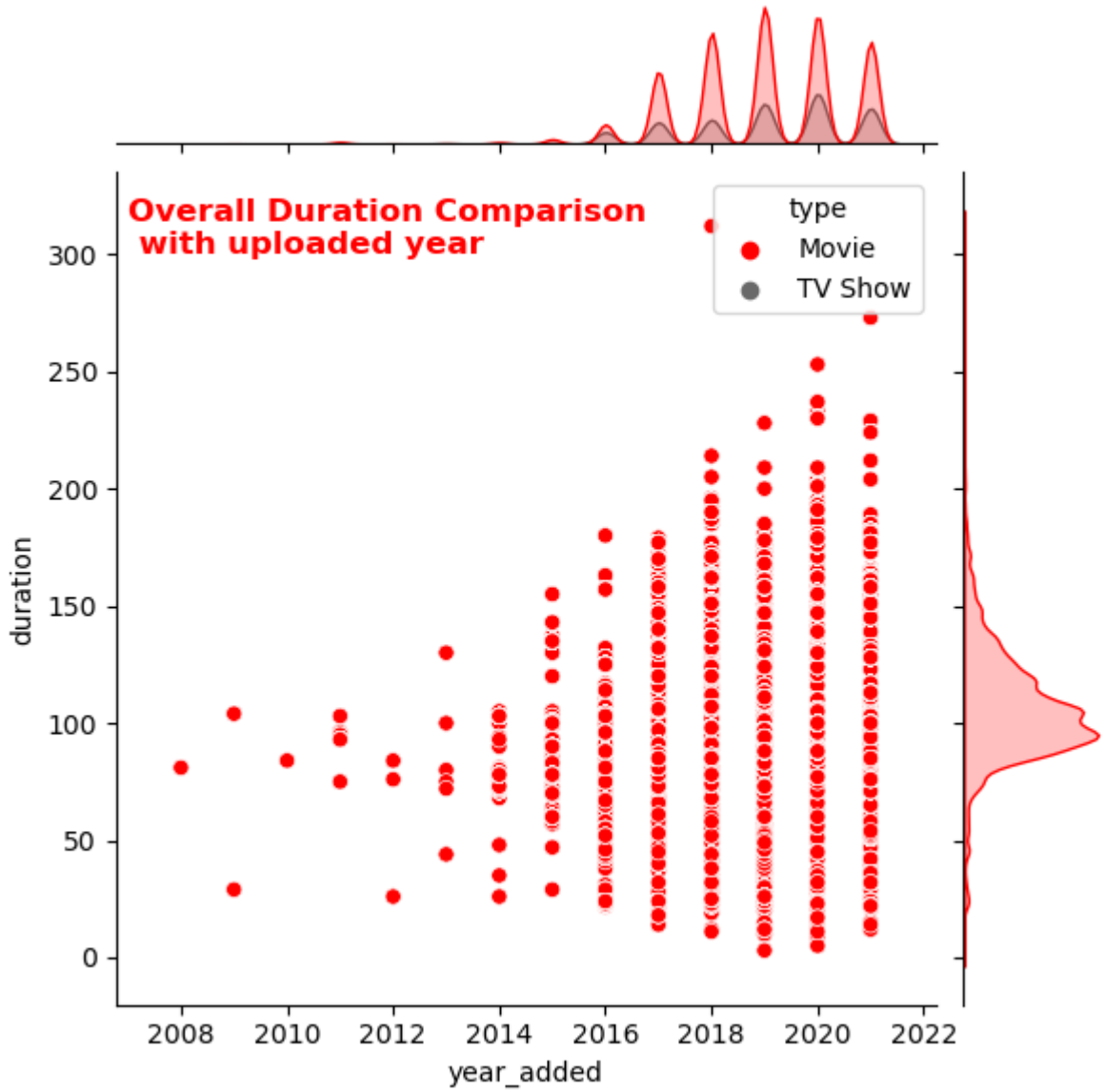
In [424…

```
sns.jointplot(nx , x='release_year' ,
              y='duration' , hue='type' ,
              palette=['red','dimgrey'])
plt.text(1948,320,'Overall Duration Comparison \n with release year',color='dimgrey
         fontsize=12,fontweight='bold')
plt.show()
```



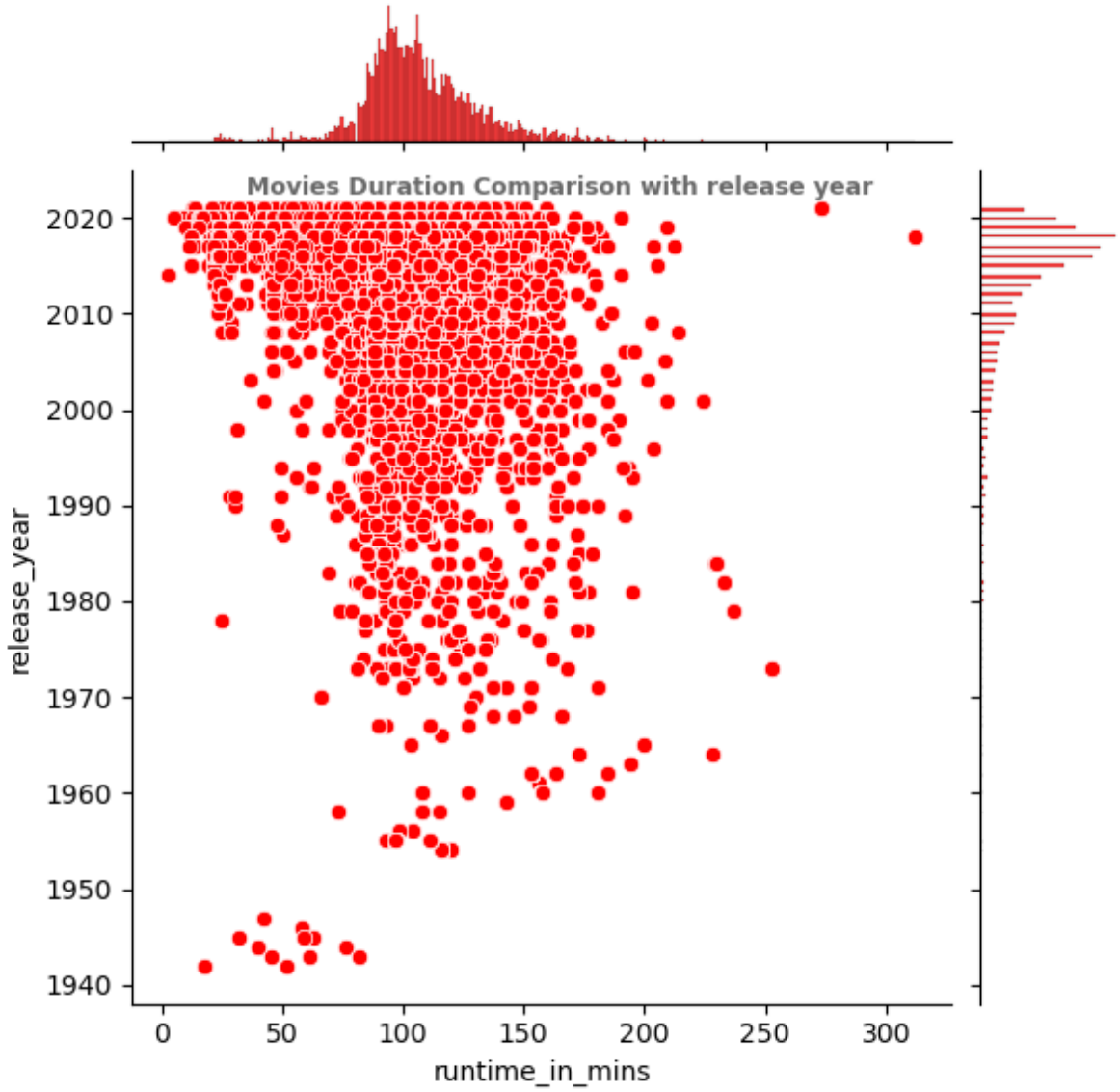
In [429…

```
sns.jointplot(nx , x='year_added' ,
              y='duration' , hue='type' ,
              palette=['red','dimgrey'])
plt.text(2007,300,'Overall Duration Comparison \n with uploaded year',color='red',
         fontsize=12,fontweight='bold')
plt.show()
```



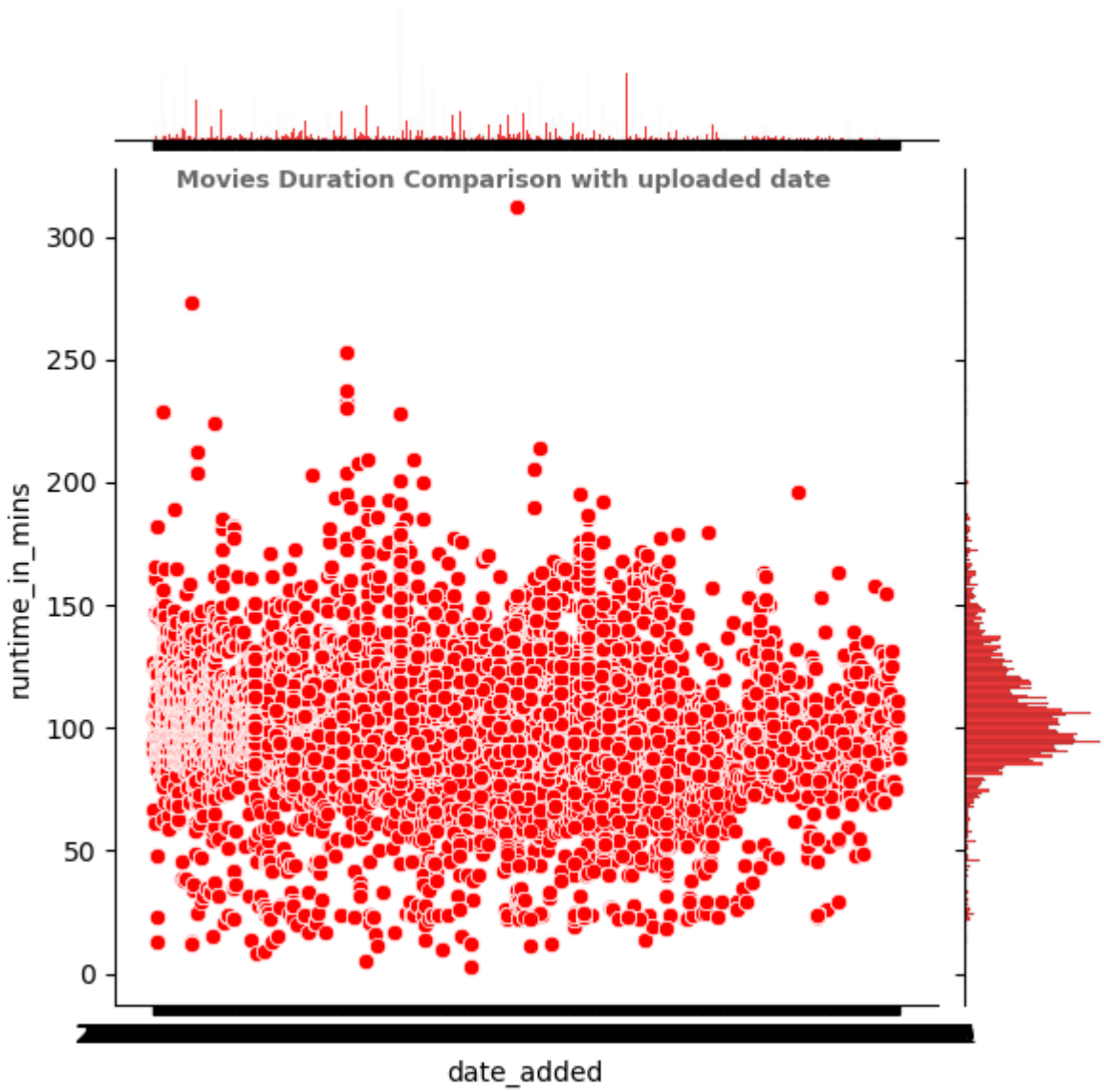
In [447…

```
sns.jointplot(md , y='release_year' , x='runtime_in_mins' , color='red')
plt.text(35,2022.5,'Movies Duration Comparison with release year',
                   color='dimgrey',fontsize=9,fontweight='bold')
plt.show()
```



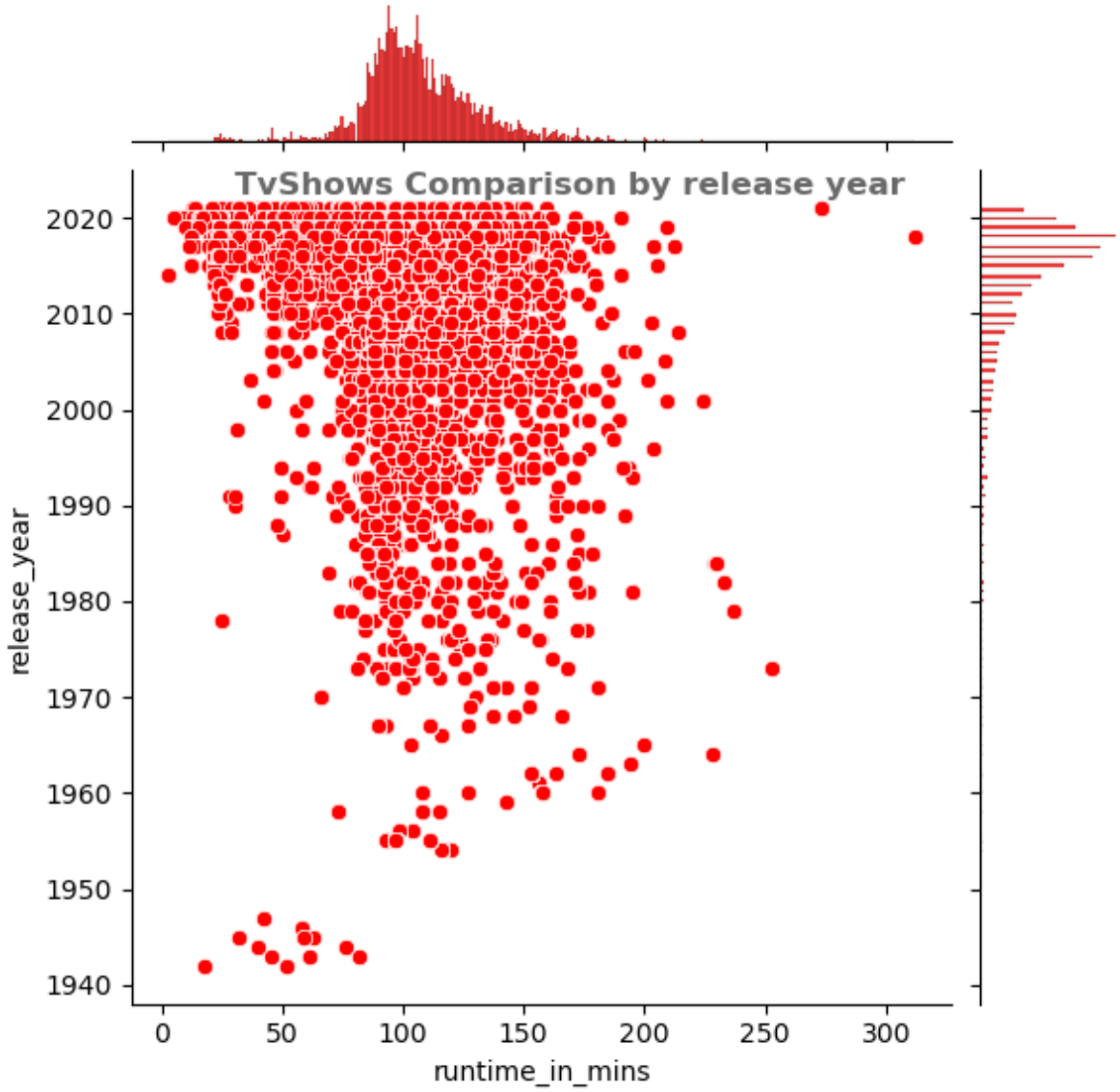
In [451…

```
sns.jointplot(md , x='date_added' , y='runtime_in_mins' , color='red')
plt.text(50,320,'Movies Duration Comparison with uploaded date',
                 color='dimgrey',fontsize=9,fontweight='bold')
plt.show()
```



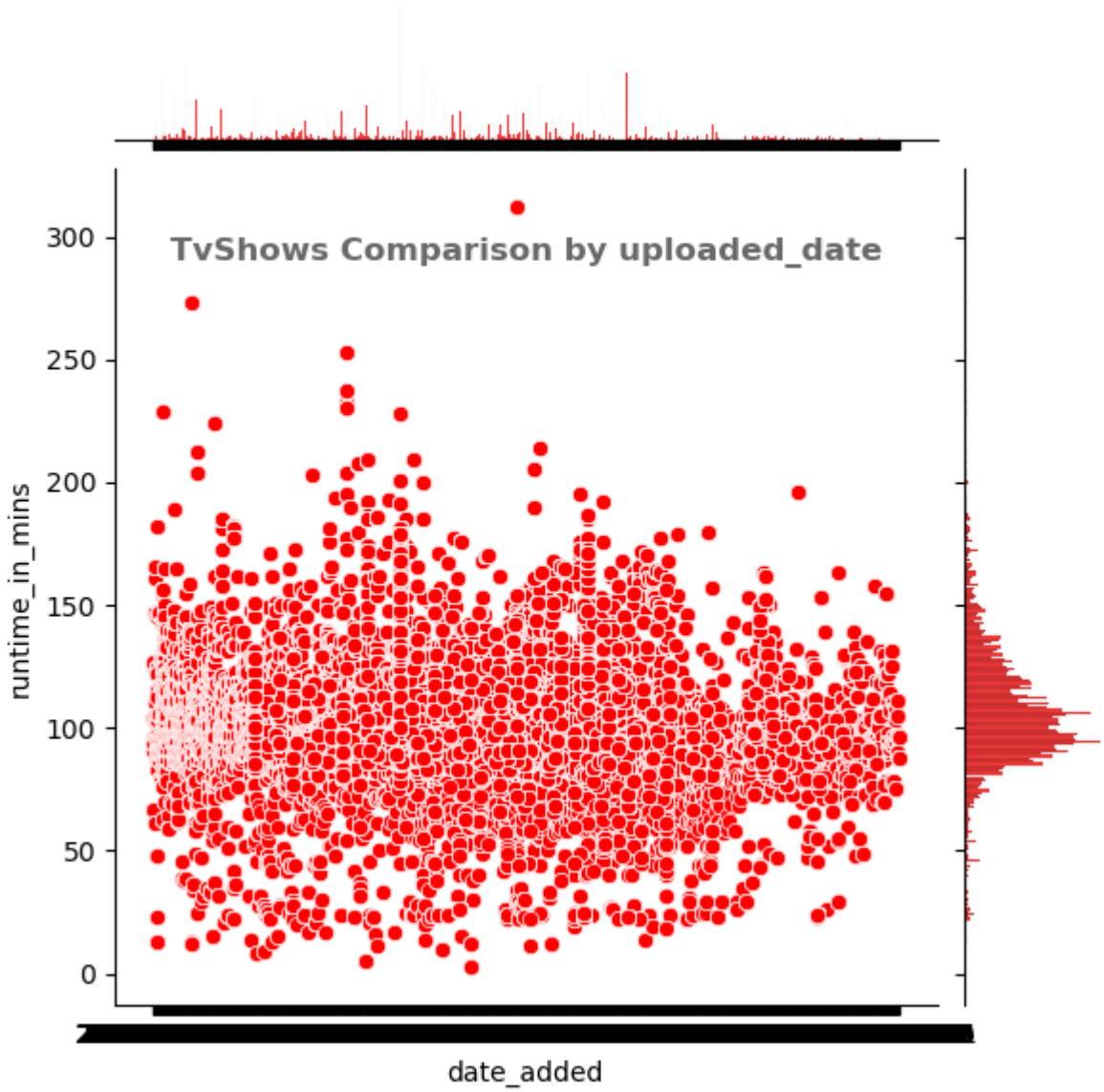
In [474…

```
sns.jointplot(md , y='release_year' , x='runtime_in_mins' , color='red')
plt.text(30,2022.4,'TvShows Comparison by release year',color='dimgrey',fontsize=12
plt.show()
```



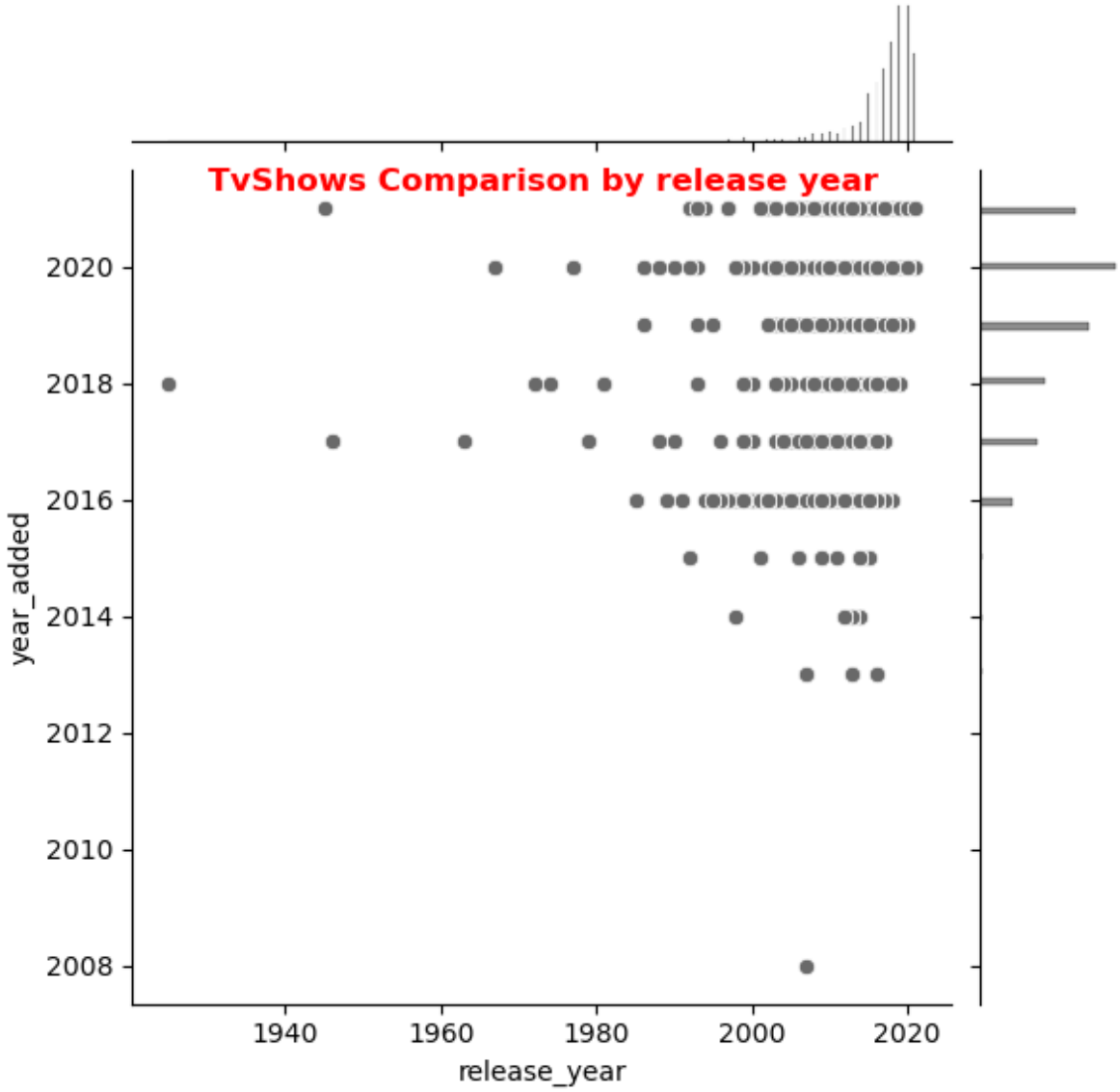
In [459…

```
sns.jointplot(md , x='date_added' , y='runtime_in_mins' , color='red')
plt.text(35,290,'TvShows Comparison by uploaded_date',color='dimgrey',fontsize=12,f
plt.show()
```



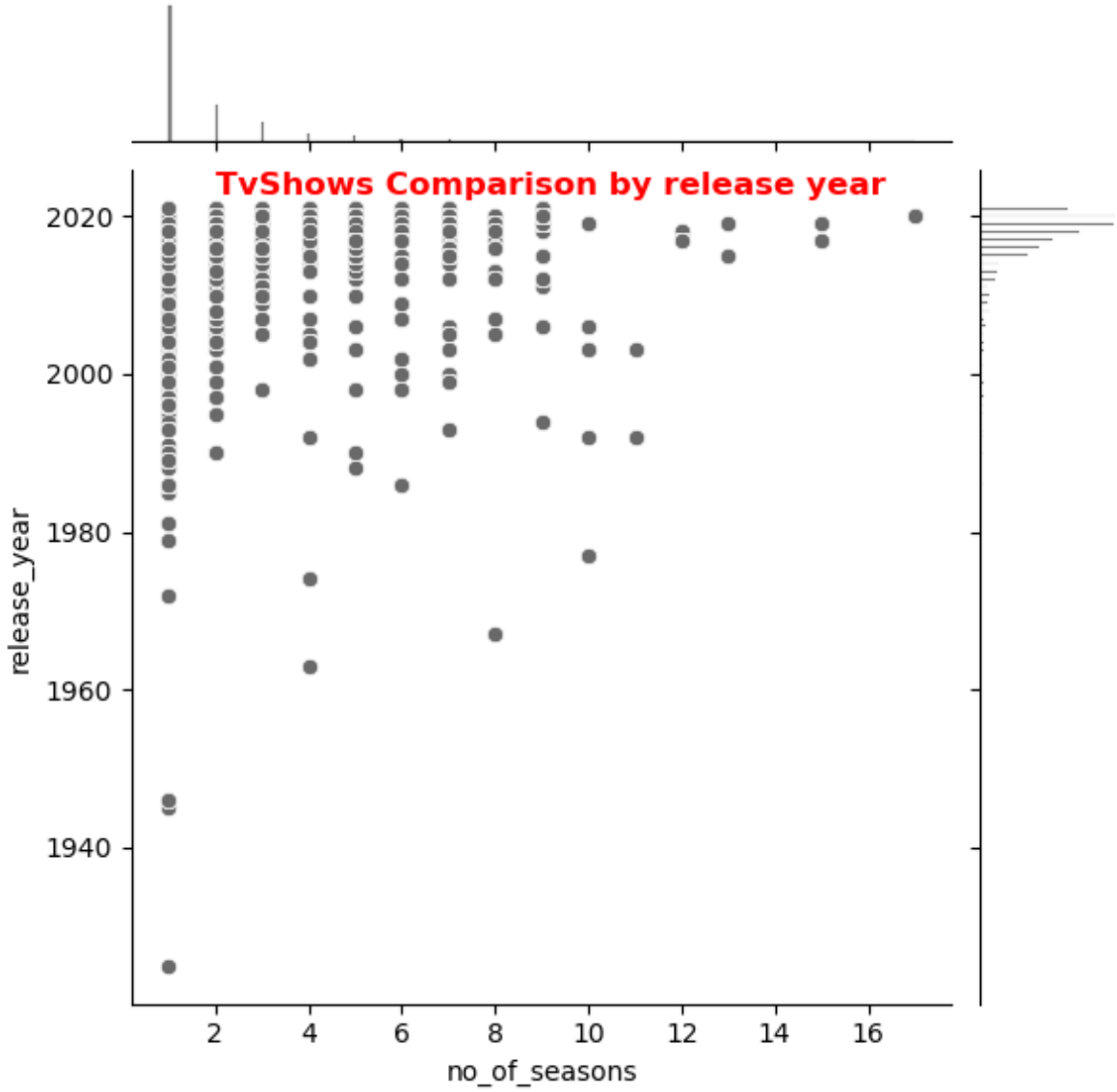
In [471…

```
sns.jointplot(tvd , x='release_year' , y='year_added' , color='dimgrey')
plt.text(1930,2021.3,'TvShows Comparison by release year',color='red',fontsize=12,f
plt.show()
```



In [469…

```
sns.jointplot(tvd , y='release_year' , x='no_of_seasons' , color='dimgrey')
plt.text(2,2022.8,'TvShows Comparison by release year',
             color='red',fontsize=12,fontweight='bold')
plt.show()
```



👉Insights :

- The majority of movies appear to have a runtime around **90-120** `minutes` . This is evident from the peak in the red line plot having highlighted the maximum value (maximum movie count) using a large silver square marker
- In the TV shows, there are a higher number of TV shows with a smaller number of seasons (e.g., **1-3** `seasons`), and the counts gradually decrease as the number of seasons increases.

📌Q. What are the ratings given for the contents uploaded on netflix ?

In [108…
```
md.groupby(['rating'])[['title']].nunique().sum()
```

Out[108]:
```
title    6131
dtype: int64
```

In [109…
```
tvd.groupby(['rating'])[['title']].nunique().sum()
```

Out[109]:
```
title    2676
dtype: int64
```

In [162…
```
movie_rating = md.groupby(['rating'])[['title']].nunique().reset_index()
movie_rating = movie_rating.sort_values(by='title',ascending=False)
```

```
movie_rating
```

Out[162]:

	rating	title
8	TV-MA	2062
6	TV-14	1427
5	R	797
9	TV-PG	540
4	PG-13	490
3	PG	287
11	TV-Y7	139
10	TV-Y	131
7	TV-G	126
2	NR	75
0	G	41
12	TV-Y7-FV	5
14	Unknown	5
1	NC-17	3
13	UR	3

In [161…

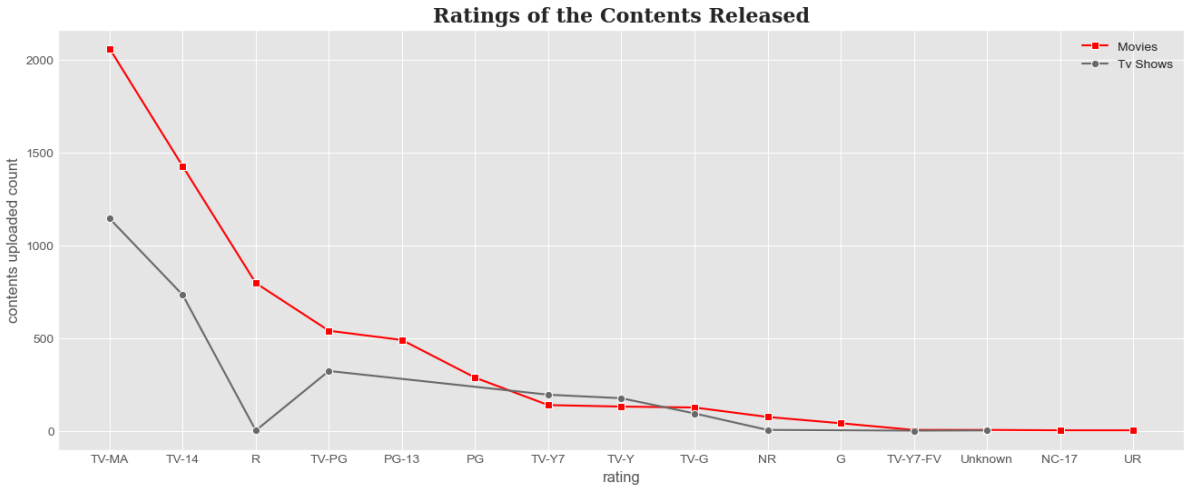
```
tv_rating = tvd.groupby(['rating'])[['title']].nunique().reset_index()
tv_rating = tv_rating.sort_values(by='title',ascending=False)
tv_rating
```

Out[161]:

	rating	title
4	TV-MA	1145
2	TV-14	733
5	TV-PG	323
7	TV-Y7	195
6	TV-Y	176
3	TV-G	94
0	NR	5
1	R	2
9	Unknown	2
8	TV-Y7-FV	1

In [164…

```
plt.figure(figsize=(16,6))
plt.style.use('ggplot')
sns.lineplot(data=movie_rating , x='rating' , y='title' , color='r' , label = 'Movi
sns.lineplot(data=tv_rating , x='rating' , y='title' , color='dimgrey', label='Tv S
plt.title('Ratings of the Contents Released',fontsize=16,fontweight="bold",fontfami
plt.ylabel('contents uploaded count')
plt.legend(loc='upper right')
plt.show()
```



👉Insights :

MOVIES

- The most common content rating is "TV-MA," with a total of 2,062 contents , typically associated with content intended for mature audiences.
- "TV-14" is the second most common rating, with 1,427 content count indicating content suitable for viewers aged 14 and older.
- "Restricted: R - Under 17 requires accompanying parent or adult guardian" is the third most common rating, with 797 titles.

TV SHOWS

- The "TV-MA" rating with 1,145 titles suggests that a significant portion of the content is intended for mature audiences.
- "TV-14" is the second most common rating, with 733 titles indicates that contents are for viewers aged 14 and older.
- "TV-PG" - parental guidance is recommended stands third with 323 contents in Tv programs.

📌Q. Diversify the actors with more contents ?

In [113…

```
movies_cast = md.groupby('cast')[['title']].nunique().sort_values(by='title',ascend
movies_cast
```

Out[113]:

cast	title
Anupam Kher	42
Shah Rukh Khan	35
Naseeruddin Shah	32
Om Puri	30
Akshay Kumar	30
Paresh Rawal	28
Julie Tejwani	28
Amitabh Bachchan	28
Rupa Bhimani	27
Boman Irani	27
Kareena Kapoor	25
Samuel L. Jackson	22
Ajay Devgn	21
Rajesh Kava	21
Nawazuddin Siddiqui	20
Nicolas Cage	20
Salman Khan	20
Adam Sandler	20
Kay Kay Menon	20

In [114…

```
tv_cast = tvd.groupby('cast')[['title']].nunique().sort_values(by='title',ascending
tv_cast
```

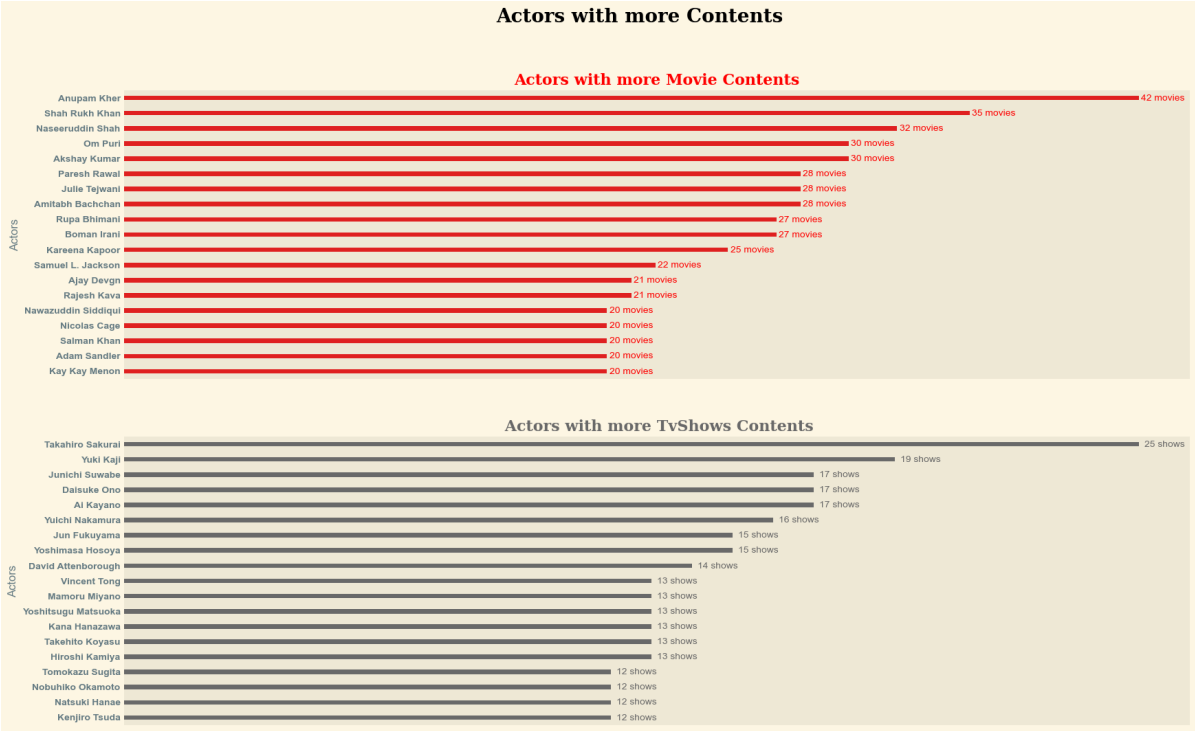
Out[114]:

cast	title
Takahiro Sakurai	25
Yuki Kaji	19
Junichi Suwabe	17
Daisuke Ono	17
Ai Kayano	17
Yuichi Nakamura	16
Jun Fukuyama	15
Yoshimasa Hosoya	15
David Attenborough	14
Vincent Tong	13
Mamoru Miyano	13
Yoshitsugu Matsuoka	13
Kana Hanazawa	13
Takehito Koyasu	13
Hiroshi Kamiya	13
Tomokazu Sugita	12
Nobuhiko Okamoto	12
Natsuki Hanae	12
Kenjiro Tsuda	12

In [116…

```
plt.figure(figsize=(20,12))
plt.suptitle('Actors with more Contents',
             fontsize=20,fontweight="bold",fontfamily='serif',color='k')
plt.style.use('Solarize_Light2')

plt.subplot(2,1,1)
c1 = sns.barplot(movies_cast, y=movies_cast.index , x='title',color='red',width=0.3
sns.despine(left=True,bottom=True,trim=True)
plt.title('Actors with more Movie Contents',
          fontsize=16,fontweight="bold",fontfamily='serif',color='r')
plt.xticks([])
plt.yticks(fontweight='bold')
plt.xlabel('')
plt.ylabel('Actors',fontsize=12)
for i in range(19):
    c1.annotate((str(movies_cast.title[i])+' movies'), (movies_cast.title[i]+1,i+0.
                ha='center' , va='bottom' , color='red')

plt.subplot(2,1,2)
c2 = sns.barplot(tv_cast, y=tv_cast.index , x='title',color='dimgrey',width=0.3)
sns.despine(left=True,bottom=True,trim=True)
plt.title('\n Actors with more TvShows Contents',
          fontsize=16,fontweight="bold",fontfamily='serif',color='dimgray')
plt.xticks([])
plt.xlabel('')
plt.yticks(fontweight='bold')
```

```
plt.ylabel('Actors',fontsize=12)
for i in range(19):
    c2.annotate((str(tv_cast.title[i])+' shows'), (tv_cast.title[i]+0.63,i+0.3),
                ha='center' , va='bottom' , color='dimgrey')

plt.show()
```



👉Insights:-

Movies

- `Anupam Kher` has appeared in the most movies (42) followed by `Shah Rukh Khan` follows closely with 35 movies, establishing himself as a prominent figure in the industry.
- `Naseeruddin Shah` , `Om Puri` , and `Akshay Kumar` all have 30 movies, indicating their significant presence in the film industry.
- Several actors and actresses, including Paresh Rawal, Julie Tejwani, Amitabh Bachchan, Rupa Bhimani, and Boman Irani, have been featured in a substantial number of movies (ranging from 27 to 28),
- The list includes actors from different film industries and countries. For instance, Samuel L. Jackson and Nicolas Cage are prominent actors from Hollywood, while others are from the Indian film industry.
- This diversity in the list reflects the global nature of the entertainment industry.

TvShows

- `Takahiro Sakurai` stands out as the Voice actor with the highest count of shows (25), indicating a prolific career in the field.
- `Yuki Kaji` , `Junichi Suwabe` , `Daisuke Ono` , and `Ai Kayano` all have significant counts, ranging from 17 to 19 titles, demonstrating their prominence in the acting industry.
- Both Japanese voice actors (seiyuu) and international actors like David Attenborough and Vincent Tong are recognized for their work in different contexts and markets.

- Many of the voice actors listed are well-known and popular among anime and animation enthusiasts.

📌Q. How much contents are being delivered by directors to content library?

In [117…
```
fmd = md.groupby('director')[['show_id']].nunique()
fmd = fmd.sort_values(by='show_id',ascending=False)[1:21]
fmd
```

Out[117]:

director	show_id
Rajiv Chilaka	22
Jan Suter	21
Raúl Campos	19
Suhas Kadav	16
Marcus Raboy	15
Jay Karas	15
Cathy Garcia-Molina	13
Martin Scorsese	12
Jay Chapman	12
Youssef Chahine	12
Steven Spielberg	11
Don Michael Paul	10
Shannon Hartman	9
Yılmaz Erdoğan	9
David Dhawan	9
Fernando Ayllón	8
Johnnie To	8
Lance Bangs	8
Hakan Algül	8
Troy Miller	8

In [118…
```
ftvd = tvd.groupby(['director'])[['show_id']].nunique()
ftvd= ftvd.sort_values(by='show_id',ascending=False)[1:21]
ftvd
```

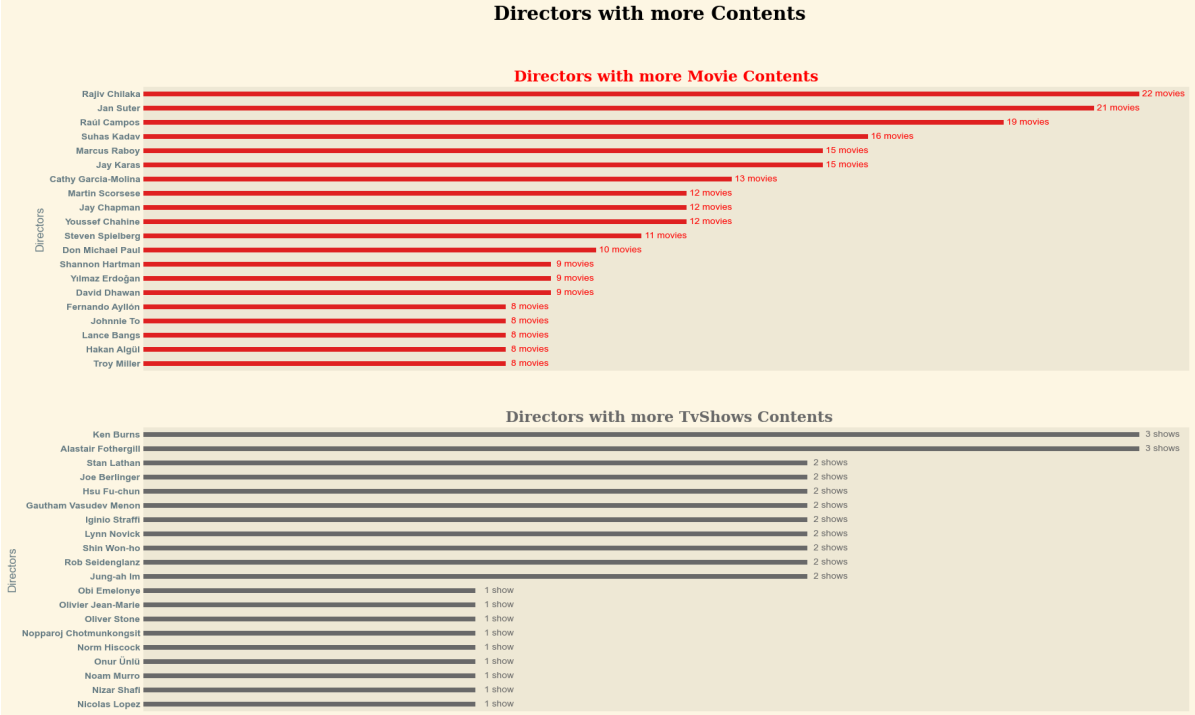
Out[118]:

director	show_id
Ken Burns	3
Alastair Fothergill	3
Stan Lathan	2
Joe Berlinger	2
Hsu Fu-chun	2
Gautham Vasudev Menon	2
Iginio Straffi	2
Lynn Novick	2
Shin Won-ho	2
Rob Seidenglanz	2
Jung-ah Im	2
Obi Emelonye	1
Olivier Jean-Marie	1
Oliver Stone	1
Nopparoj Chotmunkongsit	1
Norm Hiscock	1
Onur Ünlü	1
Noam Murro	1
Nizar Shafi	1
Nicolas Lopez	1

In [119…

```
plt.figure(figsize=(20,12))
plt.suptitle('Directors with more Contents',
             fontsize=20,fontweight="bold",fontfamily='serif',color='k')
plt.style.use('Solarize_Light2')

plt.subplot(2,1,1)
c1 = sns.barplot(fmd, y=fmd.index , x='show_id',color='red',width=0.3)
sns.despine(left=True,bottom=True,trim=True)
plt.title('Directors with more Movie Contents',
          fontsize=16,fontweight="bold",fontfamily='serif',color='r')
plt.xticks([])
plt.yticks(fontweight='bold')
plt.xlabel('')
plt.ylabel('Directors',fontsize=12)
for i in range(20):
    c1.annotate((str(fmd.show_id[i])+' movies'), (fmd.show_id[i]+0.53,i+0.3),
                ha='center' , va='bottom' , color='red')

plt.subplot(2,1,2)
c2 = sns.barplot(ftvd, y=ftvd.index , x='show_id',color='dimgrey',width=0.3)
sns.despine(left=True,bottom=True,trim=True)
plt.title('\n Directors with more TvShows Contents',
          fontsize=16,fontweight="bold",fontfamily='serif',color='dimgray')
plt.xticks([])
```

```
plt.xlabel('')
plt.yticks(fontweight='bold')
plt.ylabel('Directors',fontsize=12)
for i in range(20):
    if ftvd.show_id[i]>1:
        c2.annotate((str(ftvd.show_id[i])+' shows'),(ftvd.show_id[i]+0.07,i+0.3),
                    ha='center' , va='bottom' , color='dimgrey')
    else:
        c2.annotate((str(ftvd.show_id[i])+' show'),(ftvd.show_id[i]+0.07,i+0.3),
                    ha='center' , va='bottom' , color='dimgrey')
plt.show()
```



👉Insights:-

Movies

- `Rajiv Chilaka` is the filmmaker with the highest number of movies, having directed 22 films. This shows a long and prolific career in filmmaking career and he is the `CEO of GREEN GOLD ANIMATIONS` - Movies like ChottaBheem series,chorr police attracted a huge audiences.
- `Jan Suter` and `Raúl Campos` also have impressive directorial careers, with 21 and 19 movies, respectively.
- This includes directors from various countries, such as `India` (Rajiv Chilaka, Suhas Kadav, David Dhawan), the `United States` (Jan Suter, Marcus Raboy, Jay Karas, Cathy Garcia-Molina, Steven Spielberg, Don Michael Paul, Shannon Hartman, Troy Miller), `Egypt` (Youssef Chahine), and more.

TvShows

- Some directors have worked on multiple shows. For example, `Ken Burns` and `Alastair Fothergill` have directed three shows each, making them prolific in their contributions to these projects.
- The list of directors includes individuals from various backgrounds and countries, such as the `United States` (Ken Burns, Stan Lathan, Joe Berlinger, Lynn Novick, Rob Seidenglanz, Norm Hiscock, Oliver Stone), `South Korea` (Shin Won-ho), `India`

(Gautham Vasudev Menon), `Italy` (Iginio Straffi), and more. This reflects the international reach and diversity of directors in the entertainment industry.

- Directors like **Ken Burns, Alastair Fothergill, and Lynn Novick** have been involved in multiple projects, indicating their impact on the world of documentaries and shows.

📌 Q. Who is the Actor-Directors worked with each other the most ?

In [84]:
```
ad = nx[['cast','show_id','director','type']]
ad = ad[ad.cast!='Unknown actors']
ad = ad[ad.director!='Unknown director']
ad = ad.drop_duplicates().reset_index(drop=True)
ad
```

Out[84]:

	cast	show_id	director	type
0	Sami Bouajila	s3	Julien Leclercq	TV Show
1	Tracy Gotoas	s3	Julien Leclercq	TV Show
2	Samuel Jouy	s3	Julien Leclercq	TV Show
3	Nabiha Akkari	s3	Julien Leclercq	TV Show
4	Sofia Lesaffre	s3	Julien Leclercq	TV Show
...	...	...	...	...
51176	Manish Chaudhary	s8807	Mozez Singh	Movie
51177	Meghna Malik	s8807	Mozez Singh	Movie
51178	Malkeet Rauni	s8807	Mozez Singh	Movie
51179	Anita Shabdish	s8807	Mozez Singh	Movie
51180	Chittaranjan Tripathy	s8807	Mozez Singh	Movie

51181 rows × 4 columns

In [85]:
```
nad = ad.groupby(['cast','director','type'])[['show_id']].nunique()
new_ad = nad.reset_index().sort_values(by='show_id',ascending=False)
new_ad['ad_pair'] = new_ad['cast']+'-'+new_ad['director']
new_ad
```

Out[85]:

	cast	director	type	show_id	ad_pair
22507	Julie Tejwani	Rajiv Chilaka	Movie	19	Julie Tejwani-Rajiv Chilaka
36246	Rajesh Kava	Rajiv Chilaka	Movie	19	Rajesh Kava-Rajiv Chilaka
38580	Rupa Bhimani	Rajiv Chilaka	Movie	18	Rupa Bhimani-Rajiv Chilaka
20310	Jigna Bhardwaj	Rajiv Chilaka	Movie	18	Jigna Bhardwaj-Rajiv Chilaka
45933	Vatsal Dubey	Rajiv Chilaka	Movie	16	Vatsal Dubey-Rajiv Chilaka
...	...	...	...	...	...
16454	Harish Roy	Keerthi	Movie	1	Harish Roy-Keerthi
16455	Harish Verma	Amarpreet G S Chabbra	Movie	1	Harish Verma-Amarpreet G S Chabbra
16456	Harish Verma	Mukesh Vohra	Movie	1	Harish Verma-Mukesh Vohra
16457	Harish Verma	Saket Behl	Movie	1	Harish Verma-Saket Behl
48308	Ṣọpẹ́ Dìrísù	Remi Weekes	Movie	1	Ṣọpẹ́ Dìrísù-Remi Weekes

48309 rows × 5 columns

In [86]:
```
mad = new_ad[new_ad.type=='Movie']
tvad = new_ad[new_ad.type=='TV Show']
```

In [87]:
```
mad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 45952 entries, 22507 to 48308
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   cast      45952 non-null  object
 1   director  45952 non-null  object
 2   type      45952 non-null  object
 3   show_id   45952 non-null  int64 
 4   ad_pair   45952 non-null  object
dtypes: int64(1), object(4)
memory usage: 2.1+ MB
```

In [88]:
```
mad = mad[['ad_pair','show_id']]
mad
```

Out[88]:

	ad_pair	show_id
22507	Julie Tejwani-Rajiv Chilaka	19
36246	Rajesh Kava-Rajiv Chilaka	19
38580	Rupa Bhimani-Rajiv Chilaka	18
20310	Jigna Bhardwaj-Rajiv Chilaka	18
45933	Vatsal Dubey-Rajiv Chilaka	16
...	...	...
16454	Harish Roy-Keerthi	1
16455	Harish Verma-Amarpreet G S Chabbra	1
16456	Harish Verma-Mukesh Vohra	1
16457	Harish Verma-Saket Behl	1
48308	Ṣọpẹ́ Dìrísù-Remi Weekes	1

45952 rows × 2 columns

In [89]:
```
tvad = tvad[['ad_pair','show_id']]
tvad
```

Out[89]:

	ad_pair	show_id
10574	David Attenborough-Alastair Fothergill	3
3306	Anjali-Gautham Vasudev Menon	2
15050	Gautham Vasudev Menon-Gautham Vasudev Menon	2
10508	Dave Chappelle-Stan Lathan	2
42944	Sung Dong-il-Shin Won-ho	2
...	...	...
16552	Harry Lloyd-Michael Samuels	1
16437	Hari-Gautham Vasudev Menon	1
16438	Hari-Sudha Kongara	1
16439	Hari-Vetri Maaran	1
16440	Hari-Vignesh Shivan	1

2357 rows × 2 columns

In [90]:
```
mad.dtypes
```

Out[90]:
```
ad_pair    object
show_id     int64
dtype: object
```

In [91]:
```
fmad = mad[:25].set_index('ad_pair')
ftvad = tvad[:25].set_index('ad_pair')
```

In [92]:
```
fmad
```

Out[92]:

ad_pair	show_id
Julie Tejwani-Rajiv Chilaka	19
Rajesh Kava-Rajiv Chilaka	19
Rupa Bhimani-Rajiv Chilaka	18
Jigna Bhardwaj-Rajiv Chilaka	18
Vatsal Dubey-Rajiv Chilaka	16
Mousam-Rajiv Chilaka	13
Swapnil-Rajiv Chilaka	13
Saurav Chakraborty-Suhas Kadav	8
Anushka Shetty-S.S. Rajamouli	7
Nassar-S.S. Rajamouli	7
Houko Kuwashima-Toshiya Shinohara	7
Ramya Krishnan-S.S. Rajamouli	7
Satsuki Yukino-Toshiya Shinohara	7
Sathyaraj-S.S. Rajamouli	7
Prabhas-S.S. Rajamouli	7
Rana Daggubati-S.S. Rajamouli	7
Tamannaah Bhatia-S.S. Rajamouli	7
Yılmaz Erdoğan-Yılmaz Erdoğan	7
Koji Tsujitani-Toshiya Shinohara	7
Kappei Yamaguchi-Toshiya Shinohara	7
Kumiko Watanabe-Toshiya Shinohara	7
Smita Malhotra-Tilak Shetty	6
Ata Demirer-Hakan Algül	6
Joseph May-Joey So	6
Ricardo Quevedo-Fernando Ayllón	6

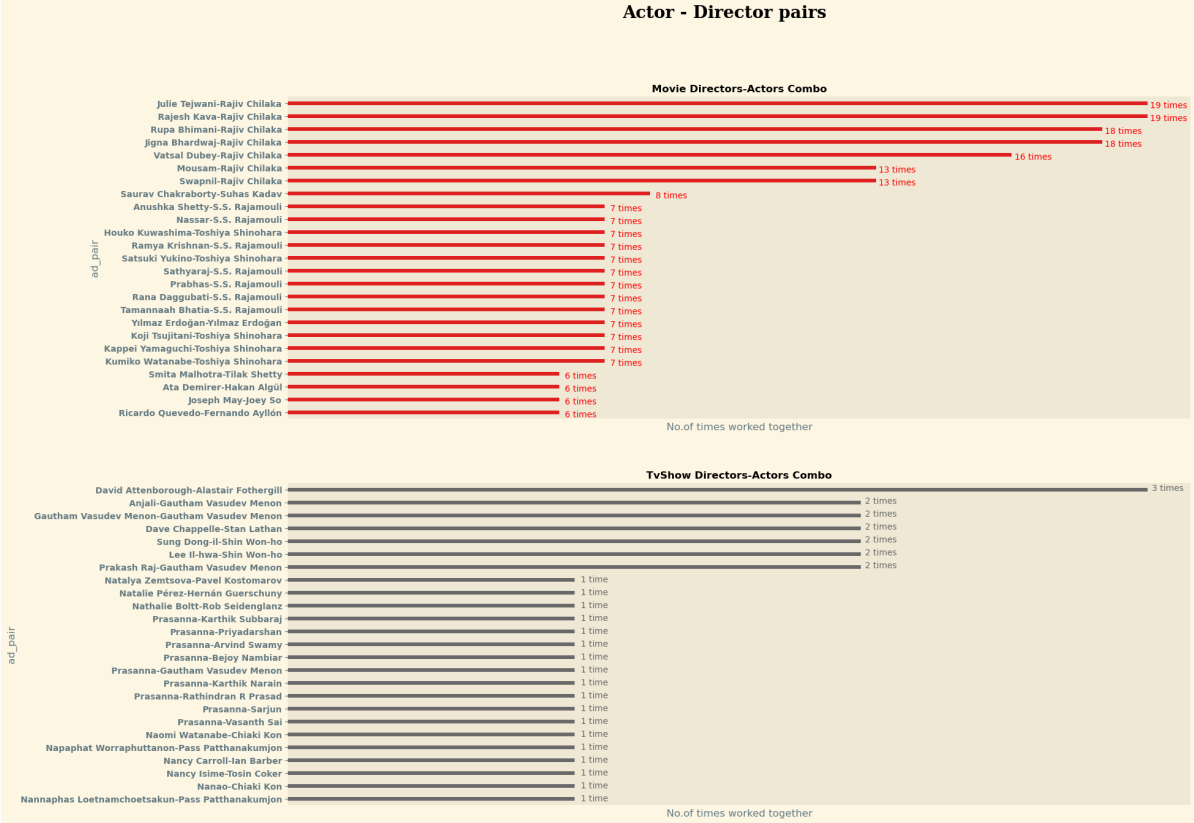
In [93]:
```
ftvad
```

Out[93]:

ad_pair	show_id
David Attenborough-Alastair Fothergill	3
Anjali-Gautham Vasudev Menon	2
Gautham Vasudev Menon-Gautham Vasudev Menon	2
Dave Chappelle-Stan Lathan	2
Sung Dong-il-Shin Won-ho	2
Lee Il-hwa-Shin Won-ho	2
Prakash Raj-Gautham Vasudev Menon	2
Natalya Zemtsova-Pavel Kostomarov	1
Natalie Pérez-Hernán Guerschuny	1
Nathalie Boltt-Rob Seidenglanz	1
Prasanna-Karthik Subbaraj	1
Prasanna-Priyadarshan	1
Prasanna-Arvind Swamy	1
Prasanna-Bejoy Nambiar	1
Prasanna-Gautham Vasudev Menon	1
Prasanna-Karthik Narain	1
Prasanna-Rathindran R Prasad	1
Prasanna-Sarjun	1
Prasanna-Vasanth Sai	1
Naomi Watanabe-Chiaki Kon	1
Napaphat Worraphuttanon-Pass Patthanakumjon	1
Nancy Carroll-Ian Barber	1
Nancy Isime-Tosin Coker	1
Nanao-Chiaki Kon	1
Nannaphas Loetnamchoetsakun-Pass Patthanakumjon	1

In [94]:
```
plt.figure(figsize=(19, 15))
plt.suptitle('Actor - Director pairs',fontsize=20,
                fontweight="bold",fontfamily='serif')
plt.style.use('default')
plt.style.use('Solarize_Light2')

plt.subplot(2,1,1)
a1 = sns.barplot(y=fmad.index, x=fmad.show_id, color='red',width=0.3)
plt.title('Movie Directors-Actors Combo',fontsize=12,fontweight="bold")
sns.despine(left=True,bottom=True,trim=True)
plt.yticks(fontweight='bold')
plt.xticks([])
plt.xlabel('No.of times worked together')
for i in range(25):
    a1.annotate((str(fmad.show_id[i])+' times'), (fmad.show_id[i]+0.47,i+0.5),
                ha='center' , va='bottom' , color='red')
```

```
plt.subplot(2,1,2)
a2 = sns.barplot(ftvad , y=ftvad.index, x=ftvad.show_id, color='dimgrey',width=0.3)
plt.title('TvShow Directors-Actors Combo',fontsize=12,fontweight="bold")
sns.despine(left=True,bottom=True,trim=True)
plt.yticks(fontweight='bold')
plt.xticks([])
plt.xlabel('No.of times worked together')
for i in range(25):
    if ftvad.show_id[i]>1:
        a2.annotate((str(ftvad.show_id[i])+' times'), (ftvad.show_id[i]+0.07,i+0.2)
                ha='center' , va='bottom' , color='dimgrey')
    else:
        a2.annotate((str(ftvad.show_id[i])+' time'), (ftvad.show_id[i]+0.07,i+0.3),
                ha='center' , va='bottom' , color='dimgrey')

plt.show()
```



👉Insights :

- `Rajiv Chilaka` , `SS.Rajamouli` & `Gautam vasudev Menon` are the most comfortable directors to work with and has the high number repeat rate by which we can interpret that they are success combo to repeat both contentwise and financial profit yielding.
- similary `David attenborough-Alister Fothergill` and `Houko Kuwashima-Toshiya Shinohara` are repeat combos and yet they are successful repeat combos.

📌Q. What is the best time to launch a movie?

In [123…

```
md.columns
```

Out[123]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'listed_in', 'year_added', 'runtime_in_mins'],
      dtype='object')
```

In [124…
```
movie_release = md[['show_id','title','date_added']]
movie_release = movie_release.reset_index(drop=True)
movie_release
```

Out[124]:

	show_id	title	date_added
0	s1	Dick Johnson Is Dead	2021-09-25
1	s7	My Little Pony: A New Generation	2021-09-24
2	s7	My Little Pony: A New Generation	2021-09-24
3	s7	My Little Pony: A New Generation	2021-09-24
4	s7	My Little Pony: A New Generation	2021-09-24
…	…	…	…
145838	s8807	Zubaan	2019-03-02
145839	s8807	Zubaan	2019-03-02
145840	s8807	Zubaan	2019-03-02
145841	s8807	Zubaan	2019-03-02
145842	s8807	Zubaan	2019-03-02

145843 rows × 3 columns

In [32]:
```
movie_release.dtypes
```

Out[32]:
```
show_id       object
title         object
date_added    object
dtype: object
```

In [125…
```
movie_release['date_added'] = pd.to_datetime(movie_release['date_added'])
```

In [34]:
```
movie_release.dtypes
```

Out[34]:
```
show_id               object
title                 object
date_added    datetime64[ns]
dtype: object
```

In [35]:
```
movie_release.isna().sum()
```

Out[35]:
```
show_id       0
title         0
date_added    0
dtype: int64
```

In [126…
```
movie_release['week_uploaded'] = movie_release['date_added'].dt.isocalendar().week
movie_release['uploaded_weekday'] = movie_release['date_added'].dt.strftime('%A')
movie_release['uploaded_month'] = movie_release['date_added'].dt.strftime('%B')
```

In [127…
```
month_order = ['January', 'February', 'March', 'April', 'May',
               'June', 'July', 'August', 'September',
               'October', 'November', 'December']
movie_release['uploaded_month']= pd.Categorical(movie_release['uploaded_month'],
                                                categories=month_order, ordered=True)
```

In [58]:
```
movie_release
```

Out[58]:

	show_id	title	date_added	week_uploaded	uploaded_weekday	uploaded_month
0	s1	Dick Johnson Is Dead	2021-09-25	38	Saturday	September
1	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	September
2	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	September
3	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	September
4	s7	My Little Pony: A New Generation	2021-09-24	38	Friday	September
...	...	...	...	...	...	...
145838	s8807	Zubaan	2019-03-02	9	Saturday	March
145839	s8807	Zubaan	2019-03-02	9	Saturday	March
145840	s8807	Zubaan	2019-03-02	9	Saturday	March
145841	s8807	Zubaan	2019-03-02	9	Saturday	March
145842	s8807	Zubaan	2019-03-02	9	Saturday	March

145843 rows × 6 columns

In [128...
```
week_movie_release=movie_release.groupby('week_uploaded')['show_id'].nunique()
week_movie_release=week_movie_release.reset_index()
week_movie_release
```

Out[128]:

	week_uploaded	show_id
0	1	316
1	2	78
2	3	81
3	4	56
4	5	135
5	6	64
6	7	106
7	8	72
8	9	207
9	10	107
10	11	115
11	12	67
12	13	174
13	14	124
14	15	100
15	16	124
16	17	109
17	18	173
18	19	73
19	20	85
20	21	76
21	22	146
22	23	112
23	24	89
24	25	101
25	26	195
26	27	154
27	28	89
28	29	94
29	30	116
30	31	185
31	32	73
32	33	105
33	34	102
34	35	189
35	36	97

	week_uploaded	show_id
36	37	114
37	38	88
38	39	111
39	40	215
40	41	84
41	42	90
42	43	88
43	44	243
44	45	61
45	46	83
46	47	85
47	48	139
48	49	95
49	50	119
50	51	86
51	52	80
52	53	61

In [50]:
```
week_movie_release.sum()
```

Out[50]:
```
6131
```

In [129…
```
monthly_movie_release=movie_release.groupby('uploaded_month')['show_id'].nunique()
monthly_movie_release = monthly_movie_release.reset_index()
monthly_movie_release = monthly_movie_release.sort_values(by='uploaded_month')
monthly_movie_release.reset_index(drop=True)
monthly_movie_release
```

Out[129]:

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

In [52]:
```
monthly_movie_release.title.sum()
```

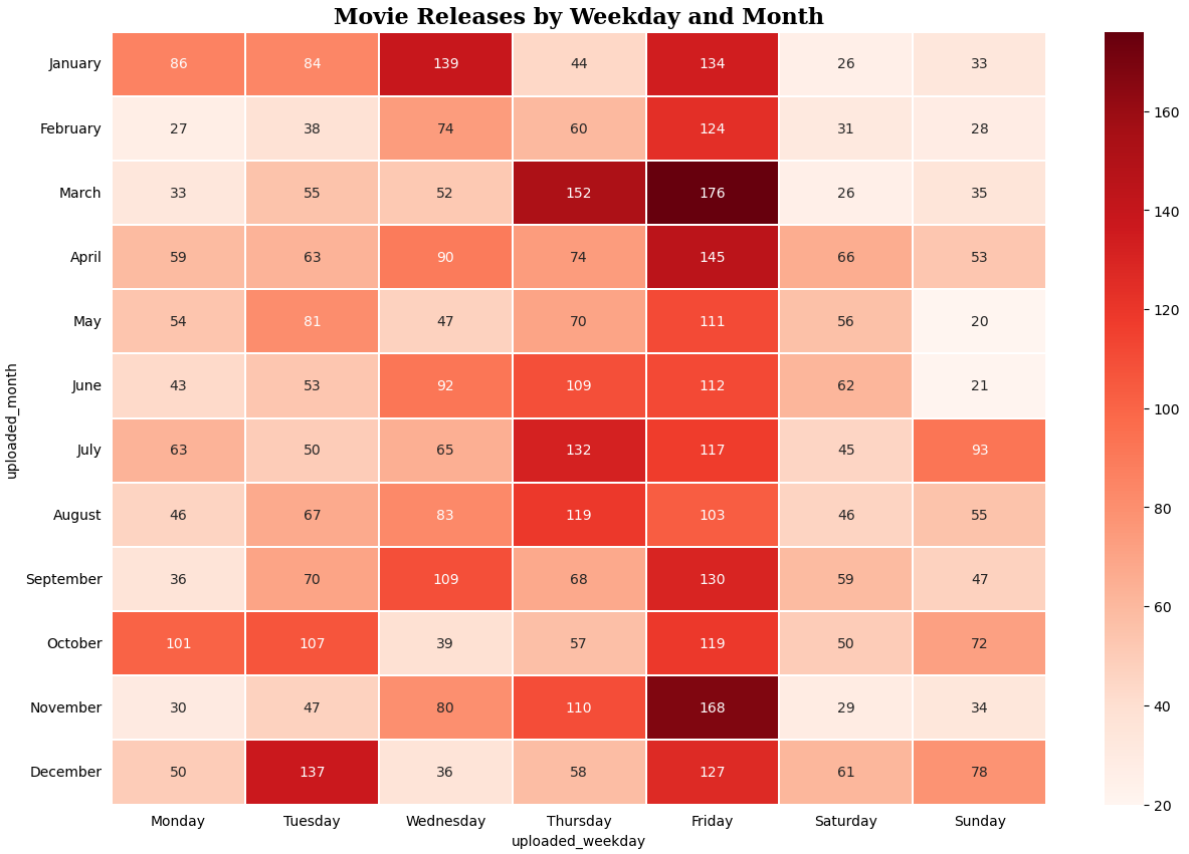
Out[52]:
```
6131
```

In [130…
```
movies_release_pivot = movie_release.pivot_table(index='uploaded_month',
                                                 columns='uploaded_weekday',
                                                 values='show_id',
                                                 aggfunc=pd.Series.nunique)

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
             'Friday', 'Saturday','Sunday']
movies_release_pivot = movies_release_pivot[day_order]
movies_release_pivot
```

Out[130]:

uploaded_weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
uploaded_month							
January	86	84	139	44	134	26	33
February	27	38	74	60	124	31	28
March	33	55	52	152	176	26	35
April	59	63	90	74	145	66	53
May	54	81	47	70	111	56	20
June	43	53	92	109	112	62	21
July	63	50	65	132	117	45	93
August	46	67	83	119	103	46	55
September	36	70	109	68	130	59	47
October	101	107	39	57	119	50	72
November	30	47	80	110	168	29	34
December	50	137	36	58	127	61	78

In [134…
```
plt.figure(figsize=(15, 10))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
sns.heatmap(movies_release_pivot, cmap='Reds',
                annot=True, fmt='d' , linewidth=0.1)
plt.title("Movie Releases by Weekday and Month",
             fontfamily='serif',fontsize=16,fontweight='bold')
plt.tick_params(axis='both', which='both', left=False, bottom=False)
plt.show()
```



In [77]:
```
movies_release_pivot.sum().sort_values(ascending=False)
```

Out[77]:
```
uploaded_weekday
Friday       1566
Thursday     1053
Wednesday     906
Tuesday       852
Monday        628
Sunday        569
Saturday      557
dtype: int64
```

In [78]:
```
movies_release_pivot.sum().sum()
```

Out[78]:
```
6131
```

📌 Q. What is the best time to launch a Tvshow ?

In [371…
```
tvs_release = tvd[['show_id','title','date_added']]
tvs_release = tvs_release.reset_index(drop=True)
tvs_release
```

Out[371]:

	show_id	title	date_added
0	s2	Blood & Water	2021-09-24
1	s2	Blood & Water	2021-09-24
2	s2	Blood & Water	2021-09-24
3	s2	Blood & Water	2021-09-24
4	s2	Blood & Water	2021-09-24
...	...	...	...
56143	s8801	Zindagi Gulzar Hai	2016-12-15
56144	s8801	Zindagi Gulzar Hai	2016-12-15
56145	s8804	Zombie Dumb	2019-07-01
56146	s8804	Zombie Dumb	2019-07-01
56147	s8804	Zombie Dumb	2019-07-01

56148 rows × 3 columns

In [80]:
```
tvs_release.dtypes
```

Out[80]:
```
show_id       object
title         object
date_added    object
dtype: object
```

In [81]:
```
tvs_release.isna().sum()
```

Out[81]:
```
show_id       0
title         0
date_added    0
dtype: int64
```

In [372…
```
tvs_release['date_added'].fillna(tvs_release['date_added'].mode()[0],inplace=True)
```

In [373…
```
tvs_release['date_added'] = pd.to_datetime(tvs_release['date_added'])
```

In [374…
```
tvs_release['date_added'].dtypes
```

Out[374]:
```
dtype('<M8[ns]')
```

In [375…
```
tvs_release['week_uploaded'] = tvs_release['date_added'].dt.isocalendar().week
tvs_release['uploaded_weekday'] = tvs_release['date_added'].dt.strftime('%A')
tvs_release['uploaded_month'] = tvs_release['date_added'].dt.strftime('%B')
```

In [376…
```
month_order = ['January', 'February', 'March', 'April', 'May',
               'June', 'July', 'August', 'September',
               'October', 'November', 'December']
tvs_release['uploaded_month']= pd.Categorical(tvs_release['uploaded_month'],
                                    categories=month_order, ordered=True)
```

In [88]:
```
tvs_release
```

Out[88]:

	show_id	title	date_added	week_uploaded	uploaded_weekday	uploaded_month
0	s2	Blood & Water	2021-09-24	38	Friday	September
1	s2	Blood & Water	2021-09-24	38	Friday	September
2	s2	Blood & Water	2021-09-24	38	Friday	September
3	s2	Blood & Water	2021-09-24	38	Friday	September
4	s2	Blood & Water	2021-09-24	38	Friday	September
...	...	...	...	...	...	...
56143	s8801	Zindagi Gulzar Hai	2016-12-15	50	Thursday	December
56144	s8801	Zindagi Gulzar Hai	2016-12-15	50	Thursday	December
56145	s8804	Zombie Dumb	2019-07-01	27	Monday	July
56146	s8804	Zombie Dumb	2019-07-01	27	Monday	July
56147	s8804	Zombie Dumb	2019-07-01	27	Monday	July

56148 rows × 6 columns

In [89]:
```
tvs_release.groupby('week_uploaded')['show_id'].nunique().sum()
```

Out[89]:
```
2676
```

In [377…
```
week_release = tvs_release.groupby('week_uploaded')['show_id'].nunique()
week_release = week_release.reset_index()
week_release
```

Out[377]:

	week_uploaded	show_id
0	1	150
1	2	26
2	3	31
3	4	31
4	5	68
5	6	33
6	7	41
7	8	37
8	9	46
9	10	28
10	11	46
11	12	40
12	13	73
13	14	48
14	15	50
15	16	34
16	17	45
17	18	60
18	19	43
19	20	44
20	21	39
21	22	56
22	23	39
23	24	75
24	25	42
25	26	69
26	27	85
27	28	40
28	29	44
29	30	43
30	31	79
31	32	49
32	33	47
33	34	40
34	35	73
35	36	44

	week_uploaded	show_id
36	37	67
37	38	50
38	39	55
39	40	69
40	41	31
41	42	45
42	43	28
43	44	67
44	45	36
45	46	51
46	47	35
47	48	56
48	49	44
49	50	65
50	51	48
51	52	50
52	53	41

In [378…

```
month_release = tvs_release.groupby('uploaded_month')['show_id'].nunique()
month_release = month_release.reset_index()
month_release
```

Out[378]:

	uploaded_month	show_id
0	January	279
1	February	175
2	March	205
3	April	209
4	May	187
5	June	232
6	July	254
7	August	230
8	September	246
9	October	210
10	November	199
11	December	250

In [94]:

```
month_release.show_id.sum()
```

Out[94]:

```
2676
```

In [379…
```
tvs_release_pivot = tvs_release.pivot_table(
            index='uploaded_month' ,
            columns='uploaded_weekday' ,
            values='show_id' ,
            aggfunc=pd.Series.nunique
    )

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
                'Friday', 'Saturday','Sunday']
tvs_release_pivot = tvs_release_pivot[day_order]
tvs_release_pivot
```

Out[379]:

uploaded_month \ uploaded_weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
January	13	19	128	19	85	5	10
February	20	13	28	27	63	17	7
March	11	25	32	22	88	16	11
April	20	20	38	38	58	21	14
May	21	26	25	15	74	17	9
June	22	16	35	29	77	47	6
July	22	50	32	29	82	29	10
August	23	41	30	26	80	18	12
September	13	29	44	34	87	15	24
October	19	25	21	35	64	26	20
November	6	29	27	21	85	11	20
December	27	37	28	39	67	24	28

In [98]:
```
tvs_release_pivot.sum(axis=1)
```

Out[98]:
```
uploaded_month
January      279
February     175
March        205
April        209
May          187
June         232
July         254
August       230
September    246
October      210
November     199
December     250
dtype: int64
```

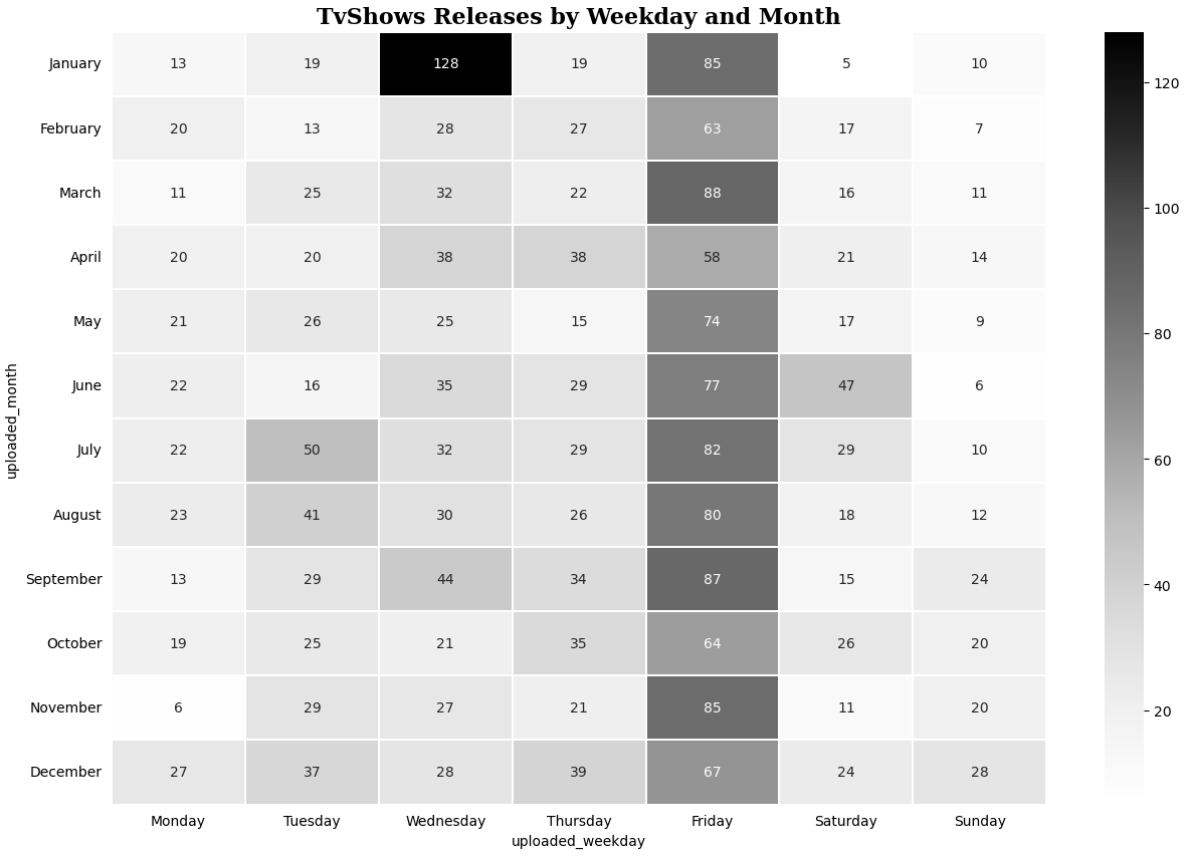
In [99]:
```
tvs_release_pivot.sum().sort_values(ascending=False)
```

Out[99]:
```
uploaded_weekday
Friday       910
Wednesday    468
Thursday     334
Tuesday      330
Saturday     246
Monday       217
Sunday       171
dtype: int64
```

In [100…
```
tvs_release_pivot.sum().sum()
```

Out[100]:
```
2676
```

In [380…
```
plt.figure(figsize=(15, 10))
plt.style.use('seaborn-v0_8-bright')
sns.heatmap(tvs_release_pivot, cmap='Greys', annot=True,
                fmt='d' , linewidth=0.1)
plt.title("TvShows Releases by Weekday and Month",
                 fontfamily='serif',fontsize=16,fontweight='bold')
plt.tick_params(axis='both', which='both', left=False, bottom=False)
plt.show()
```



In [112…
```
plt.figure(figsize=(30,15))
plt.suptitle('Releases Broader view',fontfamily='serif',
                         fontsize=20,fontweight='bold')

plt.subplot(2,2,1)
sns.pointplot(week_movie_release ,x='week_uploaded' , y='show_id' ,color='r')
plt.title('Movie Weekly Releases count',fontfamily='serif',
                      fontsize=16,fontweight='bold')
plt.xlabel('Week Number')
plt.ylabel('no.of contents released')

plt.subplot(2,2,3)
sns.pointplot(monthly_movie_release ,x='uploaded_month' , y='show_id' ,color='r')
```

```
plt.title('Movie Monthly Releases count',fontfamily='serif',
                      fontsize=16,fontweight='bold')
plt.xlabel('Month')
plt.ylabel('no.of contents released')

plt.subplot(2,2,2)
sns.pointplot(week_release ,x='week_uploaded' , y='show_id' ,color='dimgrey')
plt.title('Tv Shows Weekly Releases count',fontfamily='serif',
                      fontsize=16,fontweight='bold')
plt.xlabel('Week Number')
plt.ylabel('no.of contents released')

plt.subplot(2,2,4)
sns.pointplot(month_release ,x='uploaded_month' , y='show_id' ,color='dimgrey')
plt.title('Tv Shows Monthly Releases count',fontfamily='serif',
                           fontsize=16,fontweight='bold')
plt.xlabel('Month')
plt.ylabel('no.of contents released')

plt.show()
```

👉Insights :

- The best time to launch the Movie is on weekends `FRIDAY` , `SATURDAY` & `SUNDAY` .
- similary , Releasing on the occassional festivities would attract more audiences.
- `December` , `January` haas very high amount of releases because of the festivities and occasions.
- December month weeks like 50,51,52,53,1 are having the number collated. Hence the downfall in the december month weeks.

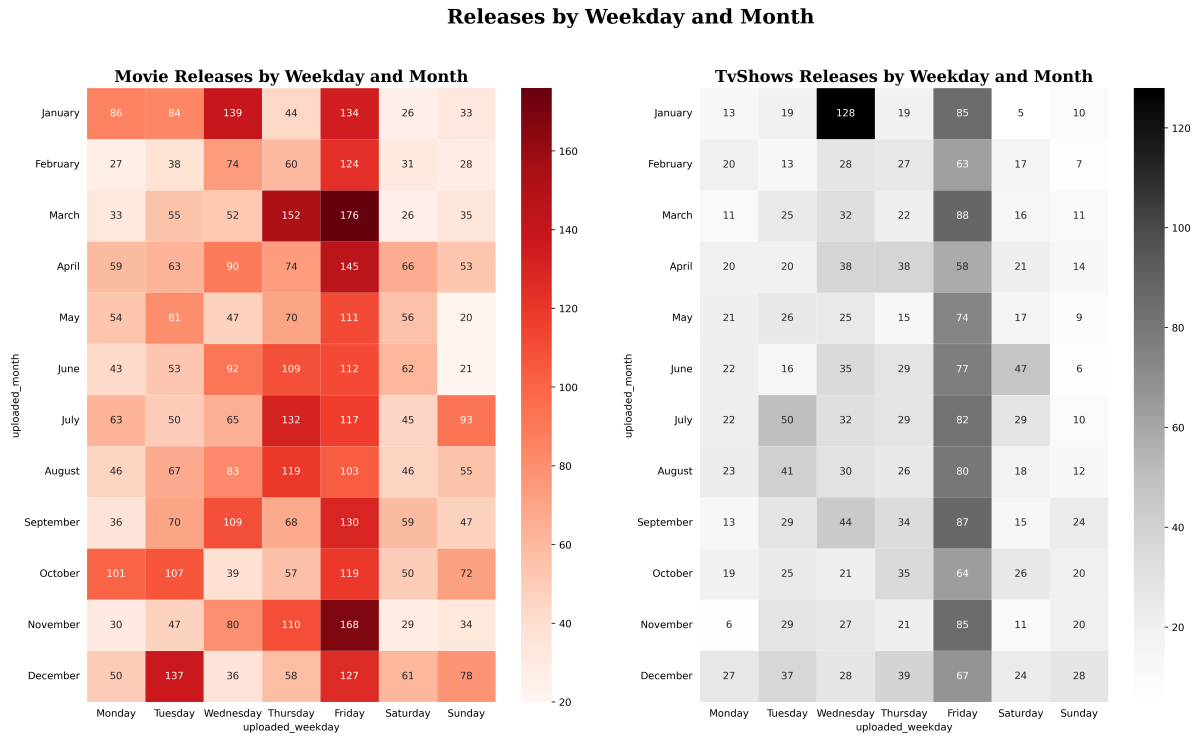
In [165…

```
# df[(df["week_uploaded"]>50) & (df["uploaded_month"]=="January")]['show_id'].nuniq
```

In [383…

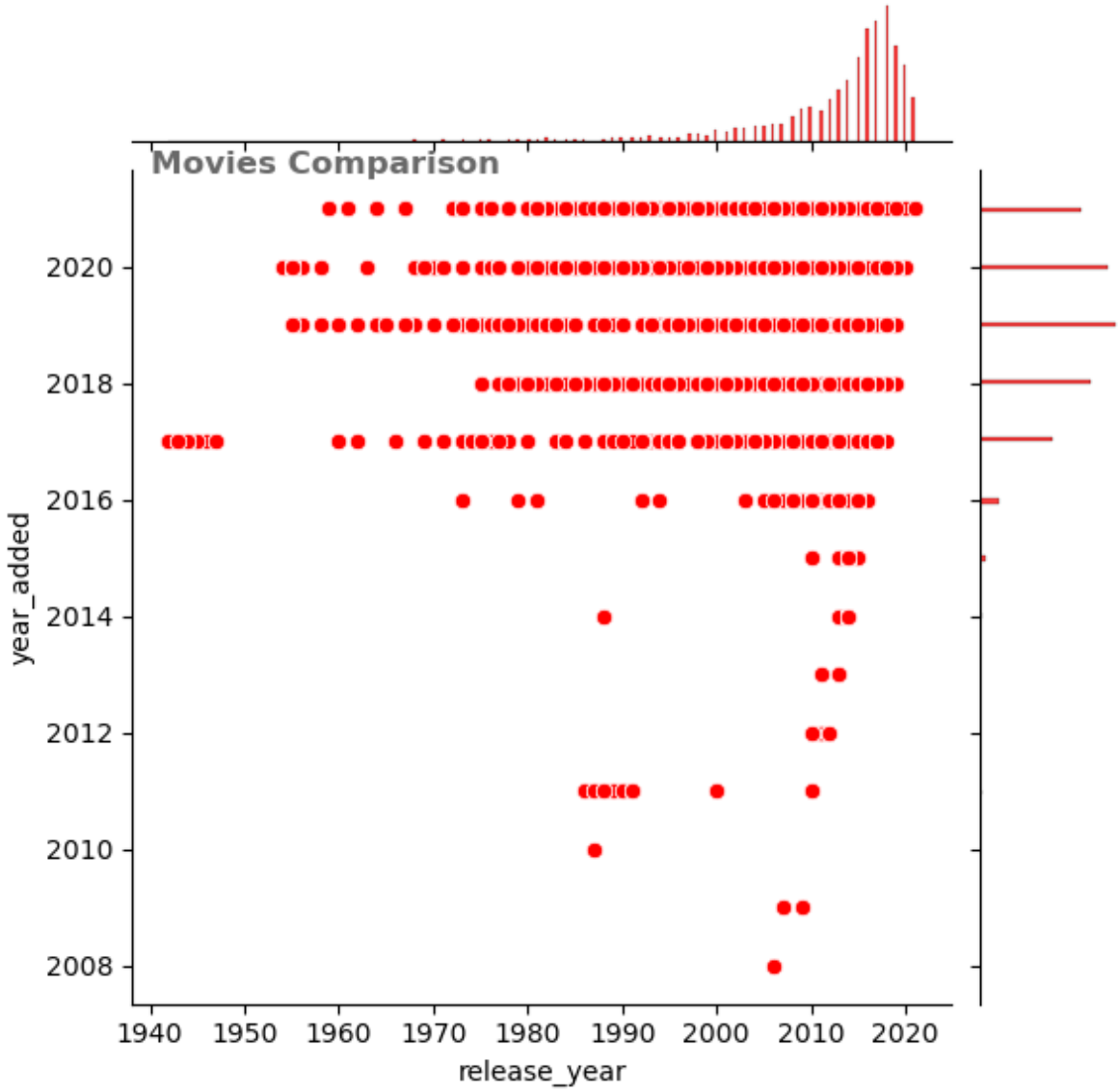
```
plt.figure(figsize=(20,11) , dpi=400)
plt.suptitle('Releases by Weekday and Month',fontfamily='serif',
                      fontsize=20,fontweight='bold')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
sns.heatmap(movies_release_pivot, cmap='Reds', annot=True,
```

```
                    fmt='d' , linewidth=0.1)
plt.title("Movie Releases by Weekday and Month",fontfamily='serif',
                  fontsize=16,fontweight='bold')
plt.yticks(rotation=0)
plt.tick_params(axis='both', which='both', left=False , bottom=False)

plt.subplot(1,2,2)
sns.heatmap(tvs_release_pivot, cmap='Greys', annot=True, fmt='d' ,
                   linewidth=0.1)
plt.title("TvShows Releases by Weekday and Month",fontfamily='serif',
                  fontsize=16,fontweight='bold')
plt.yticks(rotation=0)
plt.tick_params(axis='both', which='both', left=False, bottom=False)

plt.show()
```



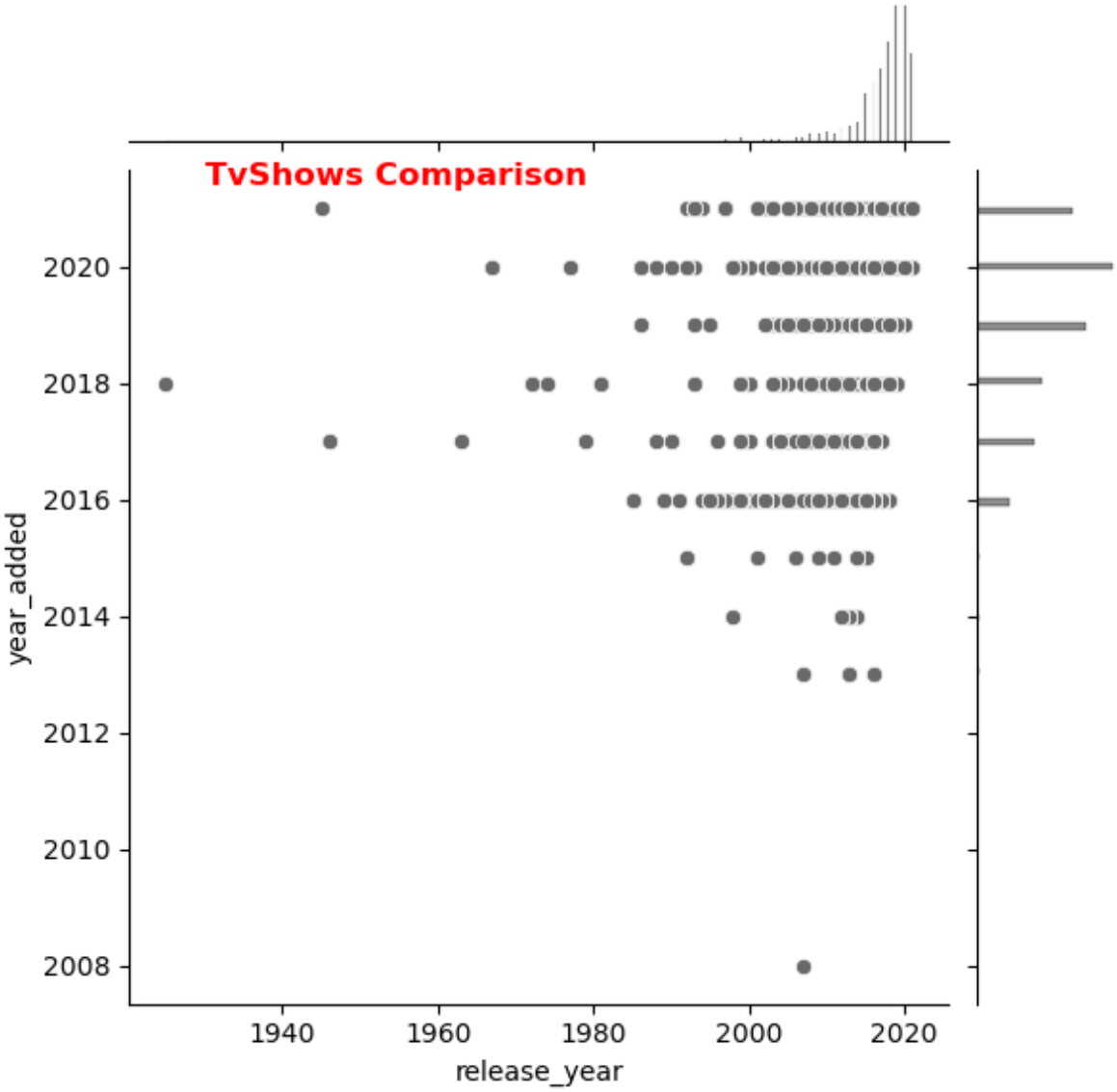
In [402…

```
sns.jointplot(md , x='release_year' , y='year_added' , color='red')
plt.text(1940,2021.6,'Movies Comparison' , color='dimgrey' , fontsize=12 , fontweig
plt.show()
```



In [407…

```
sns.jointplot(tvd , x='release_year' , y='year_added' , color='dimgrey')
plt.text(1930,2021.4,'TvShows Comparison',color='red',fontsize=12,fontweight='bold'
plt.show()
```



👉Insights:-

- `Friday` of every Week,Month throughout the year, has peaked stating that weekends are great time for movie releases.
- `Thrusday` of 3rd Quarter of the year (June , July August) and occasional holidays and festival months have peaked with movie releases.
- `March` , `April` have topped at releasing because of the summer holidays.
- `November` , `December` & `January` also have recorded high no.of releases on the occasion of **DIWALI**,**CHRISTMAS**,**NEW_YEAR**.
- `Wednesday` , `Thrusday` and `Friday` seems to an nominal days releasing a new Tv Shows.
- `June` , `July` , `August` , `September` are also having a higher than average nominal no.of TvShow releases.

📌Q. Find After how many days the contents will be added to Netflix after the release date?

MOVIES:

In [200…
```
filtered_md = md[['show_id','title','release_year','year_added']].drop_duplicates()
```

In [201…
```
filtered_md.shape
```

Out[201]:
```
(6131, 4)
```

In [216…
```
filtered_md.sample()
```

Out[216]:

	show_id	title	release_year	year_added
106514	s6908	Haapus	2010	2018

❓ NOTE:

- I can convert the release year to datetime format but it will give the Jan 1, `release_year` for all the rows which will affect the analysis as all the contents will be released on Jan 1st of every year.
- so we will proceed with the yearwise directly... if the **Time_diff is `zero` it means the content is added to OTT on the same year with some months or days difference**

In [202…
```
filtered_md['time_diff_in_yrs']=filtered_md['year_added']-filtered_md['release_year
```

In [218…
```
filtered_md.head()
```

Out[218]:

	show_id	title	release_year	year_added	time_diff_in_yrs
0	s1	Dick Johnson Is Dead	2020	2021	1
1	s7	My Little Pony: A New Generation	2021	2021	0
21	s8	Sankofa	1993	2021	28
165	s10	The Starling	2021	2021	0
187	s13	Je Suis Karl	2021	2021	0

In [219…
```
filtered_md.time_diff_in_yrs.mode()[0]
```

Out[219]:
```
0
```

👉Insights:

- Time difference is `ZERO` indicating that the contents are added to the netflix library within the same year.
- The contents are added to the Netflix OTT platform within some months or days of release.
- Now a days , as per aggrements the new content will be uploaded in OTT platforms within `50-60 DAYS`.

In [220…
```
filtered_md['time_diff_in_yrs'].value_counts()
```

Out[220]:
```
time_diff_in_yrs
0     1862
1     1178
2      491
3      347
4      273
      ...
60       1
63       1
71       1
70       1
50       1
Name: count, Length: 71, dtype: int64
```

In [221…
```
fmdg=filtered_md.groupby(['time_diff_in_yrs'])[['title']].agg(numbers_released = ('
# .agg(numbers_released = ('title','count'))
```

In [222…
```
rtf = fmdg.sort_values(by='numbers_released',ascending=False)
```

In [223…
```
rtf = rtf.reset_index()
```

In [224…
```
rtf
```

Out[224]:

	time_diff_in_yrs	numbers_released
0	0	1862
1	1	1178
2	2	491
3	3	347
4	4	273
...	...	...
66	63	1
67	60	1
68	70	1
69	71	1
70	50	1

71 rows × 2 columns

In [225…
```
rtf[rtf.time_diff_in_yrs==-1]
```

Out[225]:

	time_diff_in_yrs	numbers_released
60	-1	2

In [226…
```
filtered_md[filtered_md.time_diff_in_yrs==-1]
```

Out[226]:

	show_id	title	release_year	year_added	time_diff_in_yrs
80171	s5395	Hans Teeuwen: Real Rancour	2018	2017	-1
110062	s7064	Incoming	2019	2018	-1

👉Insights:

- Movies like `Hans Teewen` & `Incoming` are uploaded in OTT (1 year) before the Theatrical Release.
- Now a days , some Movies are produced and made specifically for direct **NETFLIX OTT RELEASE**.

TvShows:

In [227…
```
filtered_tvd = tvd[['show_id','title','release_year','year_added']].drop_duplicates
```

In [228…
```
filtered_tvd
```

Out[228]:

	show_id	title	release_year	year_added
0	s2	Blood & Water	2021	2021
57	s3	Ganglands	2021	2021
84	s4	Jailbirds New Orleans	2021	2021
86	s5	Kota Factory	2021	2021
110	s6	Midnight Mass	2021	2021
...	...	...	...	...
56034	s8796	Yu-Gi-Oh! Arc-V	2015	2018
56054	s8797	Yunus Emre	2016	2017
56090	s8798	Zak Storm	2016	2018
56118	s8801	Zindagi Gulzar Hai	2012	2016
56145	s8804	Zombie Dumb	2018	2019

2676 rows × 4 columns

In [229…
```
filtered_tvd.shape
```

Out[229]:
```
(2676, 4)
```

In [230…
```
filtered_tvd['time_diff_in_yrs']=filtered_tvd['year_added']-filtered_tvd['release_y
```

In [231…
```
filtered_tvd.tail()
```

Out[231]:

	show_id	title	release_year	year_added	time_diff_in_yrs
56034	s8796	Yu-Gi-Oh! Arc-V	2015	2018	3
56054	s8797	Yunus Emre	2016	2017	1
56090	s8798	Zak Storm	2016	2018	2
56118	s8801	Zindagi Gulzar Hai	2012	2016	4
56145	s8804	Zombie Dumb	2018	2019	1

In [232…
```
filtered_tvd['time_diff_in_yrs'].mode()
```

Out[232]:
```
0    0
Name: time_diff_in_yrs, dtype: int64
```

In [233…
```
filtered_tvd['time_diff_in_yrs'].mode()[0]
```

Out[233]:
```
0
```

👉Insights:

- Time difference is `ZERO` indicating that the contents are added to the netflix library within the same year.
- The contents are added to the Netflix OTT platform within some months or days of release.
- Now a days , as per aggrements the new content will be uploaded in OTT platforms within `24 hours` after aired on Television.

In [235…
```
ftv = filtered_tvd.groupby(['time_diff_in_yrs'])[['title']].agg(numbers_released =
# .agg(numbers_released = ('title','count'))
```

In [236…
```
rtv = ftv.sort_values(by='numbers_released',ascending=False)
```

In [237…
```
rtv
```

Out[237]:

time_diff_in_yrs	numbers_released
0	1360
1	381
2	223
3	153
4	109
5	80
6	75
7	50
8	44
9	33
10	21
11	17
15	13
14	13
13	12
-1	10
12	10
17	8
16	8
18	7
27	5
19	5
21	5
20	4
22	3
24	2
25	2
28	2
29	2
76	1
71	1
54	1
37	1
53	1
46	1

time_diff_in_yrs	numbers_released
44	1
43	1
38	1
-3	1
34	1
33	1
32	1
31	1
30	1
26	1
23	1
-2	1
93	1

In [238…
```
rtv = rtv.reset_index()
```

In [239…
```
rtv[(rtv.time_diff_in_yrs==-1)|(rtv.time_diff_in_yrs==-2)|(rtv.time_diff_in_yrs==-3
```

Out[239]:

	time_diff_in_yrs	numbers_released
15	-1	10
38	-3	1
46	-2	1

In [240…
```
filtered_tvd[filtered_tvd.time_diff_in_yrs<0]
```

Out[240]:

	show_id	title	release_year	year_added	time_diff_in_yrs
11555	s1552	Hilda	2021	2020	-1
12696	s1697	Polly Pocket	2021	2020	-1
22307	s2921	Love Is Blind	2021	2020	-1
24727	s3169	Fuller House	2020	2019	-1
26073	s3288	Maradona in Mexico	2020	2019	-1
26953	s3370	BoJack Horseman	2020	2019	-1
27389	s3434	The Hook Up Plan	2020	2019	-1
39091	s4845	Unbreakable Kimmy Schmidt	2019	2018	-1
39095	s4846	Arrested Development	2019	2018	-1
44911	s5659	Sense8	2018	2016	-2
45201	s5678	Tokyo Trial	2017	2016	-1
50538	s7113	Jack Taylor	2016	2013	-3

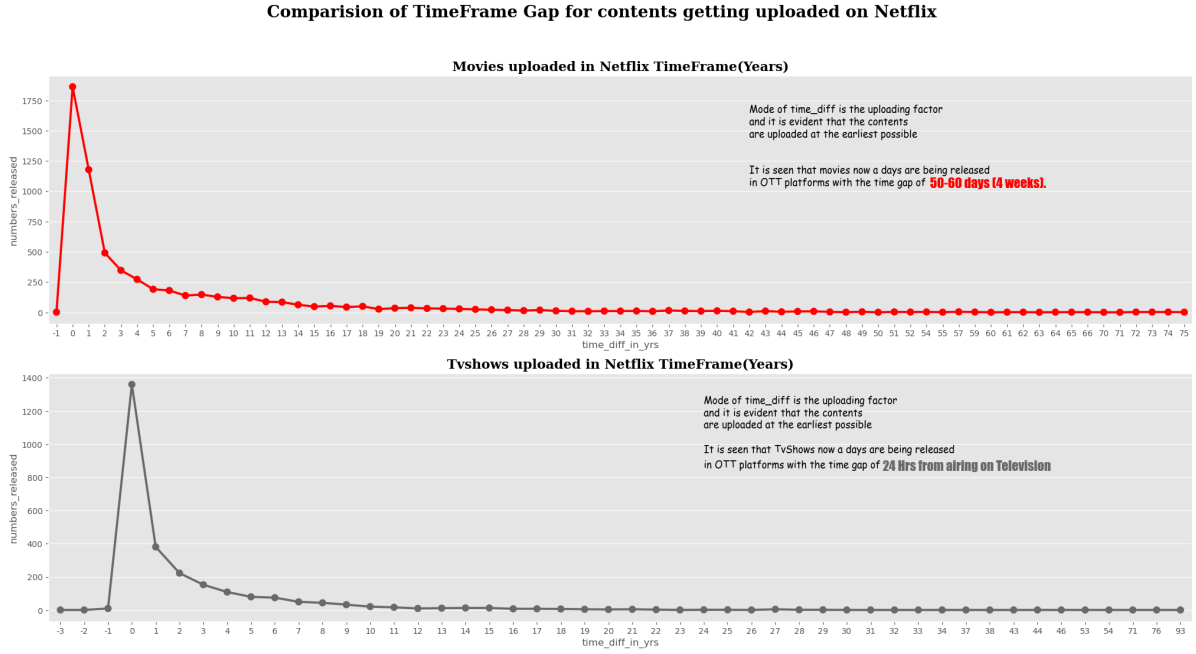
👉Insights:

- Contents like TV show episodes and web series Seasons are mostly released directly in OTT platform.
- From the day and time of Airing any contents, those will be uploaded on Netflix library within `24 Hours`.

In [246…

```
plt.figure(figsize=(25,12))
plt.suptitle('Comparision of TimeFrame Gap for contents getting uploaded on Netflix
             fontfamily='serif',fontweight='bold',fontsize=20)
plt.style.use('default')
plt.style.use('ggplot')

plt.subplot(2,1,1)
# plt.figure(figsize=(20,6))
sns.pointplot(rtf , x='time_diff_in_yrs' , y='numbers_released' , color='r')
plt.title('Movies uploaded in Netflix TimeFrame(Years)',
                  fontfamily='serif',fontweight='bold',fontsize=16)
plt.text(43,1450,'Mode of time_diff is the uploading factor \n
                and it is evident that the contents \n',
                  fontsize=12,fontfamily='cursive')
plt.text(43,1450,'are uploaded at the earliest possible',
                  fontsize=12,fontfamily='cursive')
plt.text(43,1150,'It is seen that movies now a days are being released',
                  fontsize=12,fontfamily='cursive')
plt.text(43,1050,'in OTT platforms with the time gap of',
                  fontsize=12,fontfamily='cursive')
plt.text(54.2,1030,'50-60 days (4 weeks).',
                  fontsize=16,fontfamily='fantasy',fontweight='bold',color='red')

plt.subplot(2,1,2)
# plt.figure(figsize=(20,6))
sns.pointplot(rtv , x='time_diff_in_yrs' , y='numbers_released' , color='dimgrey')
plt.title('Tvshows uploaded in Netflix TimeFrame(Years)',
                    fontfamily='serif',fontweight='bold',fontsize=16)
plt.text(27,1100,'Mode of time_diff is the uploading factor \n
                    and it is evident that the contents \n',
                  fontsize=12,fontfamily='cursive')
```

```
plt.text(27,1100,'are uploaded at the earliest possible',fontsize=12,fontfamily='cu
plt.text(27,950,'It is seen that TvShows now a days are being released',
                fontsize=12,fontfamily='cursive')
plt.text(27,860,'in OTT platforms with the time gap of',
                fontsize=12,fontfamily='cursive')
plt.text(34.5,840,'24 Hrs from airing on Television',
                fontsize=16,fontfamily='fantasy',fontweight='bold',color='dimgrey'
plt.show()
```



👉Insights:-

MOVIES

- Upon considering the recent releases in **Netflix**, it is seen that the contracts says, the Movie will be released in the window of the `50-60 days` (4 Weeks) after the theatrical releases.
- The *Mode* of the release time difference between *OTT vs Theatrical* is found to be `Zero` years so we consider it is released with some months or weeks gap.
- We can also see that movies like `Incoming` & `Hans Teeuwen: Real Rancour` is released in Netflix Platform before theatrical release, thus indicating the release time difference is negative.

TvShows

- As of Televisions shows, the contents(episodes) are added to Netflix library within `24 Hours` of airing on the Television.
- We can that the *Mode* of release time difference is `Zero` year as well indication that window is too short for OTT updation of episodes and contents.
- Shows like `Sense8` , `Jack Taylor` and much more are specifically tailor made for Netflix OTT releases.

🤛 Recently Many content creators prefer direct `Netflix OTT` releases for a massive gathering of Audience providing freedom for the users to access unlimited contents of different languages and Showcasing that **Entertainment is now in Your Home**...

💡 Days to add contents Netflix Library after acquiring rights: (Recent past data)

In [215…
```
nx.release_year.dtypes
```

Out[215]:
```
dtype('int64')
```

In [218…
```
cu = nx.copy
cu = nx.drop_duplicates(subset='show_id')
cu['date_added'] = pd.to_datetime(cu['date_added'])
cu['release_date'] = pd.to_datetime(cu['release_year'].astype(str))
cu
```

Out[218]:

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown actors	United States	2021-09-25	2020	PG
1	s2	TV Show	Blood & Water	Unknown director	Ama Qamata	South Africa	2021-09-24	2021	
58	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila	Unknown	2021-09-24	2021	
85	s4	TV Show	Jailbirds New Orleans	Unknown director	Unknown actors	Unknown	2021-09-24	2021	
87	s5	TV Show	Kota Factory	Unknown director	Mayur More	India	2021-09-24	2021	
…	…	…	…	…	…	…	…	…	
201902	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo	United States	2019-11-20	2007	
201932	s8804	TV Show	Zombie Dumb	Unknown director	Unknown actors	Unknown	2019-07-01	2018	TV
201935	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg	United States	2019-11-01	2009	
201949	s8806	Movie	Zoom	Peter Hewitt	Tim Allen	United States	2020-01-11	2006	
201967	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal	India	2019-03-02	2015	TV

8807 rows × 13 columns

In [219…
```
cu['days_to_add'] = (cu['date_added'] - cu['release_date']).dt.days
cu
```

Out[219]:

	show_id	type	title	director	cast	country	date_added	release_year	rat
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown actors	United States	2021-09-25	2020	PG
1	s2	TV Show	Blood & Water	Unknown director	Ama Qamata	South Africa	2021-09-24	2021	
58	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila	Unknown	2021-09-24	2021	
85	s4	TV Show	Jailbirds New Orleans	Unknown director	Unknown actors	Unknown	2021-09-24	2021	
87	s5	TV Show	Kota Factory	Unknown director	Mayur More	India	2021-09-24	2021	
...	...	...	...	...	...	...	...	...	
201902	s8803	Movie	Zodiac	David Fincher	Mark Ruffalo	United States	2019-11-20	2007	
201932	s8804	TV Show	Zombie Dumb	Unknown director	Unknown actors	Unknown	2019-07-01	2018	TV
201935	s8805	Movie	Zombieland	Ruben Fleischer	Jesse Eisenberg	United States	2019-11-01	2009	
201949	s8806	Movie	Zoom	Peter Hewitt	Tim Allen	United States	2020-01-11	2006	
201967	s8807	Movie	Zubaan	Mozez Singh	Vicky Kaushal	India	2019-03-02	2015	TV

8807 rows × 14 columns

In [220…

```
# considering entire data (both tvshows and movies)
day_mode = cu['days_to_add'].mode()[0]
day_mode
```

Out[220]:

```
334
```

In [221…

```
cu.dtypes
```

Out[221]:

```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added      datetime64[ns]
release_year             int64
rating                  object
duration                object
listed_in               object
year_added               int64
release_date    datetime64[ns]
days_to_add              int64
dtype: object
```

In [223…

```
# filtering the recent past data (after 2018) for movies
fcum = cu[(cu.release_year>2018) & (cu.type=='Movie')]
```

```
fcum
```

Out[223]:

	show_id	type	title	director	cast	country	date_added	release_year	r
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	Unknown actors	United States	2021-09-25	2020	I
159	s7	Movie	My Little Pony: A New Generation	Robert Cullen	Vanessa Hudgens	Unknown	2021-09-24	2021	
331	s10	Movie	The Starling	Theodore Melfi	Melissa McCarthy	United States	2021-09-24	2021	I
431	s13	Movie	Je Suis Karl	Christian Schwochow	Luna Wedler	Germany	2021-09-23	2021	
475	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Klara Castanho	Unknown	2021-09-22	2021	T
...	...	...	...	...	...	...	...	...	
195483	s8517	Movie	The Spy Who Fell to Earth	Thomas Meadmore	Unknown actors	United Kingdom	2019-04-05	2019	
197939	s8632	Movie	Trixie Mattel: Moving Parts	Nicholas Zeig-Owens	Brian Firkus	United States	2020-03-27	2019	
199919	s8719	Movie	Westside vs. the World	Michael Fahey	Ron Perlman	Unknown	2019-08-09	2019	
200631	s8753	Movie	Wish Man	Theo Davies	Andrew Steel	United States	2019-12-03	2019	T
200704	s8757	Movie	Woodstock	Barak Goodman	Unknown actors	United States	2019-08-13	2019	

1427 rows × 14 columns

In [224…

```
upload_date_interval_movie = fcum['days_to_add'].mode()[0]
upload_date_interval_movie
```

Out[224]:

```
252
```

In [225…

```
# filtering the recent past data (after 2018) for tvshows
fcutv = cu[(cu.release_year>2018) & (cu.type=='TV Show')]
fcutv
```

Out[225]:

	show_id	type	title	director	cast	country	date_added	release_year	rati
1	s2	TV Show	Blood & Water	Unknown director	Ama Qamata	South Africa	2021-09-24	2021	T M
58	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila	Unknown	2021-09-24	2021	T M
85	s4	TV Show	Jailbirds New Orleans	Unknown director	Unknown actors	Unknown	2021-09-24	2021	T M
87	s5	TV Show	Kota Factory	Unknown director	Mayur More	India	2021-09-24	2021	T M
111	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel	Unknown	2021-09-24	2021	T M
...	...	...	...	...	...	...	...	...	
185478	s8126	TV Show	Super Wings	Unknown director	Luca Padovan	United States	2020-12-01	2020	TV
185601	s8133	TV Show	Surviving R. Kelly Part II: The Reckoning	Unknown director	R. Kelly	United States	2020-04-13	2020	T M
186754	s8174	TV Show	Thackeray	Unknown director	Unknown actors	India	2019-05-25	2019	T M
191986	s8377	TV Show	The Kindness Diaries	Unknown director	Leon Logothetis	Unknown	2019-05-01	2019	TV-I
193549	s8438	TV Show	The Netflix Afterparty	Unknown director	David Spade	United States	2021-01-02	2021	T M

1148 rows × 14 columns

In [257…
```
fcutv['days_to_add'].value_counts()
```

Out[257]:
```
days_to_add
 0      10
 164    10
 79     10
 219     9
 186     9
        ..
 521     1
 104     1
 518     1
 149     1
-217     1
Name: count, Length: 438, dtype: int64
```

In [227…
```
upload_date_interval_tvs = fcutv['days_to_add'].mode()
upload_date_interval_tvs
```

Out[227]:
```
0      0
1     79
2    164
Name: days_to_add, dtype: int64
```

In [255…
```
upload_date_interval_tvs.mean()
```

Out[255]:
```
81.0
```

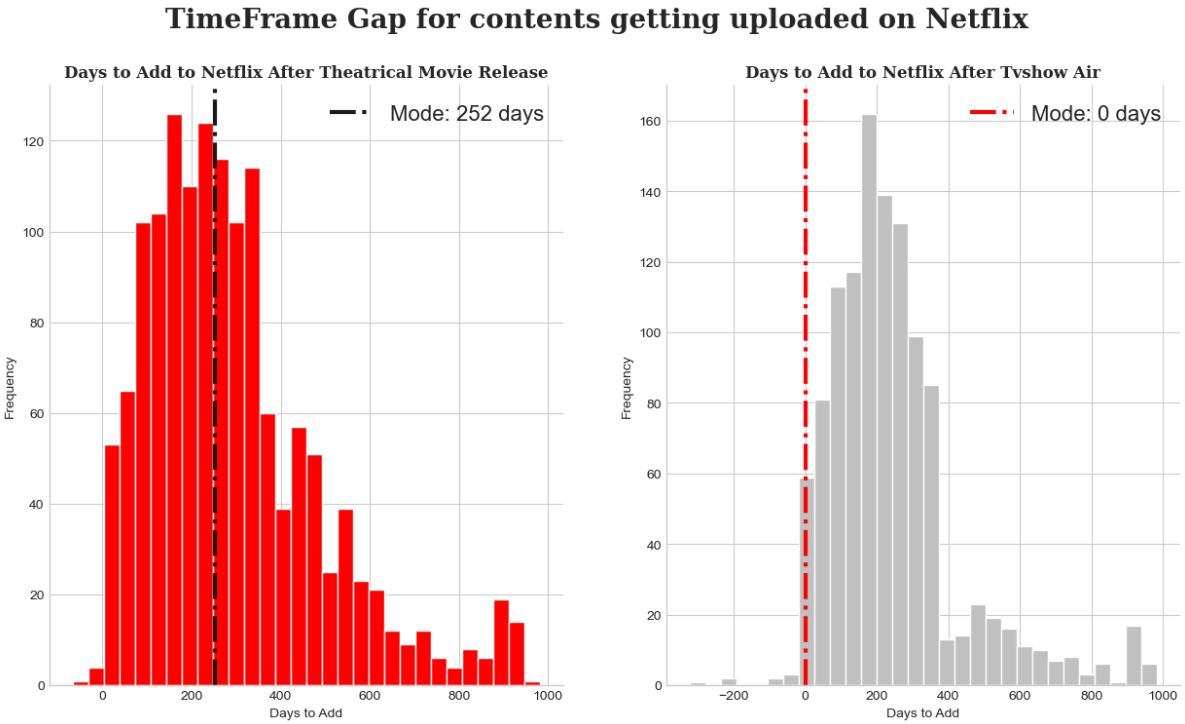
In [256…
```
upload_date_interval_tvs.median()
```

Out[256]:
```
79.0
```

In [248…
```
upload_date_interval_tvs[0]
```

Out[248]:
```
0
```

In [276…
```
plt.figure(figsize=(15,8))
plt.suptitle('TimeFrame Gap for contents getting uploaded on Netflix',
             fontfamily='serif',fontweight='bold',fontsize=20)
plt.style.use('default')
plt.style.use('seaborn-v0_8-whitegrid')

plt.subplot(1,2,1)
plt.hist(fcum['days_to_add'], bins=30, color='red', edgecolor='white')
plt.title('Days to Add to Netflix After Theatrical Movie Release',
         fontfamily='serif',fontweight='bold',fontsize=12)
plt.xlabel('Days to Add')
plt.ylabel('Frequency')
plt.axvline(upload_date_interval_movie, color='k', linestyle='-.',
            linewidth=3, label=f'Mode: {upload_date_interval_movie} days')
plt.legend(fontsize=16)

plt.subplot(1,2,2)
plt.hist(fcutv['days_to_add'], bins=30, color='silver', edgecolor='white')
plt.title('Days to Add to Netflix After Tvshow Air',
         fontfamily='serif',fontweight='bold',fontsize=12)
plt.xlabel('Days to Add')
plt.ylabel('Frequency')
plt.axvline(upload_date_interval_tvs[0], color='red', linestyle='-.',
            linewidth=3, label=f'Mode: {upload_date_interval_tvs[0]} days')
plt.legend(fontsize=16)

sns.despine()
plt.show()
```



👉Insights:

- Time difference is `ZERO` and even found the mode days to be approx `252 days` indicating that the contents are added to the netflix library within the same year i.e 365 days.
- The contents are added to the Netflix OTT platform within some months or days of release as per todays agreements it is found that netflix acquires telecasting rights after `4 weeks` of Release of the `MOVIEs` .
- Now a days , as per aggrements the new `TvShow` content will be uploaded in OTT platforms within `24 hours` after aired on Television.

📌Q. Whats the shortest and longest duration of contents ?

MOVIES

In [249…
```
md.columns
```

Out[249]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'listed_in', 'year_added', 'runtime_in_mins'],
      dtype='object')
```

In [258…
```
#Shortest Movie
shortest_movie = md.loc[(md['runtime_in_mins']==np.min(md.runtime_in_mins))] [['tit
shortest_movie
```

Out[258]:

	title	runtime_in_mins
58371	Silent	3

In [259…
```
#Longest Movie
longest_movie = md.loc[(md['runtime_in_mins']==np.max(md.runtime_in_mins))] [['titl
longest_movie
```

Out[259]:

	title	runtime_in_mins
64115	Black Mirror: Bandersnatch	312

📌 Q. Find how are the contents added to Netflix library (uploading rate)?

In [136…
```
md.sample()
```

Out[136]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	listed_ir
99310	s6519	Movie	Come and Find Me	Zack Whedon	Terry Chen	United States	2019-09-18	2016	R	Dramas

In [137…
```
df = md.drop_duplicates(subset='title')
```

In [168…
```
df.shape
```

Out[168]:
```
(6131, 12)
```

In [138…
```
df = df[['show_id','title','date_added']]
```

In [170…
```
df.shape
```

Out[170]:
```
(6131, 3)
```

In [274…
```
df.sample()
```

Out[274]:

	show_id	title	date_added
57454	s3704	El testigo	2019-06-28

In [276…
```
df.dtypes
```

Out[276]:
```
show_id       object
title         object
date_added    object
dtype: object
```

In [139…
```
df['date_added'] = pd.to_datetime(df['date_added'])
```

In [278…
```
df.dtypes
```

Out[278]:
```
show_id               object
title                 object
date_added    datetime64[ns]
dtype: object
```

In [140…
```
df['year_added'] = df['date_added'].dt.year
```

In [141…
```
df['month_added'] = df['date_added'].dt.month_name()
```

In [320…
```
df.sample()
```

Out[320]:

	show_id	title	date_added	year_added	month_added
47409	s2934	Polaroid	2020-02-09	2020	February

In [142…
```
upload_rate = df.groupby('year_added')['month_added'].value_counts()
```

In [337…
```
upload_rate
```

Out[337]:
```
year_added  month_added
2008        January          1
2009        November         1
            May              1
2010        November         1
2011        October         11
                           ...
2021        August         117
            January         96
            May             94
            March           75
            February        65
Name: count, Length: 105, dtype: int64
```

In [143…
```
month_order = ['January', 'February', 'March', 'April', 'May',
               'June', 'July','August', 'September',
               'October', 'November', 'December']
upload_rate = upload_rate.unstack()[month_order]
upload_rate
```

Out[143]:

month_added	January	February	March	April	May	June	July	August	September	October
year_added										
2008	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2009	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN
2010	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	1.0	11.0
2012	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0
2014	2.0	1.0	NaN	1.0	NaN	1.0	1.0	1.0	1.0	4.0
2015	1.0	3.0	3.0	1.0	5.0	4.0	5.0	2.0	6.0	10.0
2016	15.0	9.0	14.0	14.0	9.0	11.0	19.0	23.0	29.0	32.0
2017	58.0	65.0	87.0	66.0	63.0	65.0	45.0	77.0	81.0	97.0
2018	105.0	63.0	138.0	87.0	70.0	50.0	125.0	130.0	81.0	146.0
2019	116.0	103.0	119.0	119.0	91.0	122.0	98.0	87.0	86.0	128.0
2020	152.0	72.0	93.0	127.0	105.0	115.0	103.0	82.0	115.0	116.0
2021	96.0	65.0	75.0	135.0	94.0	124.0	169.0	117.0	118.0	NaN

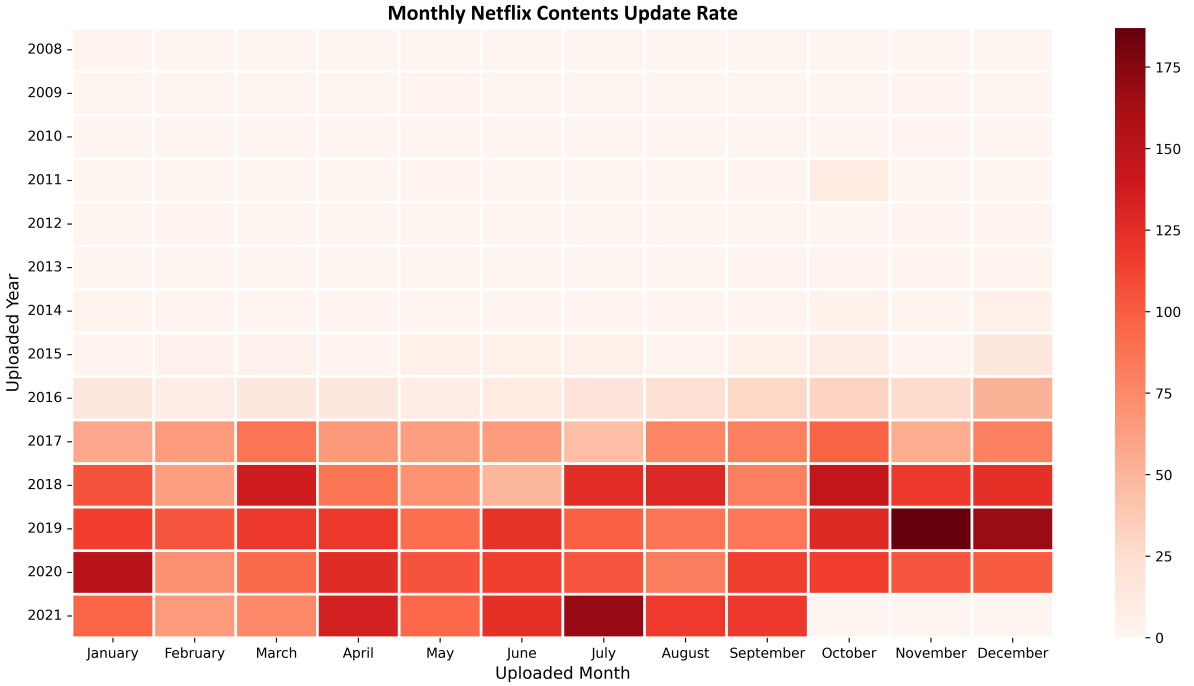
In [144…
```
upload_rate = upload_rate.fillna(0)
upload_rate
```

Out[144]:

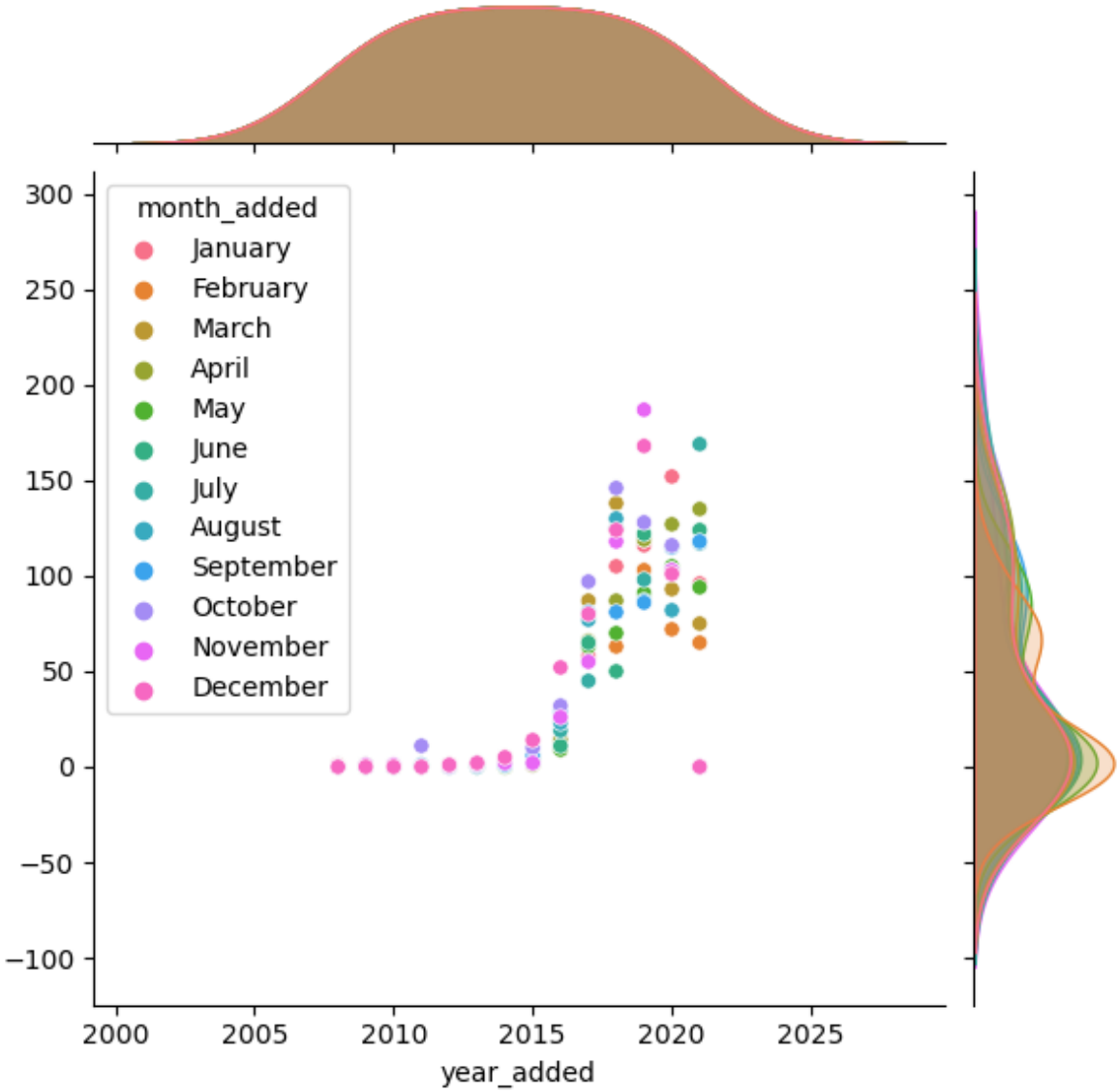
month_added year_added	January	February	March	April	May	June	July	August	September	October
2008	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2010	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2011	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	11.0
2012	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2013	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
2014	2.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	4.0
2015	1.0	3.0	3.0	1.0	5.0	4.0	5.0	2.0	6.0	10.0
2016	15.0	9.0	14.0	14.0	9.0	11.0	19.0	23.0	29.0	32.0
2017	58.0	65.0	87.0	66.0	63.0	65.0	45.0	77.0	81.0	97.0
2018	105.0	63.0	138.0	87.0	70.0	50.0	125.0	130.0	81.0	146.0
2019	116.0	103.0	119.0	119.0	91.0	122.0	98.0	87.0	86.0	128.0
2020	152.0	72.0	93.0	127.0	105.0	115.0	103.0	82.0	115.0	116.0
2021	96.0	65.0	75.0	135.0	94.0	124.0	169.0	117.0	118.0	0.0

In [326…

```
plt.figure(figsize=(16,8) , dpi=500)
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
sns.heatmap(upload_rate, cmap='Reds', edgecolors='beige', linewidths=2)
plt.title('Monthly Netflix Contents Update Rate',
          fontsize=16, fontfamily='calibri', fontweight='bold')
plt.yticks(rotation=0)
plt.xticks(rotation=0)
plt.xlabel('Uploaded Month',fontsize=12)
plt.ylabel('Uploaded Year',fontsize=12)
plt.show()
```



In [384…
```
sns.jointplot(upload_rate)
plt.show()
```



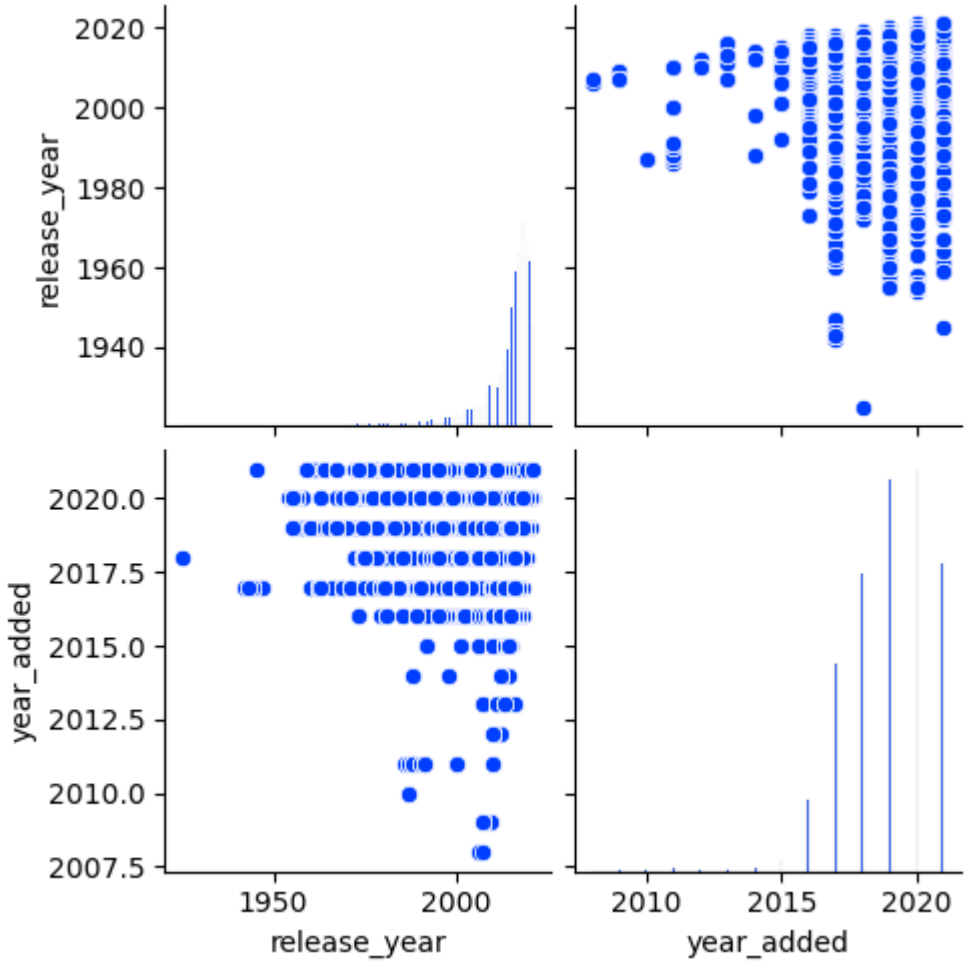
👉Insights :

- The contents are uploaded to the Netflix library as soon the contents rights are acquired in case of movies and for telecasted tvShows or episodes are uploaded within 24 hrs after airing on Television.
- It is seen the as the year are progressing the contents varities and uploading rates are too high.

🏷️ PAIRPLOT OF cleaned Raw Data

In [166…
```
sns.pairplot(nx)
```

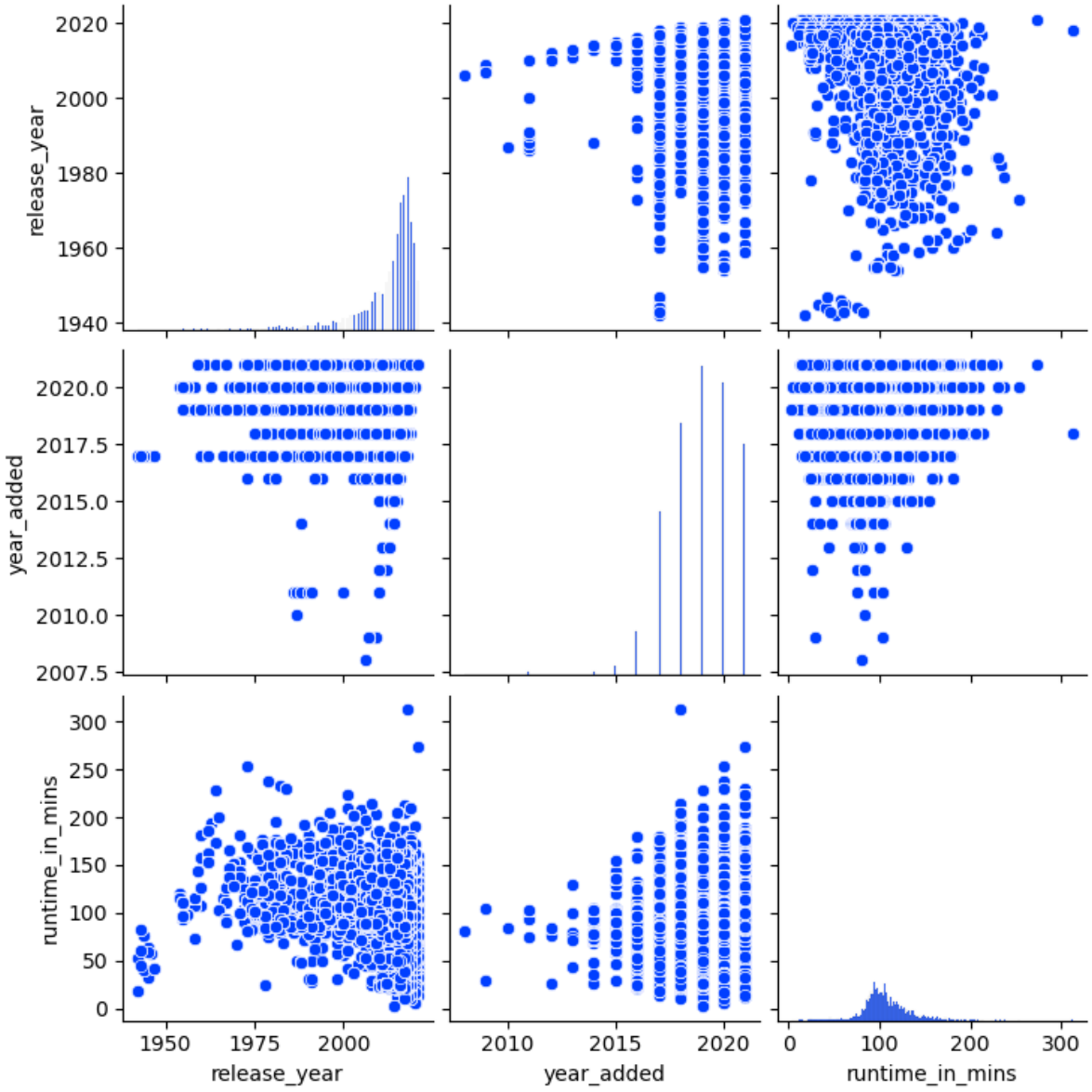
Out[166]:
```
<seaborn.axisgrid.PairGrid at 0x212b411b390>
```



🏷️ PAIRPLOT OF MOVIES Data

In [164…
```
sns.pairplot(md)
```

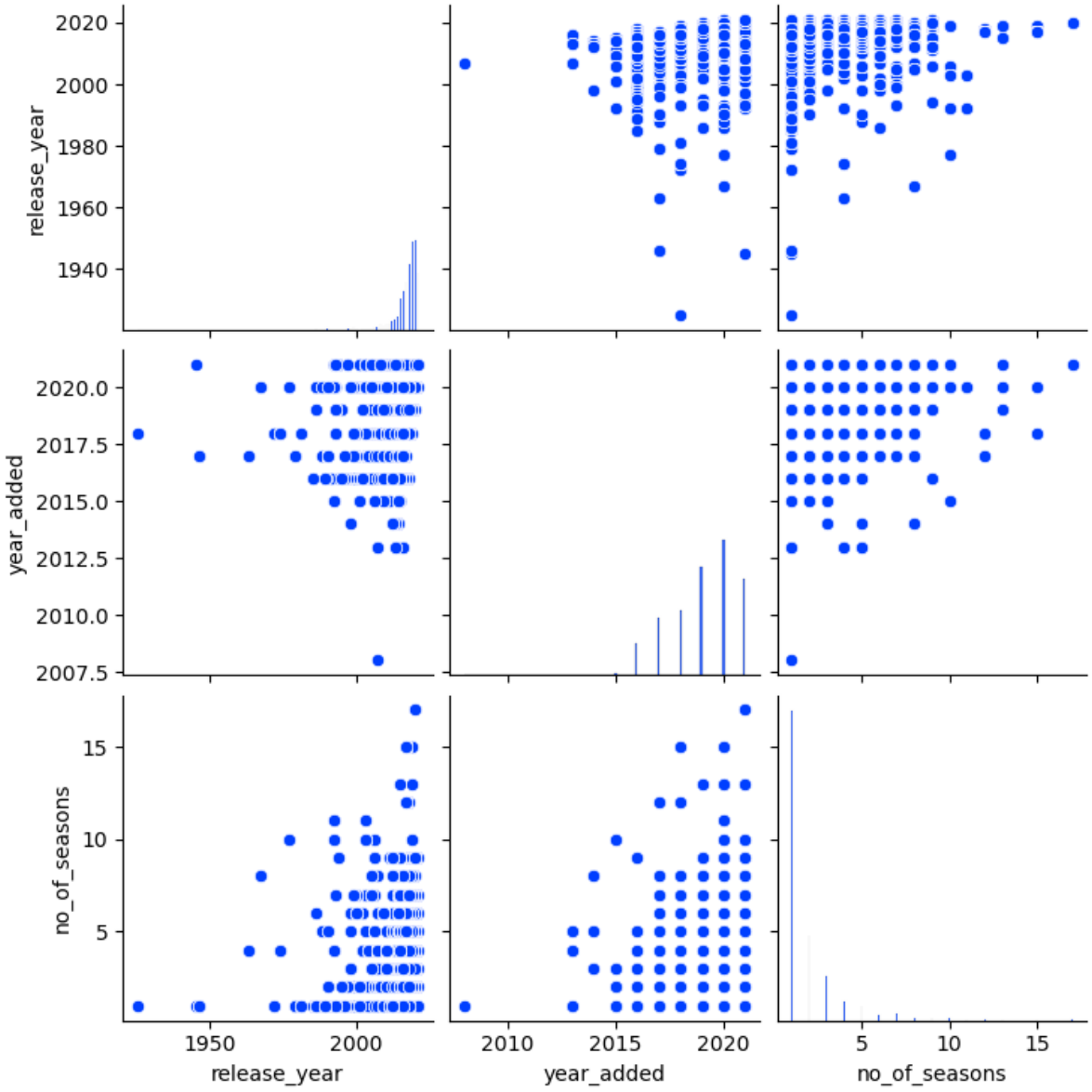
Out[164]:
```
<seaborn.axisgrid.PairGrid at 0x212b8678690>
```



🏷️ PAIRPLOT OF TvSHOWS Data

In [165…
```
sns.pairplot(tvd)
```

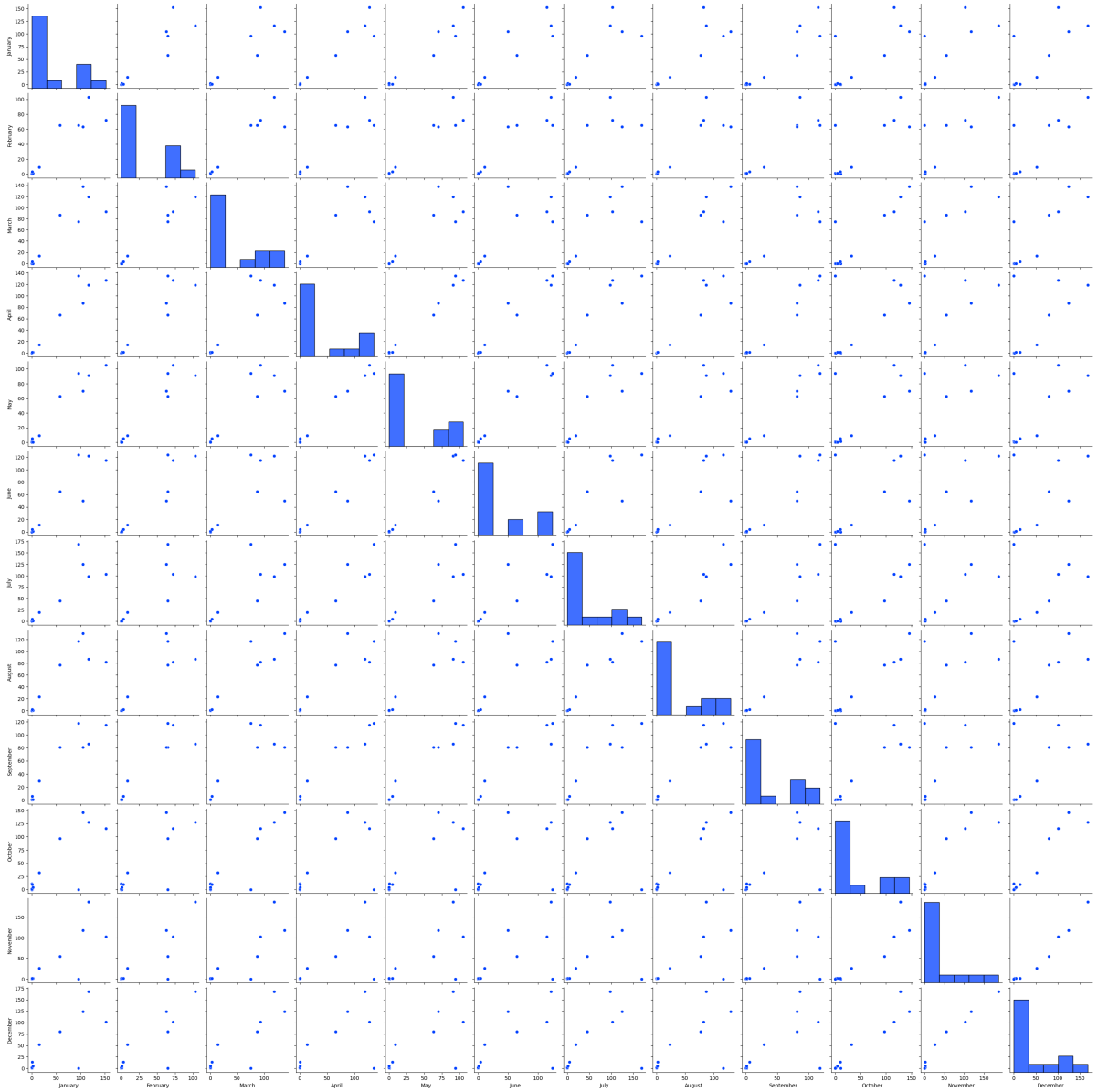
Out[165]:
```
<seaborn.axisgrid.PairGrid at 0x212b3c62dd0>
```



🏷️ Pair plotting of Upload Rate

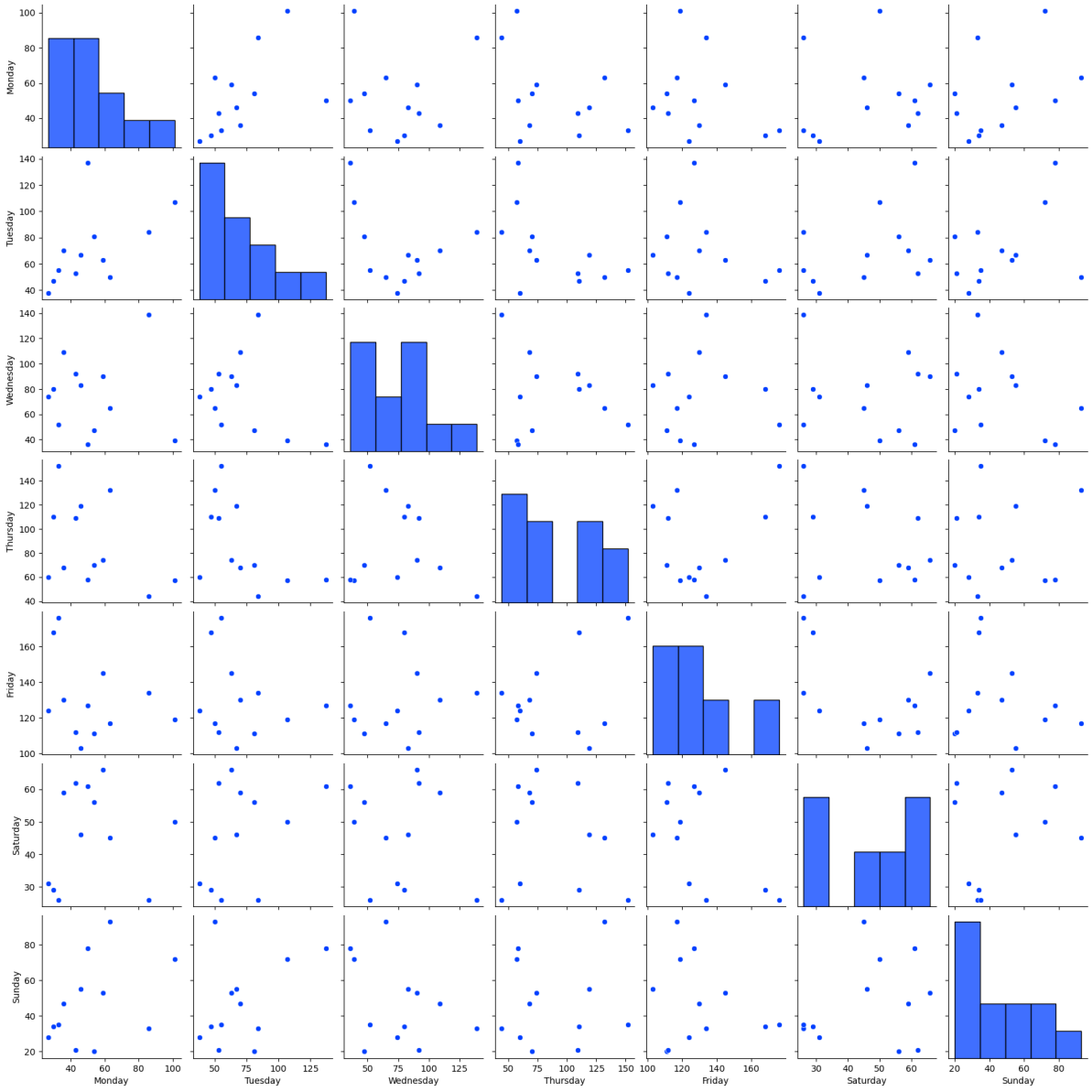
In [156…

```
sns.pairplot(upload_rate, kind='scatter')
plt.show()
```



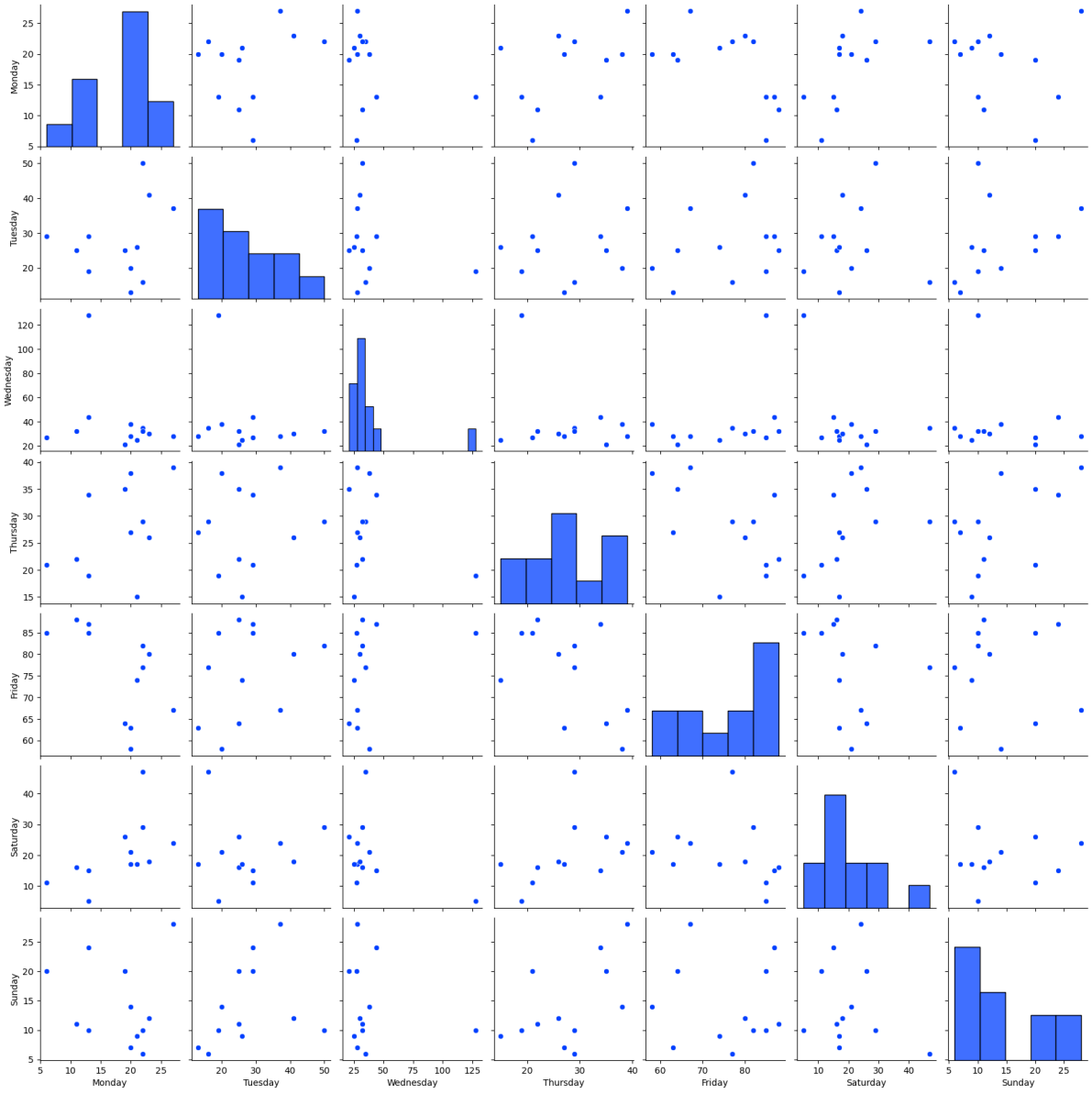
In [475…

```
sns.pairplot(movies_release_pivot)
plt.show()
```



In [476…

```
sns.pairplot(tvs_release_pivot)
plt.show()
```



In [477…
```
movies_release_pivot.corr()
```

Out[477]:

uploaded_weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
uploaded_weekday							
Monday	1.000000	0.526678	0.004534	-0.422987	-0.307633	0.110362	0.404492
Tuesday	0.526678	1.000000	-0.354611	-0.534402	-0.230047	0.378600	0.419413
Wednesday	0.004534	-0.354611	1.000000	-0.179123	0.035969	-0.175518	-0.353864
Thursday	-0.422987	-0.534402	-0.179123	1.000000	0.295514	-0.234899	0.057401
Friday	-0.307633	-0.230047	0.035969	0.295514	1.000000	-0.488785	-0.189752
Saturday	0.110362	0.378600	-0.175518	-0.234899	-0.488785	1.000000	0.251742
Sunday	0.404492	0.419413	-0.353864	0.057401	-0.189752	0.251742	1.000000

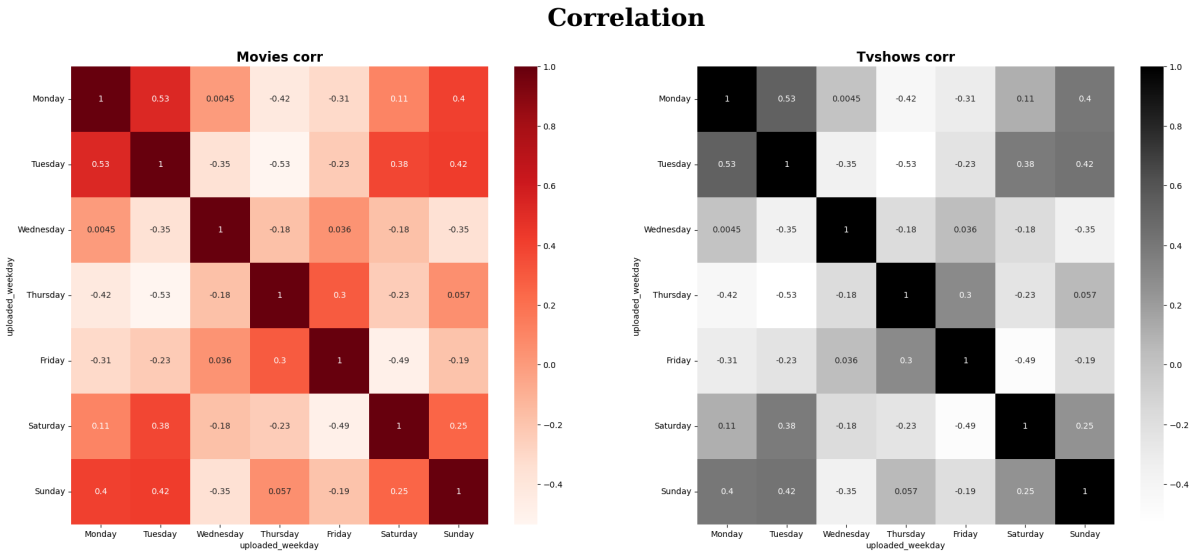
In [484…
```
tvs_release_pivot.corr()
```

Out[484]:

uploaded_weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
uploaded_weekday							
Monday	1.000000	0.253691	-0.288971	0.441016	-0.584714	0.569130	-0.077373
Tuesday	0.253691	1.000000	-0.252029	0.132365	0.301245	0.036033	0.320120
Wednesday	-0.288971	-0.252029	1.000000	-0.289991	0.329353	-0.436326	-0.176579
Thursday	0.441016	0.132365	-0.289991	1.000000	-0.538804	0.415836	0.556975
Friday	-0.584714	0.301245	0.329353	-0.538804	1.000000	-0.268256	-0.054725
Saturday	0.569130	0.036033	-0.436326	0.415836	-0.268256	1.000000	-0.186660
Sunday	-0.077373	0.320120	-0.176579	0.556975	-0.054725	-0.186660	1.000000

In [512…

```
plt.figure(figsize=(25,10))
plt.suptitle('Correlation',fontsize=30,fontfamily='serif',fontweight='bold')

plt.subplot(1,2,1)
sns.heatmap(movies_release_pivot.corr() ,cmap='Reds',annot=True)
plt.title('Movies corr',fontsize=16,fontweight='bold')
plt.xticks(rotation=0)
plt.yticks(rotation=0)

plt.subplot(1,2,2)
sns.heatmap(movies_release_pivot.corr() ,cmap='Greys',annot=True)
plt.title('Tvshows corr',fontsize=16,fontweight='bold')
plt.xticks(rotation=0)
plt.yticks(rotation=0)

plt.show()
```



👉Insights:

- The correlation are found to nominal and on an average it is found that data has a `Positive Correlation` on weekly uploading rate

In [507…

```
mdc = md[['release_year','year_added','runtime_in_mins']].corr()
mdc
```

Out[507]:

	release_year	year_added	runtime_in_mins
release_year	1.000000	-0.024616	-0.235665
year_added	-0.024616	1.000000	0.073323
runtime_in_mins	-0.235665	0.073323	1.000000

In [508…

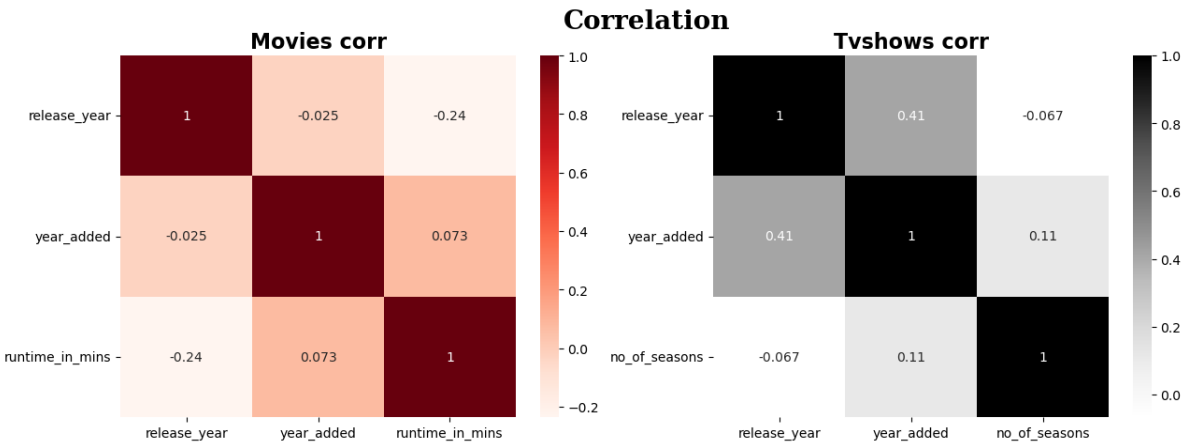
```
tvdc = tvd[['release_year','year_added','no_of_seasons']].corr()
tvdc
```

Out[508]:

	release_year	year_added	no_of_seasons
release_year	1.000000	0.412816	-0.066853
year_added	0.412816	1.000000	0.113203
no_of_seasons	-0.066853	0.113203	1.000000

In [511…

```
plt.figure(figsize=(15,5))
plt.suptitle('Correlation',fontsize=20,fontfamily='serif',fontweight='bold')

plt.subplot(1,2,1)
sns.heatmap(mdc,cmap='Reds',annot=True)
plt.title('Movies corr',fontsize=16,fontweight='bold')
plt.xticks(rotation=0)
plt.yticks(rotation=0)

plt.subplot(1,2,2)
sns.heatmap(tvdc,cmap='Greys',annot=True)
plt.title('Tvshows corr',fontsize=16,fontweight='bold')
plt.xticks(rotation=0)
plt.yticks(rotation=0)

plt.show()
```



👉Insights:

- The correlation are found to nominal and on an average it is found that data has a `least Positive Correlation` based on the coefficients of Movies and **More likely** `Positively correlated` on the TVShow data.

🧐🚀💡 *Recommendations for `Netflix` Business Growth*:💡🚀🧐

- **Invest in more original content**: Netflix has been very successful with its `original content` . The company should continue to invest in original content that is both high-

quality and appealing to a wide **Global audience** collaborating with the famous actors,directors to have a successful growth both contentwise and financially.

- **Expand into new markets**: Netflix is already available in over `190 countries`, but there are still many markets where the company could expand. The company should focus on expanding into markets where there is a large population of potential subscribers and where there is less competition from other streaming services. Prioritizing the contents rating would attract `more subscribers`.

- **Offer a lower-priced ad-supported tier**: Netflix could offer a lower-priced ad-supported tier to attract more subscribers. This would allow the company to reach a wider audience and `Generate more revenue`. These strategic decisions of release date and time would make the viewers count increase rapidly. Having the `ANIME` contents have shown as an interest for many views.

- **Personalize the user experience**: Netflix could do more to personalize the user experience. This could include recommending content based on a **user's viewing history**, offering different pricing plans based on a user's needs, and providing more localized content on the similar genre's.

- **Improve the user interface**: Netflix's user interface is generally good, but it could be improved. The company could make it easier to find content, especially when the content that is not as popular. The company could also make it easier to switch between different devices.

- **Partner with other companies**: Netflix could partner with other companies to offer `exclusive content` or to promote their services say starting their **own production house** to produce more contents based on likliness of the wide range of audiences and could expand into new genres of content, such as anime, documentaries, or stand-up comedy. This would help the company attract a wider audience.

- **Invest in technology**: Netflix could invest in new technologies, such as virtual reality or augmented reality. This would allow the company to offer new and immersive experiences to their subscribers.

- **Content Management**: Timely release of the contents ensure credibility and earns a good trust among the subscribers. The strategic Decisions of the release weekday , month matters and should be aligned with the viewers expectations.

`Netflix` **is a successful company, but there is always room for improvement**.

By following these recommendations & suggestions , `NETFLIX` can continue to Grow and attract new subscribers & Stratergize in a more effiecient way to stay ahead of its competition.....

Netflix's growth has not been without its challenges. In recent years, the company has faced increased competition from other streaming services, such as Disney+, HBO Max, and Amazon Prime Video. This has led to some declines in subscriber growth.

However, `Netflix` remains one of the ***Most Popular Streaming services** in the *world*. The company is well-positioned for future growth, as it continues to invest in original

content and expand into new markets.

- Done by:
`KASI`