In [16]:

```
#This code imports the necessary libraries for data analysis and visualization:
#Pandas, NumPy, Seaborn, and Matplotlib.
#The "%matplotlib inline" command allows for inline plotting

%matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as pltimport
import matplotlib.pyplot as plt
import nltk
from textblob import TextBlob
from plotly.offline import iplot
#import cufflinks as cf
#cf.go_offline()
import plotly.graph_objects as go
fig = go.Figure()
#from wordcloud import WordCloud
import plotly.express as px
%matplotlib inline
```

In [17]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\archana\anaconda3\lib\s
ite-packages (5.3.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\archana\anacon
da3\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: six in c:\users\archana\anaconda3\lib\site
-packages (from plotly) (1.15.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 23.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [18]:

```
!pip install plotly==5.3.1
```

```
Requirement already satisfied: plotly==5.3.1 in c:\users\archana\anaconda
3\lib\site-packages (5.3.1)


[notice] A new release of pip is available: 23.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip


Requirement already satisfied: tenacity>=6.2.0 in c:\users\archana\anacon
da3\lib\site-packages (from plotly==5.3.1) (8.2.2)
Requirement already satisfied: six in c:\users\archana\anaconda3\lib\site
-packages (from plotly==5.3.1) (1.15.0)
```

In [19]:

```
from plotly.offline import iplot
```

In [20]:

```
#load the data
data = pd.read_csv("C:/Users/ARCHANA/Downloads/Movies_on_Netflix__Prime_Video__Hulu_and_
data
```

Out[20]:

	Unnamed: 0	X	ID	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hul
0	1	0	1	Inception	2010	13+	8.8	87%	1	
1	2	1	2	The Matrix	1999	18+	8.7	87%	1	
2	3	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	
3	4	3	4	Back to the Future	1985	7+	8.5	96%	1	
4	5	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	
...	...	...	...	...	...	...	...	...	...	.
15814	16735	16734	16735	Sultan And The Rock Star	1980	NaN	5.9	NaN	0	
15815	16738	16737	16738	The Bears and I	1974	all	6.2	NaN	0	
15816	16739	16738	16739	Whispers: An Elephant's Tale	2000	all	5.0	NaN	0	
15817	16740	16739	16740	The Ghosts of Buxley Hall	1980	NaN	6.2	NaN	0	
15818	16741	16740	16741	The Poof Point	2001	7+	4.7	NaN	0	

15819 rows × 18 columns

In [21]:

```
#returns the number of elements in the data object
len(data)
```

Out[21]:

```
15819
```

In [22]:

```
#Displays column names in a pandas DataFrame object
data.columns
```

Out[22]:

```
Index(['Unnamed: 0', 'X', 'ID', 'Title', 'Year', 'Age', 'IMDb',
       'Rotten.Tomatoes', 'Netflix', 'Hulu', 'Prime.Video', 'Disney.', 'T
ype',
       'Directors', 'Genres', 'Country', 'Language', 'Runtime'],
      dtype='object')
```

In [23]:

```
#Summary statistics for each column in DataFrame.
data.describe()
```

Out[23]:

	Unnamed: 0	X	ID	Year	IMDb	Netflix
count	15819.000000	15819.000000	15819.000000	15819.000000	15819.000000	15819.000000
mean	8235.393514	8234.393514	8235.393514	2002.527404	5.904109	0.208357
std	4747.752619	4747.752619	4747.752619	20.970346	1.346653	0.406146
min	1.000000	0.000000	1.000000	1902.000000	0.000000	0.000000
25%	4226.500000	4225.500000	4226.500000	1999.000000	5.100000	0.000000
50%	8240.000000	8239.000000	8240.000000	2012.000000	6.100000	0.000000
75%	12297.500000	12296.500000	12297.500000	2016.000000	6.900000	0.000000
max	16741.000000	16740.000000	16741.000000	2020.000000	9.300000	1.000000

In [24]:

```
#Displays summary information about a DataFrame's columns.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15819 entries, 0 to 15818
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Unnamed: 0       15819 non-null  int64  
 1   X                15819 non-null  int64  
 2   ID               15819 non-null  int64  
 3   Title            15819 non-null  object 
 4   Year             15819 non-null  int64  
 5   Age              7234 non-null   object 
 6   IMDb             15819 non-null  float64
 7   Rotten.Tomatoes  5120 non-null   object 
 8   Netflix          15819 non-null  int64  
 9   Hulu             15819 non-null  int64  
 10  Prime.Video      15819 non-null  int64  
 11  Disney.          15819 non-null  int64  
 12  Type             15819 non-null  int64  
 13  Directors        15487 non-null  object 
 14  Genres           15795 non-null  object 
 15  Country          15707 non-null  object 
 16  Language         15583 non-null  object 
 17  Runtime          15819 non-null  int64  
dtypes: float64(1), int64(10), object(7)
memory usage: 2.2+ MB
```

Data Cleaning

In [25]:

```
#Looking for duplicate values
print(data.duplicated().sum())
```

```
0
```

In [26]:

```
#Looking for redundant columns
for column in data.columns:
    print(f"{column}: {data[column].nunique()}")
```

```
Unnamed: 0: 15819
X: 15819
ID: 15819
Title: 15819
Year: 109
Age: 5
IMDb: 82
Rotten.Tomatoes: 99
Netflix: 2
Hulu: 2
Prime.Video: 2
Disney.: 2
Type: 1
Directors: 10964
Genres: 1868
Country: 1274
Language: 1082
Runtime: 224
```

In [27]:

```
#Redundant column
red_col=data.T.duplicated().sum
red_col
```

Out[27]:

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of Unnamed: 0
False
X                  False
ID                  True
Title              False
Year               False
Age                False
IMDb               False
Rotten.Tomatoes    False
Netflix            False
Hulu               False
Prime.Video        False
Disney.            False
Type               False
Directors          False
Genres             False
Country            False
Language           False
Runtime            False
dtype: bool>
```

In [28]:

```
# Removing the redundant column
data = data.drop('ID', axis=1)
data
```

Out[28]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prim
0	1	0	Inception	2010	13+	8.8	87%	1	0	
1	2	1	The Matrix	1999	18+	8.7	87%	1	0	
2	3	2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	
3	4	3	Back to the Future	1985	7+	8.5	96%	1	0	
4	5	4	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	
...	...	...	...	...	...	...	...	...	...	
15814	16735	16734	Sultan And The Rock Star	1980	NaN	5.9	NaN	0	0	
15815	16738	16737	The Bears and I	1974	all	6.2	NaN	0	0	
15816	16739	16738	Whispers: An Elephant's Tale	2000	all	5.0	NaN	0	0	
15817	16740	16739	The Ghosts of Buxley Hall	1980	NaN	6.2	NaN	0	0	
15818	16741	16740	The Poof Point	2001	7+	4.7	NaN	0	0	

15819 rows × 17 columns

In [29]:

```
#Looking for null values
data.isnull().sum()
```

Out[29]:

```
Unnamed: 0              0
X                       0
Title                   0
Year                    0
Age                  8585
IMDb                    0
Rotten.Tomatoes     10699
Netflix                 0
Hulu                    0
Prime.Video             0
Disney.                 0
Type                    0
Directors             332
Genres                 24
Country               112
Language              236
Runtime                 0
dtype: int64
```

In [30]:

```
# replace the word "ALL" with 0 in column 'Age'
data['Age'] = data['Age'].replace('all', 0)
data
```

Out[30]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prime.Video	Disn
0	1	0	Inception	2010	13+	8.8	87%	1	0	0	
1	2	1	The Matrix	1999	18+	8.7	87%	1	0	0	
2	3	2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	
3	4	3	Back to the Future	1985	7+	8.5	96%	1	0	0	
			The								

In [31]:

```
#Removing the '+' sign from Age column
data['Age'] = data['Age'].str.rstrip('+').astype('float')
data
```

Out[31]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prim
0	1	0	Inception	2010	13.0	8.8	87%	1	0	
1	2	1	The Matrix	1999	18.0	8.7	87%	1	0	
2	3	2	Avengers: Infinity War	2018	13.0	8.5	84%	1	0	
3	4	3	Back to the Future	1985	7.0	8.5	96%	1	0	
4	5	4	The Good, the Bad and the Ugly	1966	18.0	8.8	97%	1	0	
...	...	...	...	...	...	...	...	...	...	
15814	16735	16734	Sultan And The Rock Star	1980	NaN	5.9	NaN	0	0	
15815	16738	16737	The Bears and I	1974	NaN	6.2	NaN	0	0	
15816	16739	16738	Whispers: An Elephant's Tale	2000	NaN	5.0	NaN	0	0	
15817	16740	16739	The Ghosts of Buxley Hall	1980	NaN	6.2	NaN	0	0	
15818	16741	16740	The Poof Point	2001	7.0	4.7	NaN	0	0	

15819 rows × 17 columns

In [32]:

```
# Replace 0 with NaN
data['Age'] = data['Age'].replace(0, np.nan)
data['Age'] = data['Age'].fillna(data['Age'].mean())
data
```

Out[32]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu
0	1	0	Inception	2010	13.000000	8.8	87%	1	0
1	2	1	The Matrix	1999	18.000000	8.7	87%	1	0
2	3	2	Avengers: Infinity War	2018	13.000000	8.5	84%	1	0
3	4	3	Back to the Future	1985	7.000000	8.5	96%	1	0
4	5	4	The Good, the Bad and the Ugly	1966	18.000000	8.8	97%	1	0
...	...	...	...	...	...	...	...	...	...
15814	16735	16734	Sultan And The Rock Star	1980	14.474587	5.9	NaN	0	0
15815	16738	16737	The Bears and I	1974	14.474587	6.2	NaN	0	0
15816	16739	16738	Whispers: An Elephant's Tale	2000	14.474587	5.0	NaN	0	0
15817	16740	16739	The Ghosts of Buxley Hall	1980	14.474587	6.2	NaN	0	0
15818	16741	16740	The Poof Point	2001	7.000000	4.7	NaN	0	0

15819 rows × 17 columns

In [33]:

```
#Filling missing values in the Age column with '0'
data['Age'] = data['Age'].fillna(0)
data
```

Out[33]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu
0	1	0	Inception	2010	13.000000	8.8	87%	1	0
1	2	1	The Matrix	1999	18.000000	8.7	87%	1	0
2	3	2	Avengers: Infinity War	2018	13.000000	8.5	84%	1	0
3	4	3	Back to the Future	1985	7.000000	8.5	96%	1	0
4	5	4	The Good, the Bad and the Ugly	1966	18.000000	8.8	97%	1	0
...	...	...	...	...	...	...	...	...	...
15814	16735	16734	Sultan And The Rock Star	1980	14.474587	5.9	NaN	0	0
15815	16738	16737	The Bears and I	1974	14.474587	6.2	NaN	0	0
15816	16739	16738	Whispers: An Elephant's Tale	2000	14.474587	5.0	NaN	0	0
15817	16740	16739	The Ghosts of Buxley Hall	1980	14.474587	6.2	NaN	0	0
15818	16741	16740	The Poof Point	2001	7.000000	4.7	NaN	0	0

15819 rows × 17 columns

In [34]:

```python
#converting the data type of the 'Age' column to integer
data['Age'] = data['Age'].astype(int)
data
```

1	2	1	The Matrix	1999	18	8.7	87%	1	0	0
2	3	2	Avengers: Infinity War	2018	13	8.5	84%	1	0	0
3	4	3	Back to the Future	1985	7	8.5	96%	1	0	0
4	5	4	The Good, the Bad and the Ugly	1966	18	8.8	97%	1	0	1
...	...	...	...	...	...	...	...	...	...	...
15814	16735	16734	Sultan And The Rock Star	1980	14	5.9	NaN	0	0	0

In [35]:

```
#replace all % will nothing
data['Rotten.Tomatoes'] = data['Rotten.Tomatoes'].str.rstrip('%').astype('float')
data
```

Out[35]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prim
0	1	0	Inception	2010	13	8.8	87.0	1	0	
1	2	1	The Matrix	1999	18	8.7	87.0	1	0	
2	3	2	Avengers: Infinity War	2018	13	8.5	84.0	1	0	
3	4	3	Back to the Future	1985	7	8.5	96.0	1	0	
4	5	4	The Good, the Bad and the Ugly	1966	18	8.8	97.0	1	0	
...	...	...	...	...	...	...	...	...	...	
15814	16735	16734	Sultan And The Rock Star	1980	14	5.9	NaN	0	0	
15815	16738	16737	The Bears and I	1974	14	6.2	NaN	0	0	
15816	16739	16738	Whispers: An Elephant's Tale	2000	14	5.0	NaN	0	0	
15817	16740	16739	The Ghosts of Buxley Hall	1980	14	6.2	NaN	0	0	
15818	16741	16740	The Poof Point	2001	7	4.7	NaN	0	0	

15819 rows × 17 columns

In [36]:

```
 # Replace 0 with NaN
data['Rotten.Tomatoes'] = data['Rotten.Tomatoes'].replace(0, np.nan)
data['Rotten.Tomatoes'] = data['Rotten.Tomatoes'].fillna(data['Rotten.Tomatoes'].mean())
data
```

Out[36]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prim
0	1	0	Inception	2010	13	8.8	87.000000	1	0	
1	2	1	The Matrix	1999	18	8.7	87.000000	1	0	
2	3	2	Avengers: Infinity War	2018	13	8.5	84.000000	1	0	
3	4	3	Back to the Future	1985	7	8.5	96.000000	1	0	
4	5	4	The Good, the Bad and the Ugly	1966	18	8.8	97.000000	1	0	
...	...	...	...	...	...	...	...	...	...	
15814	16735	16734	Sultan And The Rock Star	1980	14	5.9	65.393359	0	0	
15815	16738	16737	The Bears and I	1974	14	6.2	65.393359	0	0	
15816	16739	16738	Whispers: An Elephant's Tale	2000	14	5.0	65.393359	0	0	
15817	16740	16739	The Ghosts of Buxley Hall	1980	14	6.2	65.393359	0	0	
15818	16741	16740	The Poof Point	2001	7	4.7	65.393359	0	0	

15819 rows × 17 columns

In [37]:

```
#coverting the datatype of 'Rotten Tomatoes' column to integer
data['Rotten.Tomatoes'] = data['Rotten.Tomatoes'].astype(int)
data
```

Out[37]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prime.Video	Disn
0	1	0	Inception	2010	13	8.8	87	1	0	0	
1	2	1	The Matrix	1999	18	8.7	87	1	0	0	
2	3	2	Avengers: Infinity War	2018	13	8.5	84	1	0	0	
3	4	3	Back to the Future	1985	7	8.5	96	1	0	0	
			The								

In [38]:

```
#getting datatypes inforamtion from data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15819 entries, 0 to 15818
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Unnamed: 0       15819 non-null  int64  
 1   X                15819 non-null  int64  
 2   Title            15819 non-null  object 
 3   Year             15819 non-null  int64  
 4   Age              15819 non-null  int32  
 5   IMDb             15819 non-null  float64
 6   Rotten.Tomatoes  15819 non-null  int32  
 7   Netflix          15819 non-null  int64  
 8   Hulu             15819 non-null  int64  
 9   Prime.Video      15819 non-null  int64  
 10  Disney.          15819 non-null  int64  
 11  Type             15819 non-null  int64  
 12  Directors        15487 non-null  object 
 13  Genres           15795 non-null  object 
 14  Country          15707 non-null  object 
 15  Language         15583 non-null  object 
 16  Runtime          15819 non-null  int64  
dtypes: float64(1), int32(2), int64(9), object(5)
memory usage: 1.9+ MB
```

In [39]:

```
#Looking for null values
data.isnull().sum()
```

Out[39]:

```
Unnamed: 0           0
X                    0
Title                0
Year                 0
Age                  0
IMDb                 0
Rotten.Tomatoes      0
Netflix              0
Hulu                 0
Prime.Video          0
Disney.              0
Type                 0
Directors          332
Genres              24
Country            112
Language           236
Runtime              0
dtype: int64
```

In [40]:

```
##Data Visualization
```

In [41]:

```
#Returns a covariance matrix of all numeric columns in DataFrame.

data.cov()
```

Out[41]:

	Unnamed: 0	X	Year	Age	IMDb	Ro
Unnamed: 0	2.254115e+07	2.254115e+07	-27830.399677	-639.331133	-2595.812377	
X	2.254115e+07	2.254115e+07	-27830.399677	-639.331133	-2595.812377	
Year	-2.783040e+04	-2.783040e+04	439.755397	2.670033	-0.594473	
Age	-6.393311e+02	-6.393311e+02	2.670033	8.059442	-0.393856	
IMDb	-2.595812e+03	-2.595812e+03	-0.594473	-0.393856	1.813474	
Rotten.Tomatoes	-7.268789e+03	-7.268789e+03	-8.126071	-1.231037	5.395766	
Netflix	-1.366673e+03	-1.366673e+03	2.223143	-0.003176	0.077061	
Hulu	-2.401059e+02	-2.401059e+02	0.496675	0.008589	0.012770	
Prime.Video	1.126939e+03	1.126939e+03	-2.358639	0.086882	-0.099306	
Disney.	2.644725e+02	2.644725e+02	-0.171841	-0.087154	0.018721	
Type	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
Runtime	-2.616429e+04	-2.616429e+04	53.220035	-1.122061	3.349845	

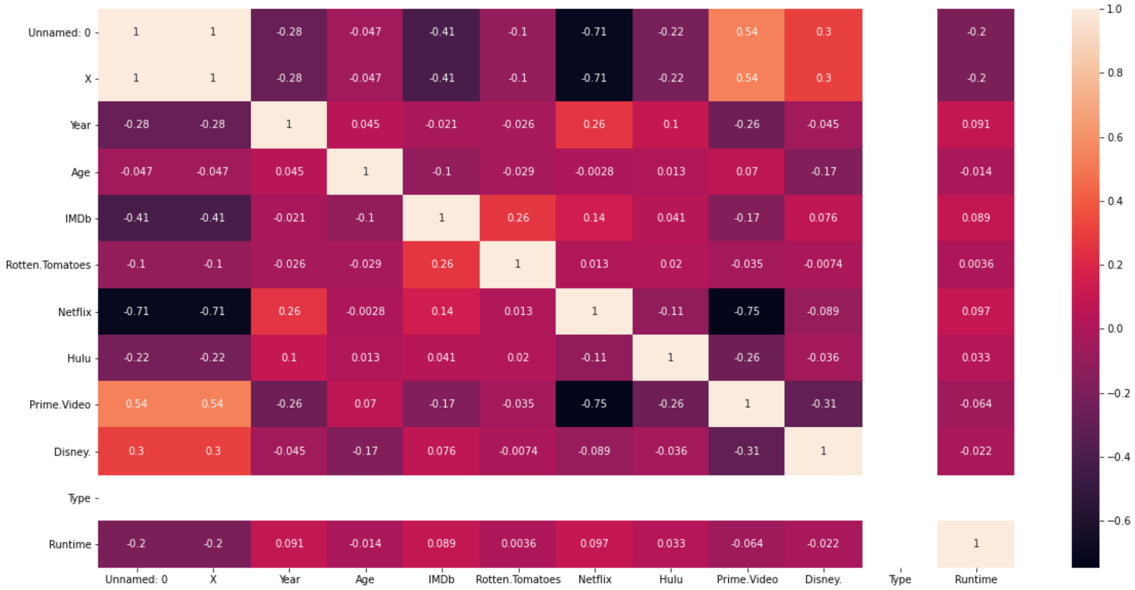
In [42]:

```
#creates a heatmap visualization of the correlation matrix, with correlation coefficient

plt.figure(figsize=(20,10))
sns.heatmap(data.corr(), annot = True)
```

Out[42]:

```
<AxesSubplot:>
```

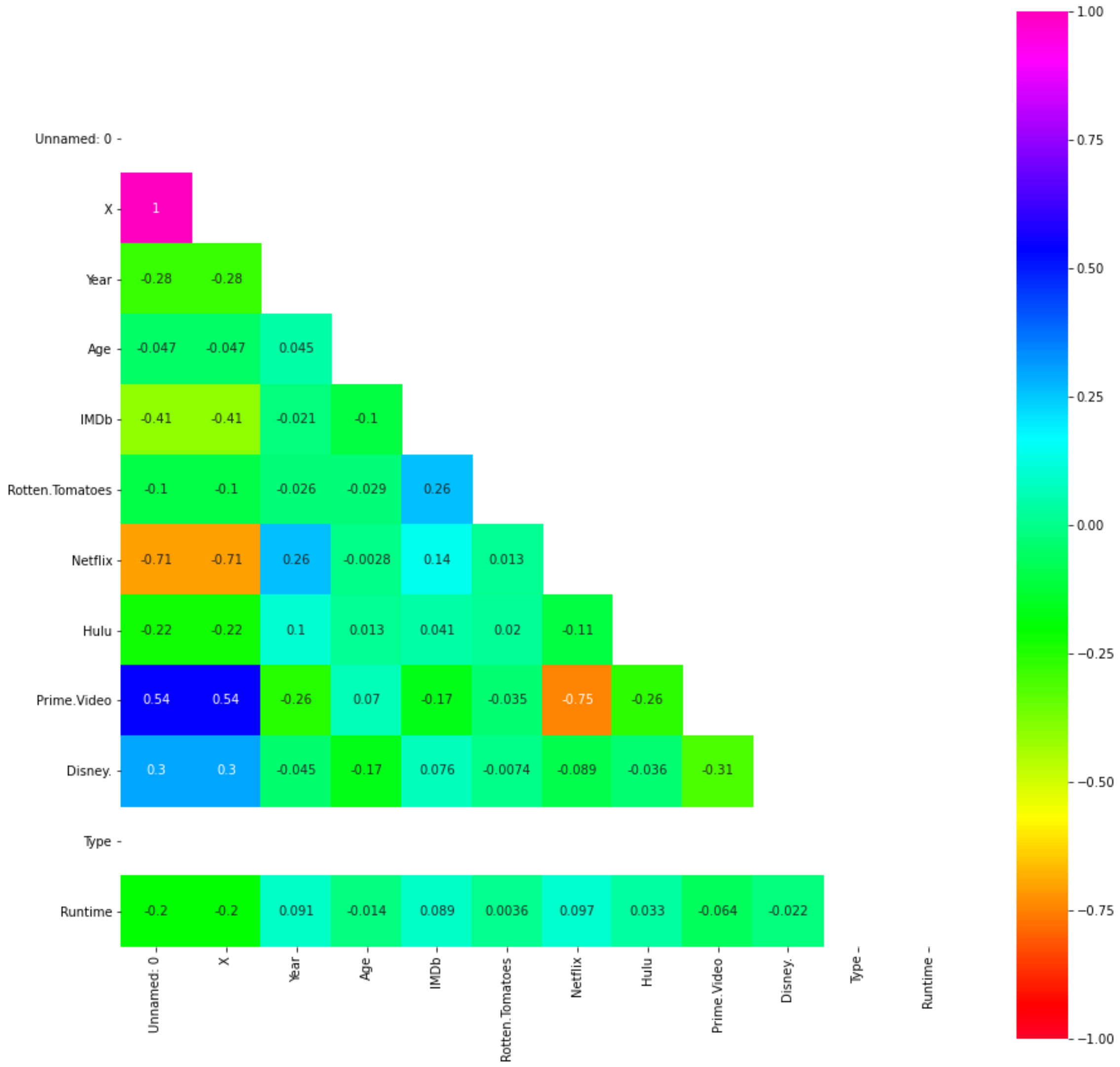


In [ ]:

In [43]:

```
#Creates a square heatmap visualization of the correlation matrix,with the upper triangl
#In a heatmap of a correlation matrix, the upper triangle is a reflection of the lower t
#Lower triangle represents unique correlation between a and b.

fig = plt.figure(figsize = (15,15))
sns.heatmap(data.corr(), mask = np.triu(data.corr()), square = True, annot=True, vmax=1,
plt.show()
```



In [ ]:

In [44]:

```
#Count of each unique value in the 'IMDb' column.

platform_counts = data['IMDb'].value_counts()
platform_counts
```

Out[44]:

```
6.5    543
6.2    535
6.4    506
6.3    503
6.1    501
      ...
9.1      3
9.0      3
1.0      2
1.5      2
1.3      1
Name: IMDb, Length: 82, dtype: int64
```

In [45]:

```
#Age Analysis
data["Age"].value_counts()
```

Out[45]:

```
14    9405
18    3433
7     1442
13    1224
16     315
Name: Age, dtype: int64
```

In [46]:

```
#IMDB Rating Data
print("TV Shows with highest IMDb ratings are= ")
print((data.sort_values("IMDb",ascending=False).head(20))['Title'])
```

```
TV Shows with highest IMDb ratings are= 
6908                                     Down, But Not Out!
4821                                        Love on a Leash
7110                                                 Bounty
6525                Steven Banks: Home Entertainment Center
6261                                             Square One
1287     My Next Guest with David Letterman and Shah Ru...
943                                               Natsamrat
7173                                         Finding Family
8122                                         Where's Daddy?
3296                                        The Dark Knight
6690                              Escape from Firebase Kate
6953                                      A Dog Named Gucci
6528                       Peter Gabriel: Secret World Live
9883                     8 Wheels & Some Soul Brotha' Music
8027                                  Stronger Than Bullets
8459                       The Jones Family Will Make a Way
10681                              Elvis: The Memphis Flash
7890                                             Lost Kites
7939                                                  Arise
7839                                           The Creators
Name: Title, dtype: object
```

In [ ]:

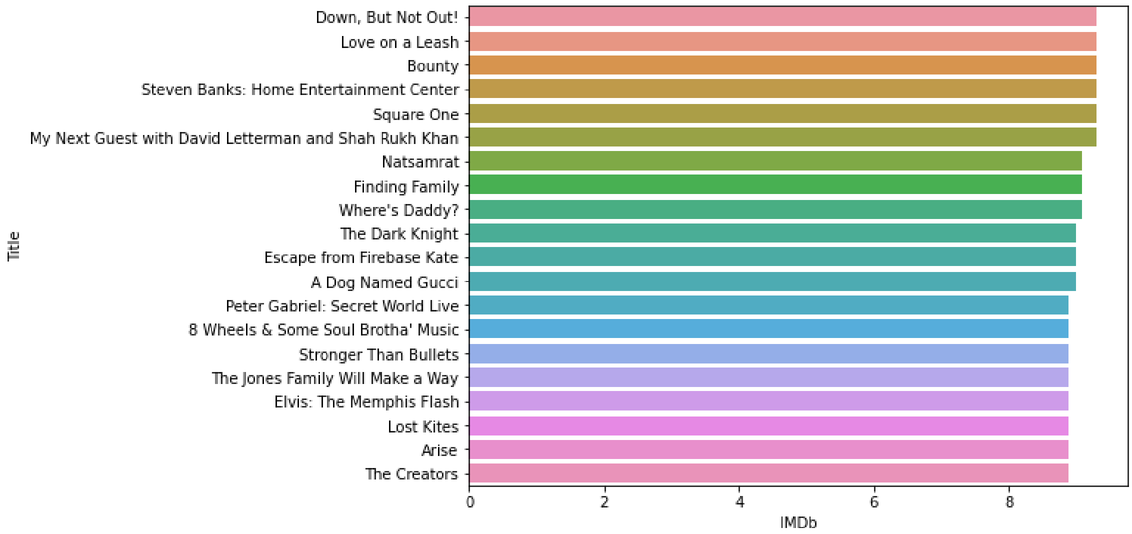
```
```

In [47]:

```
#Now, the top 20 shows with the best ratings.
#barplot of rating
plt.subplots(figsize=(8,6))
sns.barplot(x="IMDb", y="Title" , data= data.sort_values("IMDb",ascending=False).head(20
```

Out[47]:

```
<AxesSubplot:xlabel='IMDb', ylabel='Title'>
```



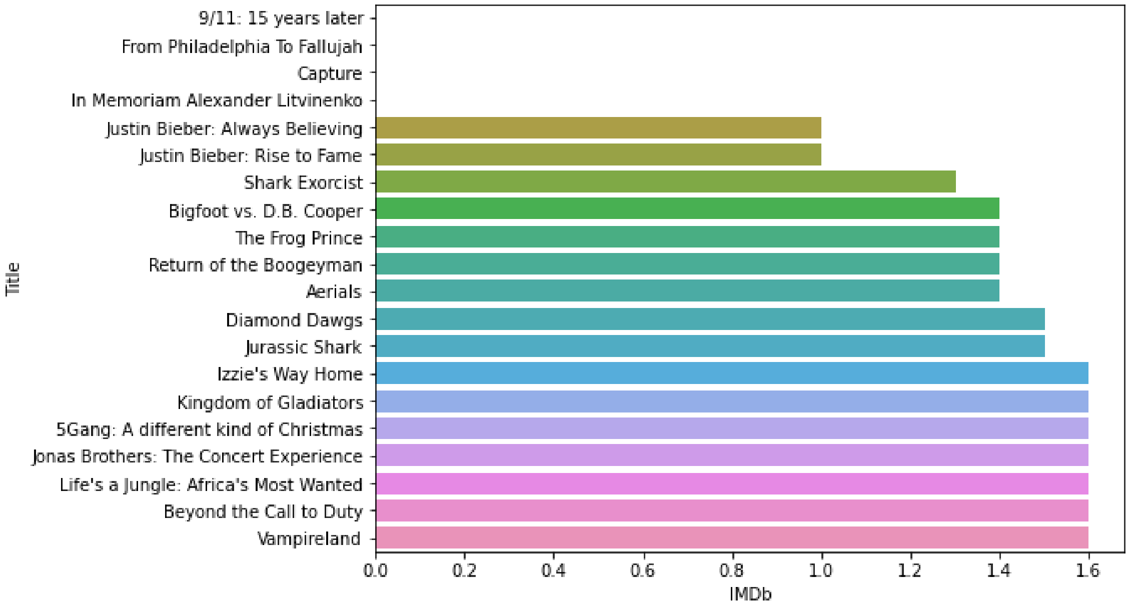
In [ ]:

In [48]:

```
#Now, the TV shows with the worst ratings.
#barplot of rating
plt.subplots(figsize=(8,6))
sns.barplot(x="IMDb", y="Title" , data= data.sort_values("IMDb",ascending=True).head(20)
```

Out[48]:

```
<AxesSubplot:xlabel='IMDb', ylabel='Title'>
```



In [ ]:

In [49]:

```
#we shall cluster the TV shows based on the IMDB rating and Rotten Tomatoes score. First
#Taking the relevant data
ratings=data[["Title",'IMDb',"Rotten.Tomatoes"]]
ratings.head()
```

Out[49]:

	Title	IMDb	Rotten.Tomatoes
0	Inception	8.8	87
1	The Matrix	8.7	87
2	Avengers: Infinity War	8.5	84
3	Back to the Future	8.5	96
4	The Good, the Bad and the Ugly	8.8	97

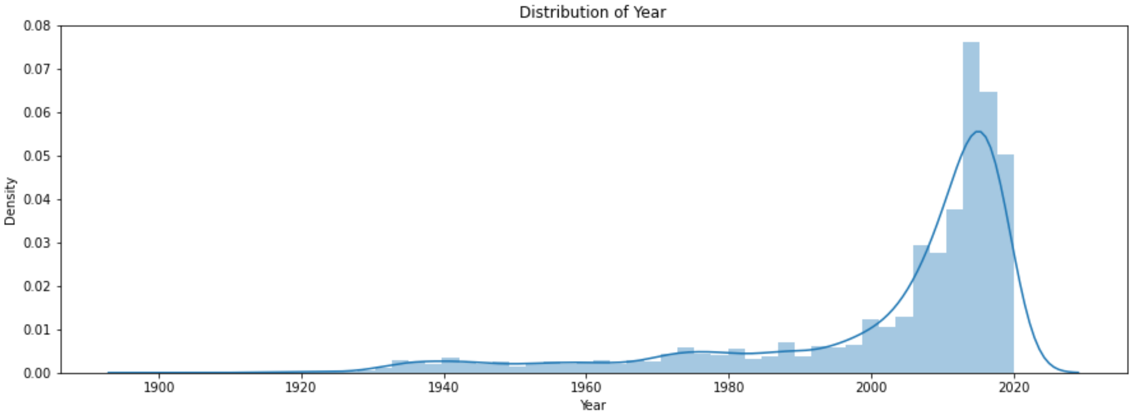
In [ ]:

In [50]:

```
#Visualization of data distribution for year variable using seaborn.
plt.figure(figsize=[15,5])
plt.title("Distribution of Year")
sns.distplot(data['Year'],)
plt.show()
```

```
C:\Users\ARCHANA\anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning:

`distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
```



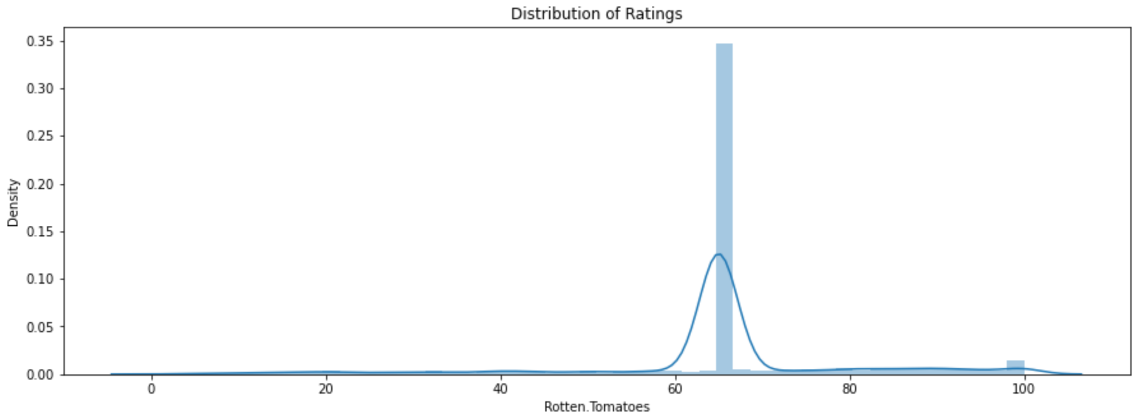
In [ ]:

In [51]:

```
#Plotting distribution of Rotten Tomatoes ratings using seaborn library.
plt.figure(figsize=[15,5])
plt.title("Distribution of Ratings")
sns.distplot(data['Rotten.Tomatoes'])
plt.show()
```

```
C:\Users\ARCHANA\anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning:

`distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
```



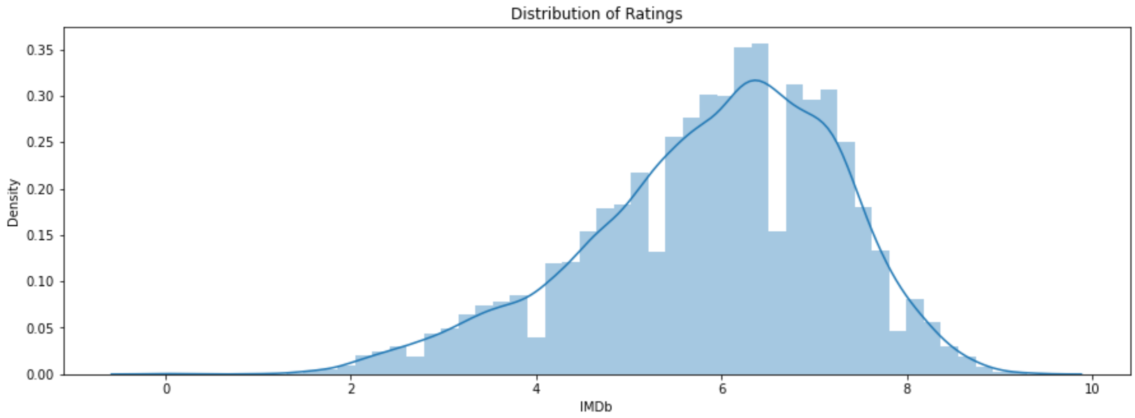
In [ ]:

In [52]:

```
#This code plots the distribution of IMDb ratings.
plt.figure(figsize=[15,5])
plt.title("Distribution of Ratings")
sns.distplot(data['IMDb'])
plt.show()
```

```
C:\Users\ARCHANA\anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning:

`distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
```



In [ ]:

```

```

In [53]:

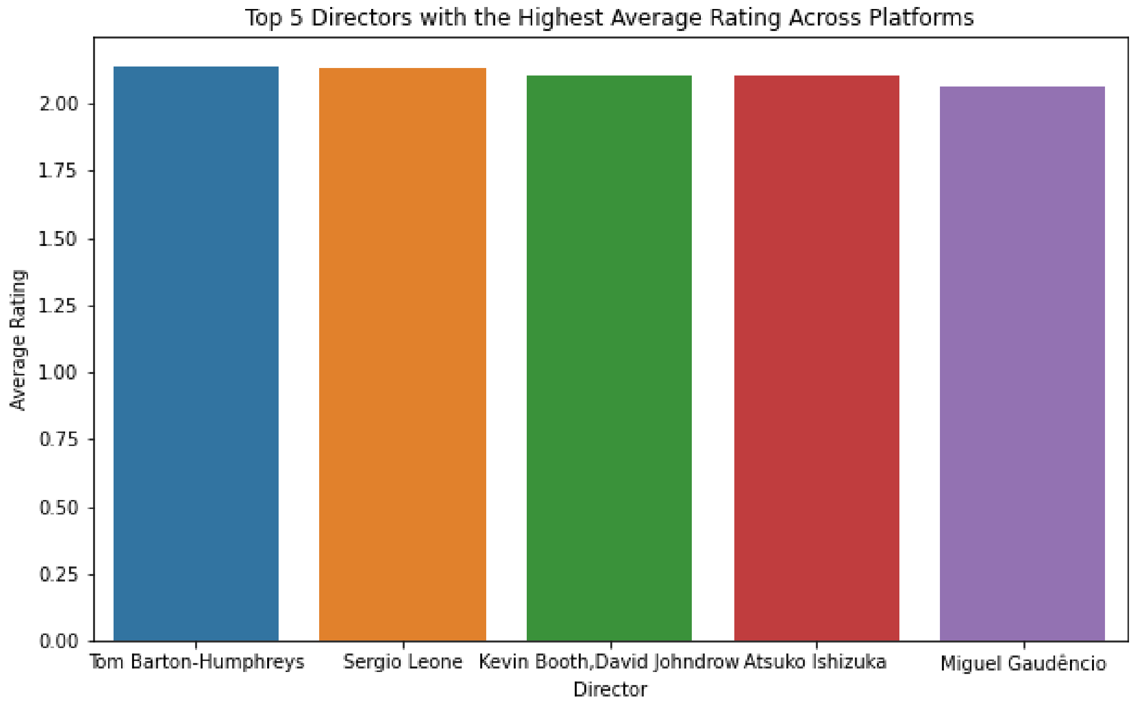
```
## Objective question
```

In [54]:

```
# calculate the average rating of each director across platforms
director_ratings = data.groupby(['Directors'])[['IMDb', 'Netflix', 'Hulu', 'Prime.Video'
# calculate the overall average rating for each director
director_ratings['Overall'] = director_ratings.mean(axis=1)
# sort the directors based on their overall average rating
top_directors = director_ratings.sort_values(by=['Overall'], ascending=False).head(5)
```

In [55]:

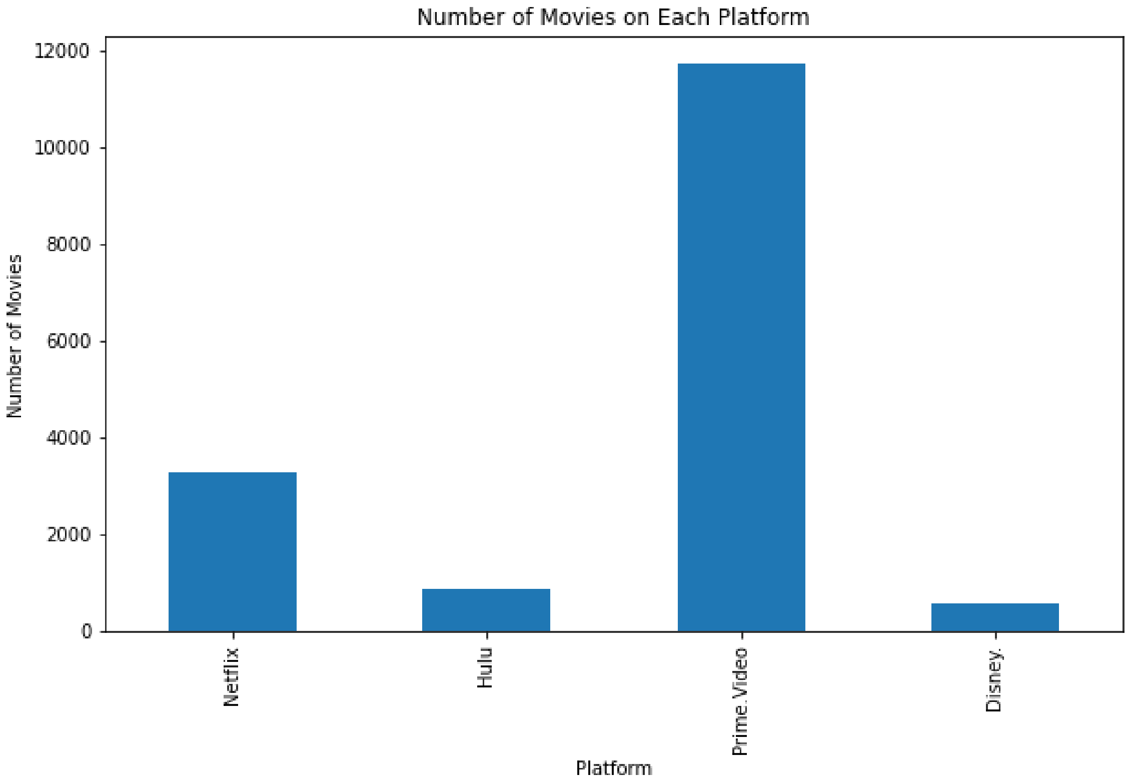
```
# plot the top 5 directors based on their overall average rating
plt.figure(figsize=(10,6))
sns.barplot(x=top_directors.index, y=top_directors['Overall'])
plt.title('Top 5 Directors with the Highest Average Rating Across Platforms')
plt.xlabel('Director')
plt.ylabel('Average Rating')
plt.show()
```



In [ ]:

In [56]:

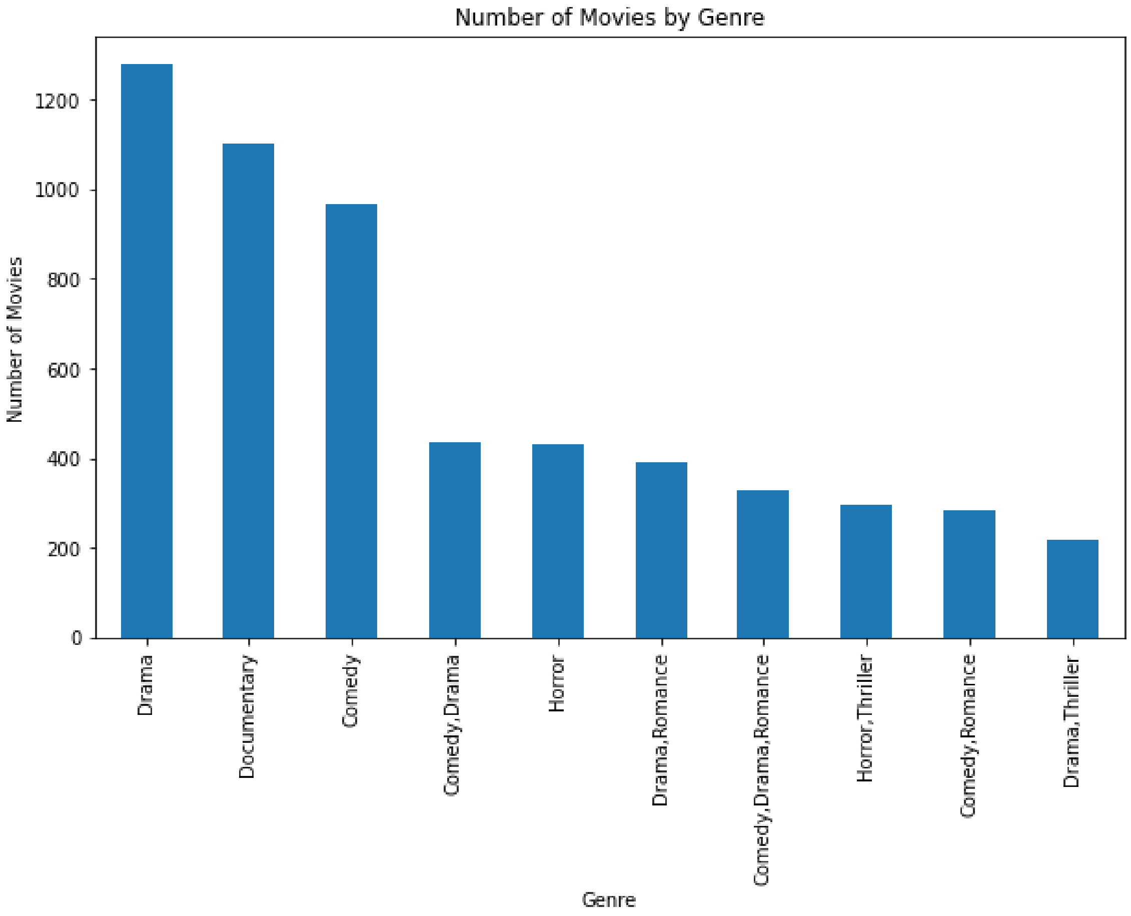
```
# plot the number of movies on each platform
platform_counts = data[['Netflix', 'Hulu', 'Prime.Video', 'Disney.']].sum()
platform_counts.plot(kind='bar', figsize=(10,6))
plt.title('Number of Movies on Each Platform')
plt.xlabel('Platform')
plt.ylabel('Number of Movies')
plt.show()
```



In [ ]:

In [57]:

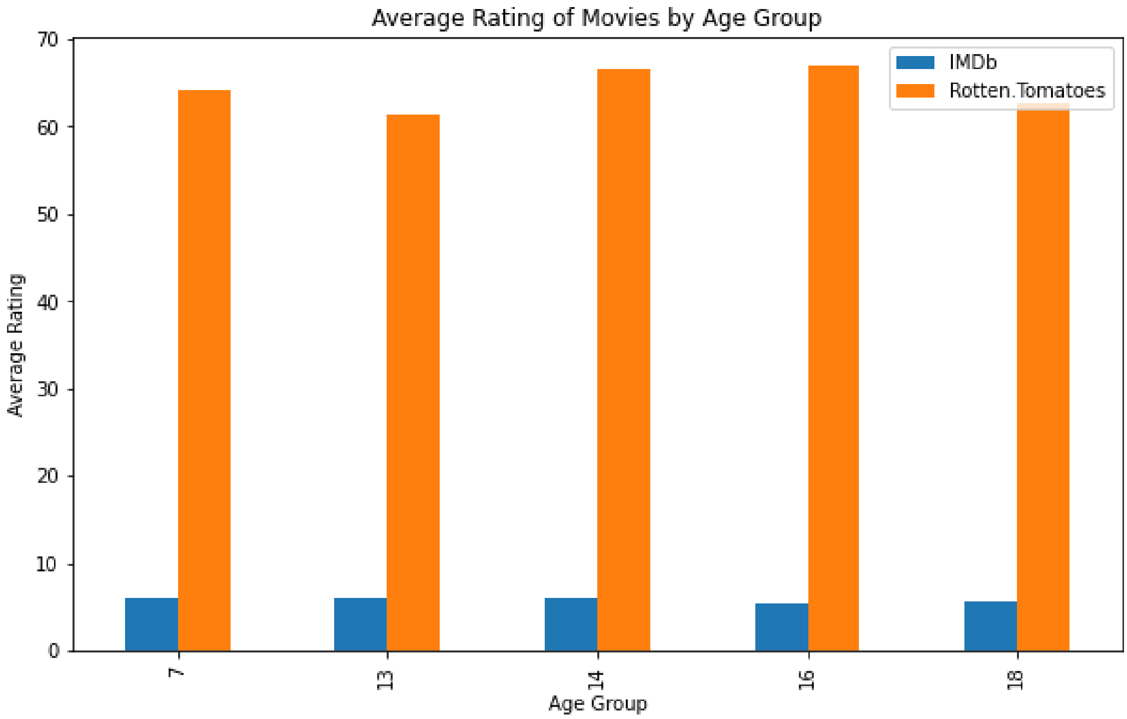
```
# plot the number of movies by genre
genre_counts = data['Genres'].value_counts().head(10)
genre_counts.plot(kind='bar', figsize=(10,6))
plt.title('Number of Movies by Genre')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.show()
```



In [ ]:

In [58]:

```
# plot the average rating of movies by age group
age_ratings = data.groupby(['Age'])[['IMDb', 'Rotten.Tomatoes']].mean()
age_ratings.plot(kind='bar', figsize=(10,6))
plt.title('Average Rating of Movies by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Rating')
plt.show()
```



In [ ]:

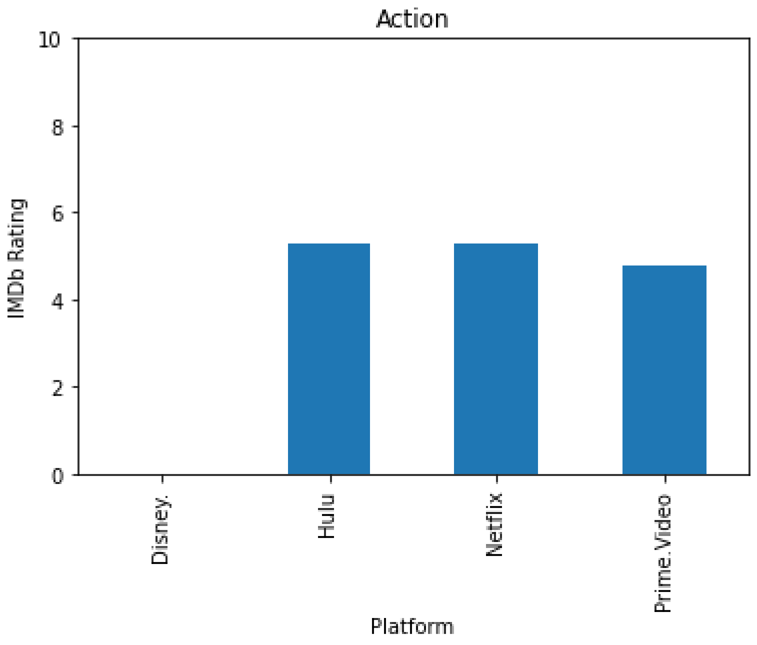
In [59]:

```
#To analyze what content resonates with the audience across different streaming platform
#we can use the movie dataset and extract information such as movie ratings and genres.
# Create a new DataFrame with only the necessary columns
platforms = ["Hulu", "Netflix", "Prime.Video", "Disney."]
cols = ["Title", "IMDb", "Genres", "Year"]
platform_data = pd.DataFrame(columns=cols)

# Loop through each platform and extract the data
for platform in platforms:
    platform_movies = data[data[platform] == 1]
    platform_movies = platform_movies[cols]
    platform_movies["Platform"] = platform
    platform_data = pd.concat([platform_data, platform_movies])

# Group the data by genre and platform and calculate the mean IMDb rating
genre_data = platform_data.groupby(["Genres", "Platform"]).mean()["IMDb"].unstack()

# Plot a bar chart for each genre
for genre in genre_data.index:
    plot_data = genre_data.loc[genre]
    plot_data.plot(kind="bar")
    plt.title(genre)
    plt.xlabel("Platform")
    plt.ylabel("IMDb Rating")
    plt.ylim(0, 10)
    plt.show()
```

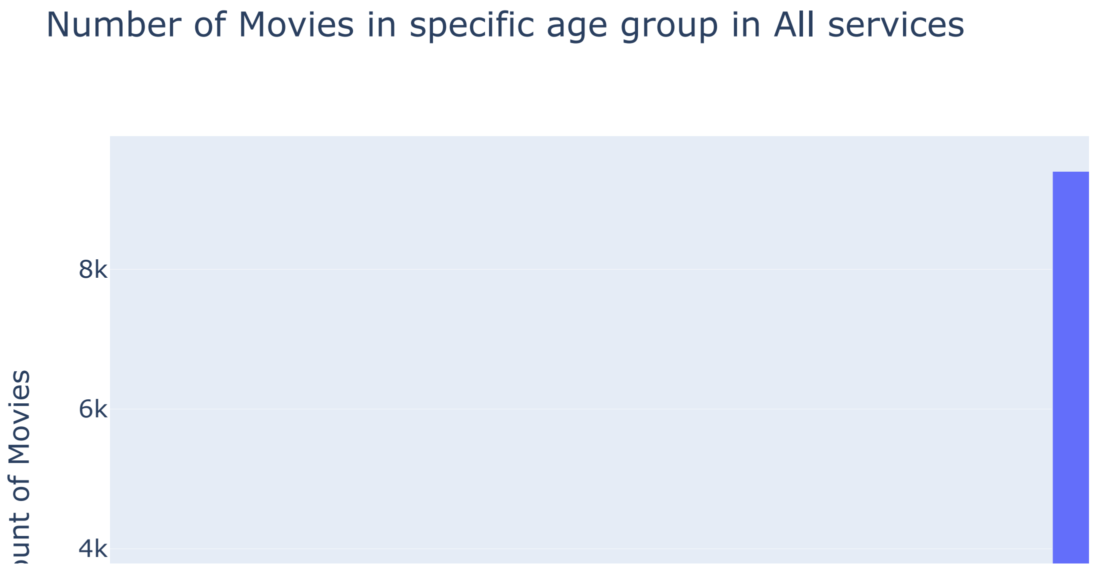


In [60]:

```
import plotly.express as px
```

In [61]:

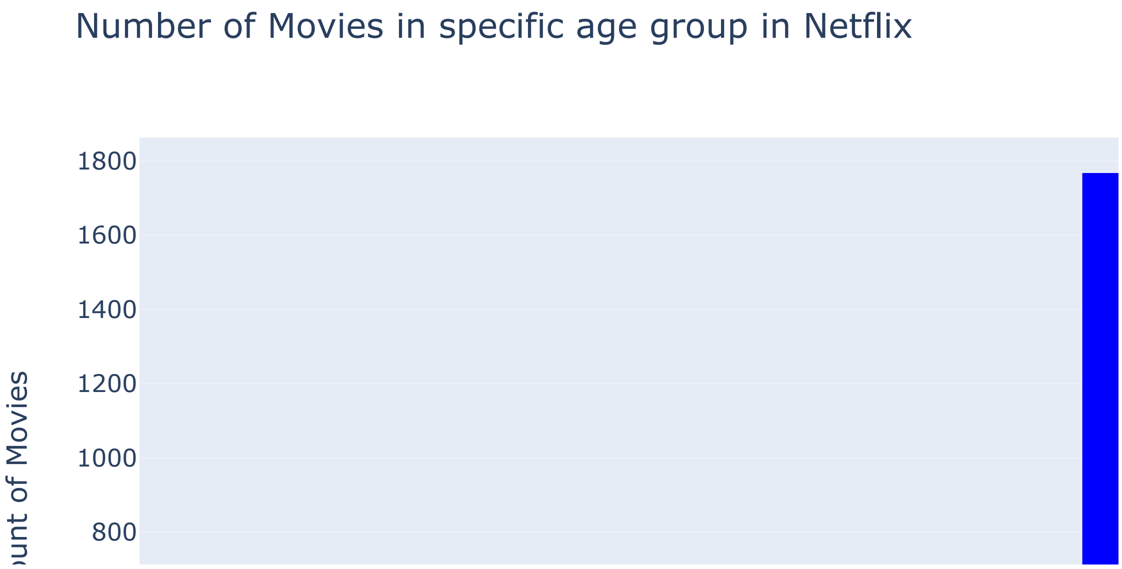
```
import plotly.express as px
#Visualizes age group movie count using Plotly Express
fig = px.bar(data['Age'].value_counts(),
             x=data['Age'].value_counts().index,
             y=data['Age'].value_counts().values,
             labels={'x': 'Age Group', 'y': 'Count of Movies'},
             title="Number of Movies in specific age group in All services")
fig.show()
```



In [ ]:

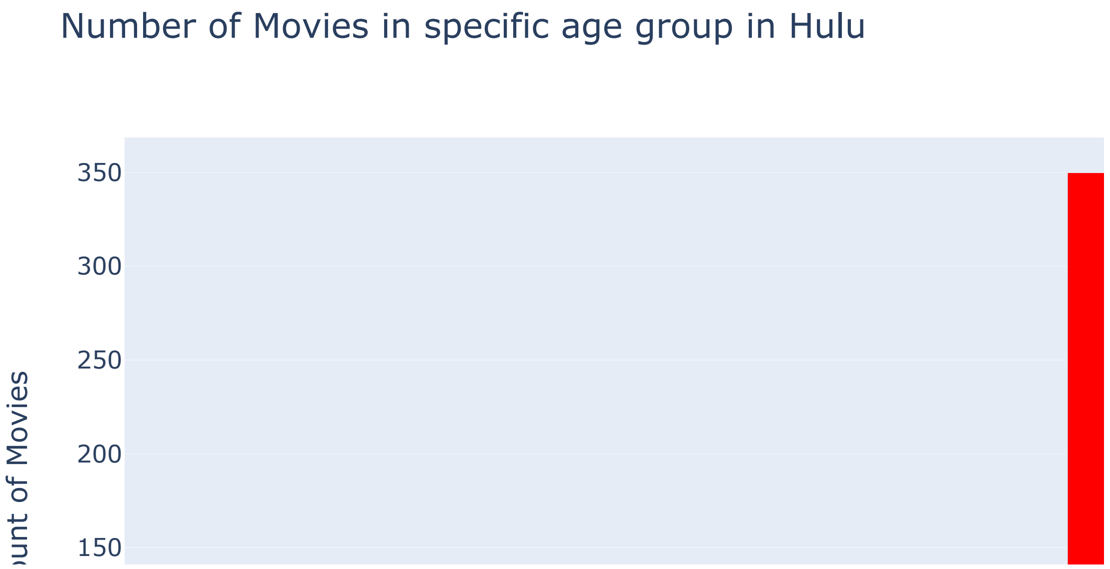
In [62]:

```
#Visualize Netflix movies count by age group using Plotly.
data_netflix = data[data['Netflix']==1]
fig = px.bar(data_netflix['Age'].value_counts(),
             x=data_netflix['Age'].value_counts().index,
             y=data_netflix['Age'].value_counts().values,
             labels={'x': 'Age Group', 'y': 'Count of Movies'},
             color_discrete_sequence=['blue'],
             title="Number of Movies in specific age group in Netflix")
fig.show()
```



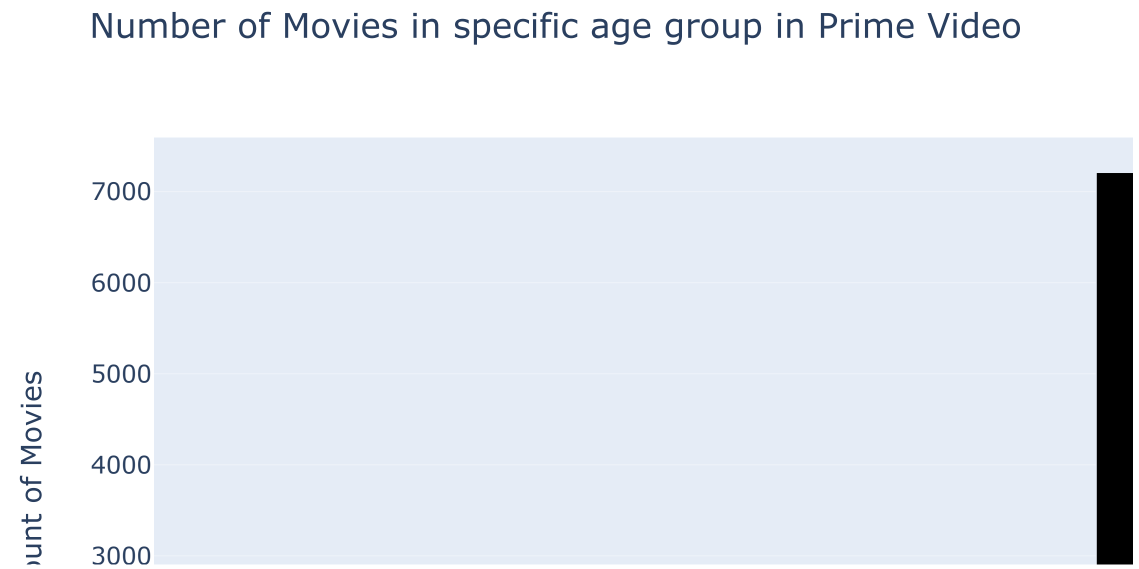
In [63]:

```
#Bar plot of movie counts in Hulu by age group.
data_hulu = data[data['Hulu']==1]
fig = px.bar(data_hulu['Age'].value_counts(),
             x=data_hulu['Age'].value_counts().index,
             y=data_hulu['Age'].value_counts().values,
             labels={'x': 'Age Group', 'y': 'Count of Movies'},
             color_discrete_sequence=['red'],
             title="Number of Movies in specific age group in Hulu")
fig.show()
```



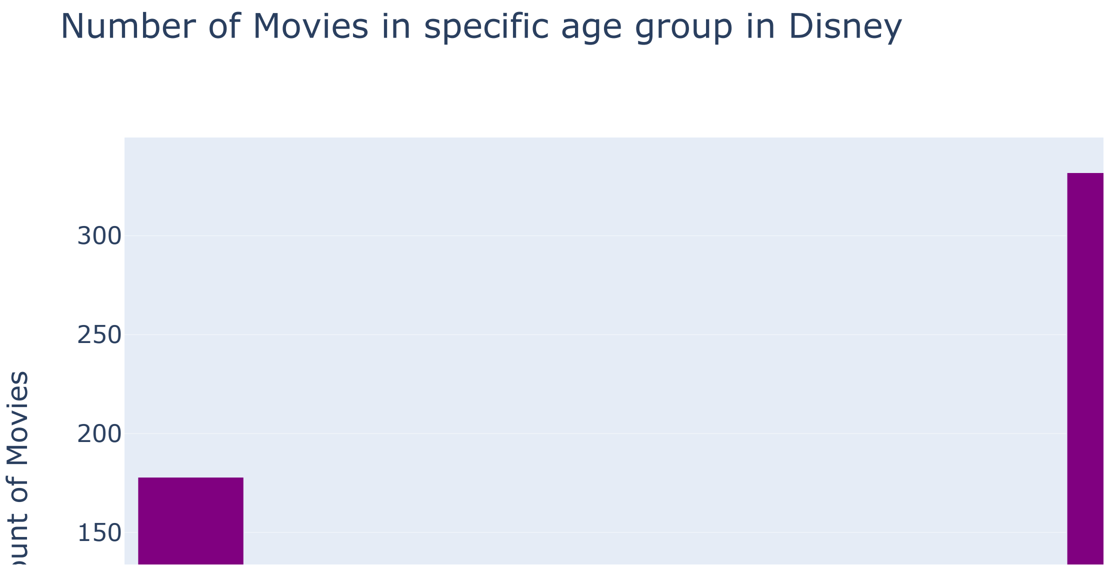
In [64]:

```
#using Plotly to visualize Prime Video movie counts.
data_prime = data[data['Prime.Video']==1]
fig = px.bar(data_prime['Age'].value_counts(),
             x=data_prime['Age'].value_counts().index,
             y=data_prime['Age'].value_counts().values,
             labels={'x': 'Age Group', 'y': 'Count of Movies'},
             color_discrete_sequence=['black'],
             title="Number of Movies in specific age group in Prime Video")
fig.show()
```



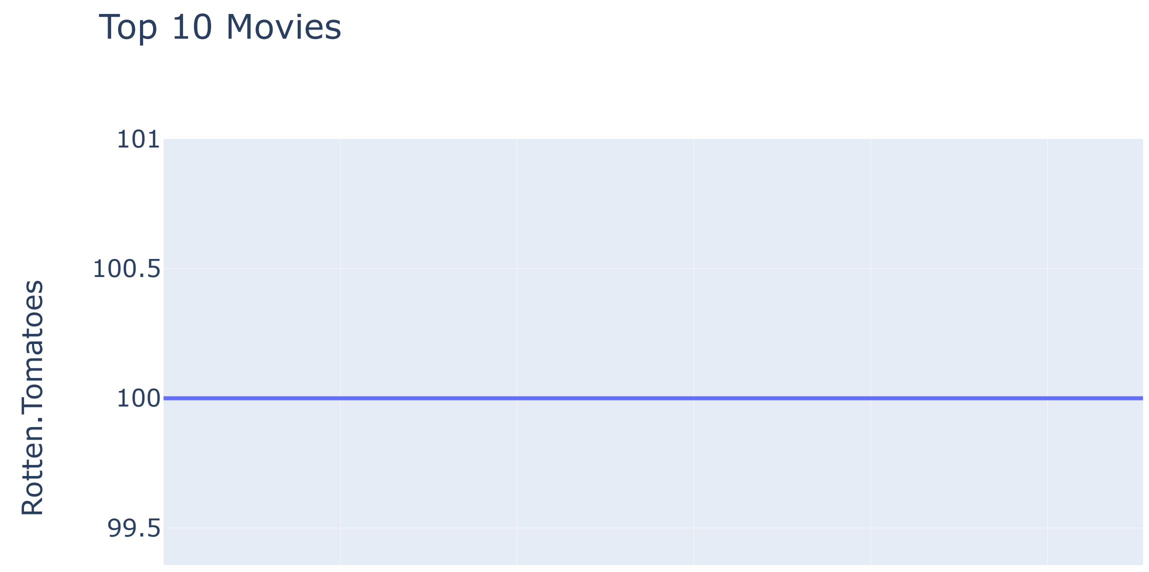
In [65]:

```
#The code uses a filtered dataset to create a bar chart showing the number of Disney mov
data_disney = data[data['Disney.']==1]
fig = px.bar(data_disney['Age'].value_counts(),
             x=data_disney['Age'].value_counts().index,
             y=data_disney['Age'].value_counts().values,
             labels={'x': 'Age Group', 'y': 'Count of Movies'},
             color_discrete_sequence=['purple'],
             title="Number of Movies in specific age group in Disney")
fig.show()
```



In [66]:

```
#Top 10 movies according to Rotten Tomatoes Ratings
rating_sorted = data.sort_values('Rotten.Tomatoes',ascending =False)
top_10_movies_by_ratings = rating_sorted.groupby(['Rotten.Tomatoes','Title']).head(10)
new_list = top_10_movies_by_ratings.head(10)
fig = px.line(new_list,x="Title",y="Rotten.Tomatoes",title="Top 10 Movies",)
fig.show()
```



In [67]:

```
#Understand what content is available on all platforms.
#Filter the data to only include movies available on all platforms
platforms = ['Netflix', 'Hulu', 'Prime.Video', 'Disney.']
available_on_all = data[data[platforms].notnull().all(axis=1)]

#Print the number of movies available on all platforms
print(f"There are {len(available_on_all)} movies available on all platforms.")
```

```
There are 15819 movies available on all platforms.
```

In [68]:

```
#Movies available in more than 1 platforms
```

In [69]:

```
#filter data with Netflix and Prime Video, output empty dataframe.
#No rows in data have both Netflix and Prime Video.
temp_data = data[data['Netflix']==1]
temp_data = temp_data[temp_data['Prime.Video']==1]
temp_data
```

Out[69]:

	Unnamed: 0	X	Title	Year	Age	IMDb	Rotten.Tomatoes	Netflix	Hulu	Prime
4	5	4	The Good, the Bad and the Ugly	1966	18	8.8	97	1	0	
6	7	6	The Pianist	2002	18	8.5	95	1	0	
11	12	11	3 Idiots	2009	13	8.4	100	1	0	
15	16	15	Once Upon a Time in the West	1968	13	8.5	95	1	0	
31	32	31	Drive	2011	18	7.8	92	1	0	
...	...	...	...	...	...	...	...	...	...	
3270	3435	3434	Contract	2008	18	4.2	65	1	0	
3273	3439	3438	Daffedar	2016	14	5.9	65	1	0	
3274	3440	3439	Hisss	2010	14	2.8	65	1	0	
3275	3441	3440	Coffee with D	2017	13	4.4	17	1	0	
3280	3446	3445	Cappuccino	2017	14	3.8	65	1	0	

336 rows × 17 columns

In [70]:

```
pip install wordcloud
```

```
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 23.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: wordcloud in c:\users\archana\anaconda3\li
b\site-packages (1.8.2.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\archana\anaconda3
\lib\site-packages (from wordcloud) (1.20.1)
Requirement already satisfied: pillow in c:\users\archana\anaconda3\lib\s
ite-packages (from wordcloud) (8.2.0)
Requirement already satisfied: matplotlib in c:\users\archana\anaconda3\l
ib\site-packages (from wordcloud) (3.3.4)
Requirement already satisfied: cycler>=0.10 in c:\users\archana\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\archana\anac
onda3\lib\site-packages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 i
n c:\users\archana\anaconda3\lib\site-packages (from matplotlib->wordclou
d) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\archana\a
naconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: six in c:\users\archana\anaconda3\lib\site
-packages (from cycler>=0.10->matplotlib->wordcloud) (1.15.0)
```

In [71]:

```
from wordcloud import WordCloud
```

In [72]:

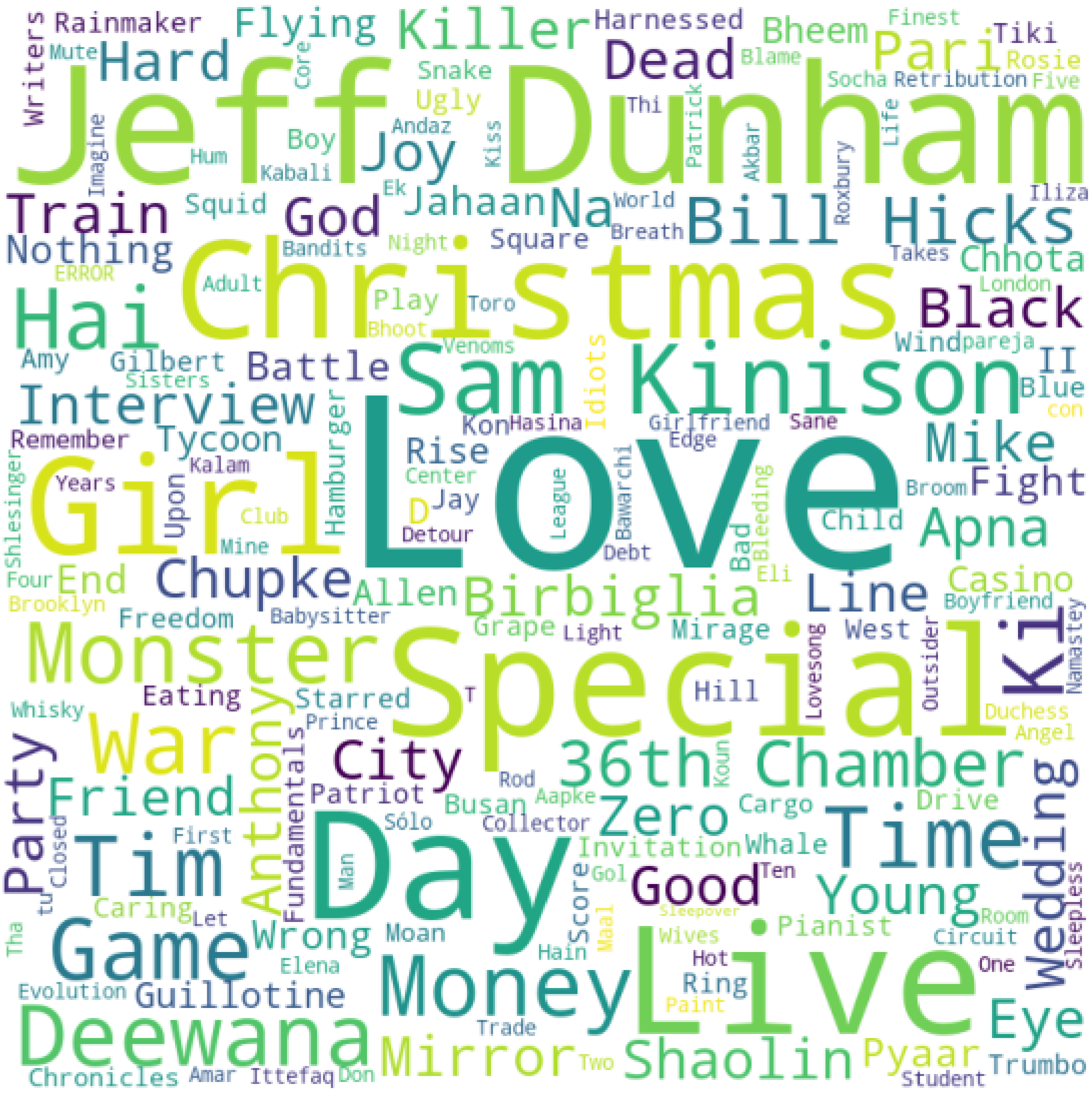
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in c:\users\archana\anaconda3\lib\site
-packages (23.1)
Collecting pip
  Downloading pip-23.1.2-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 6.9 MB/s eta 0:0
0:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1
    Uninstalling pip-23.1:
      Successfully uninstalled pip-23.1
Successfully installed pip-23.1.2
Note: you may need to restart the kernel to use updated packages.
```

In [73]:

```
#Create and display word cloud from text data.
plt.subplots(figsize = (10,10))

wordcloud = WordCloud (
                    background_color = 'white',
                    width = 720,
                    height = 720
                        ).generate(' '.join(temp_data['Title']))
plt.imshow(wordcloud) # image show
plt.axis('off') # to off the axis of x and y
plt.show()
```



In [ ]:

```

```

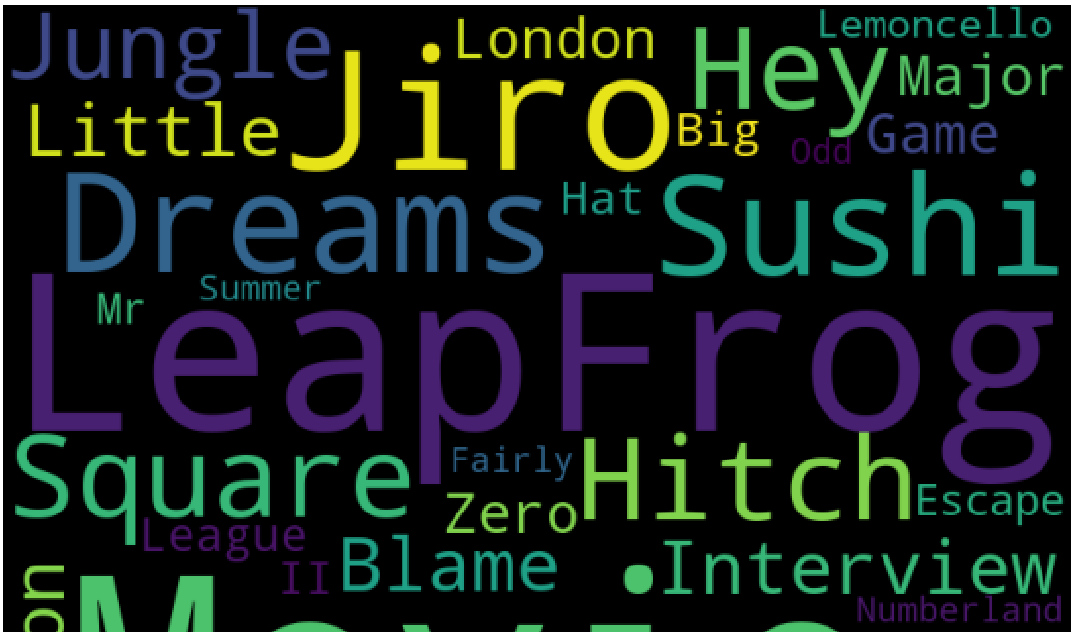
In [74]:

```
temp_data_nh = data[data['Netflix']==1]
temp_data_nh = temp_data_nh[temp_data_nh['Hulu']==1]
#list(temp_data_nh['Title'])
```

In [75]:

```
#Creates a word cloud plot from titles
plt.subplots(figsize = (10,10))

wordcloud = WordCloud (
                    background_color = 'black',
                    width = 720,
                    height = 720
                        ).generate(' '.join(temp_data_nh['Title']))
plt.imshow(wordcloud) # image show
plt.axis('off') # to off the axis of x and y
plt.show()
```



In [58]:

In [59]:

In [60]:

In [ ]:

In [ ]:

In [ ]: