Data Visualization Project

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
path = '/content/drive/MyDrive/DataVis/netflix_titles.csv'
data = pd.read_csv(path)
data.head(11)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duratio
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 m
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	Seaso
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Seas
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Seas
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	Seaso
5	s6	TV Show	Midnight Mass	Mike Flanagan	Kate Siegel, Zach Gilford, Hamish Linklater, H...	NaN	September 24, 2021	2021	TV-MA	1 Seas
			My Little	Robert	Vanessa Hudgens,					

```
data.shape
```

```
    (8807, 12)
```

```
data.describe()
```

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000

```
data.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
```

```
data.duplicated().sum()
```

```
0
```

```
data.rating.value_counts()
```

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

```
data.type.value_counts()
```

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

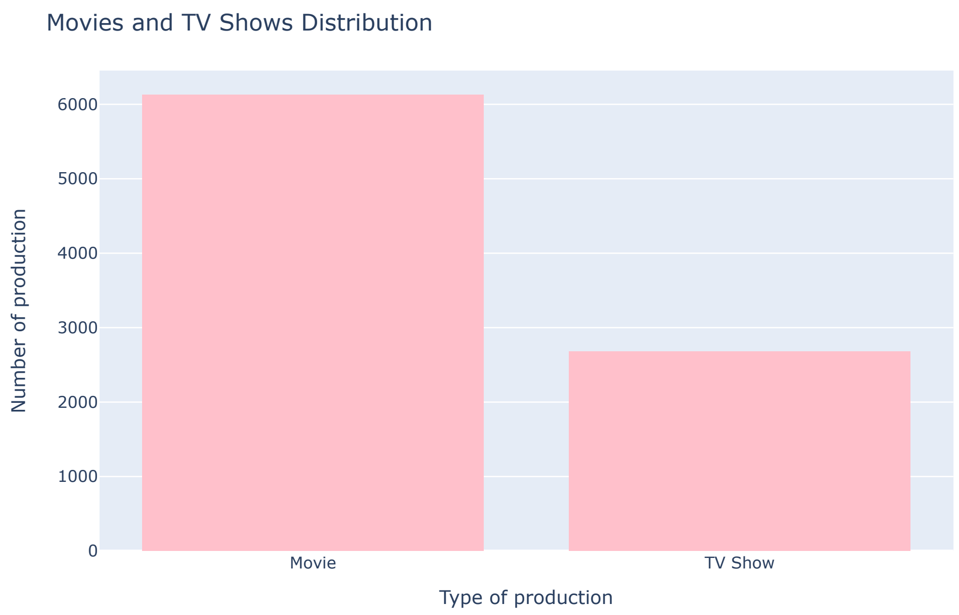
```
data.isna().sum()
```

```
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
```

```
data = data.fillna(data.mode().iloc[0])
```

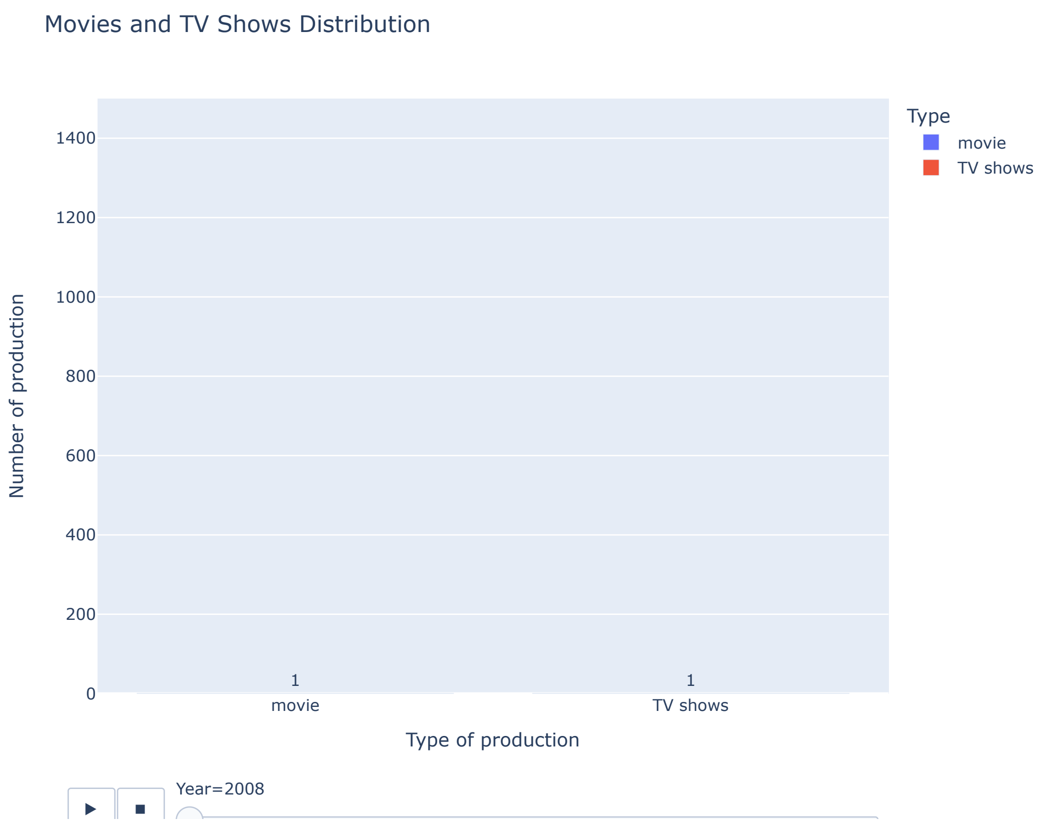
Visualization of the ratio between movies and shows

```
fig = px.histogram(data, x="type", width=800, height=600, color_discrete_sequence=['pink'])
fig.update_layout(title='Movies and TV Shows Distribution', xaxis_title='Type of production', yaxis_title='Number of production', width=800,
fig.show()
```



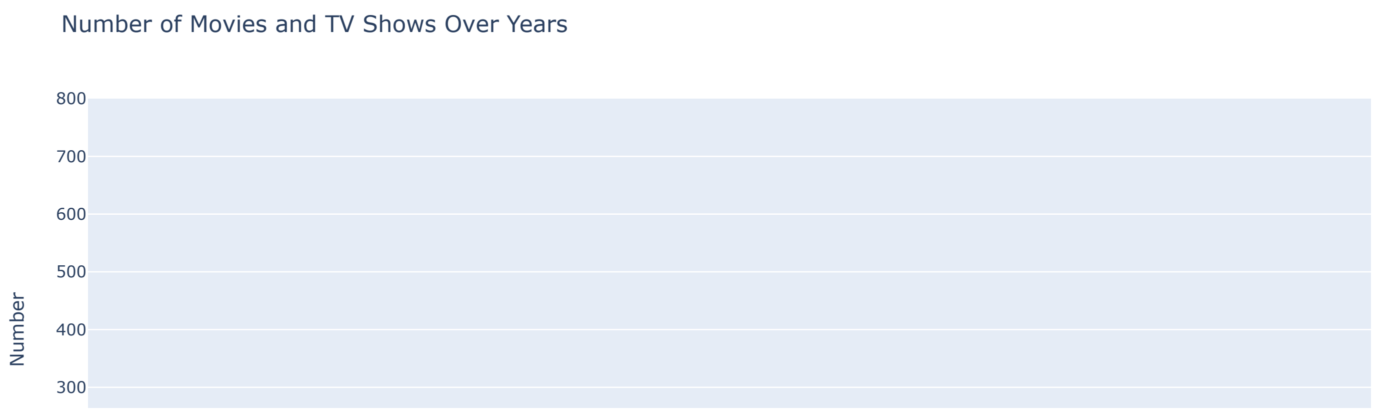
Visualization of the ratio between movies and shows that were added every year

```
data1 = data.copy()
data1['date_added'] = pd.to_datetime(data1['date_added'])
#extract the year
data1['year_added'] = data1['date_added'].dt.year

movies = data1[data1['type'] == 'Movie']
tv_shows = data1[data1['type'] == 'TV Show']

#count the number of every production type for every year
movies_count = movies['year_added'].value_counts().sort_index()
tv_shows_count = tv_shows['year_added'].value_counts().sort_index()

new_rows = pd.Series([0, 0, 0, 0], index=[2009, 2010, 2011, 2012])
tv_shows_count = pd.concat([tv_shows_count.iloc[:1], new_rows, tv_shows_count.iloc[1:]])

#create the DataFrame for the counts
counts_movies_df = pd.DataFrame({'Year': movies_count.index, 'Type': 'movie' , 'Number': movies_count.values})
counts_shows_df = pd.DataFrame({'Year': tv_shows_count.index, 'Type': 'TV shows' , 'Number': tv_shows_count.values})

counts_data = pd.concat([counts_movies_df, counts_shows_df], ignore_index=True)

sorted_data = counts_data.sort_values('Year')

fig = px.bar(counts_data.sort_values(by="Year"), x='Type', y='Number', color='Type', animation_frame='Year',
             labels={'Type': 'Type', 'Number': 'Number'},
             title='Movies and TV Shows Distribution',
             text='Number')
fig.update_layout(title='Movies and TV Shows Distribution', xaxis_title='Type of production', yaxis_title='Number of production', width=800,
fig.update_yaxes(range=[0, 1500])
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.update_layout(xaxis={'type': 'category'})
fig.show()
```



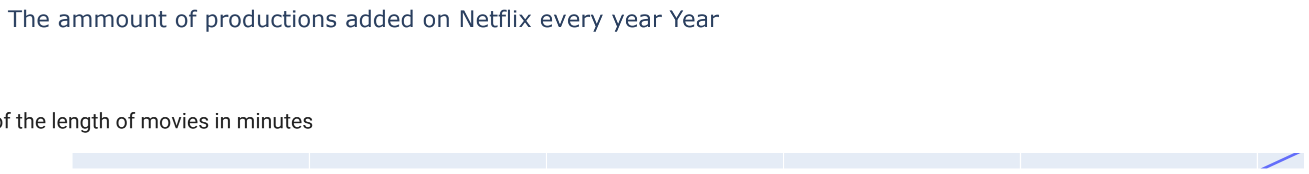
Visualization of the ratio between movies and shows from every year

```
data1 = data.copy()

movies = data1[data1['type'] == 'Movie']
tv_shows = data1[data1['type'] == 'TV Show']

#count the number of every production type for every year
movies_count = movies['release_year'].value_counts().sort_index()
tv_shows_count = tv_shows['release_year'].value_counts().sort_index()

#create the DataFrame for the counts
counts_movies_df = pd.DataFrame({'Year': movies_count.index, 'Type': 'movie' , 'Number': movies_count.values})
counts_shows_df = pd.DataFrame({'Year': tv_shows_count.index, 'Type': 'TV shows' , 'Number': tv_shows_count.values})

counts_data = pd.concat([counts_movies_df, counts_shows_df], ignore_index=True)
counts_data=counts_data[counts_data["Year"] >= 1980]

fig = px.bar(counts_data.sort_values(by="Number"), x='Type', y='Number', color='Type', animation_frame='Year',
             labels={'Type': 'Type', 'Number': 'Number'},
             title='Number of Movies and TV Shows Over Years' ,width=1200, height=600, text='Number')
fig.update_yaxes(range=[0, 800])
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.update_layout(xaxis={'type': 'category'})
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False)
fig.show()
```



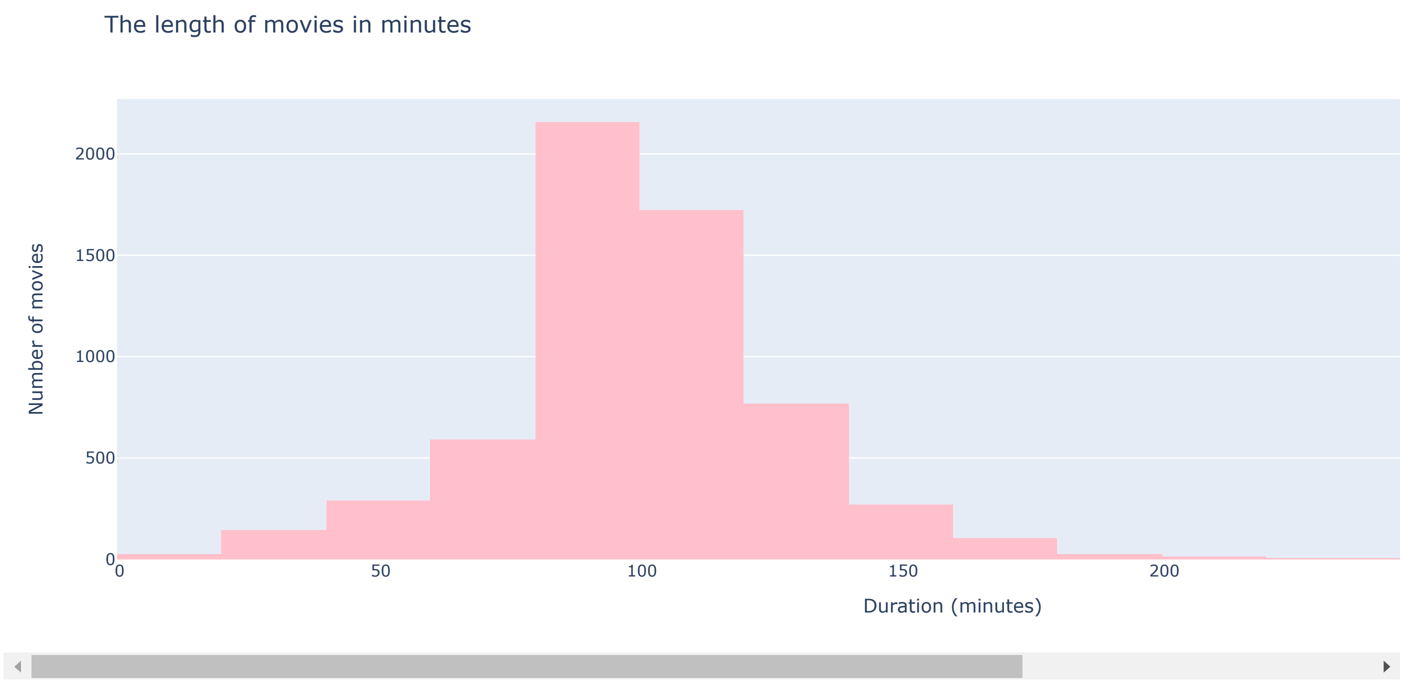
Visualization of the productions added to the dataset per years

```
data1 = data.copy()
data1['date_added'] = pd.to_datetime(data1['date_added'])
#extract the year
data1['year_added'] = data1['date_added'].dt.year

movies = data1[data1['type'] == 'Movie']
tv_shows = data1[data1['type'] == 'TV Show']

#count the number of every production type for every year
movies_count = movies['year_added'].value_counts().sort_index()
tv_shows_count = tv_shows['year_added'].value_counts().sort_index()

new_rows = pd.Series([0, 0, 0, 0], index=[2009, 2010, 2011, 2012])
tv_shows_count = pd.concat([tv_shows_count.iloc[:1], new_rows, tv_shows_count.iloc[1:]])

#create a DataFrame
counts_df = pd.DataFrame({'Year': movies_count.index, 'Movies': movies_count.values, 'TV Shows': tv_shows_count.values})

#animated line plot
fig = px.line(counts_df, x='Year', y=['Movies', 'TV Shows'], title='The ammount of productions added on Netflix every year Year')

# Add animation settings
fig.update_layout(updatemenus=[dict(type='buttons', buttons=[dict(label='Play',
                                                                  method='animate',
                                                                  args=[None, {'frame': {'duration': 500, 'redraw': True},
                                                                              'fromcurrent': True, 'transition': {'duration': 2000}}])])])

frames = [go.Frame(data=go.Scatter(x=counts_df['Year'], y=counts_df[col]),
                   name=col) for col in counts_df.columns[1:]]
fig.frames = frames
fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] = 1000
fig.show()
```



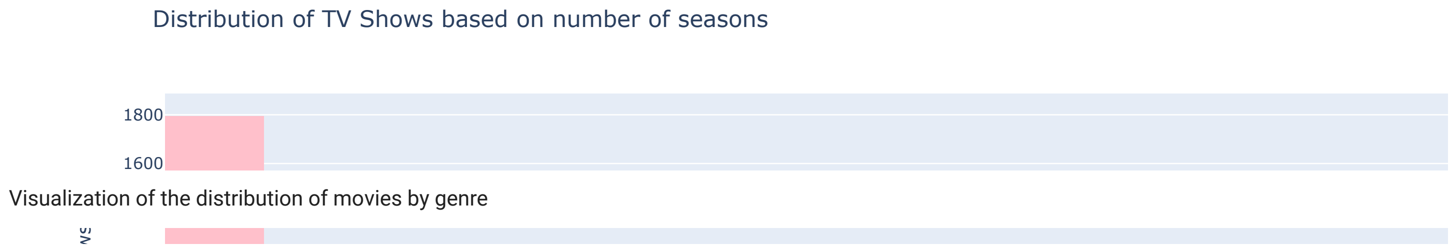
Visualization of the length of movies in minutes

```
movies = data[data['type'] == 'Movie']
movies = movies[~movies['duration'].str.contains('Season', case=False)]
movies['duration'] = movies['duration'].str.replace(' min', '').astype(int)
fig = px.histogram(movies, x='duration', nbins=20, title='The length of movies in minutes',color_discrete_sequence=['pink'])
fig.update_layout(xaxis_title='Duration (minutes)', yaxis_title='Number of movies')
fig.show()
```



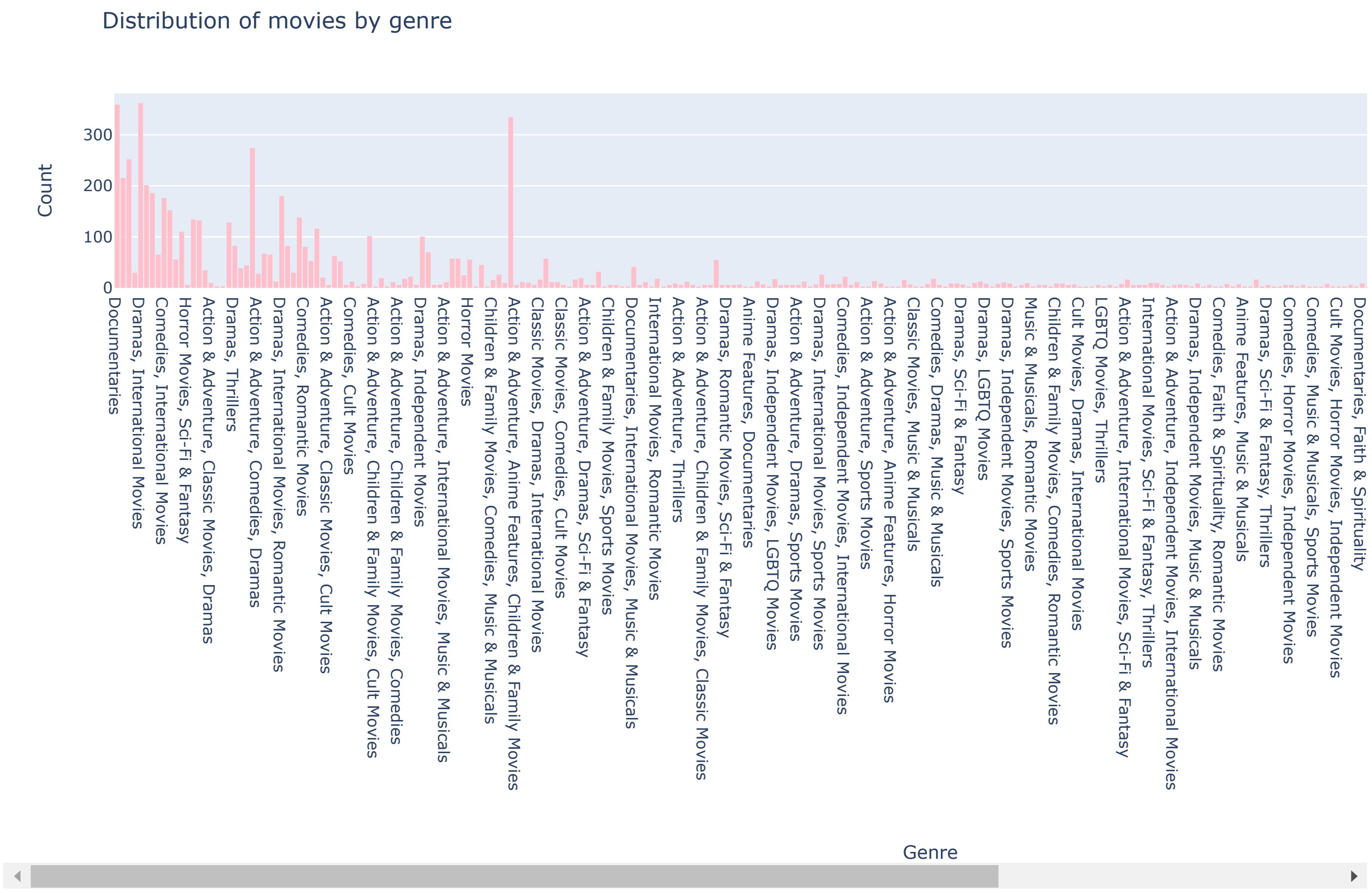
Distribution of TV Shows based on number of seasons

```
shows = data[data['type'] == 'TV Show']
shows = shows[~shows['duration'].str.contains('min', case=False)]
shows['duration'] = shows['duration'].str.replace(' Seasons', '')
shows['duration'] = shows['duration'].str.replace(' Season', '')
shows['duration'] = shows['duration'].astype(int)
fig = px.histogram(shows.sort_values(by="duration"), x='duration', nbins=20, title='Distribution of TV Shows based on number of seasons', col
fig.update_layout(xaxis_title='Number of seasons', yaxis_title='Number of TV Shows')
fig.show()
```



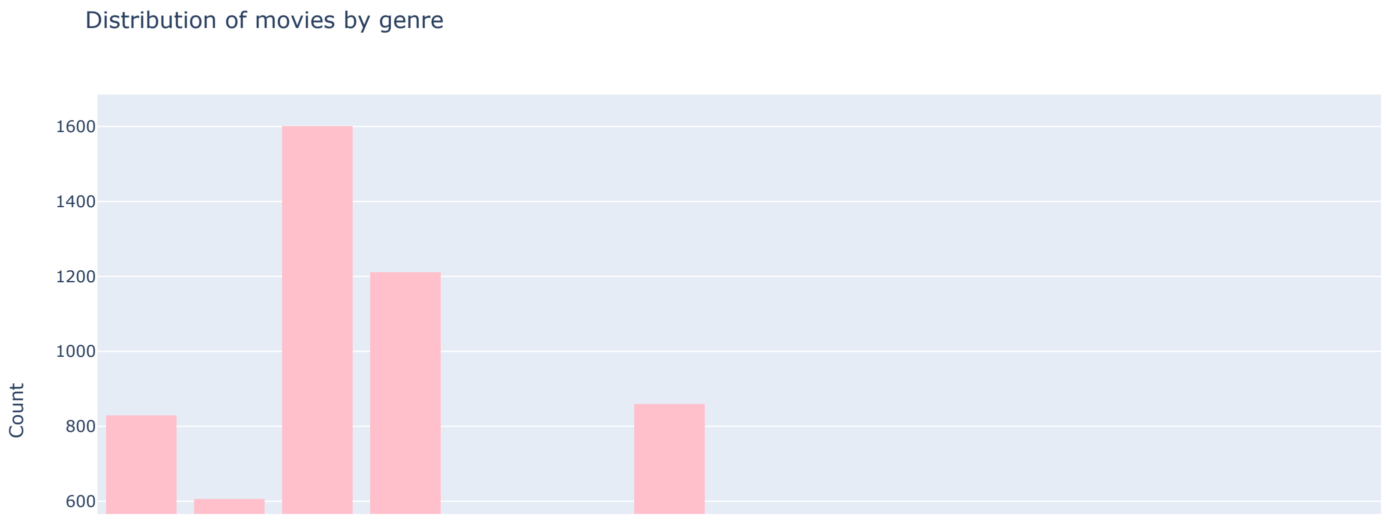
Visualization of the distribution of movies by genre

```
movies = data[data['type'] == 'Movie']
fig = px.histogram(movies, x='listed_in', title='Distribution of movies by genre',  height=700, color_discrete_sequence=['pink'])
fig.update_layout(xaxis_title='Genre', yaxis_title='Count')
fig.show()
```



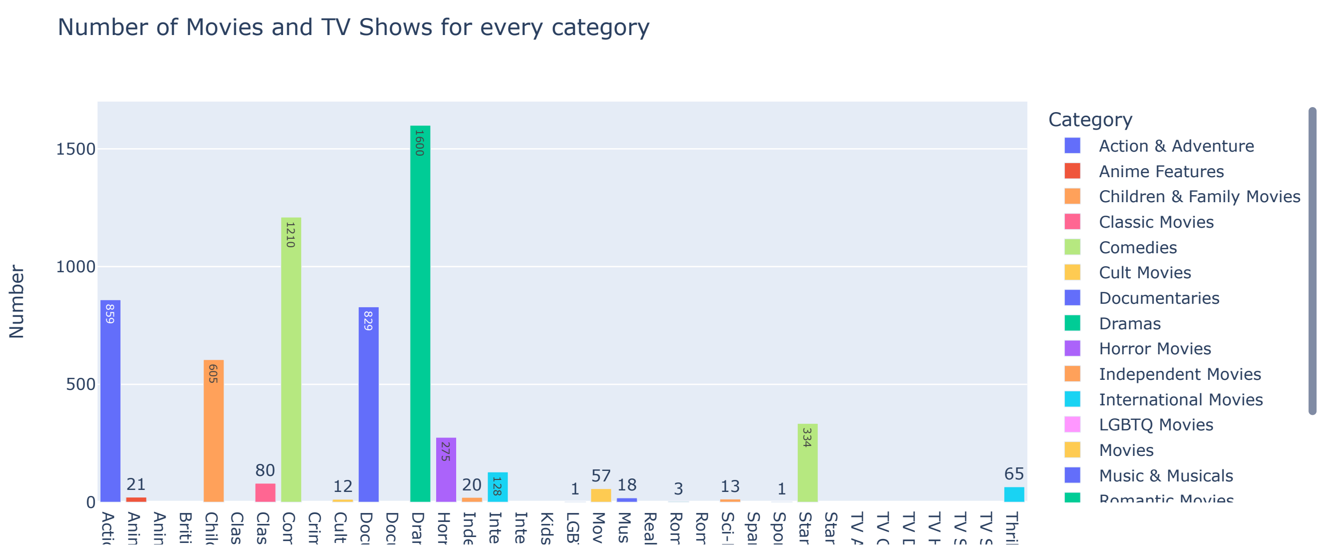
```
movies = data[data['type'] == 'Movie']
movies['listed_in'] = movies['listed_in'].str.split(',').str[0]
fig = px.histogram(movies, x='listed_in', title='Distribution of movies by genre',  height=700, color_discrete_sequence=['pink'])
fig.update_layout(xaxis_title='Genre', yaxis_title='Count')
fig.show()
```

```
<ipython-input-19-5c68a85e067f>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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



```
movies = data[data['type'] == 'Movie']
movies['listed_in'] = movies['listed_in'].str.split(',').str[0]

tv_shows = data[data['type'] == 'TV Show']
tv_shows['listed_in'] = tv_shows['listed_in'].str.split(',').str[0]

movies_count = movies['listed_in'].value_counts().sort_index()
tv_shows_count = tv_shows['listed_in'].value_counts().sort_index()

counts_movies_df = pd.DataFrame({'Category': movies_count.index, 'Type': 'movie' , 'Number': movies_count.values})
counts_shows_df = pd.DataFrame({'Category': tv_shows_count.index, 'Type': 'TV shows' , 'Number': tv_shows_count.values})

counts_data = pd.concat([counts_movies_df, counts_shows_df], ignore_index=True)

fig = px.bar(counts_data.sort_values(by="Category"), x='Category', y='Number', color='Category', animation_frame='Type',
             labels = {'Type': 'Category', 'Number': 'Number'},
             title = 'Number of Movies and TV Shows for every category', width=1000, height=900, text='Number')
fig.update_yaxes(range=[0, 1700])
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.update_layout(xaxis={'type': 'category'}, margin=dict(b=500))
fig.show()
```

```
<ipython-input-20-d3355143dead>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

<ipython-input-20-d3355143dead>:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

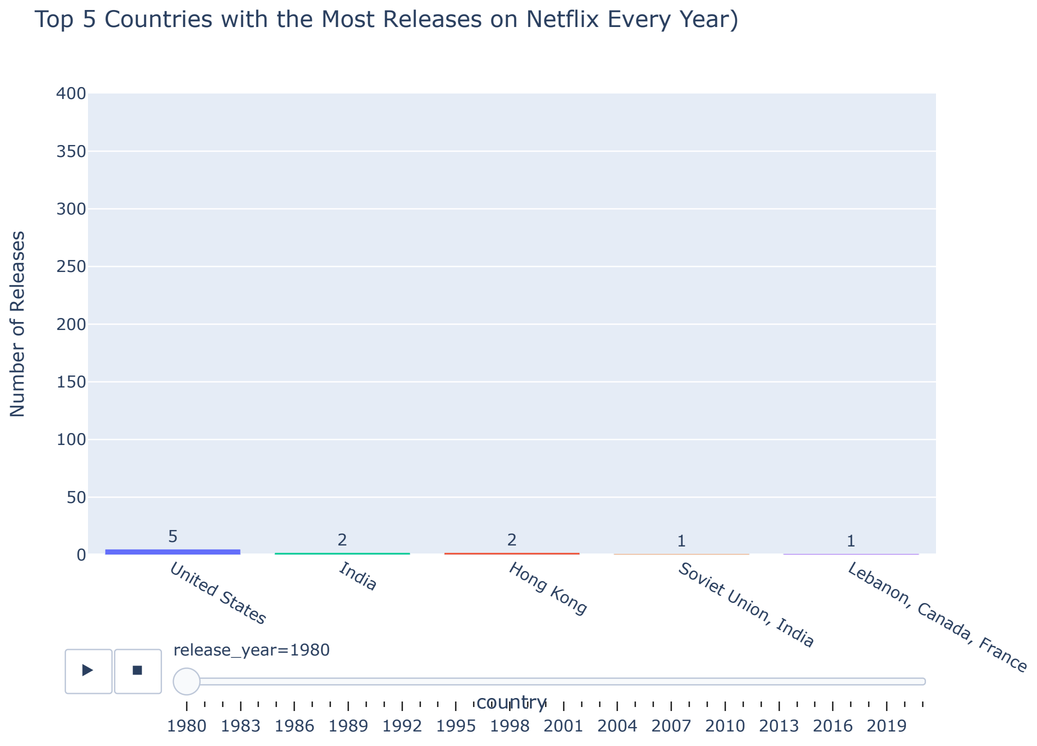


Visualization of the countries with the most releases

```
country_counts = data['country'].value_counts()

top_10_countries = country_counts.head(10)

fig = px.bar(x=top_10_countries.index, y=top_10_countries.values, color=top_10_countries.index)
fig.update_layout(
    title='Top 10 Countries with the Most Releases',
    xaxis_title='Country',
    yaxis_title='Number of Releases',
    showlegend=False
)
fig.show()
```

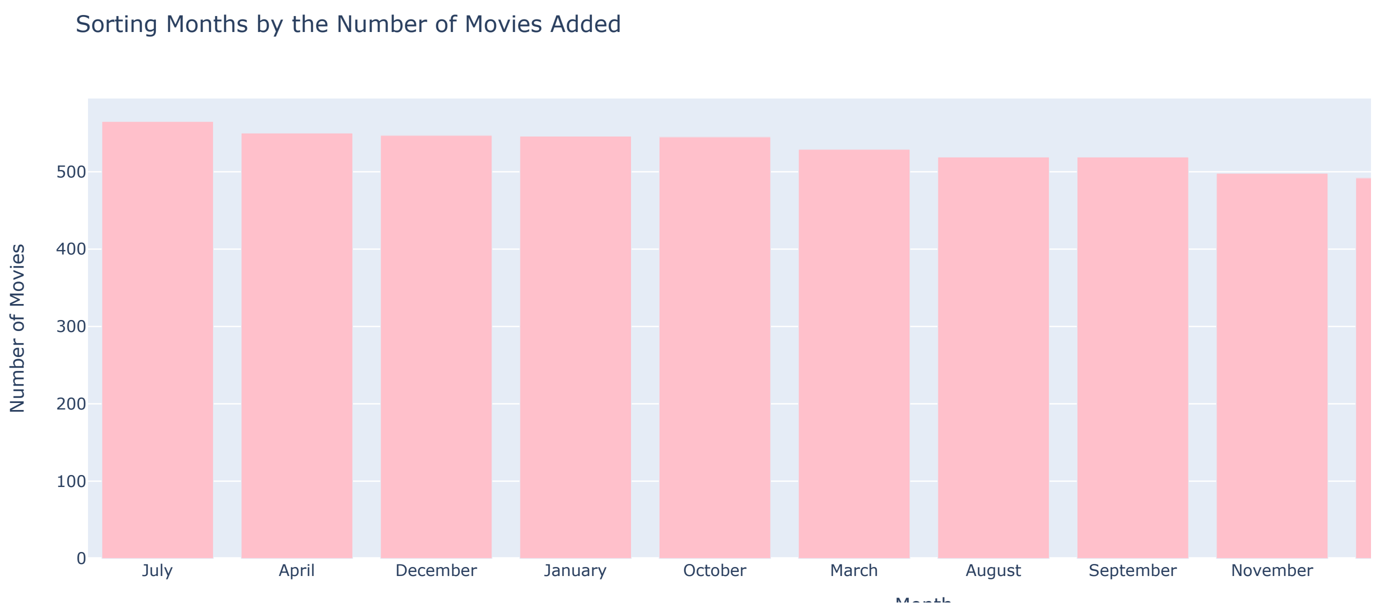
Top 5 Countries with the Most Releases on Netflix Every Year

```
grouped_data = data.groupby(['release_year', 'country']).size().reset_index(name='count')
# Sorting in descending order the data by release year
sorted_data = grouped_data.sort_values(['release_year', 'count'], ascending=[True, False])
sorted_data=sorted_data[sorted_data["release_year"] >= 1980]

top_5_countries = sorted_data.groupby('release_year').head(5)

fig = px.bar(top_5_countries.sort_values(by="release_year"), x='country', y='count', color='country', animation_frame='release_year',
             labels = {'type': 'Country', 'count': 'Number of Releases'},
             title ='Top 5 Countries with the Most Releases on Netflix Every Year)' ,width=800, height=600, text='count')
fig.update_yaxes(range=[0, 400])
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.update_layout(transition = {'duration': 1000})
fig.update_layout(xaxis={'type': 'category'})
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False)
fig.show()
```

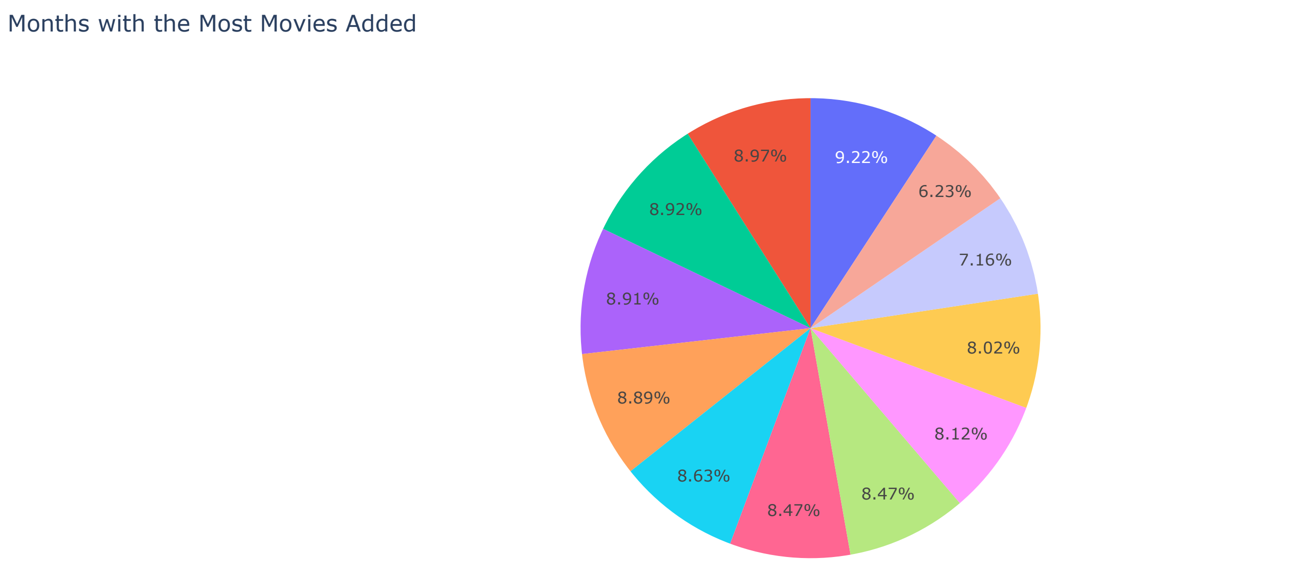


Sorting months by the number of Movies Added

```
data['date_added'] = pd.to_datetime(data['date_added'])

data['month_added'] = data['date_added'].dt.month_name()
movies = data[data['type'] == 'Movie']

month_counts = movies['month_added'].value_counts().sort_values(ascending=False)

fig = px.bar(x=month_counts.index, y=month_counts.values,
             labels={'x': 'Month', 'y': 'Number of Movies'}, title='Sorting Months by the Number of Movies Added', color_discrete_sequence=['
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False)
fig.show()
```



```
data['date_added'] = pd.to_datetime(data['date_added'])
data['month_added'] = data['date_added'].dt.month_name()
movies = data[data['type'] == 'Movie']
month_counts = movies['month_added'].value_counts().sort_values(ascending=False)
fig = px.pie(names=month_counts.index, values=month_counts.values, title='Months with the Most Movies Added')
fig.show()
```



Visualization of the oldest productions

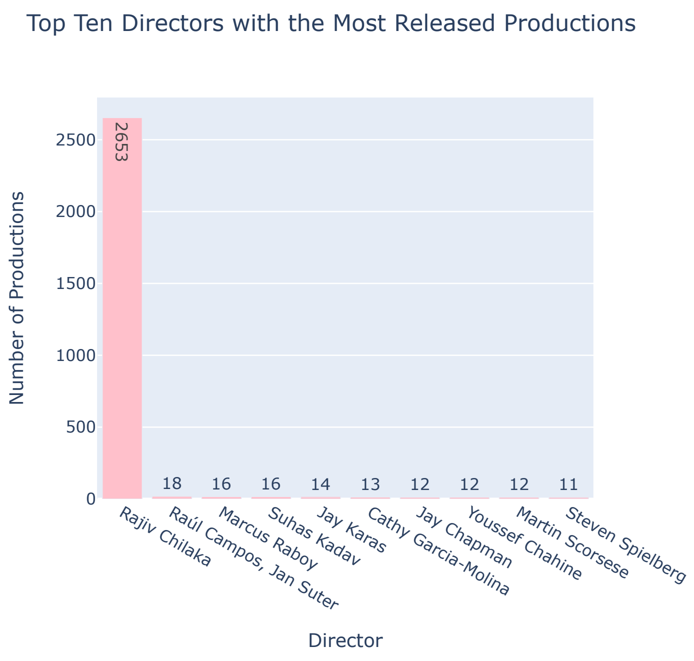
```
import pandas as pd

from tabulate import tabulate

# Calculate the age of each movie
current_year = 2023
data['movie_age'] = current_year - data['release_year']

oldest_movies = data.sort_values('release_year', ascending=True)

top_oldest_movies = oldest_movies.head(10)
```

```
print(tabulate(top_oldest_movies[['type','title','country','release_year','movie_age']], headers='keys', tablefmt='psql'))
```

```
+------+---------+---------------------------------------------+-------------------------------+----------------+-------------+
|      | type    | title                                       | country                       |   release_year |   movie_age |
|------+---------+---------------------------------------------+-------------------------------+----------------+-------------|
| 4250 | TV Show | Pioneers: First Women Filmmakers*           | United States                 |           1925 |          98 |
| 7790 | Movie   | Prelude to War                              | United States                 |           1942 |          81 |
| 8205 | Movie   | The Battle of Midway                        | United States                 |           1942 |          81 |
| 8660 | Movie   | Undercover: How to Operate Behind Enemy Lines | United States               |           1943 |          80 |
| 8739 | Movie   | Why We Fight: The Battle of Russia          | United States                 |           1943 |          80 |
| 8763 | Movie   | WWII: Report from the Aleutians             | United States                 |           1943 |          80 |
| 8640 | Movie   | Tunisian Victory                            | United States, United Kingdom |           1944 |          79 |
| 8436 | Movie   | The Negro Soldier                           | United States                 |           1944 |          79 |
| 8419 | Movie   | The Memphis Belle: A Story of a             | United States                 |           1944 |          79 |
|      |         | Flying Fortress                             |                               |                |             |
| 7930 | Movie   | San Pietro                                  | United States                 |           1945 |          78 |
+------+---------+---------------------------------------------+-------------------------------+----------------+-------------+
```

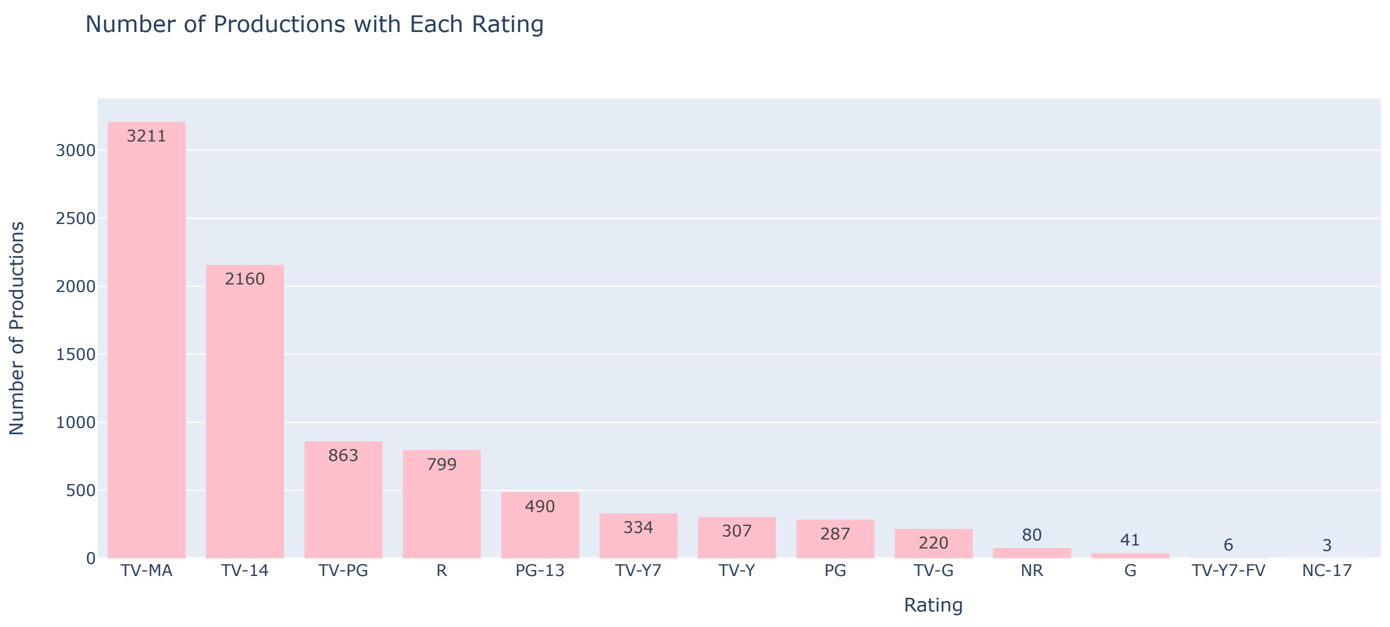
Visualization of the directors with the most productions

```
director_counts = data['director'].value_counts().sort_values(ascending=False)

top_ten_directors = director_counts.head(10)

fig = px.bar(x=top_ten_directors.index, y=top_ten_directors.values,
             labels={'x': 'Director', 'y': 'Number of Productions'},
             title='Top Ten Directors with the Most Released Productions', text=top_ten_directors.values, color_discrete_sequence=['pink'])
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False)
fig.show()
```



Visualization of the number of productions with each rating

```
import pandas as pd
import plotly.express as px

rating_counts = data['rating'].value_counts()

fig = px.bar(x=rating_counts.index, y=rating_counts.values,
             labels={'x': 'Rating', 'y': 'Number of Productions'},
             title='Number of Productions with Each Rating', text=rating_counts.values,  color_discrete_sequence=['pink'])
fig.update_layout(showlegend=False)
fig.update_traces(texttemplate='%{text}',textposition='auto')
fig.show()
```



✓ 0s completed at 2:00 AM