NETFLIX - Exploratory Data Analysis

PROBLEM STATEMENT

Neflix is one of the lasrgest OTT platform and it is operating in many countries. And New contents are addded day by day. And in the Exploratory data analaysis , I have analysed the dataset , and the Data analysis , will help to lear, how netflix is successful in content release and types of contents, mostly released , successful genre content , Successful pair of Actor and director movie released monstly. When Analysing the data , we can identify , get to know the data in past, present , and helps to predict the future. As Netflix is already running its business successfully. We can predict, how the business is successful in past years and help to run the business successful and predictable.

Following data analysis have been made in this notebook

1. Total Content
2. Total movie and TV Shows
3. Genre wise movie and TV shows released
4. Top 20 countries contributed to Netflix
5. Movies Vs TV shows released in last 5 years- Rating wise
6. Movies Vs TV shows released in last 5 years- Genre wise
7. Most content released in month wise
8. Most casted Actor in Netflix content
9. Most director contributed to the netflix content
10. Successful Actor-Director combo in Netlfix

Importing libraries

In [141…

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
import ipywidgets as w
from IPython.display import display
```

Original dataset importing

In [141…

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

In [141…

```
#duplicate data
#df=pd.read_csv("netflix_processed6.csv")
```

Getting dataset info

As we see, there are total columns and further we need to identify the null and duplicate values present in dataset

In [141…

```
#Getting info of dataset
df_o.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
```

Getting numerical computation in dataset

The Current dataset, has categorical data mostly and only few numerical datas available

In [142…

```
#Getting math of the Dataset
df_o.describe()
```

Out[1420]:

	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

Displaying the first five rows in dataset

Displaying the first five rows to analyse and identify, the data and datatype in different columns, As we see in current dataset, there ara nested datas present in cast, director,country, and listed_in columns, and there are many null values present. We need to breakdown each point to anlayse the data.

In [142…

```
#display data
df_o.head(5)
```

Out[1421]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration

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

To Check the null values present in different columns

As we see, there are null values present in director, country, cast, country, date_added,rating,duration.

In [142…

```
df_o.isna().sum()
```

Out[1422]:

```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

Filling missing value

As we see, there are'nt any rows, where more than 40% data is missing, so dropping the row is not feasible in this case

- For country column, mode value is used for filling the missing data
- For director and cast column , Anonymous value is used for filling the director and cast column
- For Rating , we input value as "Rating not available" for null values

- For duration , we input "0" for duration column

In [142…

```
df_o['country'] = df_o['country'].fillna(df_o['country'].mode()[0])
df_o['director'] = df_o['director'].fillna(value="Anonymous")
df_o['cast'] = df_o['cast'].fillna(value="Anonymous")
df_o['date_added'] = df_o['date_added'].fillna(df_o['date_added'].mode()[0])
df_o['rating'] = df_o['rating'].fillna(value="Rating not available")
df_o['duration'] = df_o['duration'].fillna(value="0")
```

Splitting duration column

We split the duration column, for splitting string value

In [142…

```
#Splitting duration and adding only numbers in new column

df_o['new_duration']=df_o['duration'].str.split(' ').str[0]
```

Casting nested datas to un-nested data's

As there are nested date's in cast,country,genre,director, we need to unnest and merge to single dataframe.

In [142…

```
constraint=df_o['director'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_o['title'])
df_new=df_new.stack()
df_director=pd.DataFrame(df_new)
df_director.reset_index(inplace=True)
df_director=df_director[['title',0]]
df_director.rename(columns={0:'director'})

constraint=df_o['cast'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_o['title'])
df_new=df_new.stack()
df_cast=pd.DataFrame(df_new)
df_cast.reset_index(inplace=True)
df_cast=df_cast[['title',0]]
df_cast.rename(columns={0:'cast'})


constraint=df_o['country'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_o['title'])
df_new=df_new.stack()
df_country=pd.DataFrame(df_new)
df_country.reset_index(inplace=True)
df_country=df_country[['title',0]]
df_country.rename(columns={0:'country'})


constraint=df_o['listed_in'].apply(lambda x: str(x).split(', ')).tolist()
df_new=pd.DataFrame(constraint,index=df_o['title'])
df_new=df_new.stack()
df_listedin=pd.DataFrame(df_new)
df_listedin.reset_index(inplace=True)
df_listedin=df_listedin[['title',0]]
df_listedin.rename(columns={0:'genre'})
```

Out[1425]:

	title	genre
0	Dick Johnson Is Dead	Documentaries
1	Blood & Water	International TV Shows

2	Blood & Water	TV Dramas
3	Blood & Water	TV Mysteries
4	Ganglands	Crime TV Shows
...	...	...
19318	Zoom	Children & Family Movies
19319	Zoom	Comedies
19320	Zubaan	Dramas
19321	Zubaan	International Movies
19322	Zubaan	Music & Musicals

19323 rows × 2 columns

In [142…

```
#Merging multiple dataframes to single
x= df_director.merge(df_cast,left_on="title",right_on="title",how="left")
x=x.rename(columns={'0_x':'director','0_y':'cast'})

y= x.merge(df_country,left_on="title",right_on="title",how="left")
y=y.rename(columns={0:'country'})

z= y.merge(df_listedin,left_on="title",right_on="title",how="left")
z=z.rename(columns={0:'genre'})

#Dropping columns in original dataset
df_o=df_o.drop(['director', 'cast','country','listed_in'], axis=1)

#Merge columns to Single dataset column

df= z.merge(df_o,left_on="title",right_on="title",how="left")
```

In [ ]:

In [142…

```
#Date_added ,changing the type to date
#Splitting the duration
df['date'] = pd.to_datetime(df['date_added'])
df['date_added'] = pd.to_datetime(df['date_added'])
df['year'] = df['date'].apply(lambda datetime: datetime.year)
df['month'] = df['date'].apply(lambda datetime: datetime.month)
```

In [142…

```
##------------------------------------------------------------------------------------------
```

Incorrect data handling and replacing values for ratings

We replace the incorrct values in rating , and further renamainig values in Rating column for better understanding. We took the information from the Netflix website

In [142…

```
df['rating'] = df['rating'].replace({'74 min': 'TV-MA', '84 min': 'TV-MA', '66 min': 'TV
df['rating'] = df['rating'].replace({'TV-Y7-FV': 'TV-Y7'})

# Replacing valus in rating for better understanding purpose
df['rating'] = df['rating'].replace({
                'PG-13': 'Teens - Age above 12',
                'TV-MA': 'Adults',
                'PG': 'Kids - with parental guidence',
                'TV-14': 'Teens - Age above 14',
                'TV-PG': 'Kids - with parental guidence',
                'TV-Y': 'Kids',
                'TV-Y7': 'Kids - Age above 7',
                'R': 'Adults',
                'TV-G': 'Kids',
                 'G': 'Kids',
                'NC-17': 'Adults',
                'NR': 'NR',
                'UR' : 'UR'
})
```

Non- Graphical Analysis

1) Total Content available in Netflix

In [143…

```
df['title'].drop_duplicates(keep='last').value_counts().value_counts()[1]
```

Out[1430]:

```
8807
```

2) Total Content released in Summer holidays

To predict ,whether summer holidays is the best time to release movie. In this prediction , summer month is assumed as May.

In [143…

```
mdm=df[df['type']=='Movie'][['title','month']]
mdm=mdm.drop_duplicates(keep='last')
mdt=df[df['type']=='TV Show'][['title','month']]
mdt=mdt.drop_duplicates(keep='last')
mdm=mdm[mdm['month']==5].value_counts().value_counts()[1]
mdt=mdt[mdt['month']==5].value_counts().value_counts()[1]
```

Total movies released in May month

In [143…

```
mdm
```

Out[1432]:

```
439
```

Total Tv shows released in May month

In [143…

```
mdt
```

Out[1433]:

```
193
```

DATA VISUALIZATION

In [143…

```
#importing Seaborn library
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
```

In [143…

```
# For Exporting graphs while downloading as PDF
import plotly.io as pio
pio.renderers.default = "notebook+pdf"  # Renderer for Notebook and HTML exports + Rende
```

```
import plotly.offline as pyo
pyo.init_notebook_mode()
```

In [ ]:

```

```

In [143…

```
##------------------------------------------------------------------------------------
```

In [143…

```
#Category wise content
md=df[df['type']=='Movie']['title']
md=md.drop_duplicates(keep='last').value_counts()
td=df[df['type']=='TV Show']['title']
td=td.drop_duplicates(keep='last').value_counts()
```

In [143…

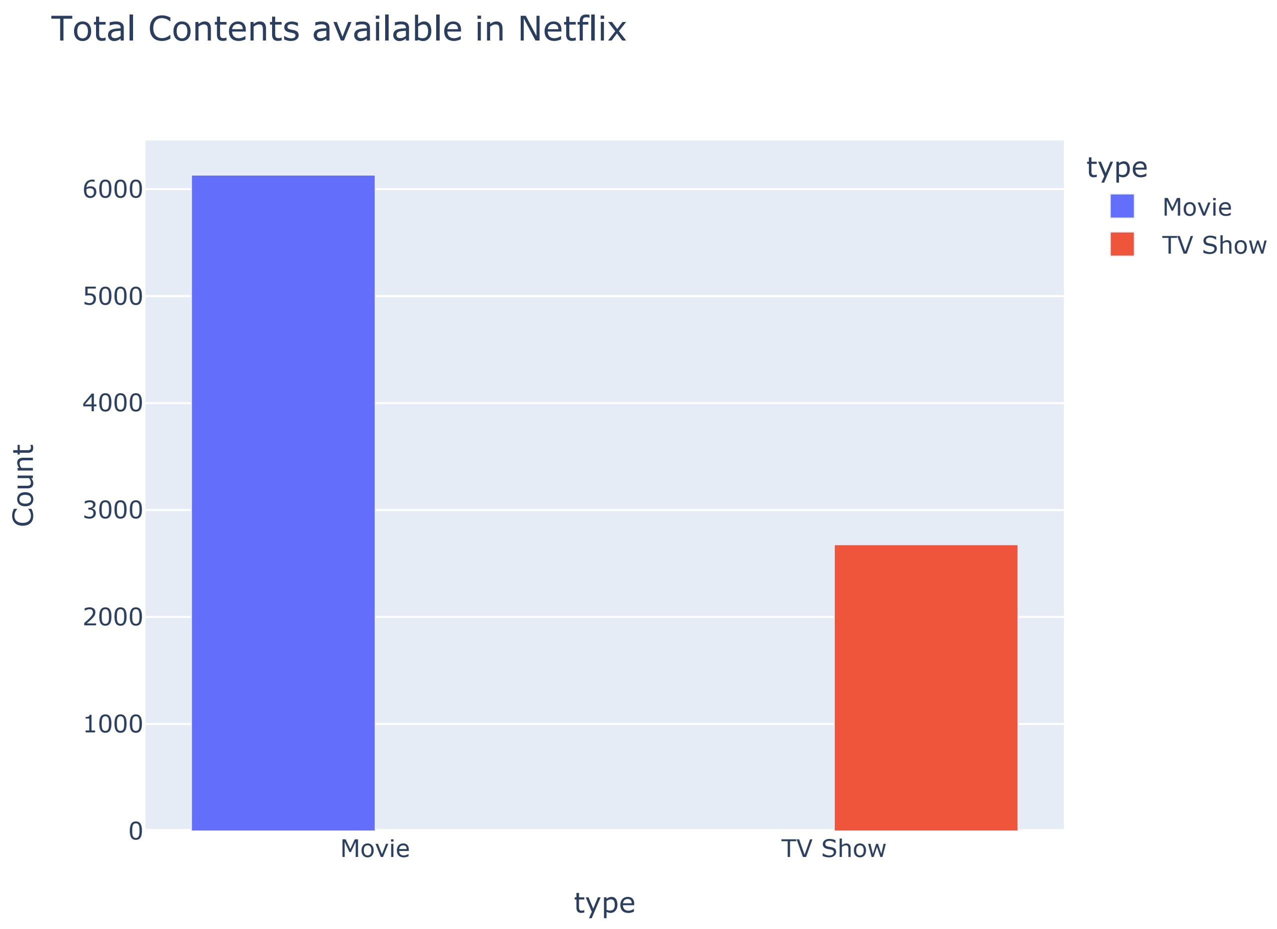
```
data_dict1 = {'Count':[md.value_counts()[1], td.value_counts()[1]], 'type': ['Movie','TV
```

In [143…

```
df_b = pd.DataFrame(data=data_dict1, columns=['Count','type'])
```

In [144…

```
px.bar(data_frame=df_b, x="type", y="Count", color="type", barmode="group",title="Total
```



In [ ]:

```

```

In [144…

```
##------------------------------------------------------------------------------------
```

In [ ]:

```

```

Top Countries contributing to Netflix

In [144…

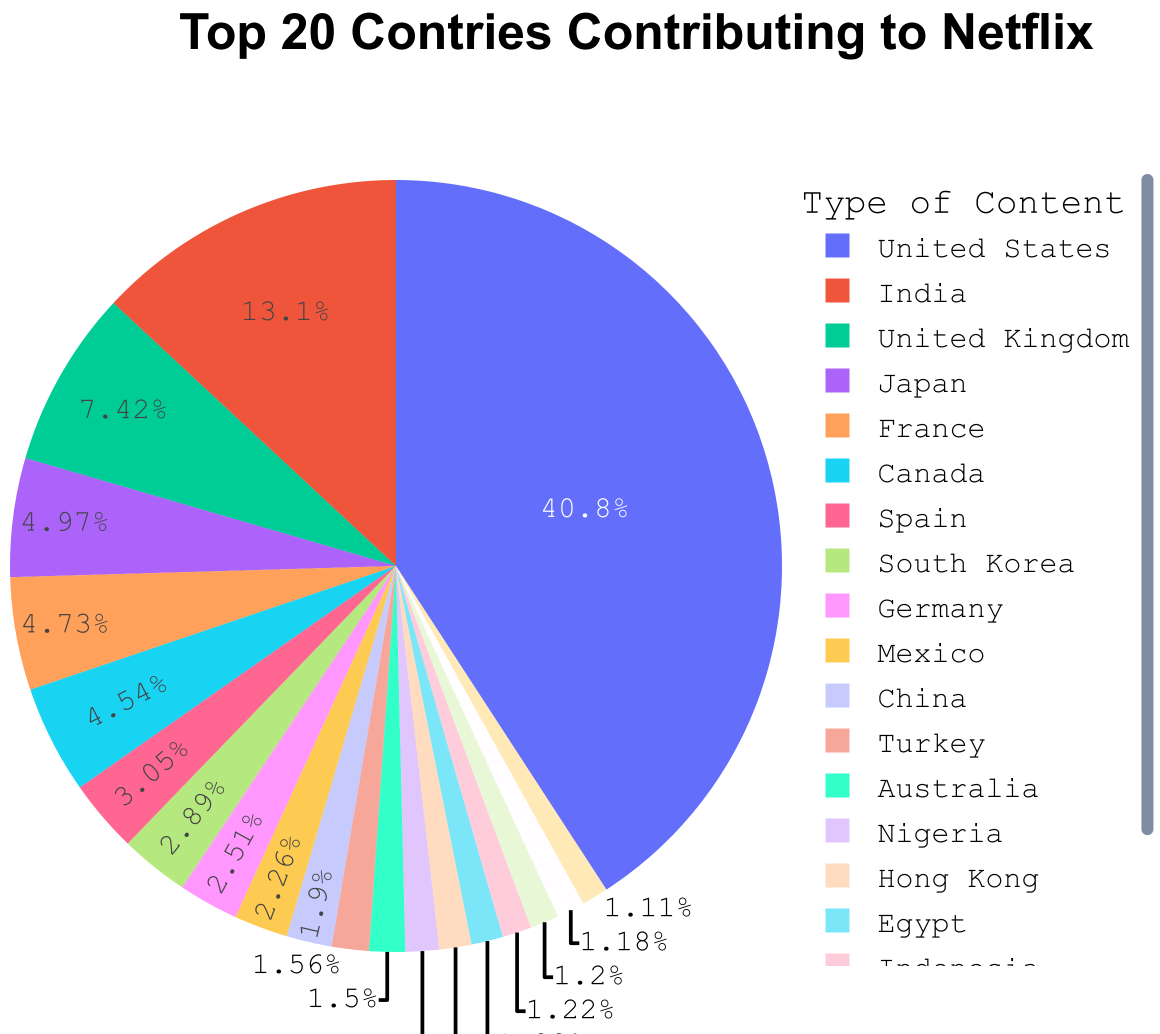
```
data_dict1 = {'country': df.groupby('country').size().sort_values(ascending=False)[:20].
              'Number of content': df.groupby('country').size().sort_values(ascending=Fal
              }
```

