Netflix

Defining Problem statement and analysing basic metrics Analyse 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 [352…

```
# Loading Netflix dataset
!wget  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/000/940/or
```

```
--2024-03-18 16:01:02--  https://d2beiqkhq929f0.cloudfront.net/public_assets/asse
ts/000/000/940/original/netflix.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.16
4.173.110, 18.164.173.18, 18.164.173.58, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.16
4.173.110|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3399671 (3.2M) [text/plain]
Saving to: ‘netflix.csv.2’

netflix.csv.2       100%[===================>]   3.24M  --.-KB/s    in 0.05s

2024-03-18 16:01:02 (60.3 MB/s) - ‘netflix.csv.2’ saved [3399671/3399671]
```

In [353…

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [354…

```
netflix = pd.read_csv("netflix.csv")
```

In [355…

```
netflix.head()
```

Out[355…

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

Checking the number of columns and rows in the dataset

In [356…

```
netflix.shape
```

Out[356…

```
(8807, 12)
```

There are 8807 rows and 12 rows in netflix dataset

Lets check what are the columns present in the data set and what are there datatypes.

In [357…
```
netflix.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
```

Observation:

We can identify there are a total of 8807 rows and 12 columns in the dataset. From the 12 columns only release_year is in “int” data type and rest all are in object data type. Date_added column is also in object data type which should to be converted to date_time data type. There are few missing values in director, cast, date_added, country columns.

Checking for Duplicates

In [358…
```
netflix.duplicated().any()
```

Out[358…
```
False
```

We can see there are no duplicates in the data set

Conversion of columns

In [359…
```
netflix["date_added"] = pd.to_datetime(netflix["date_added"]) #converting the ob
netflix['duration_num'] = netflix['duration'].str.extract('(\d+)', expand=False)
#duration column and saving it as duration_num (float datatype)
netflix.drop('duration', axis=1, inplace = True) #dropping the duration column
netflix["day"] =  netflix["date_added"].dt.day_name() #creating day column from
netflix["month"]=  netflix["date_added"].dt.month # creating a column for month
```

Checking the columns and rows in the dataset

In [360…
```
netflix.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 14 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   datetime64[ns]
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   listed_in     8807 non-null   object
 10  description   8807 non-null   object
 11  duration_num  8804 non-null   float64
 12  day           8797 non-null   object
 13  month         8797 non-null   float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(10)
memory usage: 963.4+ KB
```

Observation:

We can observe that the data type of date_added column is converted from object to date_time data type. New columns like day, month and duration_num(float) are added to the dataset. Duration_num is saved as float data type, since the duration of movies will be in minutes.

In [361…
```
netflix.head()
```

Out[361…

	show_id	type	title	director	cast	country	date_added	release_year
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2021
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	2021-09-24	2021
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2021

Observation:

We can observe that columns duration_num, day and month are stacked at the end of the dataset. We can also observe that month is in float datatype. This is because of the missing values in the column.

Checking the shape of the dataframe

In [362…

```
netflix.shape
```

Out[362…

```
(8807, 14)
```

Observation:

From the above data, we can identify there are 8807 rows and 14 columns

Statistical Inforamtion of the Data

In [363…
```
netflix.describe()
```

Out[363…

	release_year	duration_num	month
count	8807.000000	8804.000000	8797.000000
mean	2014.180198	69.846888	6.654996
std	8.819312	50.814828	3.436554
min	1925.000000	1.000000	1.000000
25%	2013.000000	2.000000	4.000000
50%	2017.000000	88.000000	7.000000
75%	2019.000000	106.000000	10.000000
max	2021.000000	312.000000	12.000000

In [364…
```
netflix.describe(include = object)
```

Out[364…

	show_id	type	title	director	cast	country	rating	listed_in
count	8807	8807	8807	6173	7982	7976	8803	8807
unique	8807	2	8807	4528	7692	748	17	514
top	s1	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	TV-MA	Dramas, International Movies
freq	1	6131	1	19	19	2818	3207	362

Checking for nulls

In [365…
```
netflix["director"].isna().sum()
```

Out[365…
```
2634
```

In [366…
```
netflix["cast"].isna().sum()
```

Out[366…
```
825
```

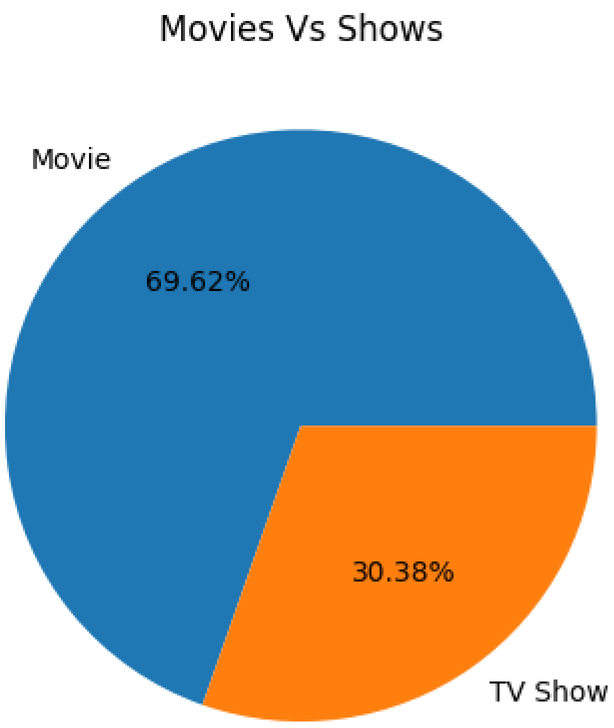
In [367…
```
netflix["country"].isna().sum()
```
Out[367…
```
831
```
In [368…
```
netflix["date_added"].isna().sum()
```
Out[368…
```
10
```
In [369…
```
netflix["duration_num"].isna().sum()
```
Out[369…
```
3
```
In [370…
```
netflix["rating"].isna().sum()
```
Out[370…
```
4
```

Observation:

There are 2634 values are missing in director column. 825, 831, 10 in cast, country and date_added columns. Since the month and day columns are formed from date_added, they will have the same number of missing values.

Non-Graphical Analysis: Value counts and unique attributes

In [371…
```
netflix["type"].value_counts()
```
Out[371…
```
Movie      6131
TV Show    2676
Name: type, dtype: int64
```
In [372…
```
plt.pie(data = netflix, x =netflix["type"].value_counts().values, labels = netfl
plt.title("Movies Vs Shows")
```
Out[372…
```
Text(0.5, 1.0, 'Movies Vs Shows')
```



Observation:

There are two categories in the type column. Movie and shows. There are 6131 movies and 2676 shows in the entire dataset.

In [373…
```
netflix["title"].value_counts().head(10)
```

Out[373…
```
Dick Johnson Is Dead                   1
Ip Man 2                               1
Hannibal Buress: Comedy Camisado       1
Turbo FAST                             1
Masha's Tales                          1
Chelsea Does                           1
Ricardo O'Farrill Abrazo Genial        1
Ip Man                                 1
Tom Segura: Mostly Stories             1
Team Foxcatcher                        1
Name: title, dtype: int64
```

In [374…
```
netflix["cast"].value_counts()
```

Out[374…
```
David Attenborough
19
Vatsal Dubey, Julie Tejwani, Rupa Bhimani, Jigna Bhardwaj, Rajesh Kava, Mousam,
Swapnil
14
Samuel West
10
Jeff Dunham
7
David Spade, London Hughes, Fortune Feimster
6

..
Michael Peña, Diego Luna, Tenoch Huerta, Joaquin Cosio, José María Yazpik, Matt
Letscher, Alyssa Diaz
1
Nick Lachey, Vanessa Lachey
1
Takeru Sato, Kasumi Arimura, Haru, Kentaro Sakaguchi, Takayuki Yamada, Kendo Ko
bayashi, Ken Yasuda, Arata Furuta, Suzuki Matsuo, Koichi Yamadera, Arata Iura,
Chikako Kaku, Kotaro Yoshida     1
Toyin Abraham, Sambasa Nzeribe, Chioma Chukwuka Akpotha, Chioma Omeruah, Chiwet
alu Agu, Dele Odule, Femi Adebayo, Bayray McNwizu, Biodun Stephen
1
Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanana, Manish Chaudhary, Meghna Mali
k, Malkeet Rauni, Anita Shabdish, Chittaranjan Tripathy
1
Name: cast, Length: 7692, dtype: int64
```

In [375…
```
netflix["director"].value_counts().head(10)
```

Out[375…
```
Rajiv Chilaka             19
Raúl Campos, Jan Suter    18
Marcus Raboy              16
Suhas Kadav               16
Jay Karas                 14
Cathy Garcia-Molina       13
Martin Scorsese           12
Youssef Chahine           12
Jay Chapman               12
Steven Spielberg          11
Name: director, dtype: int64
```

Observation:

These are the top 10 directors (both for shows and movies) from the dataset. from the above data, we can say that Rajiv Chilaka is the top director with a count of value count.

In [376…
```
#Country count
netflix["country"].value_counts().head(10)
```

Out[376…

```
United States     2818
India              972
United Kingdom     419
Japan              245
South Korea        199
Canada             181
Spain              145
France             124
Mexico             110
Egypt              106
Name: country, dtype: int64
```

Observation:

The data is for the top 10 countries. US stands out as the top country with 2818 of shows and movies together.

In [377…

```
netflix["release_year"].value_counts().head(10)
```

Out[377…

```
2018    1147
2017    1032
2019    1030
2020     953
2016     902
2021     592
2015     560
2014     352
2013     288
2012     237
Name: release_year, dtype: int64
```

Observation:

The above data is for the top 10 years with the respective movies and show counts together. In 2018, 1147 shows and movies were released on Netflix.

In [378…

```
# ratings and respective counts
top_rating = netflix["rating"].value_counts().head(10)
top_rating
```

Out[378…

```
TV-MA     3207
TV-14     2160
TV-PG      863
R          799
PG-13      490
TV-Y7      334
TV-Y       307
PG         287
TV-G       220
NR          80
Name: rating, dtype: int64
```

Observations:

These are the top 10 rating categories for both movies and show counts together. In TV-MA, is the top-rated category from the entire data and can be recommended for any age groups.

Note:

If we clearly observe there are a lot of nested values in the director, cast column which indicates that the count for those rows is applicable for all the values present in that particular row. These columns have to be unnested further to get the accurate value_counts for each director and cast respectively.

Unique values

In [379…
```
netflix["director"].nunique()
```
Out[379…
```
4528
```
In [380…
```
netflix["rating"].nunique()
```
Out[380…
```
17
```
In [381…
```
netflix["release_year"].nunique()
```
Out[381…
```
74
```
In [382…
```
netflix["country"].nunique()
```
Out[382…
```
748
```
In [383…
```
netflix["title"].nunique()
```
Out[383…
```
8807
```

Observation:

There are 4528 unique values in the director column. 17, 74, 748 and 8807 unique values in the rating, release_year, country and title columns respectively. Since the unique values in title column are equal to the length of the data set. We can consider it as the unique identifier.

Handling Missing values and nested columns

Pre-Processing the data

Filling missing values

In [384...

```python
netflix["cast"] = netflix["cast"].fillna("unknown")
netflix["listed_in"] = netflix["listed_in"].fillna("unknown")
netflix["country"] = netflix["country"].fillna("unknown")
netflix["director"] = netflix["director"].fillna("unknown")
```

The missing values of the data are filled with unkown in cast, listed_in, country and director

Unnesting the cast column

In [385...

```python
cast_data = netflix[["title", "cast"]]#creating a new data set using the title a
cast_data["listed_cast"] = cast_data["cast"].apply(lambda x: str(x).split(", "))
cast_data = cast_data.explode("listed_cast") #exploding the list values of cast
cast_data.drop("cast", inplace = True, axis = 1) #dropping the actual nested cas
cast_data.rename({"listed_cast": "cast"}, inplace = True, axis = 1) # renaming t
```

```
<ipython-input-385-a5f7d332eb11>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  cast_data["listed_cast"] = cast_data["cast"].apply(lambda x: str(x).split(",
")) #spliting the values present in rows based on ", " as new column (listed_cas
t)
```

In [386...

```python
cast_data
```

Out[386...

	title	cast
0	Dick Johnson Is Dead	unknown
1	Blood & Water	Ama Qamata
1	Blood & Water	Khosi Ngema
1	Blood & Water	Gail Mabalane
1	Blood & Water	Thabang Molaba
...	...	...
8806	Zubaan	Manish Chaudhary
8806	Zubaan	Meghna Malik
8806	Zubaan	Malkeet Rauni
8806	Zubaan	Anita Shabdish
8806	Zubaan	Chittaranjan Tripathy

64951 rows × 2 columns

Unnesting Directors column

In [387...

```
directors_data =  netflix[["title", "director"]] #creating a new data set using
directors_data["directors_list"] = directors_data["director"].apply(lambda x : s
directors_data = directors_data.explode("directors_list") #exploding the list va
directors_data.drop("director", axis =1, inplace =True) #dropping the actual nes
directors_data.rename({"directors_list" : "director"}, axis =1, inplace = True)
directors_data
```

```
<ipython-input-387-18b1b0e1aaa5>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  directors_data["directors_list"] = directors_data["director"].apply(lambda x :
str(x).split(", ")) #spliting the values present in rows based on ", " as new col
umn (director_list)
```

Out[387…

	title	director
0	Dick Johnson Is Dead	Kirsten Johnson
1	Blood & Water	unknown
2	Ganglands	Julien Leclercq
3	Jailbirds New Orleans	unknown
4	Kota Factory	unknown
...	...	...
8802	Zodiac	David Fincher
8803	Zombie Dumb	unknown
8804	Zombieland	Ruben Fleischer
8805	Zoom	Peter Hewitt
8806	Zubaan	Mozez Singh

9612 rows × 2 columns

Unnesting country column

In [388…

```
country_data = netflix[["title", "country"]]
country_data["list_country"] =  country_data["country"].apply(lambda x: str(x).s
country_data = country_data.explode("list_country")
country_data.drop("country", axis =1, inplace = True)
country_data.rename({"list_country": "counntry"})
country_data
```

```
<ipython-input-388-adacdbf7abd0>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  country_data["list_country"] =  country_data["country"].apply(lambda x: str(x).
split(", "))
```

Out[388…

	title	list_country
0	Dick Johnson Is Dead	United States
1	Blood & Water	South Africa
2	Ganglands	unknown
3	Jailbirds New Orleans	unknown
4	Kota Factory	India
...	...	...
8802	Zodiac	United States
8803	Zombie Dumb	unknown
8804	Zombieland	United States
8805	Zoom	United States
8806	Zubaan	India

10845 rows × 2 columns

Unnesting genre

In [389…

```
genre =  netflix[["title", "listed_in"]]
genre["list"] = genre["listed_in"].apply(lambda x : str(x).split(", "))
genre = genre.explode("list")
genre.drop("listed_in", inplace = True, axis =1)
genre.rename({"list": "listed_in"}, inplace= True, axis = 1)
genre
```

```
<ipython-input-389-1017de79b4c0>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  genre["list"] = genre["listed_in"].apply(lambda x : str(x).split(", "))
```

Out[389…

	title	listed_in
0	Dick Johnson Is Dead	Documentaries
1	Blood & Water	International TV Shows
1	Blood & Water	TV Dramas
1	Blood & Water	TV Mysteries
2	Ganglands	Crime TV Shows
...	...	...
8805	Zoom	Children & Family Movies
8805	Zoom	Comedies
8806	Zubaan	Dramas
8806	Zubaan	International Movies
8806	Zubaan	Music & Musicals

19323 rows × 2 columns

Meging the data

In [390…

```
cast_data = cast_data.merge(directors_data, on = "title", how="left") #Merging t
```

In [391…

```
cast_data = cast_data.merge(country_data, on = "title", how = "left") #Merging t
```

In [392…

```
cast_data = cast_data.merge(genre, on = "title", how = "left")  #Merging the gen
```

In [393…

```
netflix1 = netflix.merge(cast_data, on = "title", how = "left") # merging the ca
#creating a new data set as netflix1
```

In [394…

```
netflix1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201991 entries, 0 to 201990
Data columns (total 18 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_x    201991 non-null  object
 4   cast_x        201991 non-null  object
 5   country       201991 non-null  object
 6   date_added    201833 non-null  datetime64[ns]
 7   release_year  201991 non-null  int64
 8   rating        201924 non-null  object
 9   listed_in_x   201991 non-null  object
 10  description   201991 non-null  object
 11  duration_num  201988 non-null  float64
 12  day           201833 non-null  object
 13  month         201833 non-null  float64
 14  cast_y        201991 non-null  object
 15  director_y    201991 non-null  object
 16  list_country  201991 non-null  object
 17  listed_in_y   201991 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(14)
memory usage: 29.3+ MB
```

Observation:

All the unnested columns are merged to the actual dataset and creating a copy of it for further analysis.

Dropping the duplicated columns

In [395…

```
netflix1.drop(["director_x","cast_x","listed_in_x", "country"], axis =1, inplace
```

We are dropping the duplicated columns from the dataset and keeping only the unnested columns for further analysis.

Renaming the new columns:

In [396…

```
# renaming the columns
netflix1.rename({"cast_y": "cast","director_y": "director", "listed_in_y": "list
```

In [397…

```
netflix1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201991 entries, 0 to 201990
Data columns (total 14 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   date_added    201833 non-null  datetime64[ns]
 4   release_year  201991 non-null  int64         
 5   rating        201924 non-null  object        
 6   description   201991 non-null  object        
 7   duration_num  201988 non-null  float64       
 8   day           201833 non-null  object        
 9   month         201833 non-null  float64       
 10  cast          201991 non-null  object        
 11  director      201991 non-null  object        
 12  country       201991 non-null  object        
 13  listed_in     201991 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(10)
memory usage: 23.1+ MB
```

Filling null values in month column

In [398…
```
netflix1[["month", "date_added"]].isna().sum()
```

Out[398…
```
month         158
date_added    158
dtype: int64
```

In [399…
```
netflix1["month"] = netflix1["month"].fillna(0)# filling nulls with
```

In [400…
```
netflix1["month"] = netflix1["month"].astype(int) #converting the data type to i
```

Top10 genres

In [401…
```
top10_genres = netflix1.groupby('listed_in')['show_id'].nunique().sort_values(as
top10_genres
```

Out[401…
```
listed_in
International Movies        2752
Dramas                      2427
Comedies                    1674
International TV Shows      1351
Documentaries                869
Action & Adventure           859
TV Dramas                    763
Independent Movies           756
Children & Family Movies     641
Romantic Movies              616
Name: show_id, dtype: int64
```

These are the top 10 genres for shows and movies. International movies and Dramas tops the list with a count of 2752 and 2427 respectively.

Visual Analytics

Creating two seprate datasets for movies and shows

In [402…
```
movies = netflix1[netflix1["type"] == "Movie"] # Creating movies data set

shows = netflix1[netflix1["type"]== "TV Show"] # creating shows data set
```

Movies

In [403…
```
# How many different countries does netflix produce movies?
movies["country"].nunique()
```

Out[403…
```
123
```

Observation:

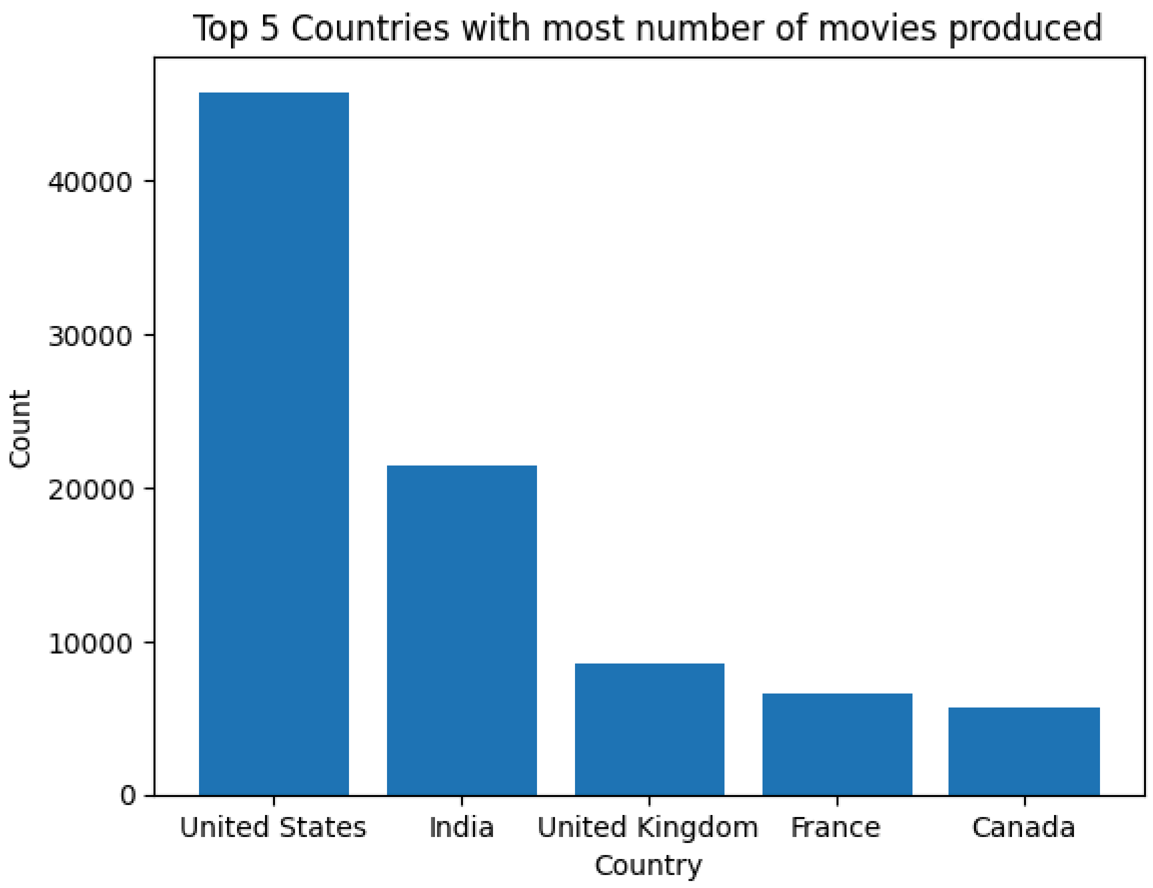
There are 123 unique countries in the movies data frame.

In [404…
```
#what are those countries ?
movies["country"].unique()
```

Out[404…
```
array(['United States', 'unknown', 'Ghana', 'Burkina Faso',
       'United Kingdom', 'Germany', 'Ethiopia', 'Czech Republic', 'India',
       'France', 'China', 'Canada', 'South Africa', 'Japan', 'Nigeria',
       'Spain', 'Australia', 'Mexico', 'Italy', 'Romania', 'Argentina',
       'Venezuela', 'Hong Kong', 'Nepal', 'New Zealand', 'Brazil',
       'Greece', 'Colombia', 'Belgium', 'Switzerland', 'Bulgaria', '',
       'Algeria', 'Poland', 'Israel', 'Saudi Arabia', 'Thailand',
       'Indonesia', 'Egypt', 'Denmark', 'Kuwait', 'Netherlands',
       'Singapore', 'Malaysia', 'South Korea', 'Vietnam', 'Hungary',
       'Lebanon', 'Syria', 'Philippines', 'United Arab Emirates',
       'Sweden', 'Qatar', 'Mauritius', 'Austria', 'Turkey', 'Russia',
       'Taiwan', 'Cameroon', 'Palestine', 'Ireland', 'United Kingdom,',
       'Kenya', 'Chile', 'Uruguay', 'Cambodia', 'Bangladesh', 'Portugal',
       'Cayman Islands', 'Norway', 'Iceland', 'Serbia', 'Malta',
       'Luxembourg', 'Namibia', 'Angola', 'Peru', 'Mozambique',
       'Cambodia,', 'Jordan', 'Zimbabwe', 'Pakistan', 'Guatemala',
       'Senegal', 'Finland', 'Iraq', 'Malawi', 'Paraguay', 'Iran',
       'United States,', 'Albania', 'Georgia', 'Soviet Union', 'Morocco',
       'Slovakia', 'West Germany', 'Ukraine', 'Bermuda', 'Ecuador',
       'Armenia', 'Mongolia', 'Bahamas', 'Sri Lanka', 'Latvia',
       'Liechtenstein', 'Nicaragua', 'Croatia', 'Poland,', 'Slovenia',
       'Dominican Republic', 'Samoa', 'Botswana', 'Vatican City',
       'Jamaica', 'Kazakhstan', 'Lithuania', 'Afghanistan', 'Somalia',
       'Sudan', 'Panama', 'Uganda', 'East Germany', 'Montenegro'],
      dtype=object)
```

In [405…

```
#Top5 countires
top_country = movies["country"].value_counts() #value counts of each country
top_country.drop("unknown", axis = 0, inplace = True) #dropping the missing valu
x = top_country.head(5).index  # setting x axis
y = top_country.head(5).values # setting y axis
plt.bar(x,y) # plotting a bar graph between x and y
plt.title('Top 5 Countries with most number of movies produced')
plt.xlabel("Country")
plt.ylabel("Count")
plt.show()
```



Obseravtion:

These are the top5 countries with highest movie count. A bar plot is plot using the data for the top 5 countries with movie count. The United States and India, UK were found to have the top3 places respectively in the movies data.

In [406…

```
#Top directors
top_directors= movies.groupby("director")["show_id"].nunique().sort_values(ascen
top_directors.drop("unknown", axis = 0, inplace = True) #dropping the unknown(nu
top_directors.head(5)
```

Out[406…

```
director
Rajiv Chilaka     22
Jan Suter         21
Raúl Campos       19
Suhas Kadav       16
Marcus Raboy      15
Name: show_id, dtype: int64
```

Observation:

These are the top 5 movie directors in the movies data. Rajvi Chilaka tops the list with a count of 22 movies followed by Jan Suter, Rahul Campos and others.

In [407…

```
# Based on the genre of the movie, in which country did most number of movies we

# counting the movies per country
movies_per_country = movies.groupby(["listed_in","country"])["show_id"].nunique(

# ranking based on count for each genre
movies_per_country["rank"] =  movies_per_country.groupby("listed_in")["cnt"].ran

movies_per_country = movies_per_country[movies_per_country["rank"]==1] # filteri
movies_per_country = movies_per_country.drop("rank", axis = 1) #dropping the ran
movies_per_country.reset_index(inplace = True)
movies_per_country.drop("index", axis = 1)
```

Out[407…

	listed_in	country	cnt
0	International Movies	India	864
1	Dramas	United States	835
2	Comedies	United States	680
3	Documentaries	United States	511
4	Action & Adventure	United States	404
5	Children & Family Movies	United States	390
6	Independent Movies	United States	390
7	Thrillers	United States	292
8	Romantic Movies	United States	225
9	Stand-Up Comedy	United States	216
10	Horror Movies	United States	201
11	Sci-Fi & Fantasy	United States	181
12	Music & Musicals	United States	147
13	Sports Movies	United States	113
14	Classic Movies	United States	81
15	LGBTQ Movies	United States	63
16	Anime Features	Japan	61
17	Cult Movies	United States	52
18	Faith & Spirituality	United States	42
19	Movies	unknown	23

Observation:

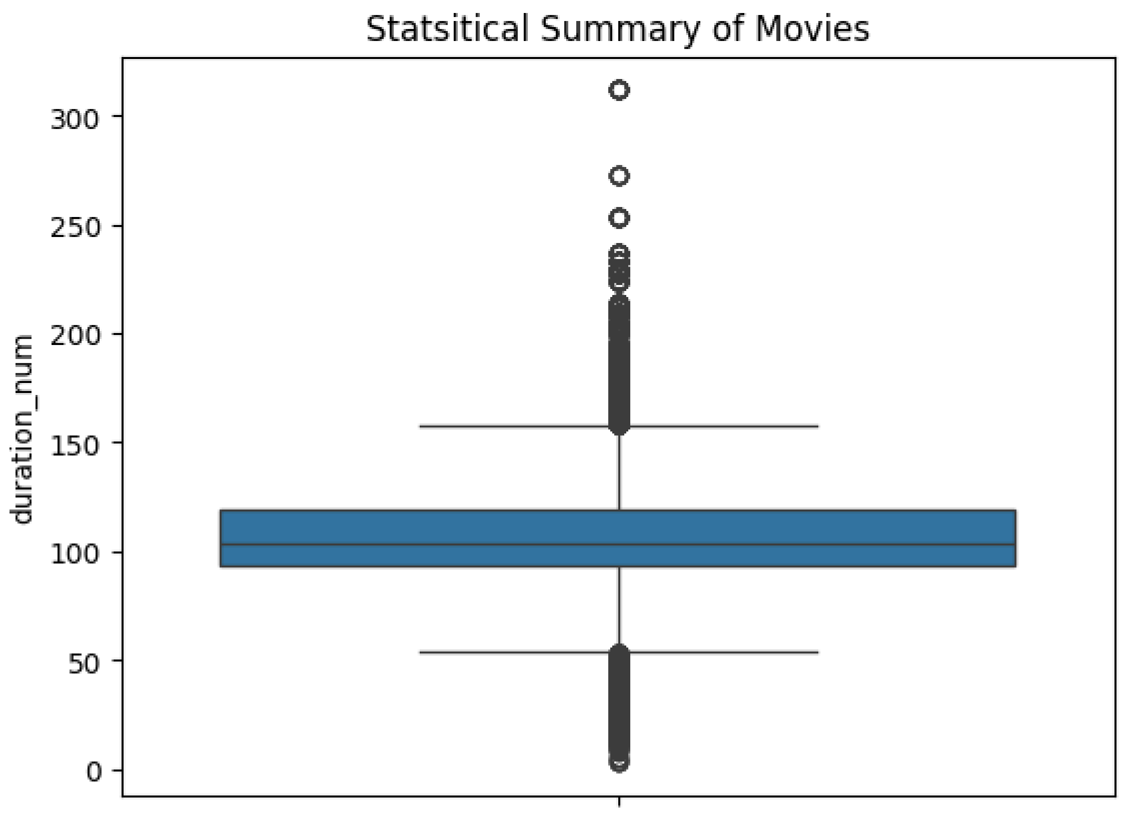
The above data shows the countries with highest movie counts per each genre. India is found to have the highest movie count in the international movies. Japan stands out as the top country for Anime Features. United States was found have the top place in most the genres. So, we can assume that US has a great market for Movies.

In [408…

```
# Detecting the outliers in Duration Column
sns.boxplot(data = movies, y= "duration_num")
plt.title("Statsitical Summary of Movies")
```

Out[408…

```
Text(0.5, 1.0, 'Statsitical Summary of Movies')
```



Observation:

From the above boxplot we can depict that there are lot of outliers in duration on both the negative and positive side.

In [409…

```
# In which years did most number of movies in netflix were originally released?
movies_per_year = movies.groupby(["release_year"])["show_id"].nunique().sort_val
movies_per_year.head(10)
```

Out[409…

	release_year	cnt
0	2018	767
1	2017	767
2	2016	658
3	2019	633
4	2020	517
5	2015	398
6	2021	277
7	2014	264
8	2013	225
9	2012	173

Observation:

2018 and 2017 has the most releases of movies with a count of 767.

In [410…

```
# What is the average, min and max duration of the movies per genre ?
movies.groupby("listed_in")["duration_num"].agg([np.mean, np.min, np.max])
```

Out[410…

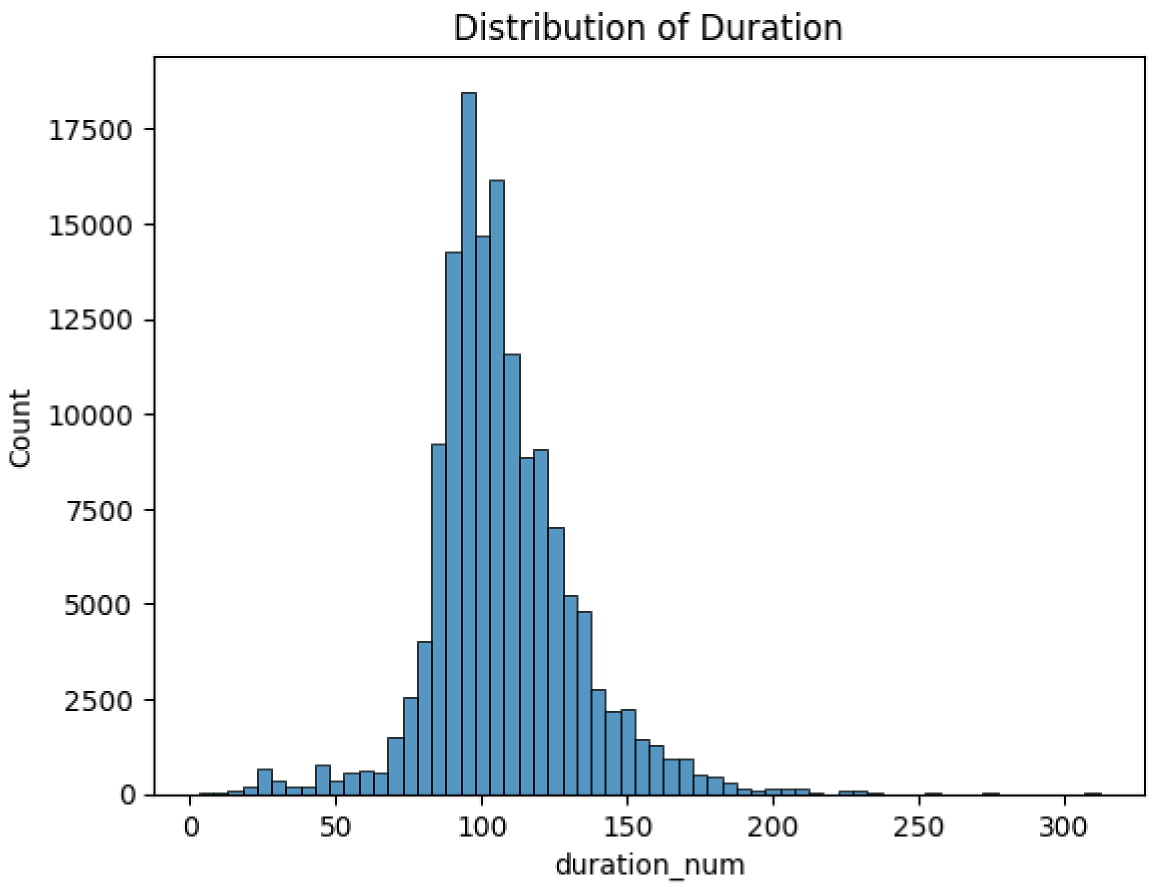
listed_in	mean	min	max
Action & Adventure	113.166339	5.0	214.0
Anime Features	95.920574	5.0	140.0
Children & Family Movies	85.431788	3.0	152.0
Classic Movies	126.979777	18.0	229.0
Comedies	102.024245	13.0	253.0
Cult Movies	111.034355	47.0	172.0
Documentaries	86.816369	10.0	273.0
Dramas	113.302872	8.0	312.0
Faith & Spirituality	109.006954	32.0	205.0
Horror Movies	99.019033	29.0	171.0
Independent Movies	102.783201	13.0	189.0
International Movies	112.020701	5.0	312.0
LGBTQ Movies	100.285203	17.0	143.0
Movies	48.838631	19.0	115.0
Music & Musicals	112.045499	15.0	224.0
Romantic Movies	110.013256	46.0	233.0
Sci-Fi & Fantasy	108.727025	3.0	312.0
Sports Movies	101.289353	12.0	161.0
Stand-Up Comedy	68.844444	28.0	146.0
Thrillers	108.326439	28.0	171.0

Observation:

The average, min and max values of duration(mins) for each genre are listed in the above table. Among all the genres movies that belong to Classic Movies genre have the highest average duration with 126.78 minutes. Movies from Dramas and International Movies genres have the maximum duration with 312 minutes. Movies from Children & Family Movies and Sci-Fi & Fantasy genres have the minimum duration with 3 minutes.

In [411…
```
# Check the distribution of the Duration column
sns.histplot(data = movies, x= "duration_num", binwidth = 5)
plt.title("Distribution of Duration")
```

Out[411…
```
Text(0.5, 1.0, 'Distribution of Duration')
```

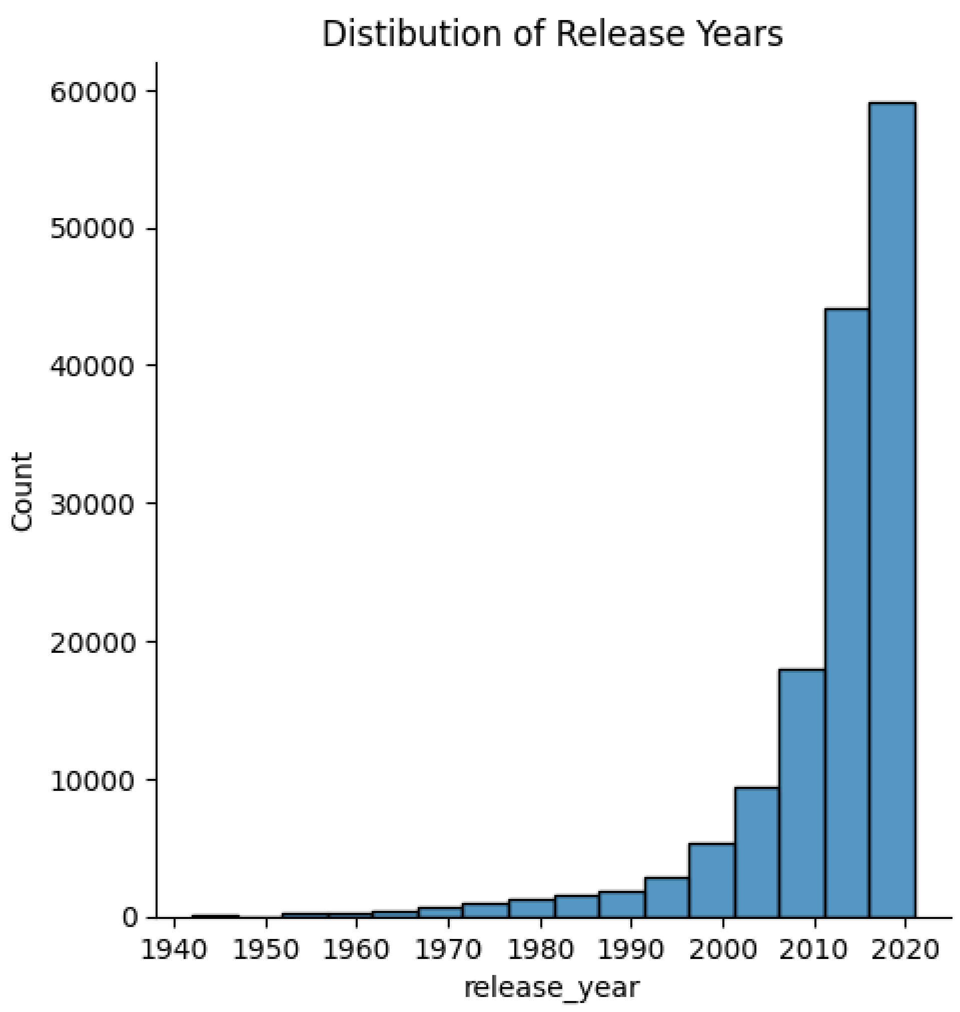


Observation:

A histplot is plotted for duration column and was found that movies with duration between ~80 and 120 are more in the movies data. We can predict that the ideal runtime of a movie should be 120 mins.

In [412…
```
# Check the distribution of the Release Year column
sns.displot(data = movies, x= "release_year", binwidth = 5)
plt.title("Distibution of Release Years")
```

Out[412…
```
Text(0.5, 1.0, 'Distibution of Release Years')
```

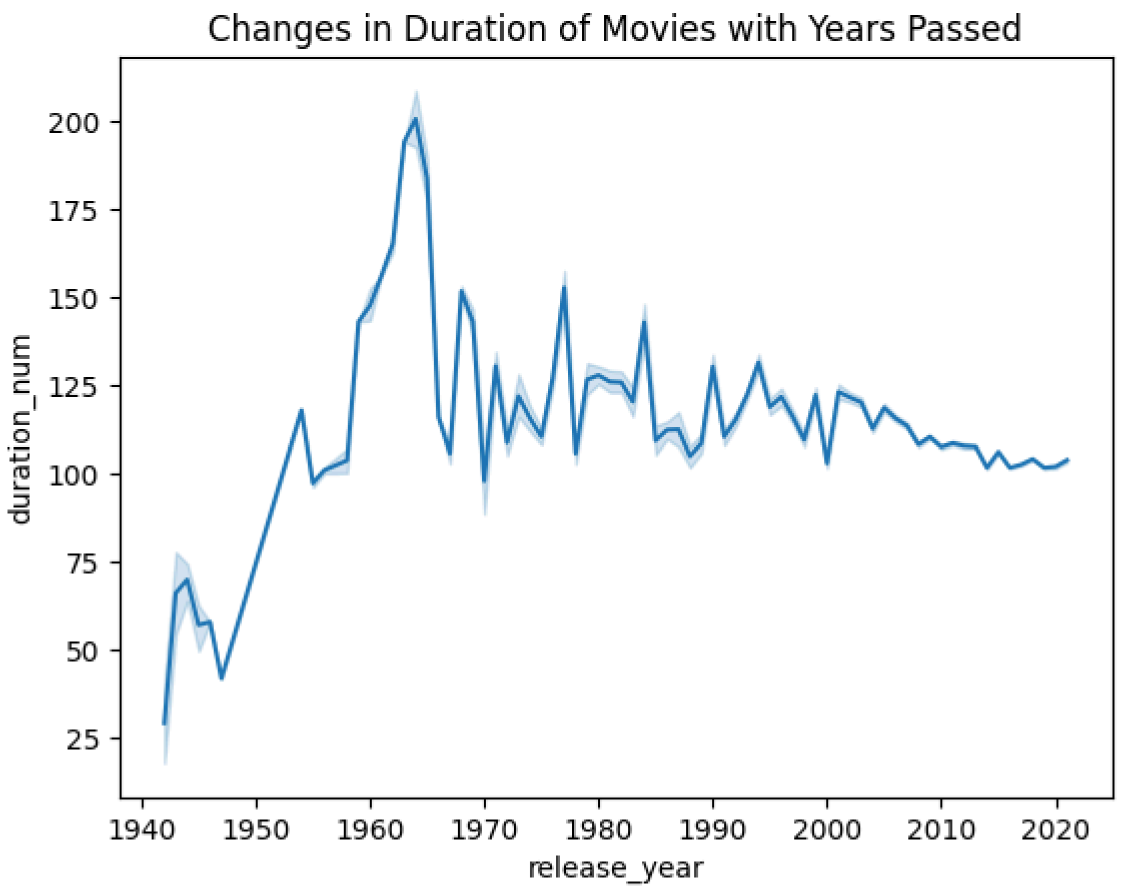


Observation:

A distplot is plotted between release_year and their respective count. The height of the hist bar is found to be high for years 2010 to 2020 which indicates that the greatest number of movies were released in the last decade.

In [413…
```
# How has the duration of movies released per year changed over time?

sns.lineplot(data= movies, x= "release_year", y = "duration_num")
plt.title("Changes in Duration of Movies with Years Passed")
```

Out[413…
```
Text(0.5, 1.0, 'Changes in Duration of Movies with Years Passed')
```



Observation:

We can see there is sudden change in the duration in 1960 and decreased in 1970. There is a fluctuation in duration from 1970 to 1990 and got stable from 1990 to 2020.

In [414…

```
# What are the names of the directors that directed most number of movies per ge

# grouping the directors and respective count based on genre
directors_per_genre = movies.groupby(["listed_in","director"])["show_id"].nuniqu

directors_per_genre = directors_per_genre.reset_index(name = "cnt") # setting an

# filtering out all the unknown values(missing values)
directors_per_genre = directors_per_genre[directors_per_genre["director"]!= "unk

#ranking the directors based on the count
directors_per_genre["rank"] = directors_per_genre.groupby("listed_in")["cnt"].ra

#filtering all the top directors from each genre
directors_per_genre = directors_per_genre[directors_per_genre["rank"]==1]

directors_per_genre= directors_per_genre[["listed_in", "director", "cnt"]]
directors_per_genre.reset_index(inplace = True) #resetting the index
directors_per_genre.drop("index", axis =1)
```

Out[414…

	listed_in	director	cnt
0	Children & Family Movies	Rajiv Chilaka	22
1	Stand-Up Comedy	Jan Suter	21
2	International Movies	Cathy Garcia-Molina	13
3	Dramas	Youssef Chahine	12
4	Comedies	David Dhawan	9
5	Action & Adventure	Don Michael Paul	9
6	Romantic Movies	Cathy Garcia-Molina	8
7	Classic Movies	Youssef Chahine	8
8	Anime Features	Toshiya Shinohara	7
9	Horror Movies	Rocky Soraya	6
10	Music & Musicals	Matt Askem	6
11	Documentaries	Vlad Yudin	6
12	Independent Movies	Paul Thomas Anderson	5
13	Independent Movies	Noah Baumbach	5
14	Faith & Spirituality	David Batty	5
15	Sci-Fi & Fantasy	Lilly Wachowski	4
16	Sci-Fi & Fantasy	Lana Wachowski	4
17	Thrillers	David Fincher	4
18	Sports Movies	Vlad Yudin	4
19	Thrillers	Rathindran R Prasad	4
20	Movies	Louis C.K.	3
21	Cult Movies	Mike Clattenburg	3
22	LGBTQ Movies	Jun Lana	2
23	LGBTQ Movies	Leigh Janiak	2
24	LGBTQ Movies	Saratswadee Wongsomphet	2
25	LGBTQ Movies	Matt Kugelman	2

In [415…

```
# What are the names of the actors that acted in most number of movies per genre
actor_per_genre = movies.groupby(["listed_in","cast"])["show_id"].nunique().sort
actor_per_genre = actor_per_genre[actor_per_genre["cast"]  != "unknown"]
actor_per_genre["rank"]= actor_per_genre.groupby("listed_in")["Count"].rank(meth
actor_per_genre = actor_per_genre[actor_per_genre["rank"]==1]
actor_per_genre= actor_per_genre.drop("rank", axis = 1)
actor_per_genre.reset_index(inplace = True) #resetting the index
actor_per_genre.drop("index", axis =1).head(10)
```

Out[415…

	listed_in	cast	Count
0	International Movies	Anupam Kher	38
1	Dramas	Anupam Kher	28
2	Dramas	Shah Rukh Khan	28
3	Dramas	Naseeruddin Shah	28
4	Children & Family Movies	Julie Tejwani	26
5	Comedies	Anupam Kher	20
6	Action & Adventure	Bruce Willis	13
7	Anime Features	Yuki Kaji	10
8	Independent Movies	Naseeruddin Shah	10
9	Documentaries	Samuel West	10

Adding Year column to Movies data

In [416…

```
movies["year"] = movies["date_added"].dt.year
```

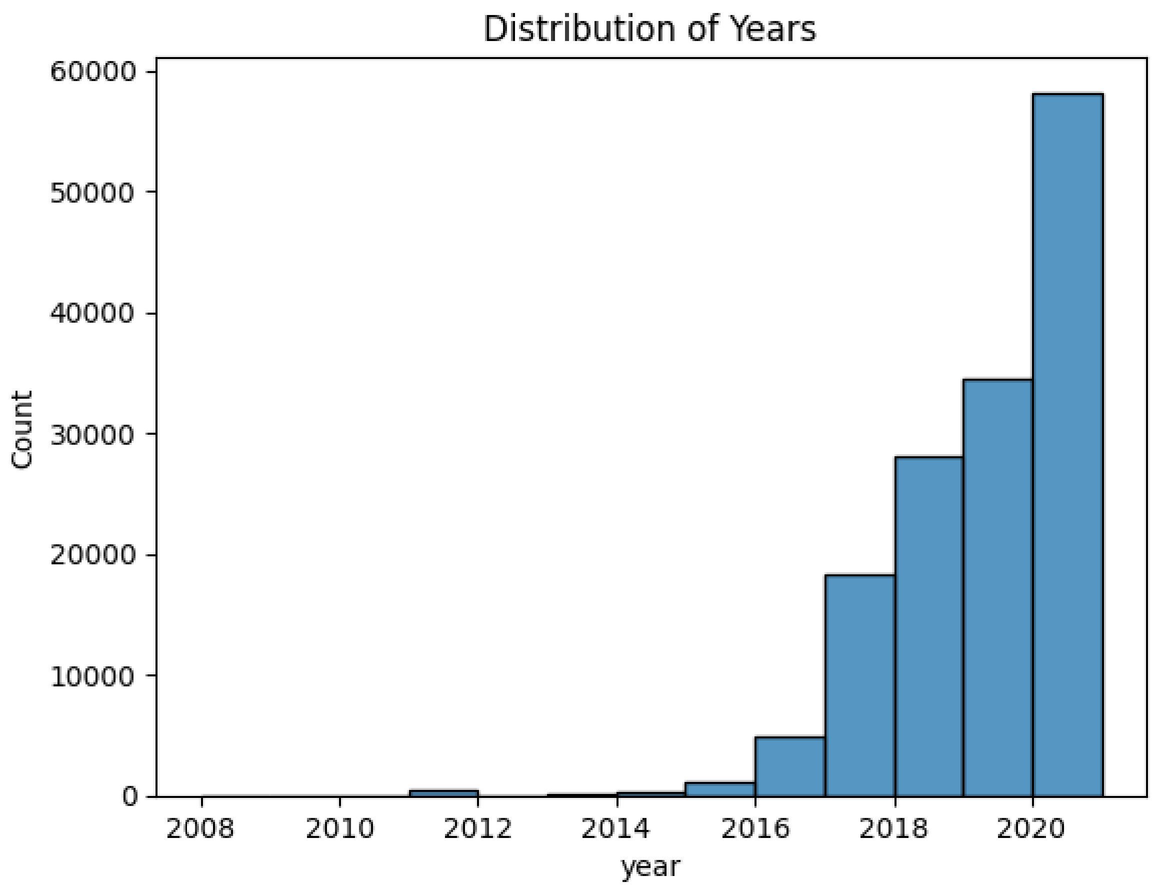
```
<ipython-input-416-20044d41b597>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  movies["year"] = movies["date_added"].dt.year
```

In [417…

```
# Check the distribution of the Year Added column
sns.histplot(movies["year"], binwidth = 1)
plt.title("Distribution of Years")
```

Out[417…

```
Text(0.5, 1.0, 'Distribution of Years')
```



Observation:

There is huge increase in number of movies added to Netflix in 2020. Whoever, we can assume that Netflix more active from 2016.

In [418…

```
# In which years did most number of movies were added to netflix per genre?

#grouping the listed_in and year and counting the values
movies_per_genre = movies.groupby(["listed_in", "year"])["show_id"].nunique().sc

movies_per_genre = movies_per_genre.reset_index(name = "cnt") # reseting the ind

# ranking the years and respective values using dense rank
movies_per_genre["rank"] = movies_per_genre.groupby("listed_in")["cnt"].rank(met

movies_per_genre = movies_per_genre[movies_per_genre["rank"]==1] #filtering the
movies_per_genre[["listed_in", "year", "cnt"]] #extracting the listed_in and yea
```

Out[418…

	listed_in	year	cnt
0	International Movies	2018	668
3	Dramas	2019	564
6	Comedies	2019	420
14	Documentaries	2017	206
15	Action & Adventure	2019	202
16	Independent Movies	2019	201
20	Romantic Movies	2020	173
23	Children & Family Movies	2020	170
29	Thrillers	2019	135
39	Horror Movies	2019	97
41	Music & Musicals	2018	96
43	Stand-Up Comedy	2018	89
51	Sci-Fi & Fantasy	2019	70
61	Sports Movies	2019	56
66	Classic Movies	2019	44
76	LGBTQ Movies	2020	28
77	Anime Features	2021	23
79	Cult Movies	2019	22
81	Faith & Spirituality	2018	20
84	Movies	2018	19

Observation:

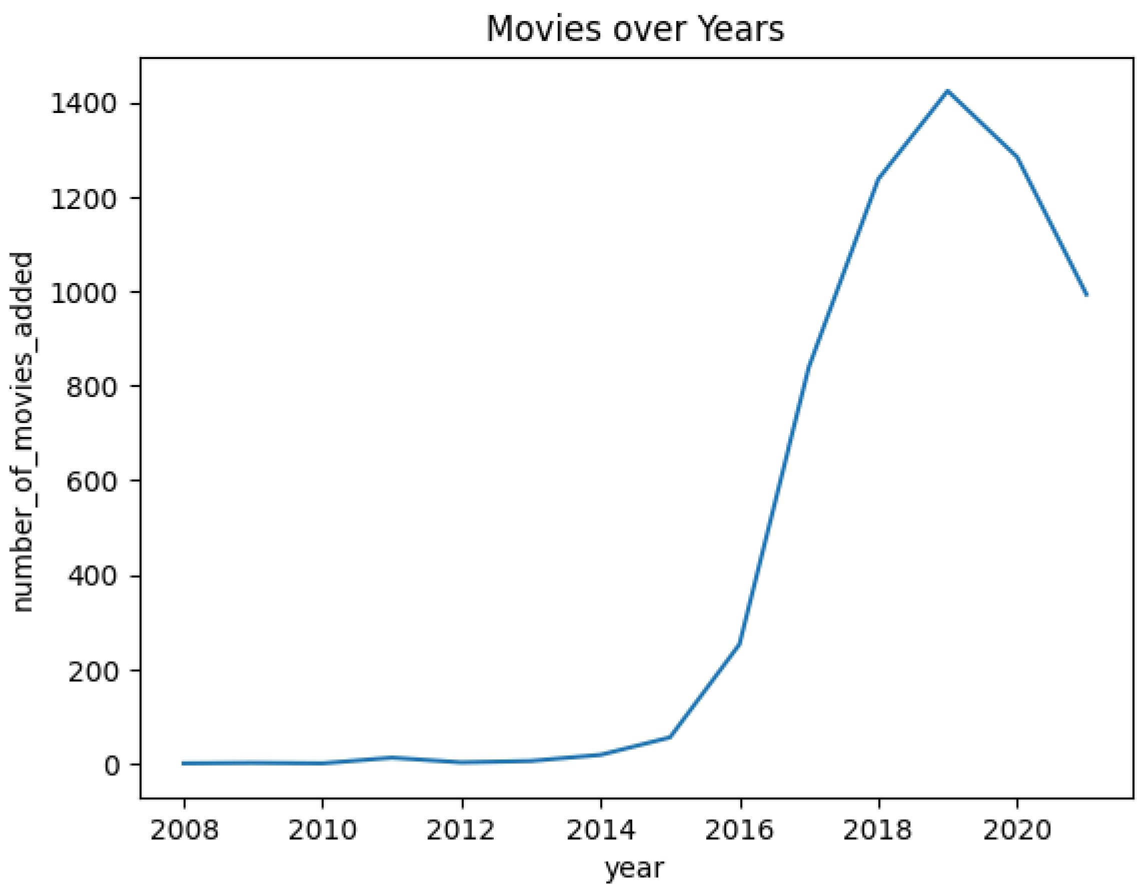
The table represent the year with highest movie releases per each genre. Most of the genres have highest releases during 2020 and 2019.

In [419…

```
# Q: How many number of movies were made per year?
movies_per_year = movies.groupby('year')['title'].nunique().sort_values(ascendin
sns.lineplot(data = movies_per_year, x= "year", y = "number_of_movies_added")
plt.title("Movies over Years")
```

Out[419…

```
Text(0.5, 1.0, 'Movies over Years')
```



Observation:

Number of movies that were added to Netflix was calculated. We can observe that 2019 tops the table with a count of 1424 and followed by 2020, 2018 with 1284 and 1237 respectively. A line plot is plot between the number of movies added per year to Netflix where we can see a huge peak at 2015 and decreasing gradually.

In [420…

```
# How many number of movies are available in netflix per genre?
top_genres = movies.groupby("listed_in")["show_id"].nunique().sort_values(ascend
top_genres = top_genres.reset_index(name ="cnt")
top_genres
```

Out[420…

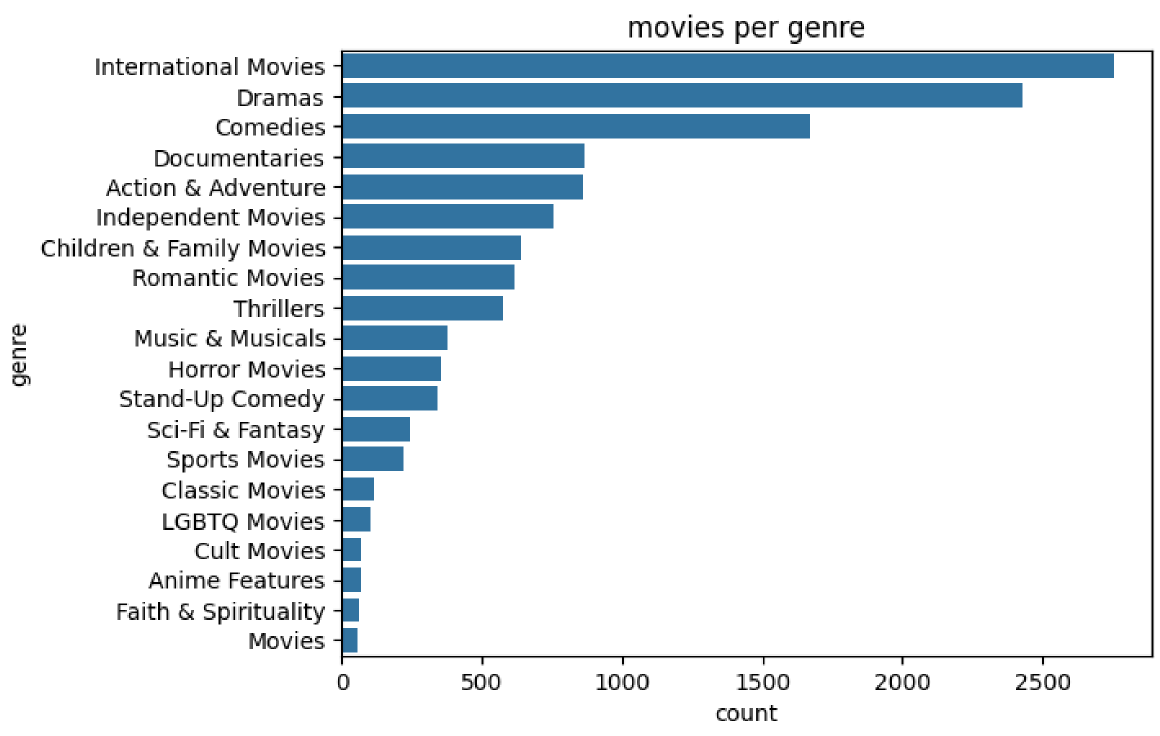
	listed_in	cnt
0	International Movies	2752
1	Dramas	2427
2	Comedies	1674
3	Documentaries	869
4	Action & Adventure	859
5	Independent Movies	756
6	Children & Family Movies	641
7	Romantic Movies	616
8	Thrillers	577
9	Music & Musicals	375
10	Horror Movies	357
11	Stand-Up Comedy	343
12	Sci-Fi & Fantasy	243
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 [421…

```
sns.barplot(data = top_genres, y= "listed_in", x= "cnt")
plt.title("movies per genre")
plt.ylabel("genre")
plt.xlabel("count")
```

Out[421…

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



Observation:

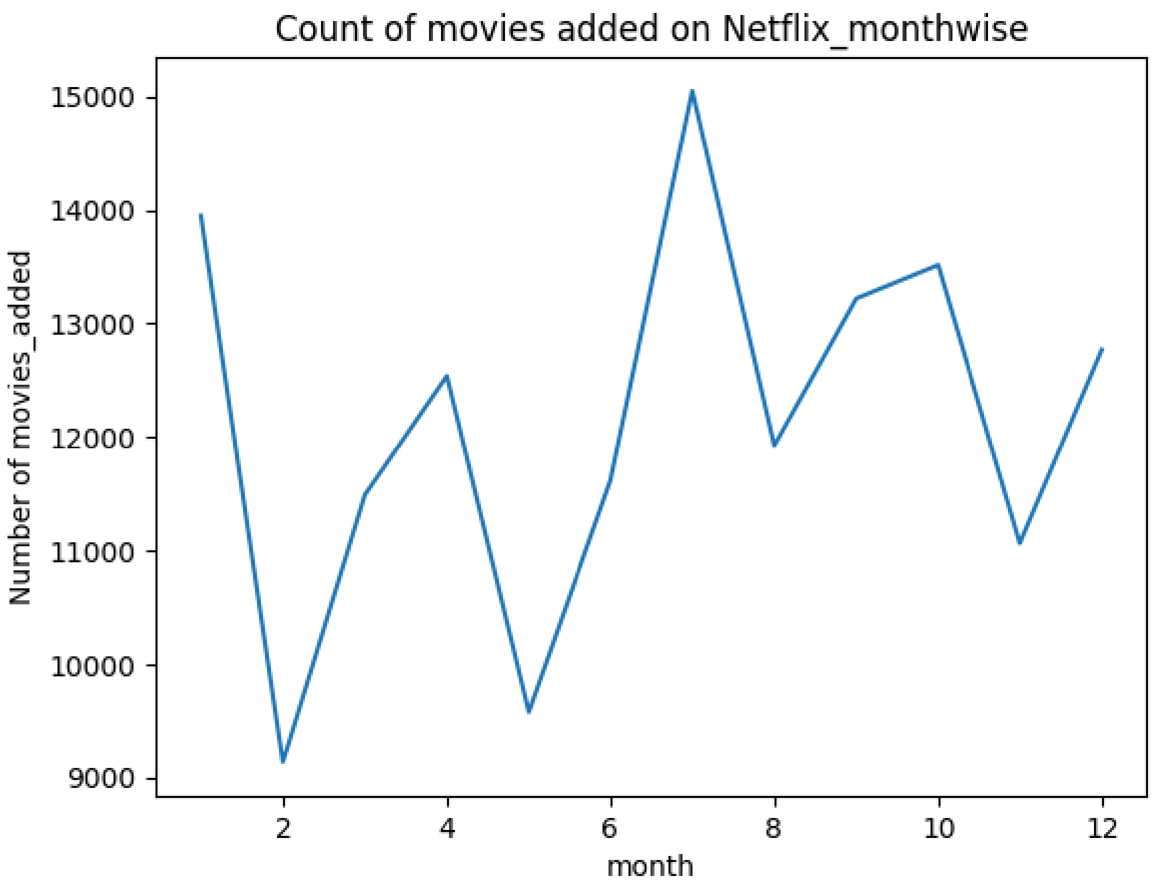
Number of movies per genre added to Netflix was calculated. International movies was found to have the highest number of movies count which is followed by Dramas , comedies respectively.

In [422…

```
#months with Highest movies release
movies_per_month = movies.groupby(["month"])["title"].count().sort_values(ascend
movies_per_month = movies_per_month.reset_index(name = "count")
sns.lineplot(data= movies_per_month, x="month", y="count")
plt.xlabel("month")
plt.ylabel("Number of movies_added")
plt.title("Count of movies added on Netflix_monthwise")
```

Out[422…

```
Text(0.5, 1.0, 'Count of movies added on Netflix_monthwise')
```

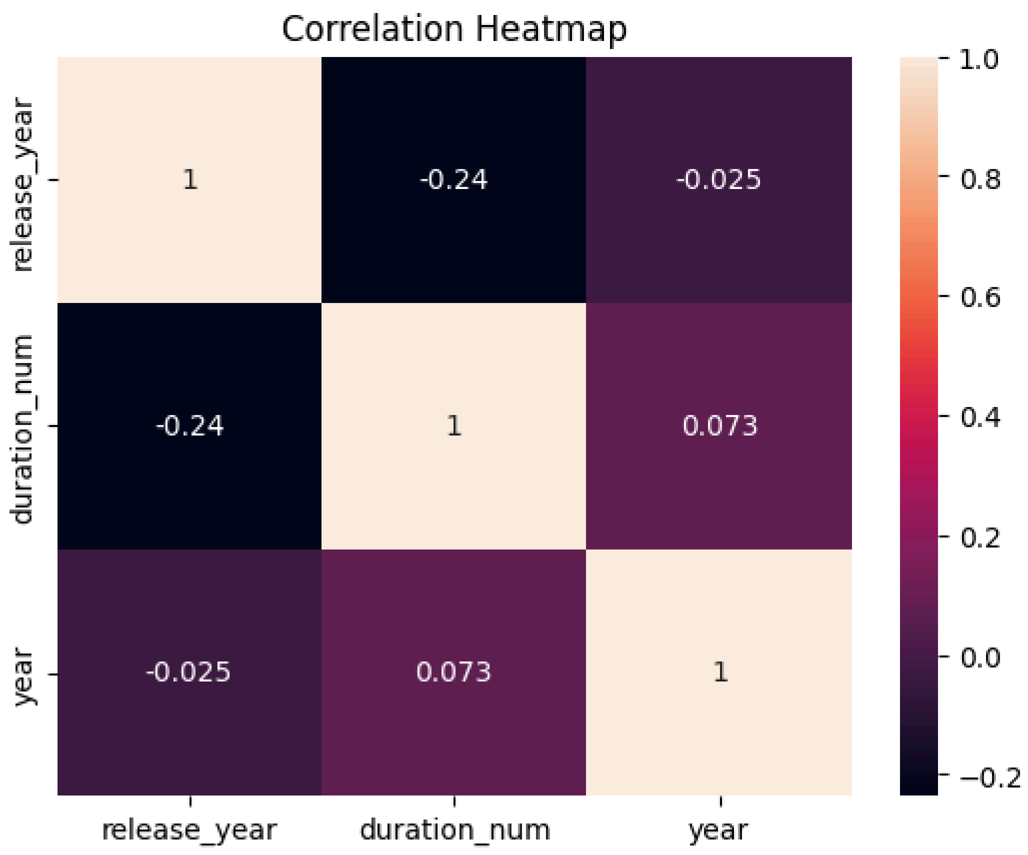


Observation:

The number of movies released of particular month of year are calculated. We can observe that more number of movies were released during the month of July and Jan. July falls under summer in many of the countires and students will have summer vactions which would help in gaining more audience. Jan follows the christmas week and a starts with a new year. There are high chance of people in countires like US, Canda and Uk will be having free time. So these two months are best for movie releases.

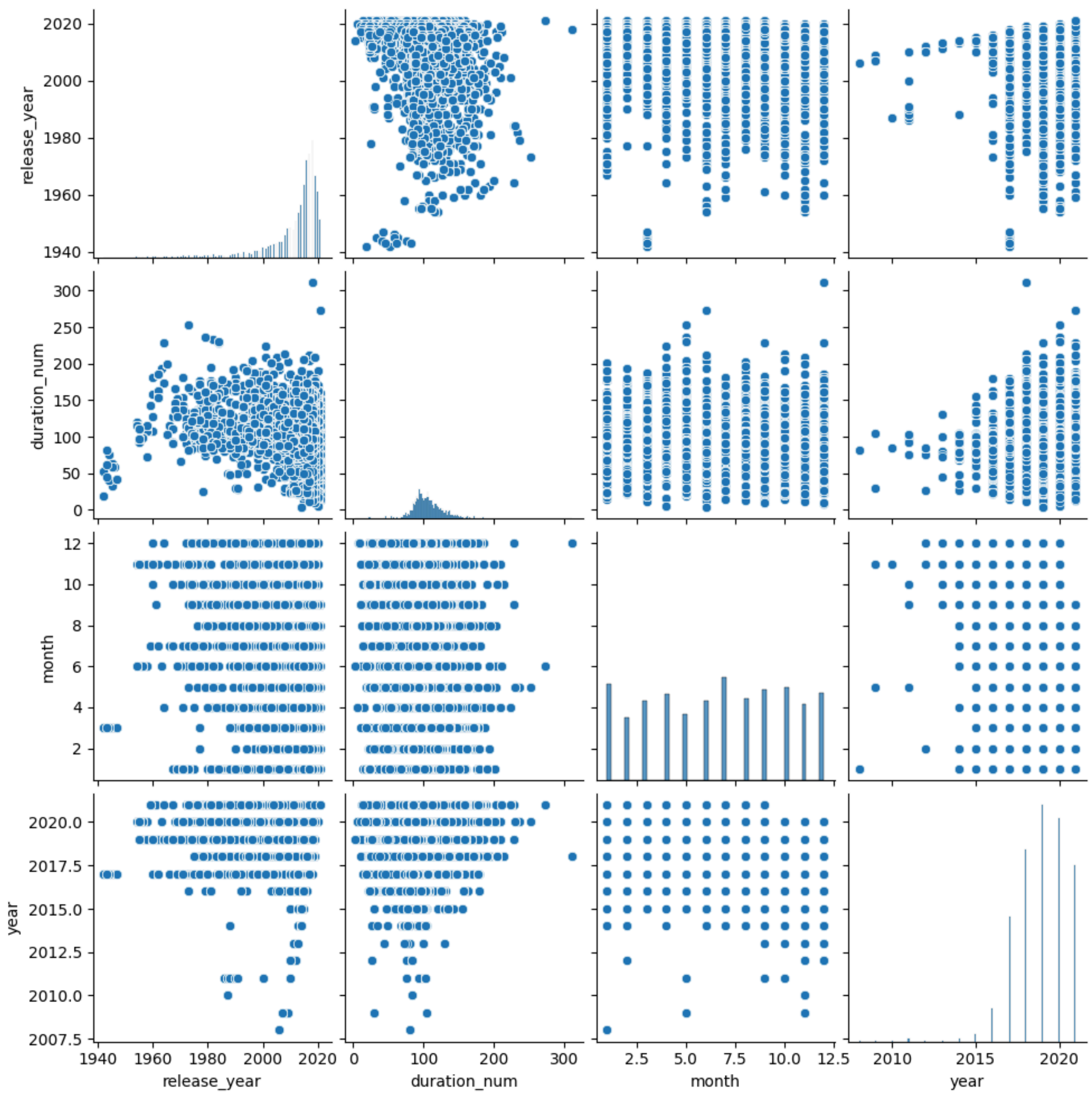
In [423…
```
#Check for correlations among numerical features
sns.heatmap(movies[['release_year','duration_num','year']].corr(), annot =True)
plt.title('Correlation Heatmap')
```

Out[423…
```
Text(0.5, 1.0, 'Correlation Heatmap')
```



In [424…
```
sns.pairplot(data= movies)
```

Out[424…
```
<seaborn.axisgrid.PairGrid at 0x7e25b7196710>
```



Observation:

1. Duration and year_added columns are the only pair that have positive correlation.
2. Duration and year_added columns have weak positive correlation.
3. Release year and duration columns have weak negative correlation.
4. Release year and year_added columns have weak negative correlation.

SHOWS

In [425…
```
# How many different countries does netflix produce tv_shows and what are those
shows["country"].nunique()
```

Out[425…
```
67
```

Observation: There are total 67 unique shows present in the Netflix dataset

In [426…
```
shows["country"].unique()
```

Out[426…

```
array(['South Africa', 'unknown', 'India', 'United Kingdom',
       'United States', 'Mexico', 'Turkey', 'Australia', 'Finland',
       'Nigeria', 'Japan', 'Belgium', 'France', 'South Korea', 'Spain',
       'Singapore', 'Russia', '', 'Ireland', 'Italy', 'Argentina',
       'Jordan', 'Colombia', 'Israel', 'Taiwan', 'Germany', 'Canada',
       'Poland', 'Thailand', 'New Zealand', 'Netherlands', 'Sweden',
       'China', 'Iceland', 'Denmark', 'Philippines', 'Indonesia',
       'United Arab Emirates', 'Norway', 'Czech Republic', 'Lebanon',
       'Brazil', 'Uruguay', 'Egypt', 'Luxembourg', 'Senegal',
       'Saudi Arabia', 'Kuwait', 'Belarus', 'Chile', 'Malta',
       'Puerto Rico', 'Austria', 'Cyprus', 'Malaysia', 'Mauritius',
       'Hong Kong', 'Croatia', 'West Germany', 'Syria', 'Hungary', 'Cuba',
       'Greece', 'Pakistan', 'Azerbaijan', 'Ukraine', 'Switzerland'],
      dtype=object)
```

In [427…

```
# What are the top 5 countries that produced the most number of tv_shows?
top5_shows = shows.groupby(["country"])["show_id"].nunique().sort_values(ascendi
top5_shows = top5_shows.drop("unknown", axis =0)
top5_shows.head(5)
```

Out[427…

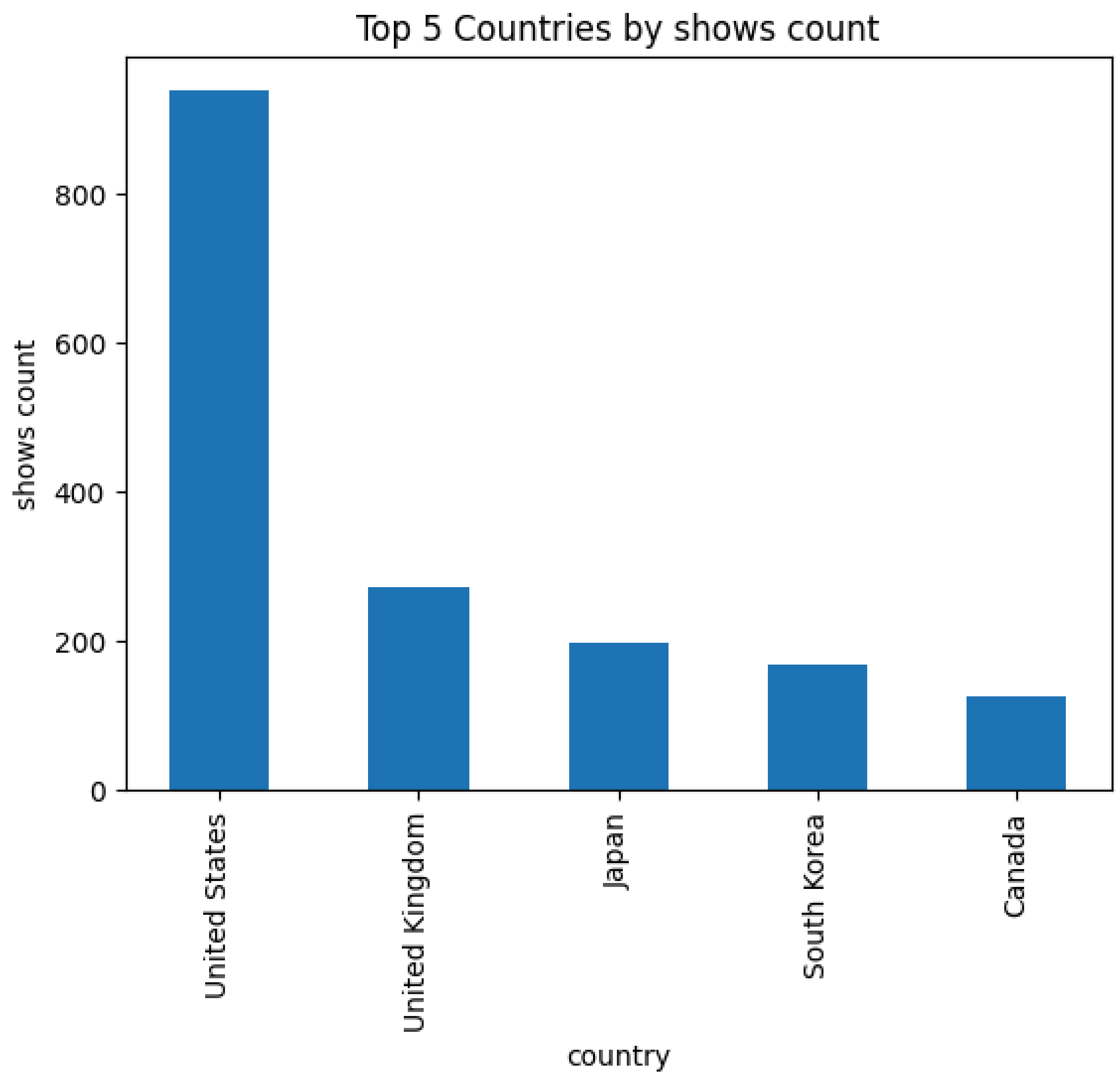
```
country
United States     938
United Kingdom    272
Japan             199
South Korea       170
Canada            126
Name: show_id, dtype: int64
```

In [428…

```
x = top5_shows.head(5).index
y = top5_shows.head(5).values
plt.bar(x,y, width = 0.5)
plt.xticks(rotation = 90)
plt.title("Top 5 Countries by shows count")
plt.xlabel("country")
plt.ylabel("shows count")
```

Out[428…

```
Text(0, 0.5, 'shows count')
```



Observation:

The top5 countries with the highest number of shows made are calculated and plotted as a graph. United States stands at the top of the list with a count of 938 shows followed by United Kingdom, Japan, south Korea and Canada.

In [429…

```
# Based on the genre of the tv_show, in which country did most number of tv_show
shows_per_country = shows.groupby(["listed_in", "country"])["show_id"].count().s
shows_per_country = shows_per_country.reset_index(name= "cnt")
shows_per_country["rank"] = shows_per_country.groupby("listed_in")["cnt"].rank(m
shows_per_country= shows_per_country[shows_per_country["rank"]==1]
shows_per_country[["listed_in", "country"]]
```

Out[429…

	listed_in	country
0	TV Dramas	United States
1	TV Comedies	United States
2	International TV Shows	Japan
3	Kids' TV	United States
4	Anime Series	Japan
6	British TV Shows	United Kingdom
8	TV Action & Adventure	United States
9	Crime TV Shows	United States
10	Korean TV Shows	South Korea
14	TV Sci-Fi & Fantasy	United States
17	Romantic TV Shows	South Korea
19	Spanish-Language TV Shows	Mexico
20	TV Mysteries	United States
30	TV Horror	United States
31	Docuseries	United States
45	TV Thrillers	United States
47	Teen TV Shows	United States
56	Reality TV	United States
69	TV Shows	India
80	Stand-Up Comedy & Talk Shows	United States
82	Classic & Cult TV	United States
143	Science & Nature TV	United States

Observation:

The table implicates the countries with highest number of shows per each genre. US was found to have the highest number of shows in most of the genres. Japan and South Korea occupying few.

In [430…

```
# In which years did most number of tv_shows in netflix were originally released
shows.groupby("release_year")["show_id"].nunique().sort_values(ascending = False
```

Out[430…

```
release_year
2020    436
2019    397
2018    380
2021    315
2017    265
2016    244
2015    162
2014     88
2012     64
2013     63
Name: show_id, dtype: int64
```

Observation:

The number of series released per year were calculated and Top10 out of them were listed down. 2020 has the highest number of shows with a count of 436. It was followed by 2019, 2018 with a count of 397 and 380 respectively.

Adding year column to the dataset by splitting it from the date_added column:

In [431…

```
shows["year"] = shows["date_added"].dt.year
shows["year"] = shows["year"].fillna(0)
shows["year"]= shows["year"].astype(int)
```

```
<ipython-input-431-3bde9db9458c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  shows["year"] = shows["date_added"].dt.year
<ipython-input-431-3bde9db9458c>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  shows["year"] = shows["year"].fillna(0)
<ipython-input-431-3bde9db9458c>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  shows["year"]= shows["year"].astype(int)
```

In [432…

```
# In which years did most number of tv_shows were added to netflix per genre?
top_years = shows.groupby(["listed_in","year"])["show_id"].count().sort_values(a
top_years =top_years.reset_index(name = "cnt")
top_years["rank"]= top_years.groupby("listed_in")["cnt"].rank(method = "dense",
top_years = top_years[top_years["rank"]==1]
top_years = top_years[["listed_in", "year"]].reset_index()
top_years.drop("index", axis =1)
```

Out[432…

	listed_in	year
0	International TV Shows	2019
1	TV Dramas	2020
2	Kids' TV	2020
3	TV Comedies	2020
4	Crime TV Shows	2020
5	Romantic TV Shows	2019
6	Anime Series	2020
7	TV Action & Adventure	2020
8	Spanish-Language TV Shows	2019
9	Korean TV Shows	2019
10	British TV Shows	2019
11	TV Mysteries	2020
12	TV Horror	2020
13	TV Sci-Fi & Fantasy	2020
14	TV Thrillers	2019
15	TV Shows	2021
16	Reality TV	2018
17	Docuseries	2021
18	Teen TV Shows	2020
19	Stand-Up Comedy & Talk Shows	2019
20	Classic & Cult TV	2017
21	Science & Nature TV	2021

Observation:

A bar plot was plotted against the number of movies released per year. Year with the highest number of shows per each genre added to Netflix was calculated and listed down. Almost all the genre have highest releases in 2020 or 2019

In [433…

```
#  What is the average and max duration of the tv_shows per genre ?
show_agg = shows.groupby("listed_in")["duration_num"].agg([np.mean, np.max])
show_agg
```

Out[433…

listed_in	mean	max
Anime Series	1.532209	9.0
British TV Shows	2.247788	10.0
Classic & Cult TV	6.055147	15.0
Crime TV Shows	1.937460	15.0
Docuseries	1.478107	9.0
International TV Shows	1.510938	12.0
Kids' TV	2.126532	10.0
Korean TV Shows	1.235294	6.0
Reality TV	1.968707	9.0
Romantic TV Shows	1.482781	17.0
Science & Nature TV	1.331210	9.0
Spanish-Language TV Shows	1.708843	7.0
Stand-Up Comedy & Talk Shows	3.552239	13.0
TV Action & Adventure	2.579545	15.0
TV Comedies	2.298610	13.0
TV Dramas	2.002237	17.0
TV Horror	2.321998	15.0
TV Mysteries	2.427791	15.0
TV Sci-Fi & Fantasy	2.740670	13.0
TV Shows	1.000000	1.0
TV Thrillers	2.260417	9.0
Teen TV Shows	2.518868	7.0

Observation:

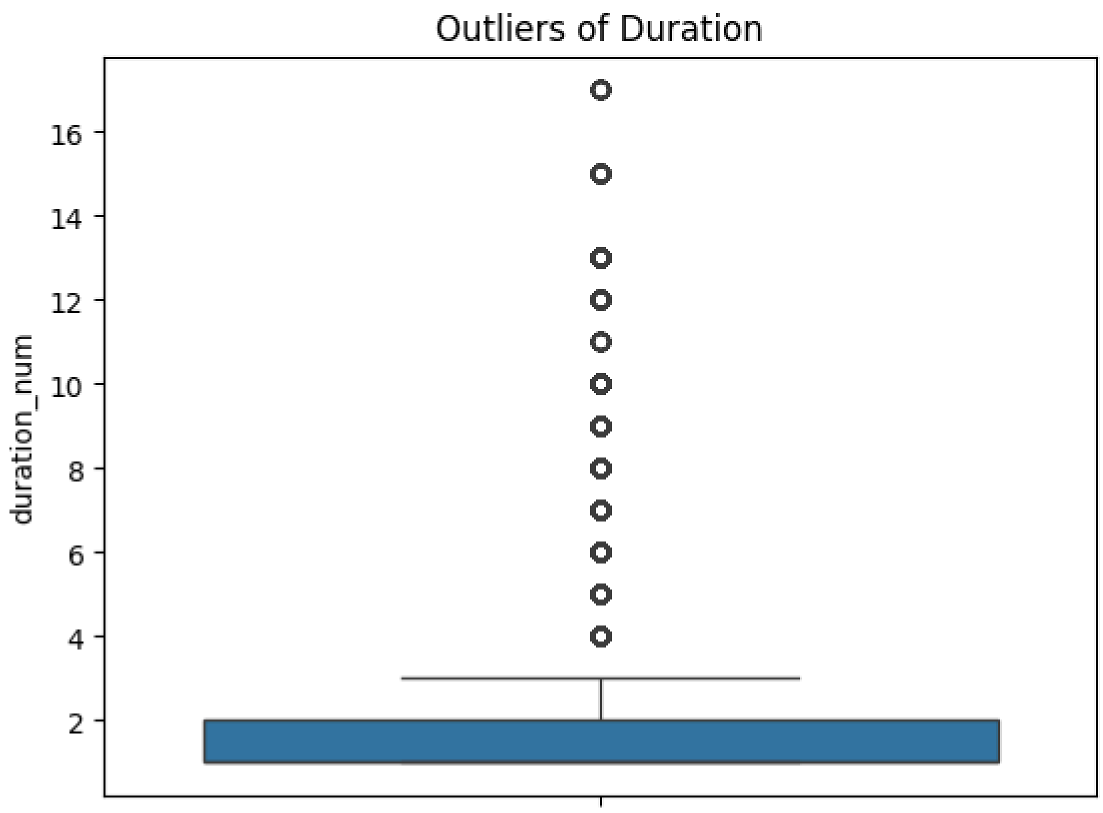
The mean and max of durations for shows for each genre was listed down. Romantic TV shows and TV Dramas are genres with max number of seasons. We are avoiding to calculate the min values of duration. Since, the shows will be in seasons, we will end up getting 1 for every genre.

In [434…

```
# Detecting the outliers in Duration Column
sns.boxplot(data = shows, y= "duration_num")
plt.title("Outliers of Duration")
```

Out[434…

```
Text(0.5, 1.0, 'Outliers of Duration')
```



Observation:

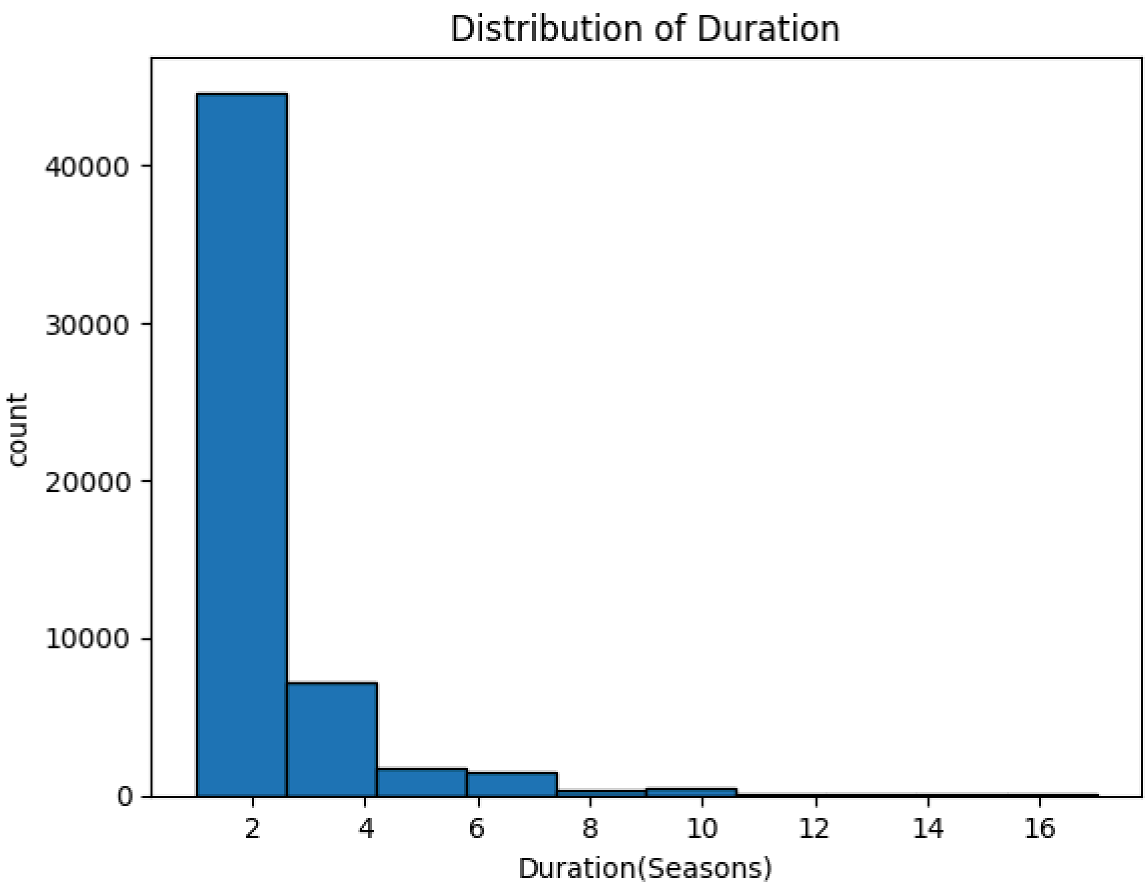
We can observe only the outliers on the positive side. most of the shows have only 1 to 2 season.

In [435…

```
# Check the distribution of the Duration column
plt.hist(shows["duration_num"], edgecolor= "black")
plt.xlabel("Duration(Seasons)")
plt.ylabel("count")
plt.title("Distribution of Duration")
```

Out[435…

```
Text(0.5, 1.0, 'Distribution of Duration')
```



Observation:

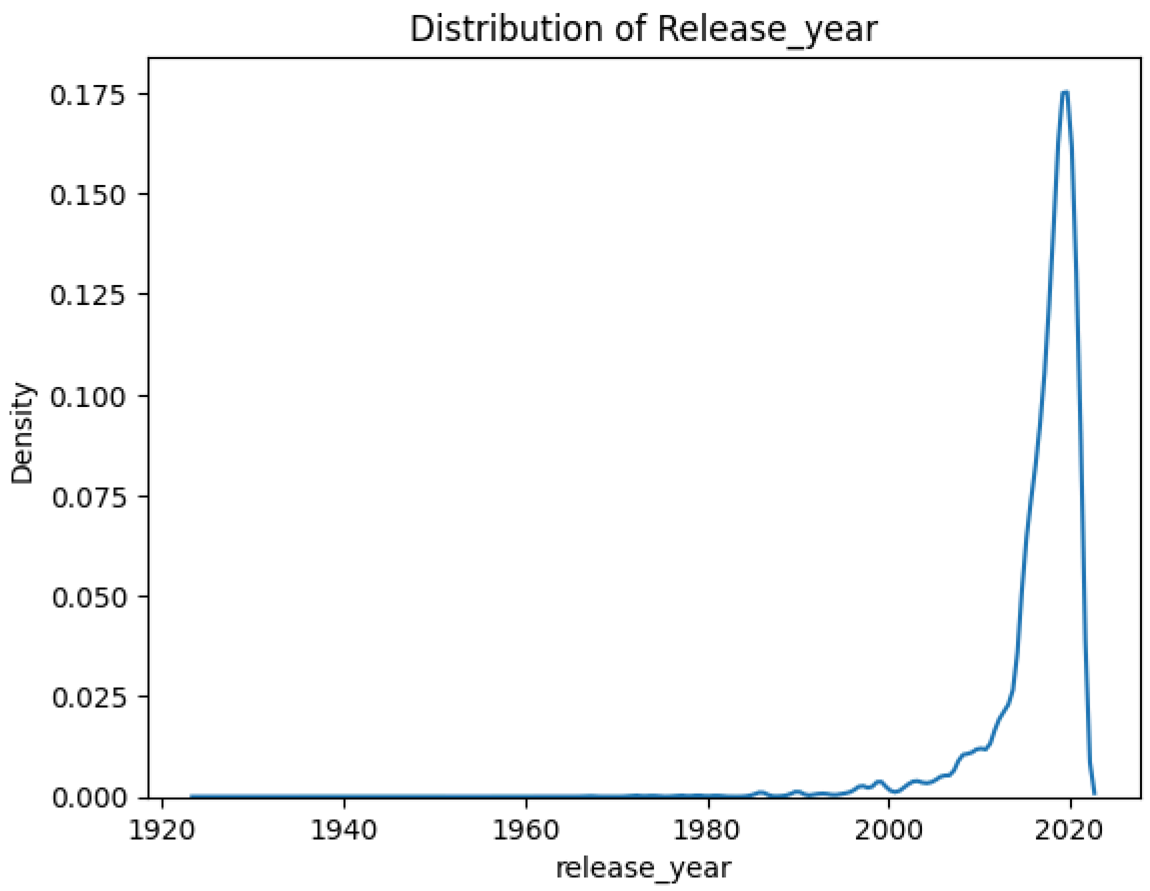
A histplot was plotted to check the distribution od duration in Shows. From the plot we can say that most of the shows have 1 or 2 seasons.

In [436…

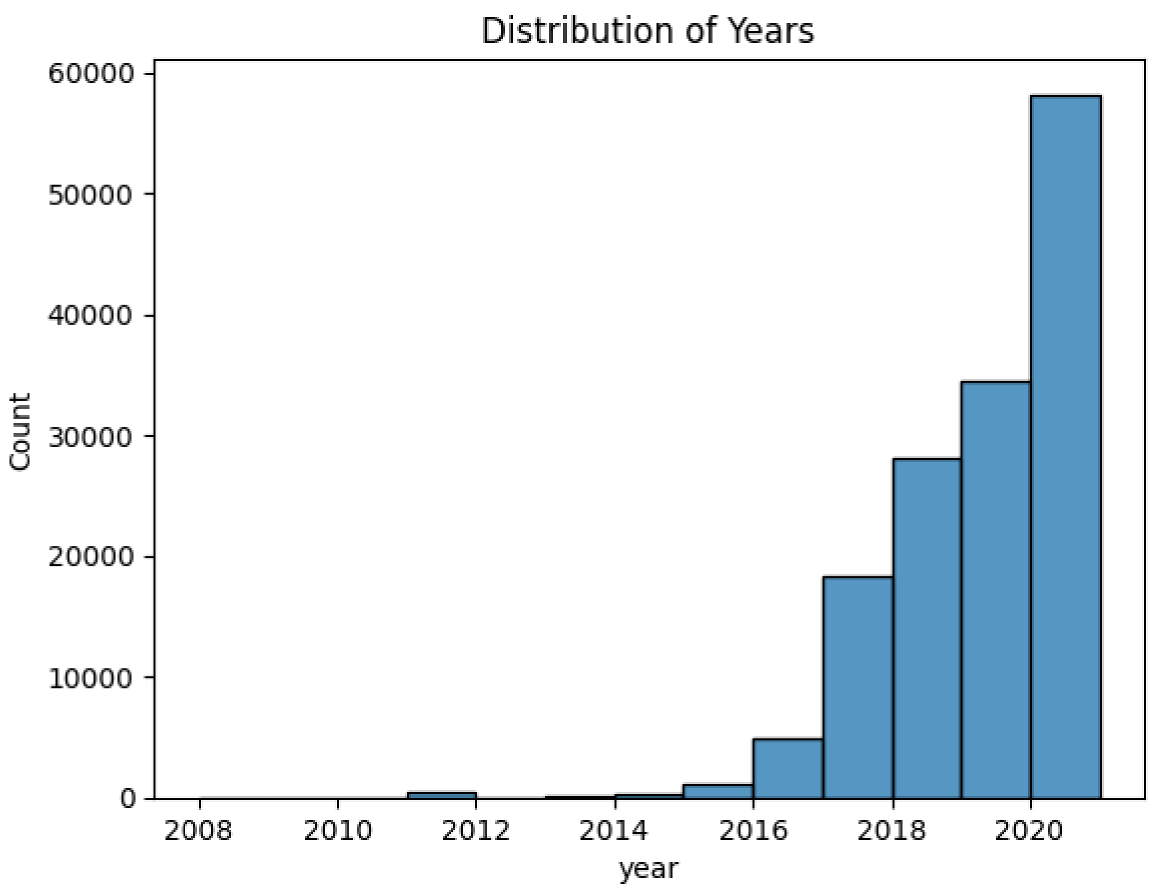
```
# Check the distribution of the Release Year column
sns.kdeplot(data = shows, x= "release_year")
plt.title("Distribution of Release_year")
```

Out[436…

```
Text(0.5, 1.0, 'Distribution of Release_year')
```



```
In [437…  # Check the distribution of the Year Added column
          sns.histplot(movies["year"], binwidth = 1)
          plt.title("Distribution of Years")

Out[437…  Text(0.5, 1.0, 'Distribution of Years')
```



Observation:

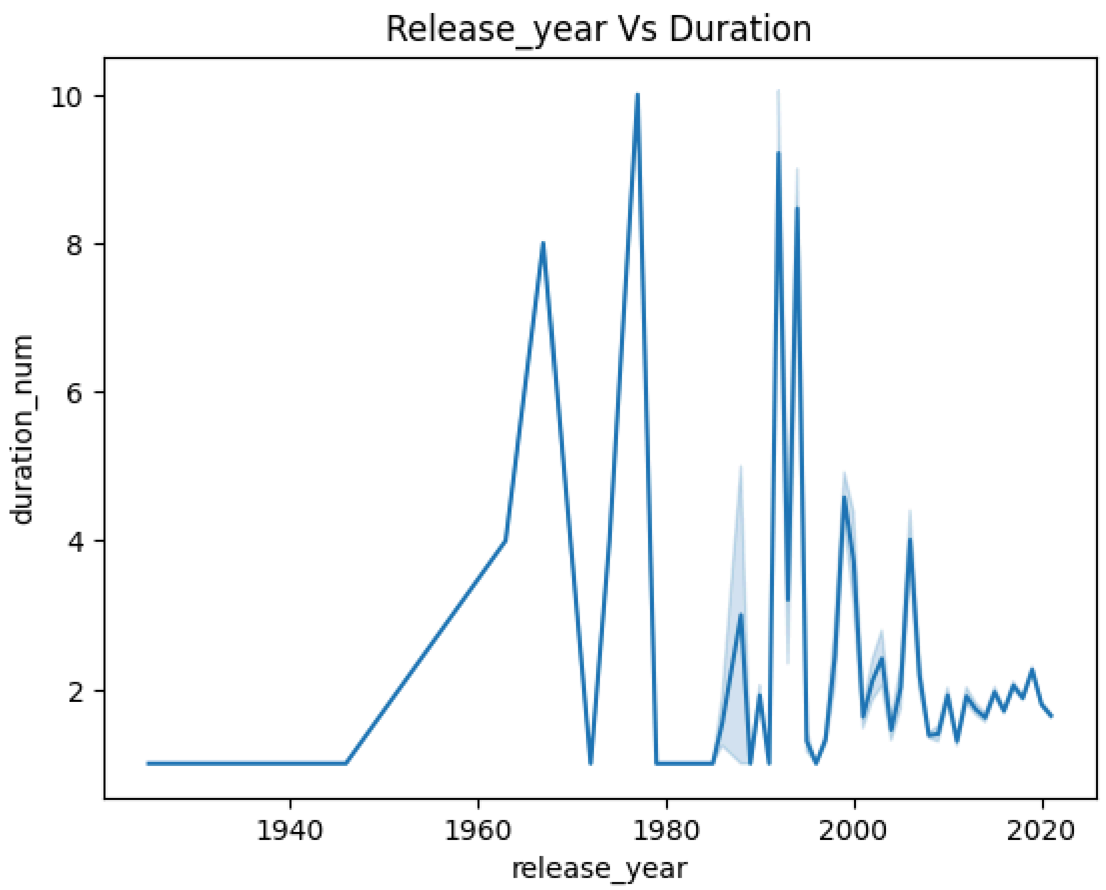
Netflix added most number of shows between 2016 and 2020. We can also observe that highest number of shows were added during 2020.

In [438…

```
#  How has the duration of movies released per year changed over time?
sns.lineplot(data = shows, x= "release_year", y = "duration_num")
plt.title("Release_year Vs Duration")
```

Out[438…

```
Text(0.5, 1.0, 'Release_year Vs Duration')
```



Observation:

A line graph is plotted using the release_year and duration_num columns. We can observe there is an increase in peak between 1960 to 1980 and also from 1990 to 2000. Later on, the duration shows a little fluctuation with years passing and attain stability somewhere around 2010.

In [439…

```
# What are the names of the directors that directed most number of movies per ge

# grouping the directors and respective count based on genre
directors_per_genre = shows.groupby(["listed_in","director"])["show_id"].nunique

# filtering out all the unknown values(missing values)
directors_per_genre = directors_per_genre[directors_per_genre["director"]!= "unk

#ranking the directors based on the count
directors_per_genre["rank"] = directors_per_genre.groupby("listed_in")["cnt"].ra

#filtering all the top directors from each genre
directors_per_genre = directors_per_genre[directors_per_genre["rank"]==1]
directors_per_genre= directors_per_genre[["listed_in", "director", "cnt"]]

directors_per_genre.reset_index(inplace = True) #resetting the index
directors_per_genre.drop("index", axis =1)
```

Out[439…

	listed_in	director	cnt
0	International TV Shows	Alastair Fothergill	3
1	Docuseries	Ken Burns	3
2	Docuseries	Alastair Fothergill	3
3	British TV Shows	Alastair Fothergill	3
4	Stand-Up Comedy & Talk Shows	Stan Lathan	2
...	...	...	...
313	Crime TV Shows	Ellena Wood	1
314	Crime TV Shows	Elías León	1
315	Crime TV Shows	Eric Goode	1
316	Crime TV Shows	Felipe Cano	1
317	Anime Series	Hayato Date	1

318 rows × 3 columns

Observation:

Directors with highest show count are listed above from each genre. We can see multiple people of the same genre are having same number of highest shows.

In [440…

```
# What are the names of the actors that acted in most number of movies per genre
actor_per_genre = shows.groupby(["listed_in","cast"])["show_id"].nunique().sort_
actor_per_genre = actor_per_genre[actor_per_genre["cast"]  != "unknown"]
actor_per_genre["rank"]= actor_per_genre.groupby("listed_in")["Count"].rank(meth
actor_per_genre = actor_per_genre[actor_per_genre["rank"]==1]
actor_per_genre= actor_per_genre.drop("rank", axis = 1)
actor_per_genre.reset_index(inplace = True) #resetting the index
actor_per_genre.drop("index", axis =1, inplace = True)
actor_per_genre.head(10)
```

```
<ipython-input-440-8f2cad59f14c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  actor_per_genre["rank"]= actor_per_genre.groupby("listed_in")["Count"].rank(met
hod = "dense", ascending = False).astype(int)
```

Out[440…

	listed_in	cast	Count
0	Anime Series	Takahiro Sakurai	24
1	International TV Shows	Takahiro Sakurai	22
2	Docuseries	David Attenborough	14
3	British TV Shows	David Attenborough	13
4	Kids' TV	Vincent Tong	13
5	TV Dramas	Tay Ping Hui	10
6	Science & Nature TV	David Attenborough	9
7	Spanish-Language TV Shows	Juan Pablo Urrego	6
8	Romantic TV Shows	Amanda Chou	6
9	Teen TV Shows	Takahiro Sakurai	5

Observation:

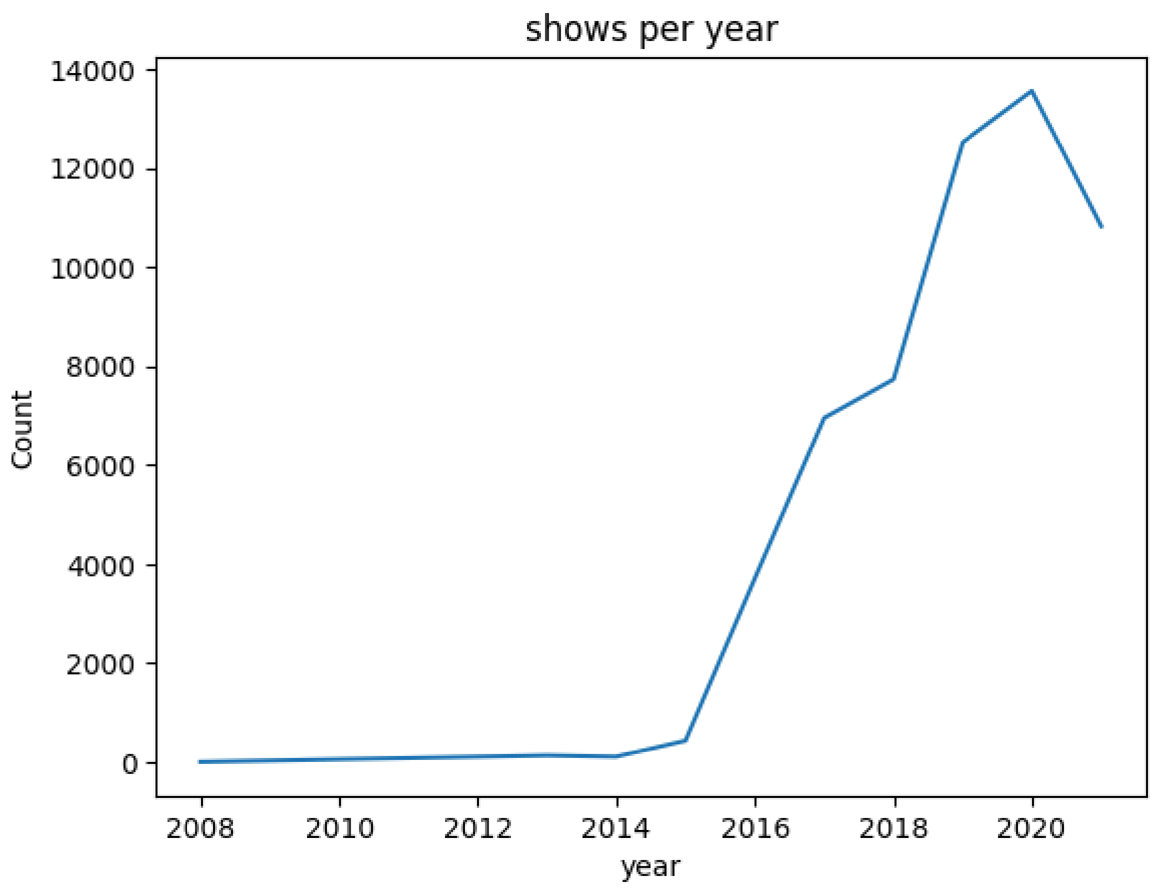
Popular actor of each genre was calculated and the Actor Takahiro Sakurai was found to be famous in genres like Anime and International TV Shows.

In [441…

```
# How many number of tv_shows were made per year?
shows_per_year = shows.groupby(["year"])["title"].count().reset_index(name = "Co
shows_per_year.drop(0, axis = 0, inplace = True)
sns.lineplot(data= shows_per_year, x= "year", y = "Count")
plt.title("shows per year")
```

Out[441…

```
Text(0.5, 1.0, 'shows per year')
```



Observation:

The number of shows made are listed as a table. Form the above table, we estimate that 2020 has the highest number of shows with a count of around 13500. A line plot was plotted for the same. We can see the line has taken upward direction from 2016 and growing rapidly till, which shows that the highest number of shows were made between 2016 and 2020.

In [442…

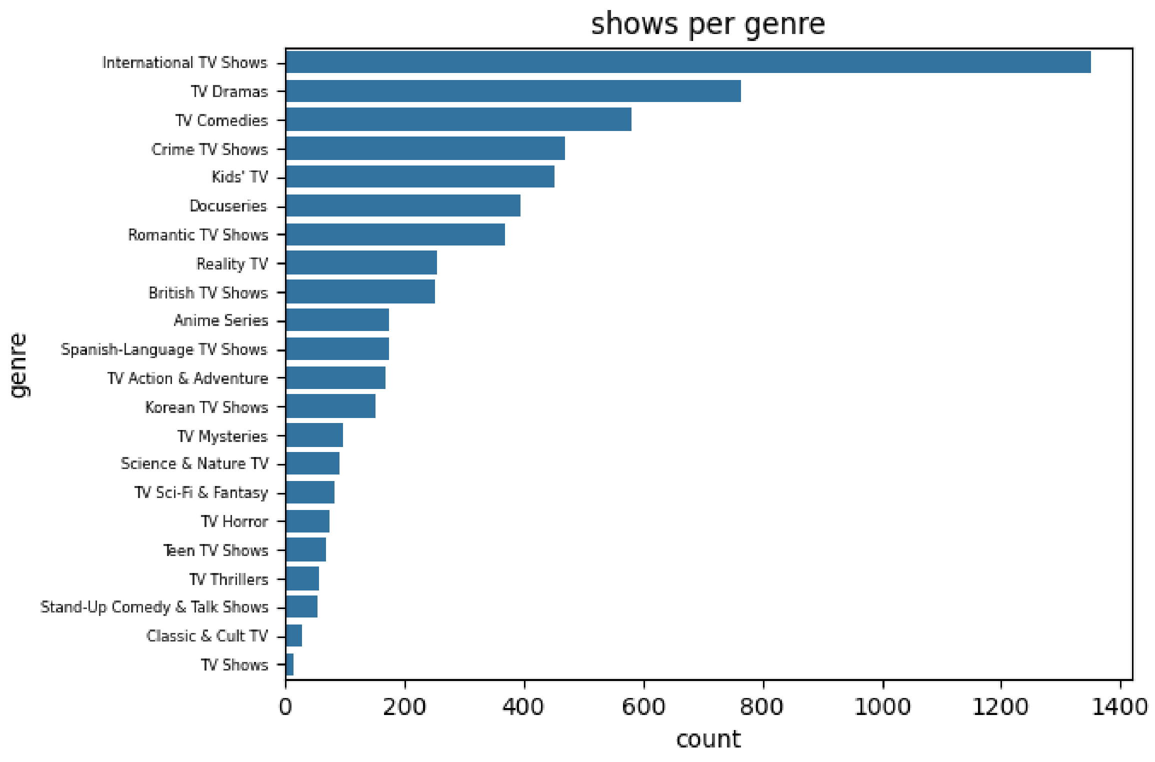
```
# Q: How many number of tv_shows are available in netflix per genre?
shows_per_genre = shows.groupby(["listed_in"])["title"].nunique().sort_values(as
shows_per_genre
```

Out[442…

	listed_in	count
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 [443…

```
sns.barplot(data = shows_per_genre, y= "listed_in", x= "count")
plt.title("shows per genre")
plt.ylabel("genre")
plt.xlabel("count")
plt.yticks(fontsize = 6)
plt.show()
```



Observation:

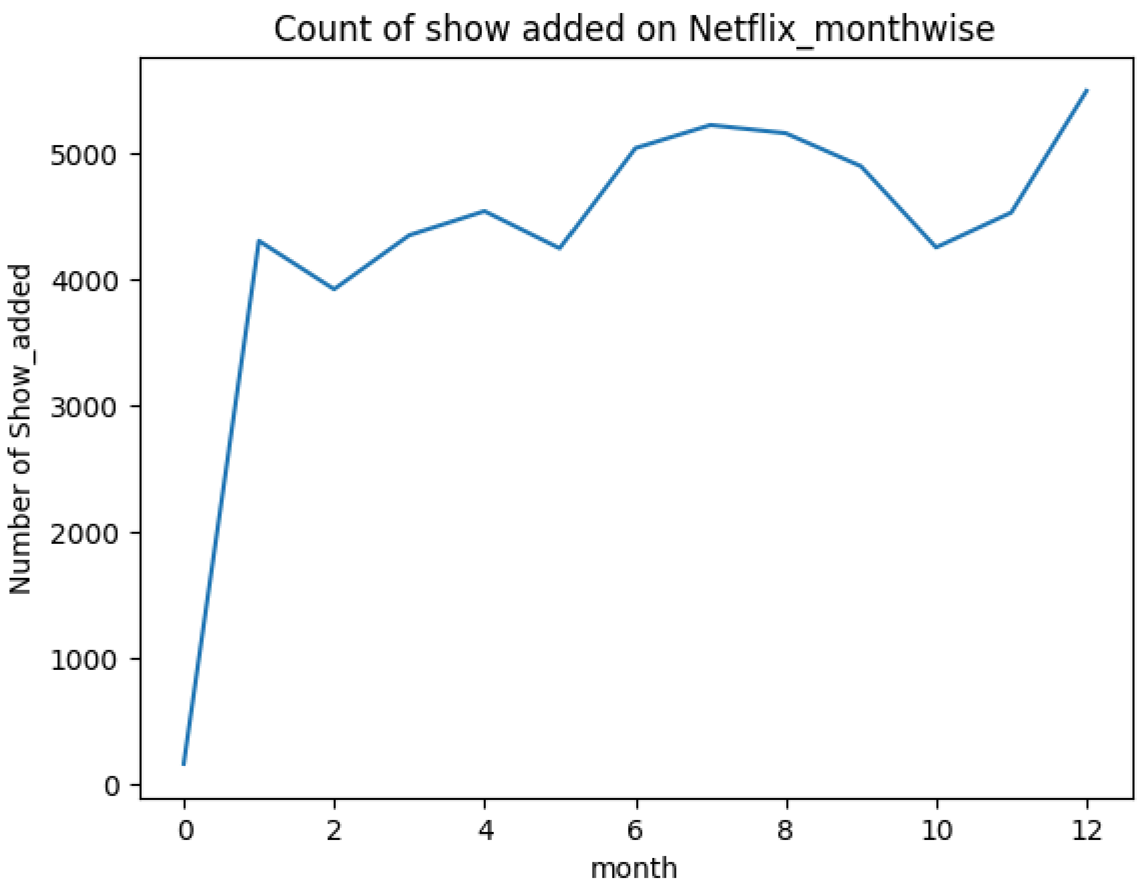
Popular genres of TV Shows are listed above. International TV shows tops the list with 1351 followed by Dramas, comedies, crime Tv shows and Kids TV. A barplot representing the same has been plotted.

In [444…

```
#months with Highest shows release
shows_per_month = shows.groupby(["month"])["title"].count()
shows_per_month = shows_per_month.sort_values(ascending = False).reset_index(nam
sns.lineplot(data=shows_per_month, x="month", y="count")
plt.xlabel("month")
plt.ylabel("Number of Show_added")
plt.title("Count of show added on Netflix_monthwise")
```

Out[444…

```
Text(0.5, 1.0, 'Count of show added on Netflix_monthwise')
```

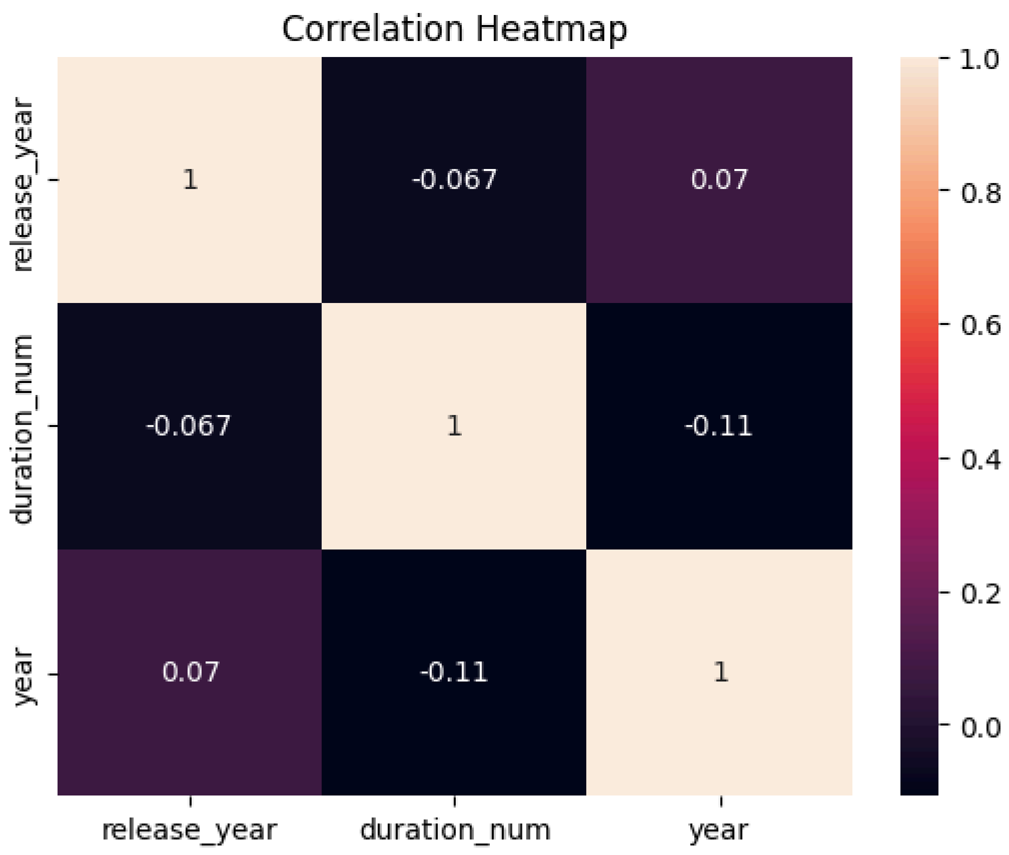


Observation:

A line plot was plotted between the months and number of shows added. Most number number of shows was found to be released during the month of December and July. We can consider that these are the best months to release the shows.

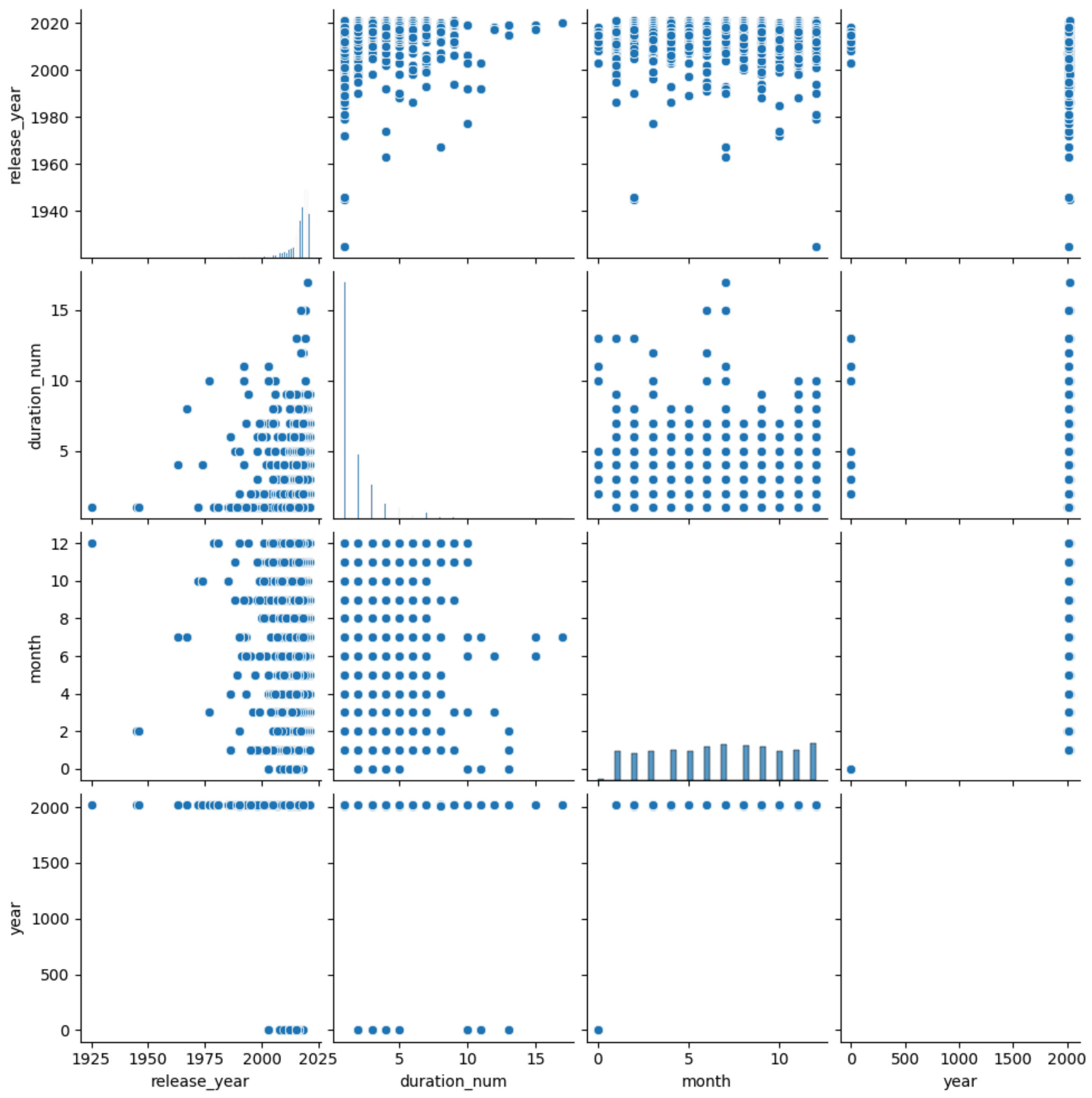
```
In [445…
#Check for correlations among numerical features
sns.heatmap(shows[['release_year','duration_num','year']].corr(), annot =True)
plt.title('Correlation Heatmap')
```

```
Out[445…
Text(0.5, 1.0, 'Correlation Heatmap')
```



```
In [446…  sns.pairplot(data = shows)
```

```
Out[446…  <seaborn.axisgrid.PairGrid at 0x7e25b6252290>
```



Observation:

1. Release year and year_added columns have moderate positive correlation, since

majority of the tv_shows were originally released as well as added to netflix in recent years. 2. Duration and year_added columns have weak positive correlation, since many tv_shows in recent years are increasing their overall duration. 3. Release year and duration columns have weak negative correlation.

Business Insights:

1. Netflix have a huge market in United states almost in every genre they make. There is a high chance of producing a greater number of movies and shows in United States.
2. In countries like Japan, most of the audience watch anime genre. Investing on this particular genre more in Japan would gain profits.

3. Actors like Anupam Kher, Sharukh Khan, Julie Tejwani and Naseeruddin Shah are highly popular with highest number of movies made. Making movies with these actors will gain profits.
4. Actor Takahiro Sakurai had made highest number of shows in Anime genre. Netflix can cast him in making more Anime with him.
5. Directors like Rajiv Chilaka, Jan Suter, Rahul Campos are the top directors with highest number of movies or shows directed. Rajiv Chilaka is the top director in the Children & Family movies. So, Netflix can produce more movies with this director especially in this genre.
6. Most of the content available is for the adult audience (TV-MA). Another large portion is the TV-14 classification, that is, programs that may contain material considered inappropriate for children under 14 years of age because they may contain moderate violence and offensive language. We can say with this that the massive audience of Netflix is made up of the adult audience.
7. United States tops in both shows and TV Shows release count. There is a huge gap between the US and other countires in terms of number of shows released. Netflix Should increase there global market by making the number of releases in other countries.

Recommendations:

1. Basing the distribution of duration, we can estimate that a movie with duration of 90 to 140 mins (which is 1.5 to 2.5 hrs of time) and for TV Shows 1 or 2 Seasons are ideal and has a great chance of making audience to be engaged in watching the movie.
2. Shows have 1 or 2 seasons as ideal duration, since the hit or flop status the of the show would decide the fate of its sequel.
3. Although Netflix platform is started from 2008, but the frequency of adding content to the platform has increased only after 2016 and shown positive growth till 2020. However, there is slow decrease in the number of movies or shows added to the platform from 2021. This is may be due to pandemic situations after 2020. This shows an impact on the business profits.
4. The number of shows released are very high the month of December. Since it is a Christmas month people would get free time to binge watching. This indicates that launching a show during this particular month would gain more audience globally.
5. Least number of shows are produced or released during February. Since February has a high chance of having two major award shows like The Golden Globe and The Oscars, usually there are a smaller number of releases during this month. If the show belongs to the romantic genre Netflix can opt to release the around the valentine week.
6. Incase Movies, July is considered as the best time to release a movie. Since July falls under summer in many parts of the world, it is the best season to score summer blockbuster during July.

7. Content with TV- MV and TV-14 are the two categories of rating with the highest number of movies or shows. One making the movies or shows that fall under these categories would help in reaching the wide range of audience.
8. TV Shows with less seasons and movies with 90-140 minutes duration on 'Dramas' & 'Comedies' Genre is preferable.
9. Netflix can make more number of movies and shows with the most popular genre of each country. This way,Netflix can expand the global market by increasing the number of shows and movies release in other countries as well.