In [144…

```
df_P = pd.DataFrame(data=data_dict1, columns=['country', 'Number of content'])
```

In [144…

```
fig = px.pie(df_P, values='Number of content', names='country',title="Top 20 Contries Co
fig.update_layout(xaxis_title="Year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
                  height=600,
    title=dict(
        text='<b>Top 20 Contries Contributing to Netflix</b>',
        x=0.25,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=15,
        color='#000000'
    )
)
fig.show()
```



```
    │ └1.33%
    └1.35%
└1.4%
```

In [144…
```
##----------------------------------------------------------------------------------------
```

In [144…
```
df7
```

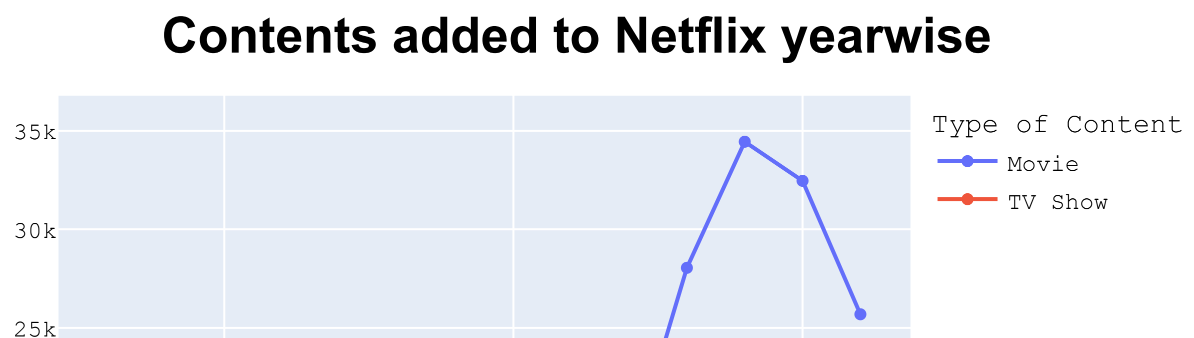
Out[1446]:

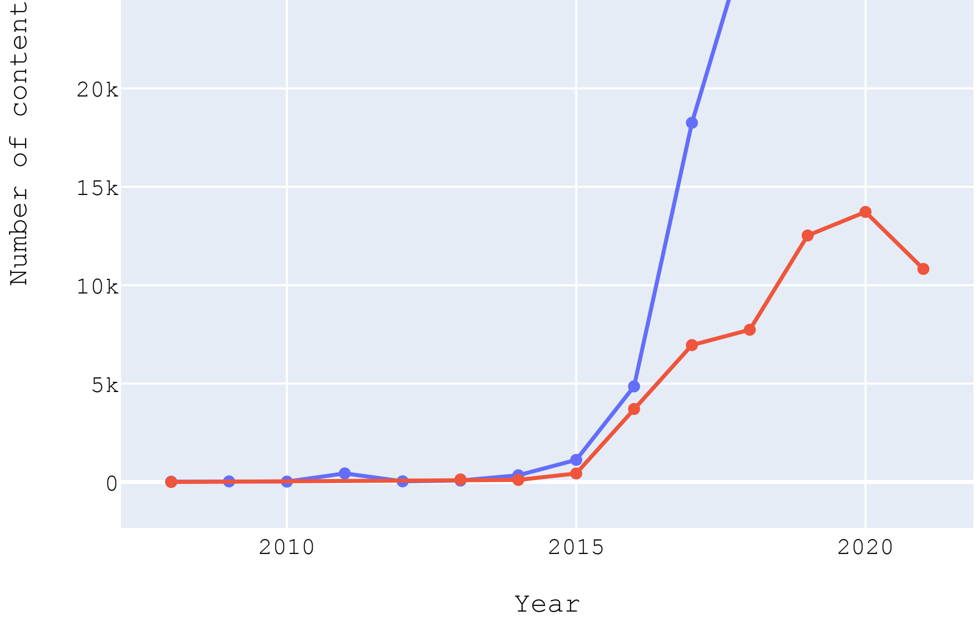
	rating	type	counts
0	Adults	Movie	5506
1	Kids	Movie	100
2	Kids - Age above 7	Movie	127
3	Kids - with parental guidence	Movie	3268
4	NR	Movie	132
5	Teens - Age above 12	Movie	233
6	Teens - Age above 14	Movie	12045

In [144…
```
##----------------------------------------------------------------------------------------
```

In [144…
```
#Yearwise Content added to netflix
```

In [144…
```
type_of_contents=df.groupby('type').size().index.tolist()
df6=df.loc[df['type'].isin(type_of_contents)]
df_6_upd=df6.groupby('year')['type'].value_counts().reset_index(name='counts')
fig = px.line(df_6_upd, x="year", y="counts", color='type',
             markers=True)
fig.update_layout(xaxis_title="Year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Contents added to Netflix yearwise</b>',
        x=0.20,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)
```





In [145…

```
##-----------------------------------------------------------------------------------------
```

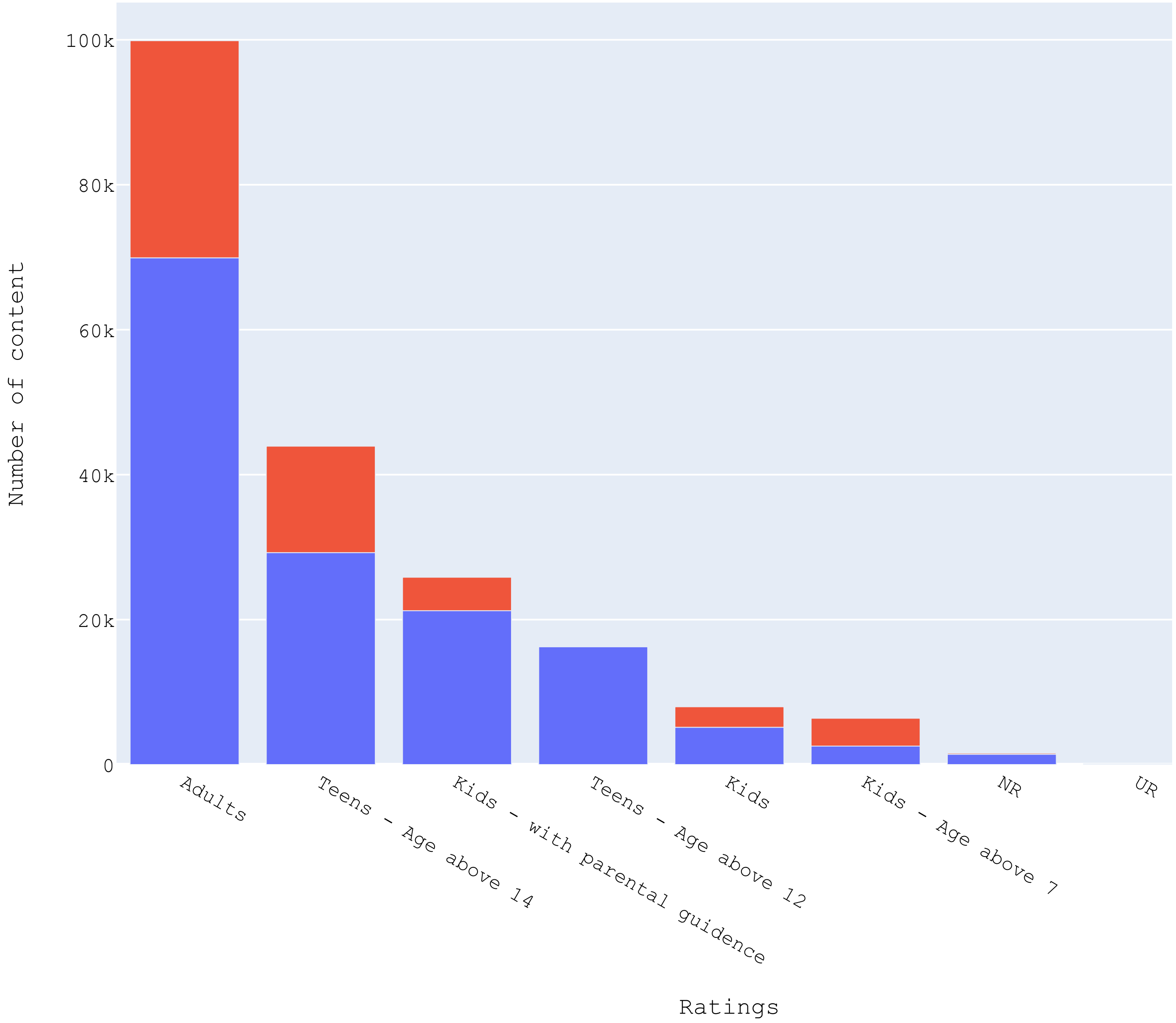
In [145…

```
#contents added in Category wise and  Rating wise
```

In [145…

```
type_of_rating=df.groupby('rating').size().index.tolist()
df7=df.loc[df['rating'].isin(type_of_rating)]
df7=df7.groupby(['type'])['rating'].value_counts().reset_index(name='counts')
data_dict1 = {'Ratings': df7['rating'],
             'Number of content': df7['counts'],'type': df7['type']
             }
df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Number of content','type'])
fig = px.bar(df_R, x="Ratings",
                   y="Number of content",
                   title="Rating wise and Category (Movie / TV Shows) wise content added
                   color='type')
fig.update_layout(autosize=False, width=950, height=700, xaxis_title="Ratings",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Rating wise and Category (Movie / TV Shows) wise content added in Netfl
        x=0.10,
        y=0.94,
        font=dict(
            family="Arial",
            size=20,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)


fig.show()
```

Rating wise and Category (Movie / TV Shows) wise content add

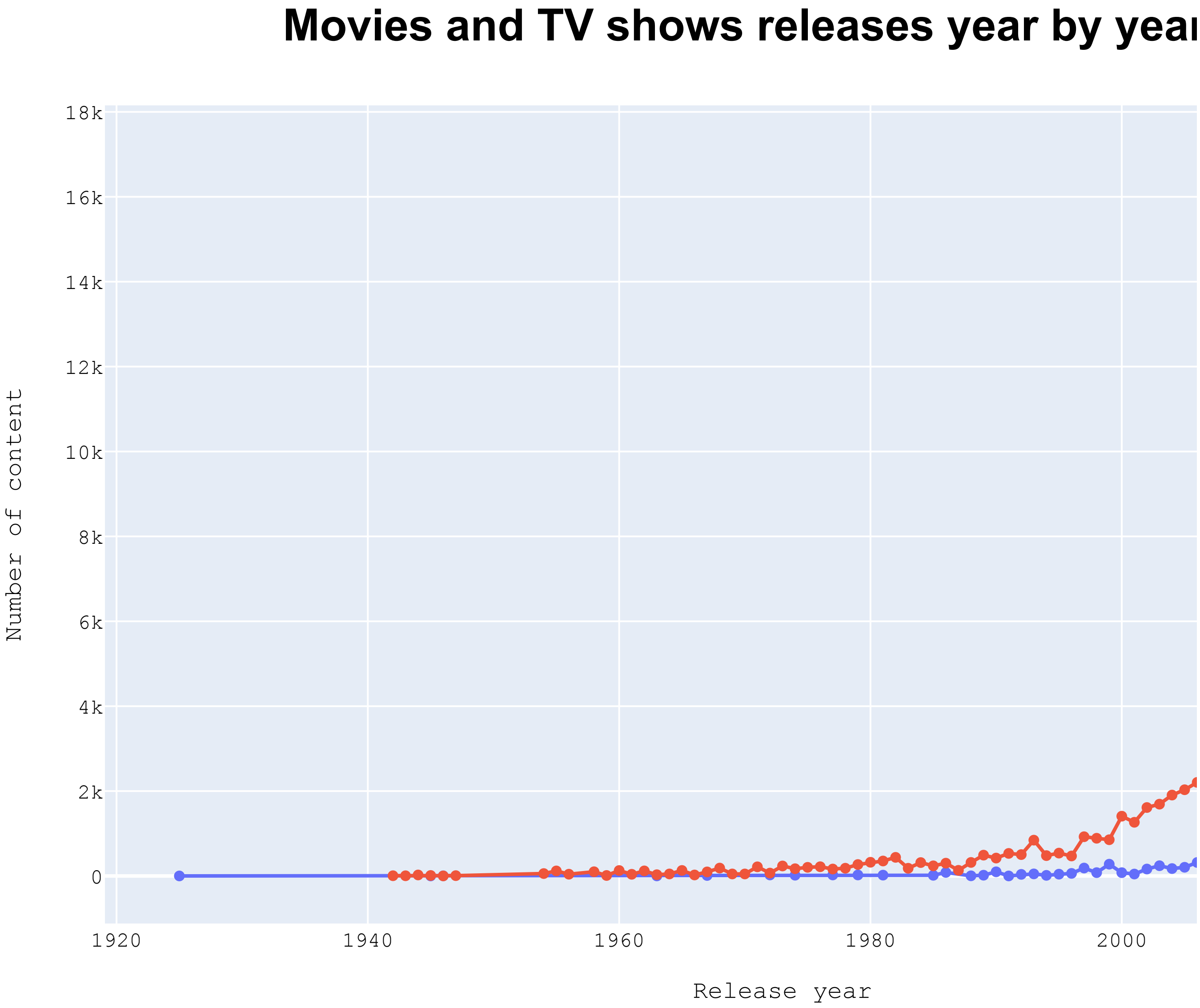


In [145…

```
# Movies and TV shows Releases year by year
```

In [145…

```
type_of_contents=df.groupby('type').size().index.tolist()
df6=df.loc[df['type'].isin(type_of_contents)]
df_6_upd=df6.groupby('release_year')['type'].value_counts().reset_index(name='counts')
fig = px.line(df_6_upd, x="release_year", y="counts", color='type',
              title='',
             markers=True)
fig.update_layout(xaxis_title="Release year",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    height=600,
    width=1000,
    title=dict(
        text='<b>Movies and TV shows releases year by year</b>',
        x=0.18,
        y=0.99,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=12,
        color='#000000'
    )
)
```



As we see in the above graph, the following points can be infered:

- Movies released more than TV Shows
- TV shows and movies contents are released more after 2015
- And during pandemic period- 2021, the movie and tv shows are released less and the curve dropping depicts the same

Total Watch hours content available

We need to identify , how much watch hour content available in Movies and TV shows, In movies, we can wasily find that. But in case of TV shows , we can only graph the seasons availeble, since we dont have duration of episodes in TV shows, We can't able to preedict that.

In [145…

```
fd=df[df['type']=='Movie'][['title','new_duration']]
fd.duplicated().sum()
fd.loc[fd.duplicated(), :]
```

```
fd=fd.drop_duplicates(keep='last')

#changing the datatype
fd=fd.astype({'new_duration': 'int32'})

moviehrs=fd['new_duration'].sum()
```

Total seasons released in TV Shows category

In [145…

```
od=df[df['type']=='TV Show'][['title','new_duration']]
od.duplicated().sum()
od.loc[od.duplicated(), :]
od=od.drop_duplicates(keep='last')

#changing the datatype
od=od.astype({'new_duration': 'int32'})

tvseasons=od['new_duration'].sum()
```

In [145…

```
##---------------------------------------------------------------------------------------
```

Pie Chart to display content available for TV shows an Movies

In [ ]:

In [145…

```
dic_hrs={'type':['Movies','Seasons'],'Total Count':[moviehrs,tvseasons]}
df_pie = pd.DataFrame(data=dic_hrs, columns=['type', 'Total Count'])
```

In [145…

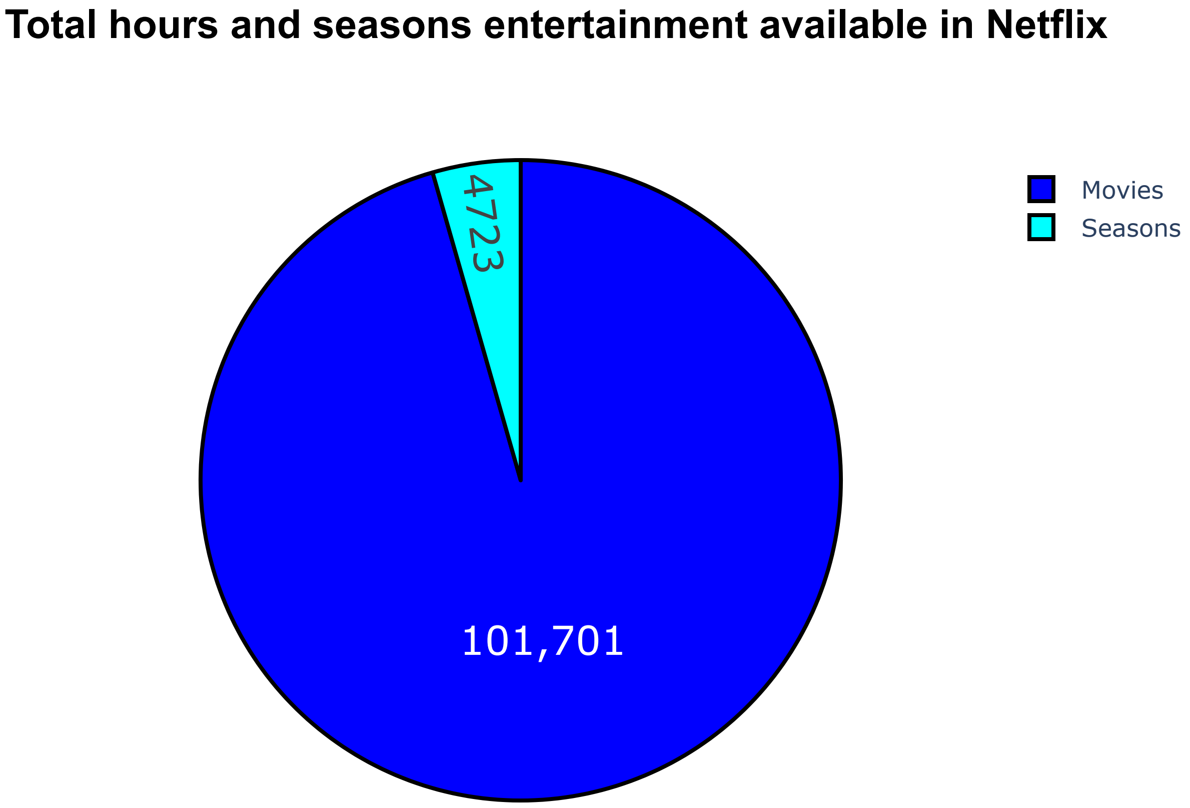
```
#Changing Duration of Movies from minutes to hours approximately %60
df_pie['Total Count'][0]=df_pie['Total Count'][0]//6
```

```
/var/folders/05/kccybxn570gcdt_n5bzs9_280000gn/T/ipykernel_8614/2001946693.py:2: Setting
WithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

We plot the data of total hour content in movies and seasons in tv shows using plotly. Plotly helps to plot the pie charts in an interactive manner. Here we ll be using graph_objects library of plotly to plot the same.

In [146…

```
colors = ['blue', 'cyan']
fig = go.Figure(go.Pie(
    name = "",
    values = df_pie['Total Count'],
    labels = df_pie['type'],
    text = ["Total Hour content in movies", "Total Seasons in TV Shows"],
    hovertemplate = "%{label}: <br>Content available: %{value} </br> %{text}",
    marker=dict(colors=colors, line=dict(color='#000000', width=2)))
)
fig.update_traces(textinfo='value',textfont_size=20)
fig.update_layout(
    height=500,
    title=dict(
        text='<b>Total hours and seasons entertainment available in Netflix</b>',
        x=0.5,
        y=0.95,
        font=dict(
            family="Arial",
            size=20,
            color='#000000'
        )
    ),

)
fig.show()
```



In [146…

```
##-----------------------------------------------------------------------------------------
```

In [146…

```
##-----------------------------------------------------------------------------------------
```

In [146…

```
##-----------------------------------------------------------------------------------------
```

In [ ]:

In [147…

```
type_of_rating=ir.groupby('genre').size().index.tolist()
```

In [147…

```
type_of_rating
```

Out[1473]:

```
['Action & Adventure',
 'British TV Shows',
 'Children & Family Movies',
 'Classic Movies',
 'Comedies',
 'Crime TV Shows',
 'Cult Movies',
 'Documentaries',
 'Docuseries',
 'Dramas',
 'Faith & Spirituality',
 'Horror Movies',
 'Independent Movies',
 'International Movies',
 'International TV Shows',
 "Kids' TV",
 'LGBTQ Movies',
 'Music & Musicals',
 'Reality TV',
 'Romantic Movies',
 'Romantic TV Shows',
 'Sci-Fi & Fantasy',
 'Sports Movies',
 'Stand-Up Comedy',
 'Stand-Up Comedy & Talk Shows',
 'TV Action & Adventure',
 'TV Comedies',
 'TV Dramas',
 'TV Horror',
 'TV Mysteries',
 'TV Sci-Fi & Fantasy',
 'TV Shows',
 'TV Thrillers',
 'Teen TV Shows',
 'Thrillers']
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Overall Movies and TV Shows released trend in past years

We have a count of release year of tv shows and movies released. And we can predict, how well the world film industry is performing with the release count. As we see, in past years to current, the release rate is increased year by year. And in Pandemic duration, the rate has fallen drastically. In pandemic duration , the film industry has facen major loss, due to this Tv shows and movies content added to netflix falled during this duration.

df_8 = df_o.query("release_year >= 2010") df_8 = df_8.groupby("release_year")["show_id"].count().reset_index() fig = px.area(df_8, x='release_year', y='show_id')

fig.update_layout(xaxis_title="Release year", yaxis_title="Number of content",

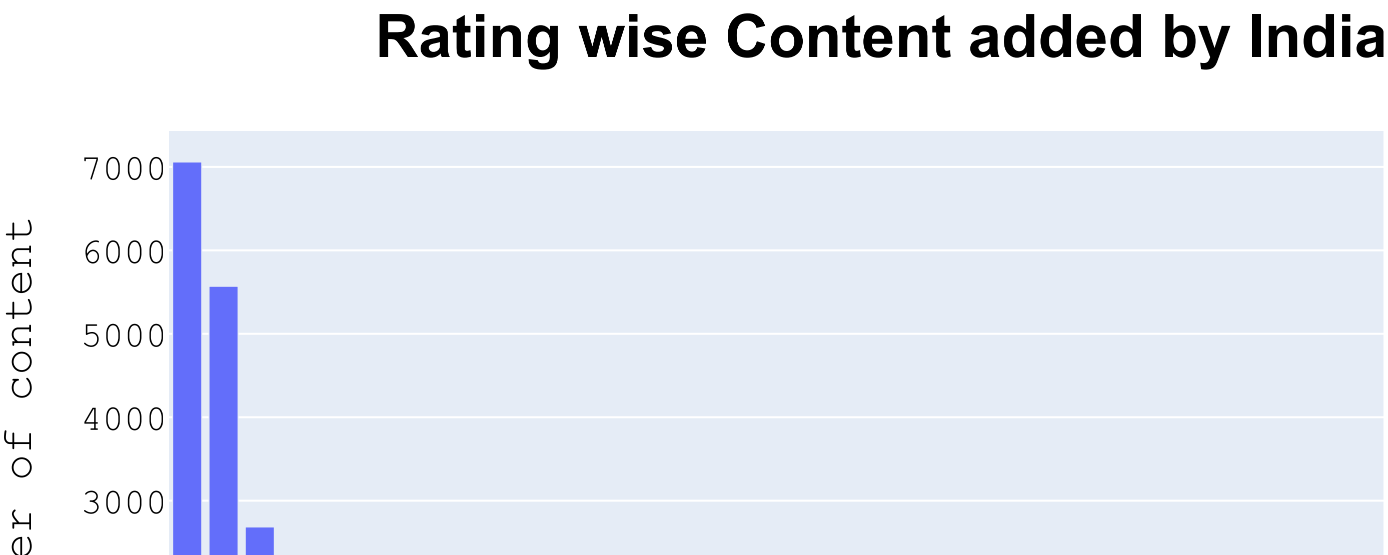
```
height=500,
title=dict(
    text='<b>Overall Movies and TV shows release Trend in past
years</b>',
    x=0.5,
    y=0.95,
    font=dict(
        family="Arial",
        size=20,
        color='#000000'
    )
),
```

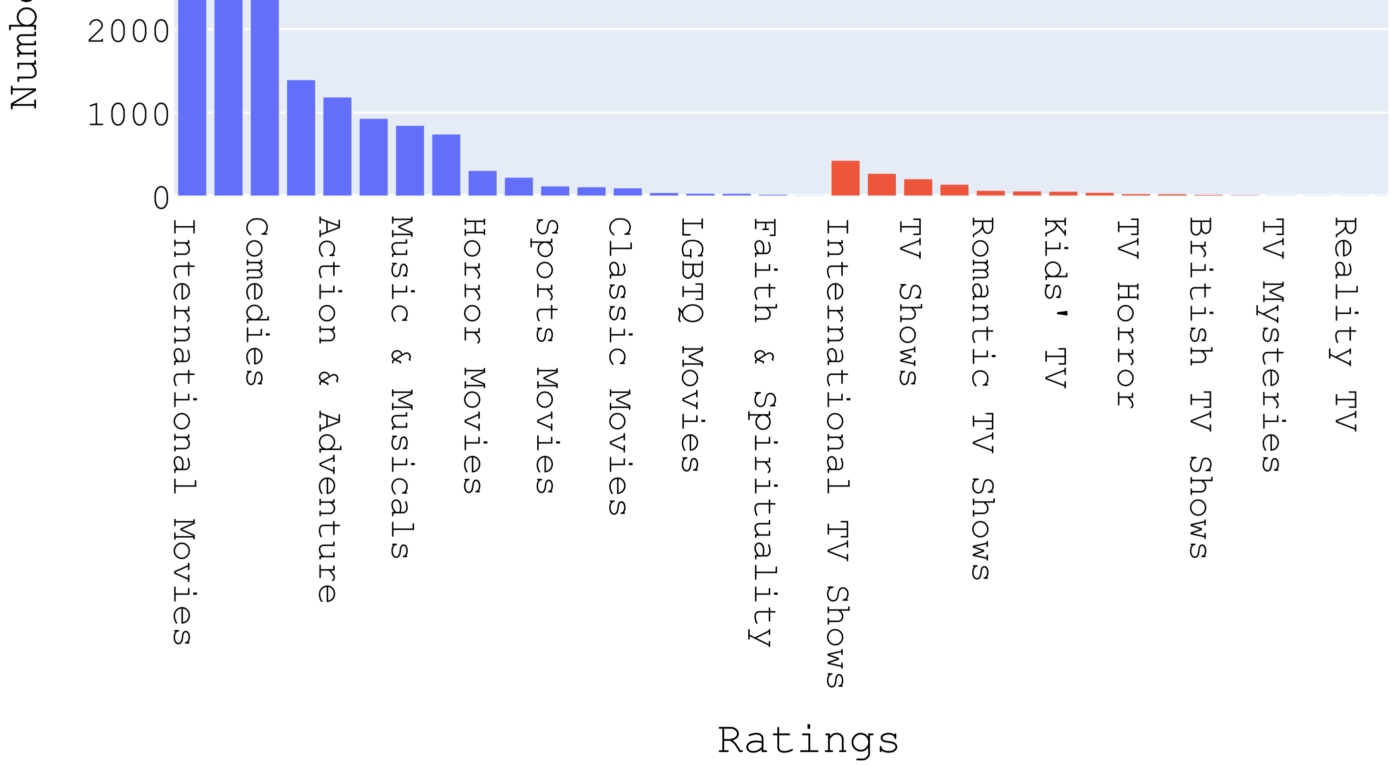
) fig.show()

Total count of movies and TV shows released in India - Genre wise

In [147…

```
ir=df[df['country']=="India"]
type_of_rating=ir.groupby('genre').size().index.tolist()
df7=ir.loc[ir['genre'].isin(type_of_rating)]
df7=df7.groupby(['type'])['genre'].value_counts().reset_index(name='counts')
data_dict1 = {'Genre': df7['genre'],
             'Number of content': df7['counts'],'type': df7['type']
             }
df_R = pd.DataFrame(data=data_dict1, columns=['Genre', 'Number of content','type'])
fig = px.bar(df_R, x="Genre",
                   y="Number of content",
                   title="Genre wise and Category (Movie / TV Shows) wise content added
                   color='type')
fig.update_layout(autosize=False, width=950, height=700, xaxis_title="Ratings",
                  yaxis_title="Number of content",
                  legend_title='Type of Content',
    title=dict(
        text='<b>Rating wise Content added by India</b>',
        x=0.20,
        y=0.94,
        font=dict(
            family="Arial",
            size=30,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=18,
        color='#000000'
    )
)
fig.show()
```





In [ ]:

In [ ]:

In [ ]:

In [ ]:

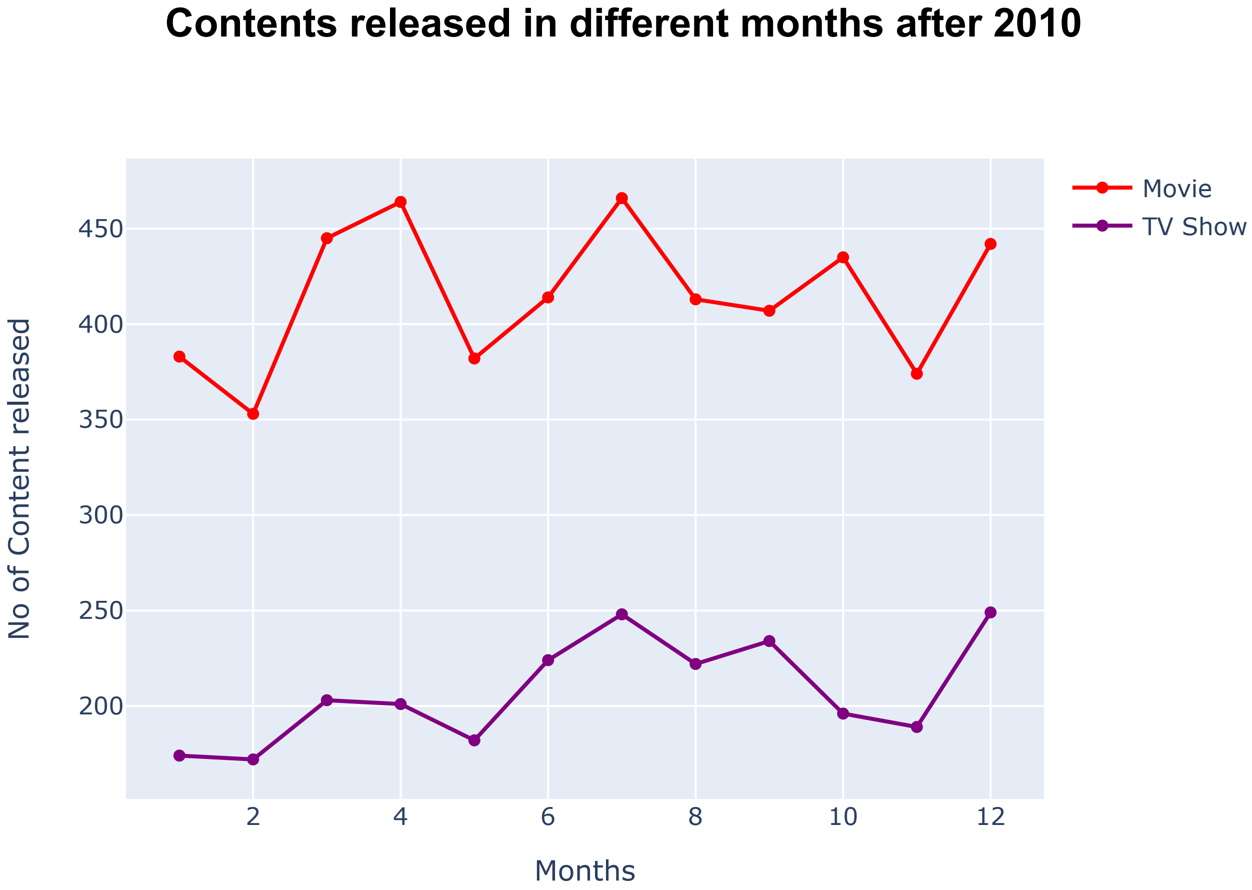
In [ ]:

In [ ]:

Contents released in different months after 2010

In [146…

```
mdmm=df[(df['type']=='Movie')& (df["release_year"] >= 2010)][['title','month']]
mdmm=mdmm.drop_duplicates(keep='last')
mdtt=df[(df['type']=='TV Show')& (df["release_year"] >= 2010)][['title','month']]
mdtt=mdtt.drop_duplicates(keep='last')
movie_month=mdmm.groupby(["month"])["title"].count()
tvshow_month=mdtt.groupby(["month"])["title"].count()

fig = go.Figure()
fig.add_trace(go.Scatter(
x= [1,2,3,4,5,6,7,8,9,10,11,12],
y= movie_month,
showlegend=True,
text = mdmm,
name='Movie',
marker_color='Red'
))
fig.add_trace(go.Scatter(
x=[1,2,3,4,5,6,7,8,9,10,11,12],
y= tvshow_month,
showlegend=True,
text = movie_month,
name='TV Show',
marker_color='Purple'
))
fig.update_layout(xaxis_title="Months", yaxis_title="No of Content released ",
height=500,
title=dict(
    text='<b>Contents released in different months after 2010</b>',
    x=0.5,
    y=0.95,
    font=dict(
        family="Arial",
        size=20,
        color='#000000'
    )
),
)
fig.show()
```



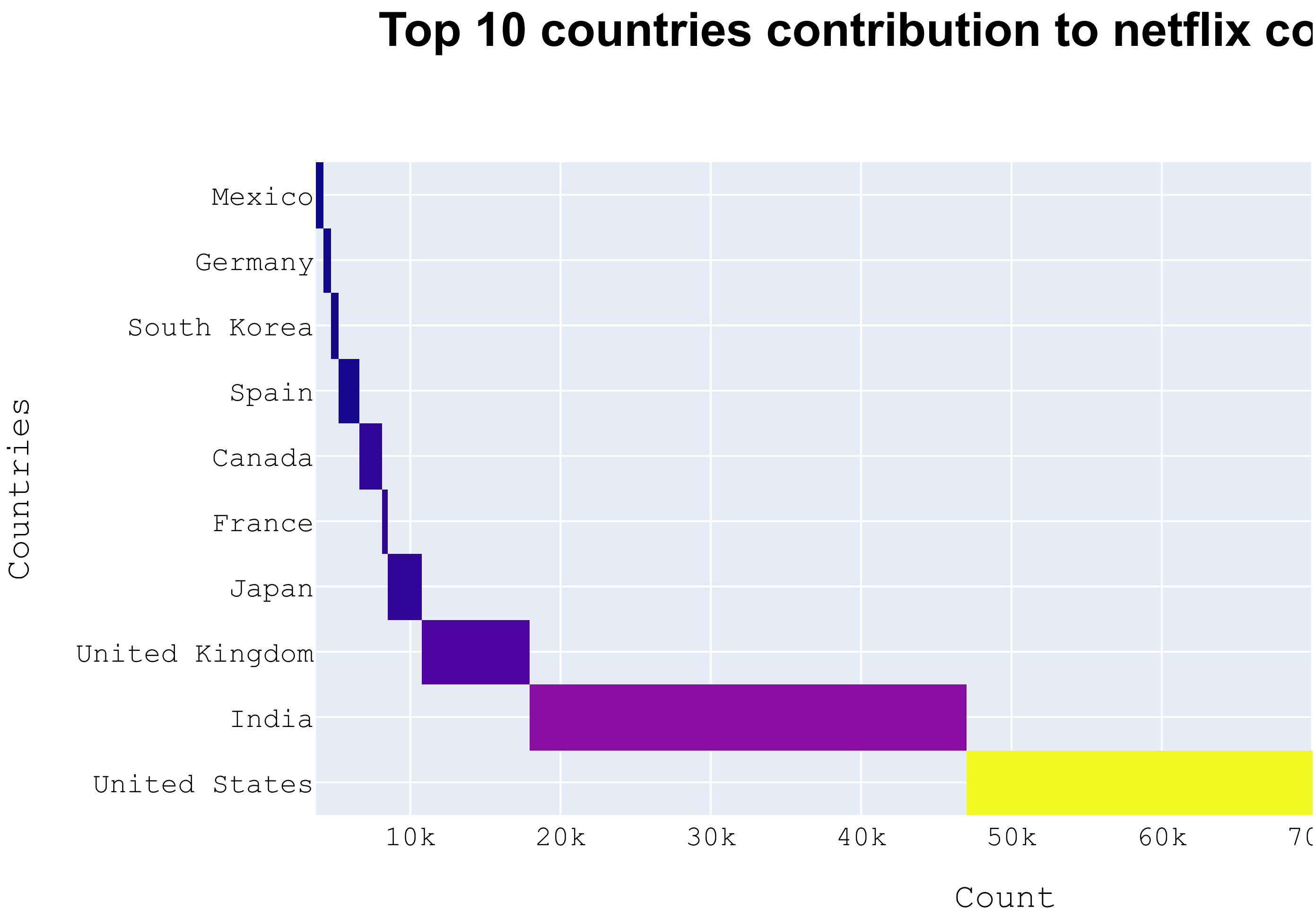
Top 10 countries contribution to netflix content

Top 10 Countries contributes most of the content in netflix content. And this is shown is heatmap. As from the heatmap, we infer, most of the content is from USA.

In [147…

```
data_heat = {'country': df.groupby('country').size().sort_values(ascending=False)[:10].i
             'Number of content': df.groupby('country').size().sort_values(ascending=Fal
             }
df_heat = pd.DataFrame(data=data_heat, columns=['country', 'Number of content'])
fig = go.Figure(data=go.Heatmap(
                   z=df_heat['Number of content'],
                   x=df_heat['Number of content'],
                   y=df_heat['country'],
                   hoverongaps = False))
```

```
fig.update_layout(xaxis_title="Count",
                  yaxis_title="Countries",
                  width=1000,

    title=dict(
        text='<b>Top 10 countries contribution to netflix content</b>',
        x=0.20,
        y=0.96,
        font=dict(
            family="Arial",
            size=25,
            color='#000000'
        )
    ),
    font=dict(
        family="Courier New, Monospace",
        size=15,
        color='#000000'
    )
)
fig.show()
```



Rating wise content added by top 20 countries

This Graph contains Rating wise content added by multiple countries. This graph is one of the greatest feature available in plotly library. And has great feature added which is drop down integration to graph. This has been implemented here for analysing the contents of different countries

```
In [147…  ir=df[df['country']=="India"]
          type_of_rating=ir.groupby('rating').size().index.tolist()

          df7_m=ir[ir['type']=="Movie"].loc[ir['rating'].isin(type_of_rating)]
          df7_m=df7_m.groupby(['rating'])['type'].value_counts().reset_index(name='counts')

          df7_t=ir[ir['type']=="TV Show"].loc[ir['rating'].isin(type_of_rating)]
          df7_t=df7_t.groupby(['rating'])['type'].value_counts().reset_index(name='counts')


          data_dict1 = {'Ratings': df7['rating'],
                       'Movie': df7_m['counts'],'TV Shows': df7_t['counts']
                       }

          df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Movie','TV Shows'])


          x  = 'Ratings'
          y = 'Movie'
          y1='TV Shows'

          trace1 = {
              'x': df_R['Ratings'],
              'y': df_R['Movie'],
              'type': 'bar',
              'name':'Movies Released'
          }
          trace2 = {
              'x': df_R['Ratings'],
              'y': df_R['TV Shows'],
              'type': 'bar',
              'name':'TV Shows released'
          }

          data = [trace1,trace2]

          # Create layout for the plot
          layout=dict(


              title=dict(
                  text='<b>Ratingwise content released in differnt countries</b>',
                  x=0.25,
                  y=0.96,
                  font=dict(
                      family="Arial",
                      size=25,
                      color='#000000'
                  )
              ),
              font=dict(
                  family="Courier New, Monospace",
                  size=15,
                  color='#000000'
              ),
              width=900, height=700, title_x=0.5,
              paper_bgcolor='#fff',
              plot_bgcolor="#fff",
              xaxis=dict(
                  title='Rating',
                  gridcolor='rgb(255,255,255)',
                  zeroline= True,
              ),
              yaxis=dict(
                  title='Number of content released',
                  zeroline= False
```

```
            )
        )

fig = go.FigureWidget(data=data, layout=layout)


def update_fig(change):
    dc=change['new']
    ir=df[df['country']==dc[0]]
    type_of_rating=ir.groupby('rating').size().index.tolist()

    df7_m=ir[ir['type']=="Movie"].loc[ir['rating'].isin(type_of_rating)]
    df7_m=df7_m.groupby(['rating'])['type'].value_counts().reset_index(name='counts')

    df7_t=ir[ir['type']=="TV Show"].loc[ir['rating'].isin(type_of_rating)]
    df7_t=df7_t.groupby(['rating'])['type'].value_counts().reset_index(name='counts')


    data_dict1 = {'Ratings': df7['rating'],
             'Movie': df7_m['counts'],'TV Shows': df7_t['counts']
             }

    df_R = pd.DataFrame(data=data_dict1, columns=['Ratings', 'Movie','TV Shows'])

    with fig.batch_update():
        for trace, column in zip(fig.data,["Movie","TV Shows"]):
            trace.y = df_R[column]

drop = w.Dropdown(options=[
    ('India', ['India']),
    ('Japan', ['Japan']),
    ('United States', ['United States']),
    ('United Kingdom', ['United Kingdom']),
    ('France', ['France']),
    ('Canada', ['Canada']),
    ('South Korea', ['South Korea']),
    ('Germany', ['Germany']),
    ('Mexico', ['Mexico']),
    ('Turkey', ['Turkey']),
    ('Mexico', ['Mexico']),
    ('Australia', ['Australia']),
    ('Nigeria', ['Nigeria']),
    ('Hong Kong', ['Hong Kong']),
    ('Egypt', ['Egypt']),
    ('Indonesia', ['Indonesia']),
    ('Taiwan', ['Taiwan']),
    ('Belgium', ['Belgium']),
    ('Thailand', ['Thailand']),
    ('China', ['China'])

])
drop.observe(update_fig, names='value')
display(w.VBox([drop, fig]))
```

```
VBox(children=(Dropdown(options=(('India', ['India']), ('Japan', ['Japan']), ('United St
ates', ['United States…
```

Business Insights

- India stands second in content creation, as this will increase in coming years
- In fourth place , japan is there and contribures less than 15k contents, and most of the contents is kids based, Anime, comics . This will increase in coming years

- The grown country Chine is not in first 10 top countries which added content to netflix
- In Pandemic, the contents released ,and added to netflix became less. This infers, if any such situation happens , there will be a major loss and there must be a backup plan for such period
- We are not sure on the subscribers count and view count of each contents, so we can't able to identify the best/worst contents in netflix

Recommendations

Below are the recommendations , that can be recommended for the business to sustain/grow in the future years

- As infer from graph, movies contents released more in April and july months, and TV shows released more in july and december.
- July may be the right month for content release
- Backup plan should be there in tough situations like pandameic to combat the loss
- Adult only rated movies has highest released rate. Such movies are not suitable to release in countries like india
- In india Teens- above 14 age rated movies released in highest rate and this type of movies should continue to release in higher rates
- Japan has more contents in kids sections , as they are more liked by people of Japan. Anime and comic releated movies, Animated characters movies are mostly released , and this can be continued.
- In Nigeria , Adult movies and Kids - with parental guidance movies has same number. So Adult movies can be replaced with Kids movies and more Teens-above age 12 and 14 category movies should be released, as there are no such data releated to those category available in above Analysis graphs.
- In whole, New category movies, should be released more, instead of releasing same category movies. This will create a positive impact on Netflix. And more movies should be dubbed in regional languages

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: