```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.offline as pyo
pyo.init_notebook_mode()
import plotly.express as px
```

```
data=pd.read_csv("MoviesOnStreamingPlatforms_updated.csv")
```

```
data.head()
```

	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres
0	1	Inception	2010	13+	8.8	87%	1.0	0	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller
1	2	The Matrix	1999	18+	8.7	87%	1.0	0	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi
2	3	Avengers: Infinity War	2018	13+	8.5	84%	1.0	0	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi
3	4	Back to the Future	1985	7+	8.5	96%	1.0	0	0	0	0	Robert Zemeckis	Adventure,Comedy,Sci-Fi
4	5	The Good, the	1966	18+	8.8	97%	1.0	0	1	0	0	Sergio Leone	Western

Here 16744 represents Number of Samples and 16 represents Total Number of Features taken

```
data.shape
```

```
(16744, 16)
```

```
data.columns
```

```
Index(['ID', 'Title', 'Year', 'Age', 'IMDb', 'Rotten Tomatoes', 'Netflix',
       'Hulu', 'Prime Video', 'Disney+', 'Type', 'Directors', 'Genres',
       'Country', 'Language', 'Runtime'],
      dtype='object')
```

```
cols=data.columns.tolist()
```

```
cols
```

```
['ID',
 'Title',
 'Year',
 'Age',
 'IMDb',
 'Rotten Tomatoes',
 'Netflix',
 'Hulu',
 'Prime Video',
 'Disney+',
 'Type',
 'Directors',
 'Genres',
 'Country',
 'Language',
 'Runtime']
```

CHECKING MISSING VALUES

Python Recognizes Missing values as NaN

```
data.isna()
```

	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres	Country
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16739	False	False	False	True	False	True	False	False	False	False	False	False	False	False
16740	False	False	False	False	False	True	False	False	False	False	False	False	False	False
16741	False	False	False	True	False	True	False	False	False	False	False	False	False	False
16742	False	False	False	True	False	True	False	False	False	False	False	False	False	False
16743	False	False	False	True	True	True	False	False	False	False	False	False	False	False

16744 rows × 16 columns

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

```
ID                     0
Title                  0
Year                   0
Age                 9390
IMDb                 571
Rotten Tomatoes    11586
Netflix                0
Hulu                   0
Prime Video            0
Disney+                0
Type                   0
Directors            726
Genres               275
Country              435
Language             599
Runtime              592
dtype: int64
```

Let's Remove "+" sign attached to AGE column

```
data.dtypes
```

```
ID                   int64
Title               object
Year                 int64
Age                 object
IMDb               float64
Rotten Tomatoes     object
Netflix            float64
Hulu                 int64
Prime Video          int64
Disney+              int64
Type                 int64
Directors           object
Genres              object
Country             object
Language            object
Runtime            float64
dtype: object
```

```
data['Age']
```

```
0        13+
1        18+
2        13+
3         7+
4        18+
            ...
    16739    NaN
    16740     7+
    16741    NaN
    16742    NaN
    16743    NaN
    Name: Age, Length: 16744, dtype: object
```

```
age_map={'13+':13,'18+':18,'7+':7,'All':0,'16':16}
data["AgeCopy"] = data["Age"].map(age_map)
data["AgeCopy"]
```

```
    0        13.0
    1        18.0
    2        13.0
    3         7.0
    4        18.0
             ...
    16739     NaN
    16740     7.0
    16741     NaN
    16742     NaN
    16743     NaN
    Name: AgeCopy, Length: 16744, dtype: float64
```

```
data['Age'].unique()
```

```
    array(['13+', '18+', '7+', nan, 'all', '16+'], dtype=object)
```

```
data["Age"]
```

```
    0        13+
    1        18+
    2        13+
    3         7+
    4        18+
            ...
    16739    NaN
    16740     7+
    16741    NaN
    16742    NaN
    16743    NaN
    Name: Age, Length: 16744, dtype: object
```

Let's Remove "%" sign attached to Rotten Tomatoes column

```
data["Rotten Tomatoes"]=data["Rotten Tomatoes"].str.replace('%','')
```

```
for i in data["Rotten Tomatoes"]:
    if i==str:
        i.astype(int)
```

```
data["Rotten Tomatoes"]
```

```
    0         87
    1         87
    2         84
    3         96
    4         97
            ...
    16739    NaN
    16740    NaN
    16741    NaN
    16742    NaN
    16743    NaN
    Name: Rotten Tomatoes, Length: 16744, dtype: object
```

Visualisations

1.What is the Nummber of Movies for each group?

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

```
    18+    3474
    7+     1462
    13+    1255
```

```
all     843
16+     320
Name: Age, dtype: int64
```

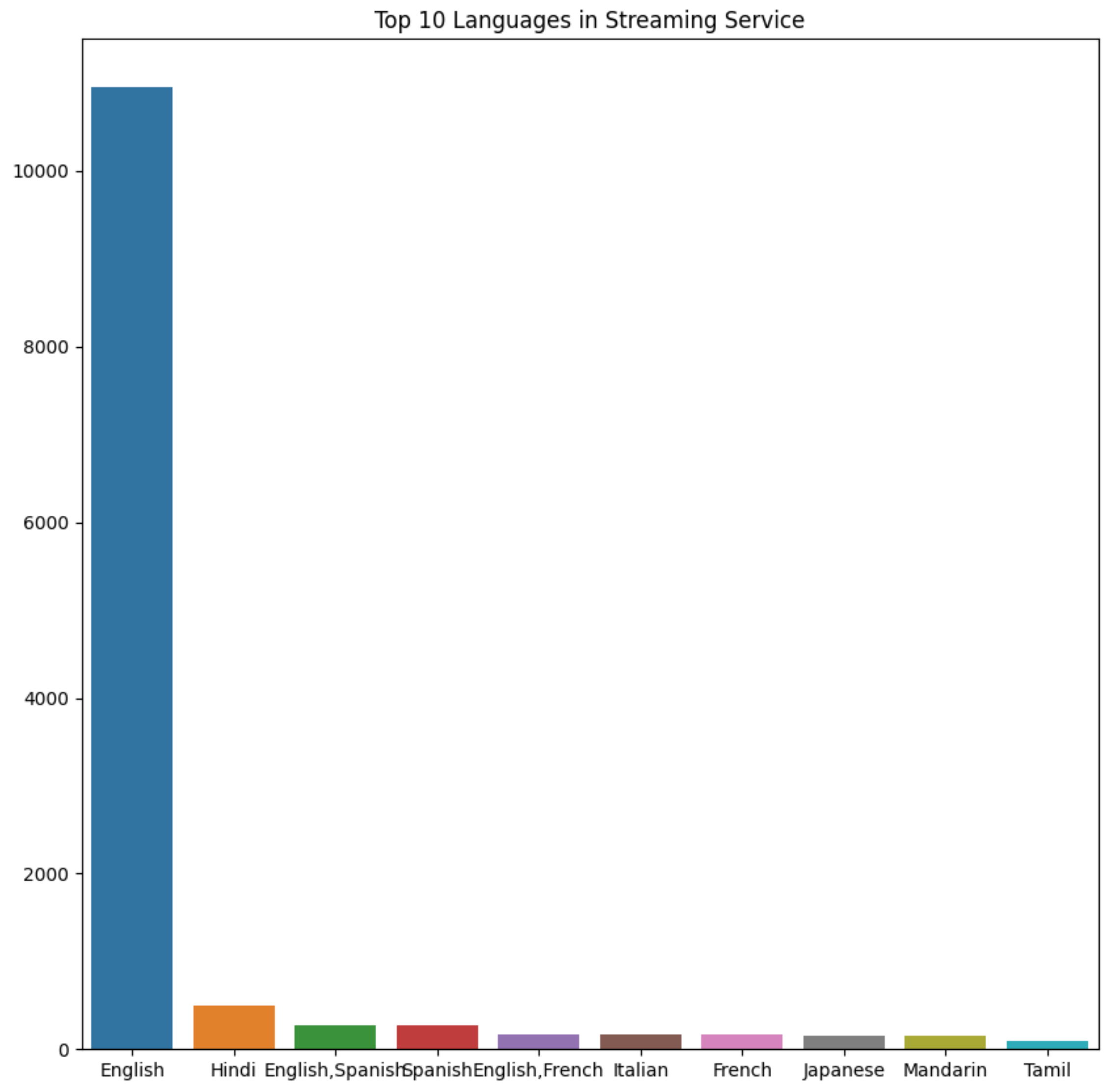
2.Top 10 Languages in Streaming Service

```
data.Language.value_counts()
```

```
English                                                                      10955
Hindi                                                                          503
English,Spanish                                                                276
Spanish                                                                        267
English,French                                                                 174
                                                                             ...
English,German,Hungarian,Romanian                                                1
English,Spanish,Chinese,Latin                                                    1
English,Danish,Malay,Dutch,Indonesian,Finnish,Luxembourgish,French Sign Language   1
Dutch,French                                                                     1
English,Algonquin                                                                1
Name: Language, Length: 1102, dtype: int64
```

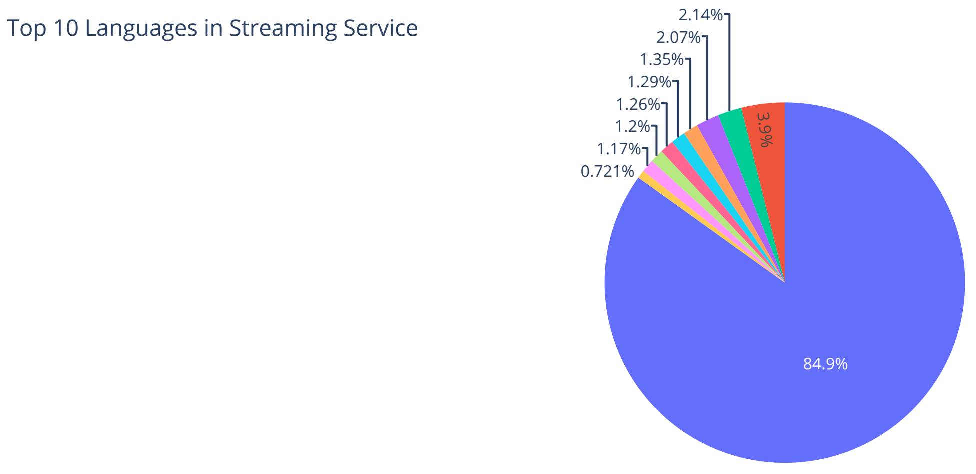
```
language=data.Language.value_counts().head(10)
plt.figure(figsize=(10,10))
plt.title('Top 10 Languages in Streaming Service')
sns.barplot(x=language.index,y=language.values)
```

```
<Axes: title={'center': 'Top 10 Languages in Streaming Service'}>
```



```
from IPython.display import HTML
import plotly.express as px
fig=px.pie(data,
          values=language.values,
          names=language.index,
          title="Top 10 Languages in Streaming Service")

HTML(fig.to_html())
```



Number of Movies in specific age group in All services

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

```
18+    3474
7+     1462
13+    1255
all     843
16+     320
Name: Age, dtype: int64
```

```
from IPython.display import HTML
import plotly.express as px


fig= px.bar(data,
            x=data["Age"].value_counts().index,
            y=data["Age"].value_counts(),
            title="Number of Movies in specific age group in All services",
            text=data["Age"].value_counts(),
            height=600)
fig.update_traces(texttemplate="%{text:.2s}",textposition="outside")

HTML(fig.to_html())
```

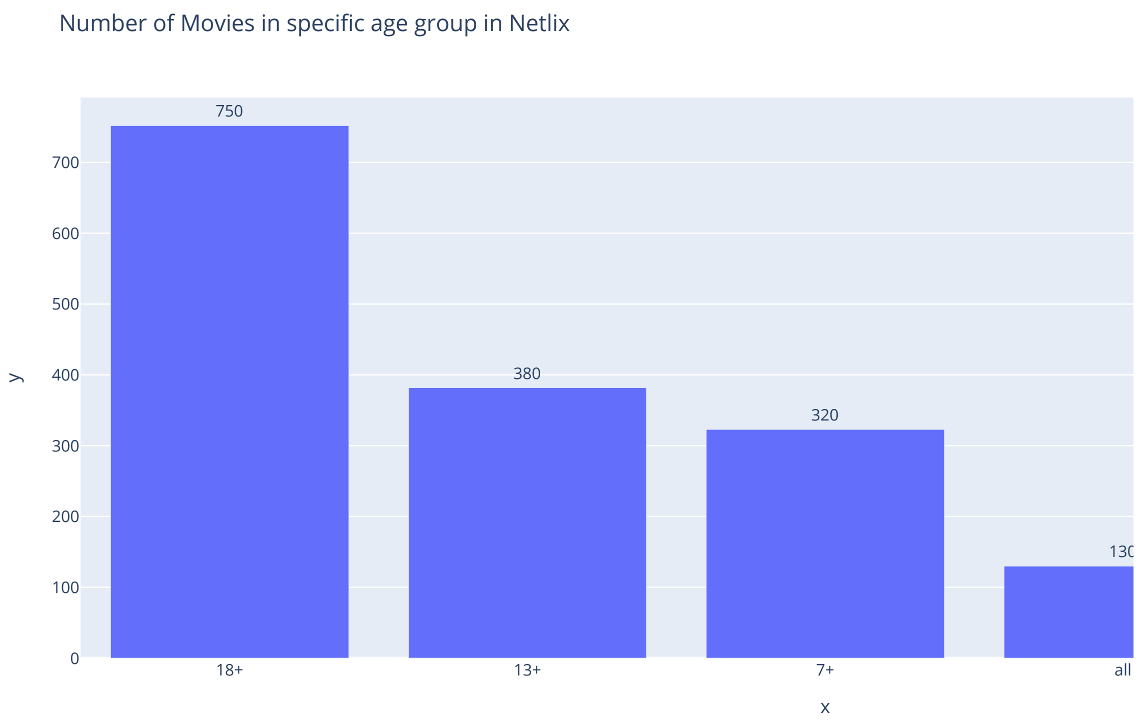
Number of Movies in specific age group in Netflix

```
from IPython.display import HTML
import plotly.express as px

netflix_data=data[data["Netflix"]==1]

fig= px.bar(data,
            x=netflix_data["Age"].value_counts().index,
            y=netflix_data["Age"].value_counts(),
            title="Number of Movies in specific age group in Netlix",
            text=netflix_data["Age"].value_counts(),
            height=600)
fig.update_traces(texttemplate="%{text:.2s}",textposition="outside")

HTML(fig.to_html())
```



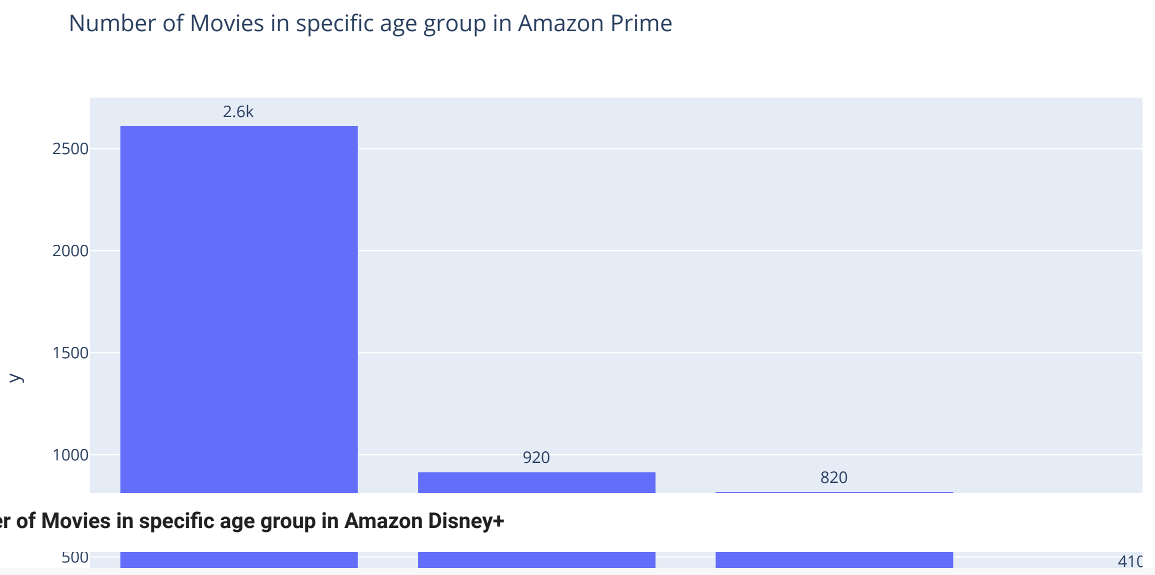
Number of Movies in specific age group in Amazon Prime Video

```
from IPython.display import HTML
import plotly.express as px

prime_data=data[data["Prime Video"]==1]

fig= px.bar(data,
            x=prime_data["Age"].value_counts().index,
            y=prime_data["Age"].value_counts(),
            title="Number of Movies in specific age group in Amazon Prime",
            text=prime_data["Age"].value_counts(),
            height=600)
fig.update_traces(texttemplate="%{text:.2s}",textposition="outside")

HTML(fig.to_html())
```



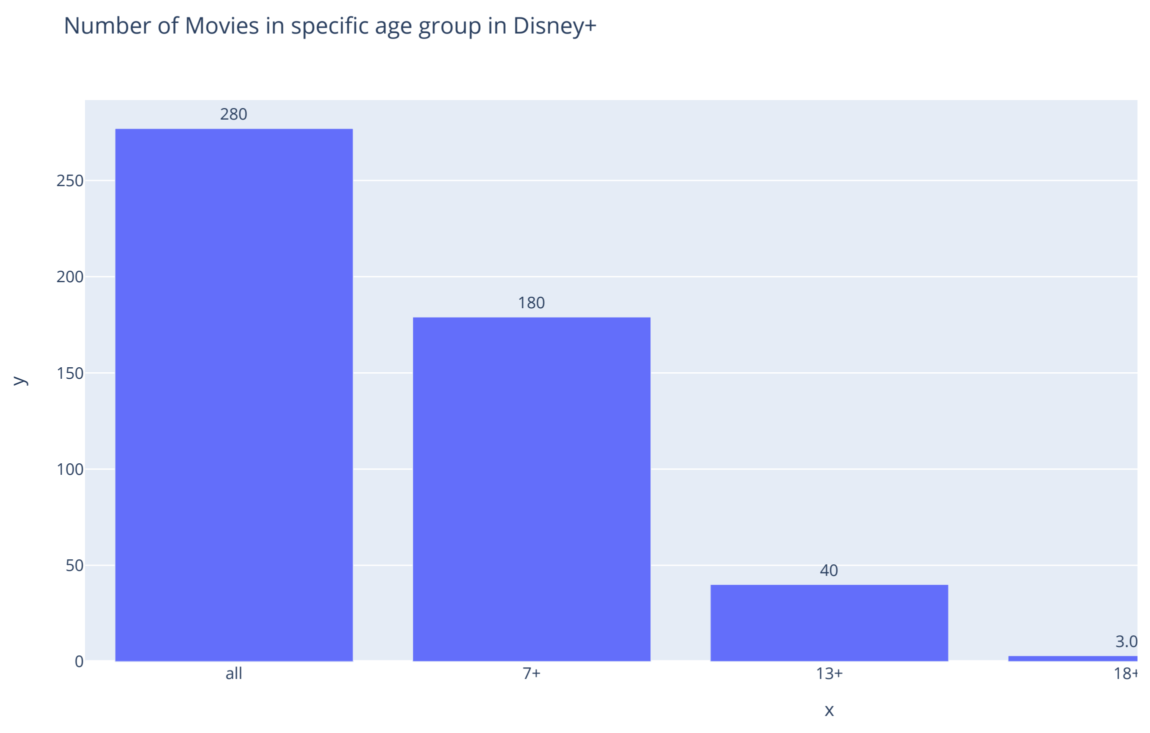
Number of Movies in specific age group in Amazon Disney+

```
from IPython.display import HTML
import plotly.express as px

Disney_data=data[data["Disney+"]==1]

fig= px.bar(data,
            x=Disney_data["Age"].value_counts().index,
            y=Disney_data["Age"].value_counts(),
            title="Number of Movies in specific age group in Disney+",
            text=Disney_data["Age"].value_counts(),
            height=600)
fig.update_traces(texttemplate="%{text:.2s}",textposition="outside")

HTML(fig.to_html())
```



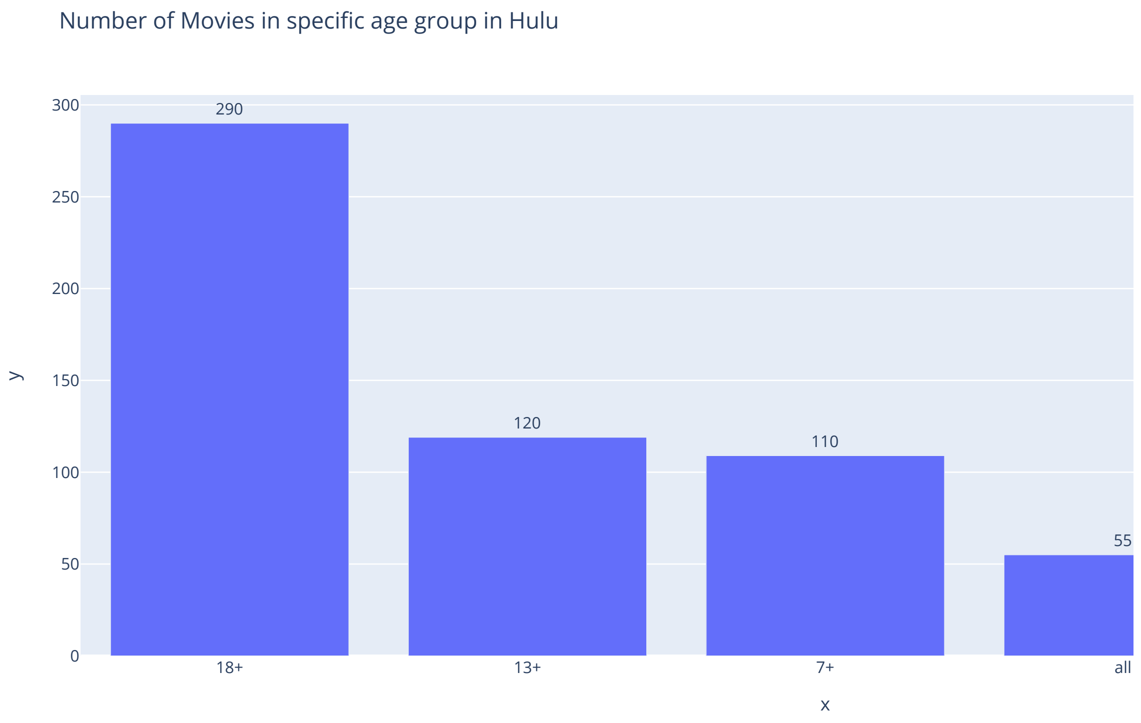
Number of Movies in specific age group in Amazon Hulu

```
from IPython.display import HTML
import plotly.express as px

Hulu_data=data[data["Hulu"]==1]

fig= px.bar(data,
            x=Hulu_data["Age"].value_counts().index,
            y=Hulu_data["Age"].value_counts(),
            title="Number of Movies in specific age group in Hulu",
            text=Hulu_data["Age"].value_counts(),
            height=600)
fig.update_traces(texttemplate="%{text:.2s}",textposition="outside")

HTML(fig.to_html())
```

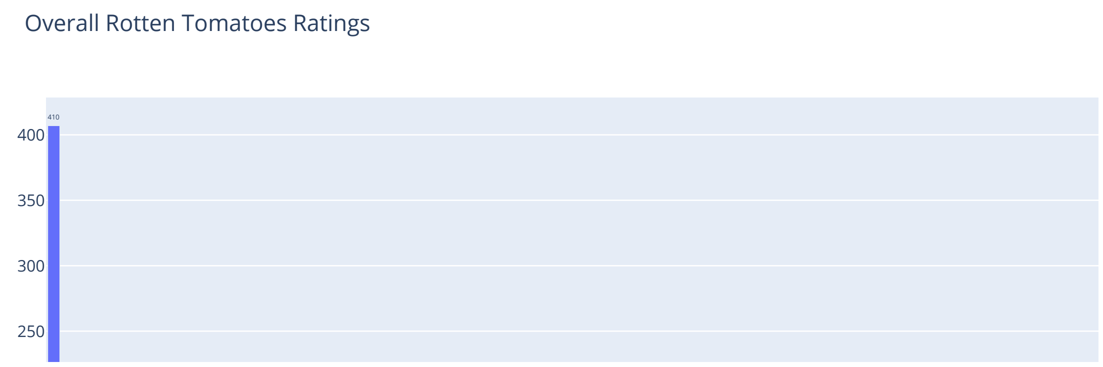


Rotten Tomatoes Score:

A Tomatometer score is calculated for a movie or TV show after it receives at least five reviews. When at least 60% of reviews for a movie or TV show are positive, a red tomato is displayed to indicate its Fresh status. Rotten Tomatoes gives films a score out of 100 based on the averaged reviews of professional film critics. If a film gets a rating of 60 or more it gets a 'fresh' red tomato on the site. Less than 60 and it gets a rotten tomato.

Rotten Tomatoes Ratings For All Services

```
from IPython.display import HTML
import plotly.express as px
Hulu_data=data [data [ 'Hulu']==1]
fig= px.bar (data,
x=data ['Rotten Tomatoes'].value_counts().index,
y=data ['Rotten Tomatoes'].value_counts(),
title="Overall Rotten Tomatoes Ratings",
text=data ['Rotten Tomatoes'].value_counts (),
height=600)
fig.update_traces (texttemplate='%{text:.2s}', textposition='outside')
HTML (fig.to_html())
```



Rotten Tomatoes Ratings For Each Services

```
netflix_data [ 'Rotten Tomatoes' ].value_counts() [0]
```

```
129
```

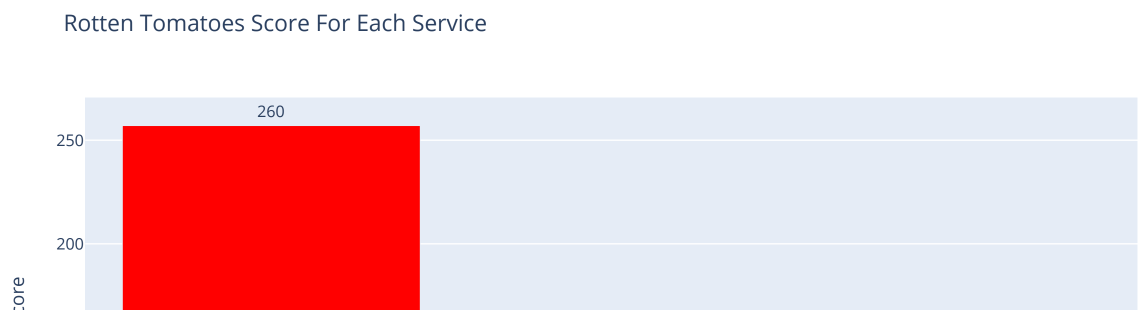
```
rt_scores=pd. DataFrame({ 'Streaming Service': [ 'Prime Video', 'Hulu', 'Disney+', 'Netflix'],
'Rotten Tomatoes Score': [ netflix_data [ 'Rotten Tomatoes' ].value_counts() [0],
prime_data[ 'Rotten Tomatoes'].value_counts() [0],
Disney_data [ 'Rotten Tomatoes' ].value_counts()[0],
Hulu_data [ 'Rotten Tomatoes'].value_counts()[0]
]})
rt_scores.head()
```

	Streaming Service	Rotten Tomatoes Score
0	Prime Video	129
1	Hulu	257
2	Disney+	19
3	Netflix	18

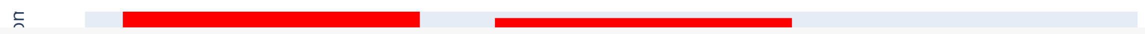
```
sort_rt_scores=rt_scores.sort_values(ascending=False,by="Rotten Tomatoes Score")
sort_rt_scores
```

	Streaming Service	Rotten Tomatoes Score
1	Hulu	257
0	Prime Video	129
2	Disney+	19
3	Netflix	18

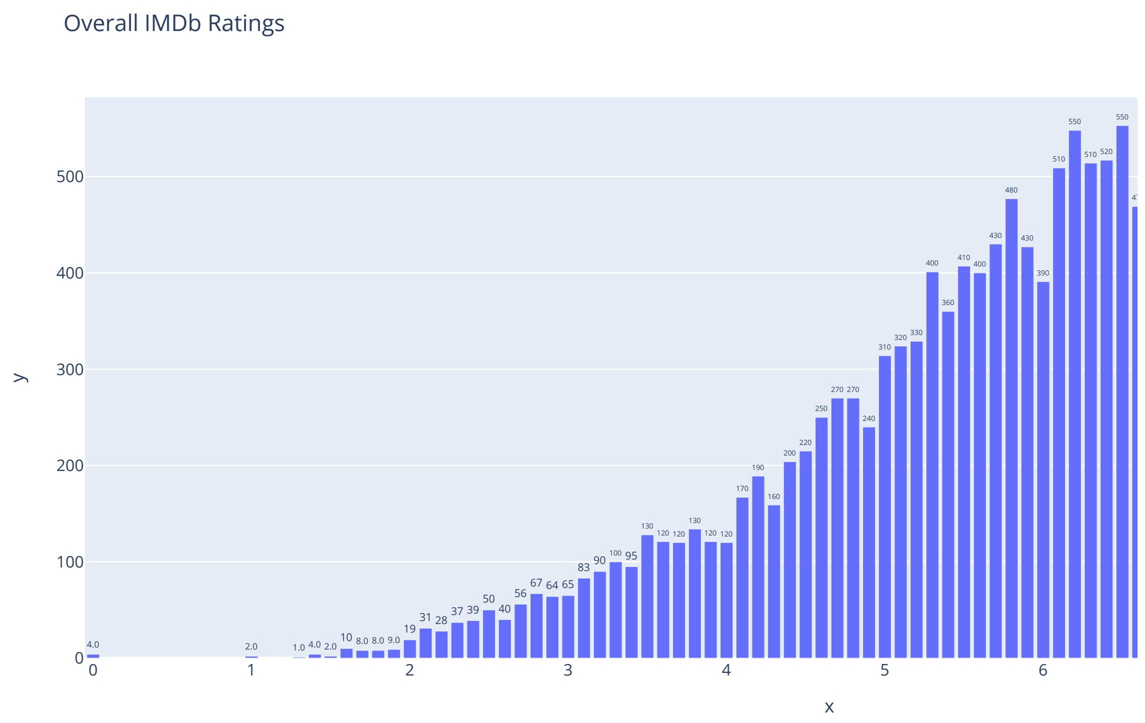
```
fig= px.bar (sort_rt_scores,
x=sort_rt_scores ['Streaming Service'],
y=sort_rt_scores ['Rotten Tomatoes Score'],
title="Rotten Tomatoes Score For Each Service",
text=sort_rt_scores ['Rotten Tomatoes Score'],
height=600)
fig.update_traces(marker_color="red",texttemplate='%{text:.2s}', textposition='outside')
HTML (fig.to_html())
```



IMDB Ratings



```
from IPython.display import HTML
import plotly.express as px
Hulu_data=data [data [ 'Hulu']==1]
fig= px.bar (data,
x=data ['IMDb'].value_counts().index,
y=data ['IMDb'].value_counts(),
title="Overall IMDb Ratings",
text=data ['IMDb'].value_counts (),
height=600)
fig.update_traces (texttemplate='%{text:.2s}', textposition='outside')
HTML (fig.to_html())
```

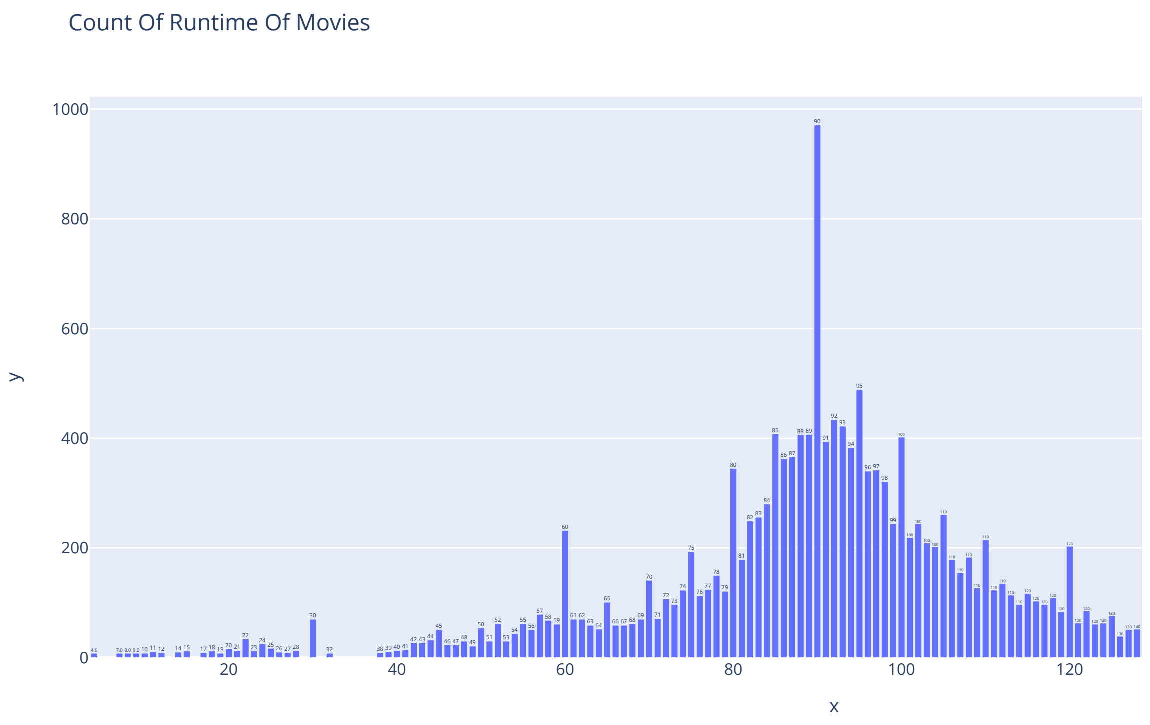


Runtimes Of Movies

```
RuntimeCount=pd.DataFrame (dict(data [ 'Runtime'].value_counts().sort_values (ascending=False) [:10]).items (),
columns=['Runtime', 'Count'])
RuntimeCount
```

	Runtime	Count
0	90.0	971
1	95.0	489
2	92.0	434
3	93.0	422
4	85.0	408

```
fig= px.bar(data,
x=RuntimeCount [ 'Runtime'],
y=RuntimeCount [ 'Count'],
title="Count Of Runtime Of Movies",
text=RuntimeCount [ 'Runtime'],
height=600)
fig.update_traces (texttemplate='%{text:.2s}', textposition='outside')
HTML (fig.to_html())
```



Exploring Genres

```
genres_=dict (data [ 'Genres'].value_counts())
```

```
genres_count = dict()
for g, count in genres_.items():
  g = g.split(",")
  for i in g:
    if i in genres_count.keys ():
      genres_count[i] = genres_count.get(i) + 1
    else:
      genres_count[i] = 1
```

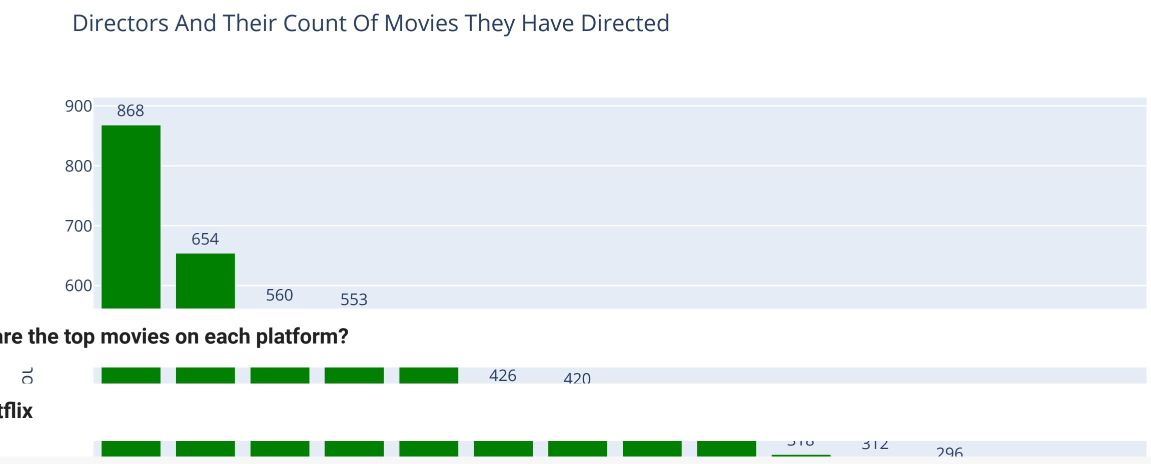
```
genres_count
```

```
{'Drama': 868,
 'Documentary': 249,
 'Comedy': 654,
 'Horror': 296,
 'Romance': 420,
 'Thriller': 467,
 'Action': 553,
 'Crime': 347,
 'Music': 171,
 'Mystery': 318,
 'Western': 168,
 'Family': 426,
 'Sci-Fi': 312,
 'Biography': 190,
 'History': 198,
 'War': 170,
 'Sport': 126,
 'Short': 141,
 'Animation': 265,
 'Adventure': 560,
 'Fantasy': 371,
 'Musical': 171,
 'News': 36,
 'Film-Noir': 25,
 'Reality-TV': 8,
 'Talk-Show': 8,
 'Game-Show': 6}
```

```
count_genres_df=pd.DataFrame (genres_count.items (), columns=['Genre', 'Count'])
count_genres_df=count_genres_df.sort_values (by='Count', ascending=False).head (20)
count_genres_df
```

	Genre	Count
0	Drama	868
2	Comedy	654
19	Adventure	560
6	Action	553
5	Thriller	467
11	Family	426
4	Romance	420
20	Fantasy	371
7	Crime	347
9	Mystery	318
12	Sci-Fi	312
3	Horror	296
18	Animation	265
1	Documentary	249
14	History	198
13	Biography	190
8	Music	171
21	Musical	171
15	War	170
10	Western	168

```
fig = px.bar (count_genres_df,
x=count_genres_df ['Genre'],
y=count_genres_df ['Count'],
title="Directors And Their Count Of Movies They Have Directed",
text=count_genres_df [ 'Count' ],
height=600)
fig.update_traces (marker_color='green', texttemplate='%{text: .2s}', textposition= 'outside')
HTML (fig.to_html())
```



What are the top movies on each platform?

On Netflix

```
data_netflix_top=netflix_data[netflix_data['IMDb']>8]
data_netflix_top=data_netflix_top[['Title', 'IMDb']].sort_values (ascending=False, by='IMDb')
data_netflix_top
```

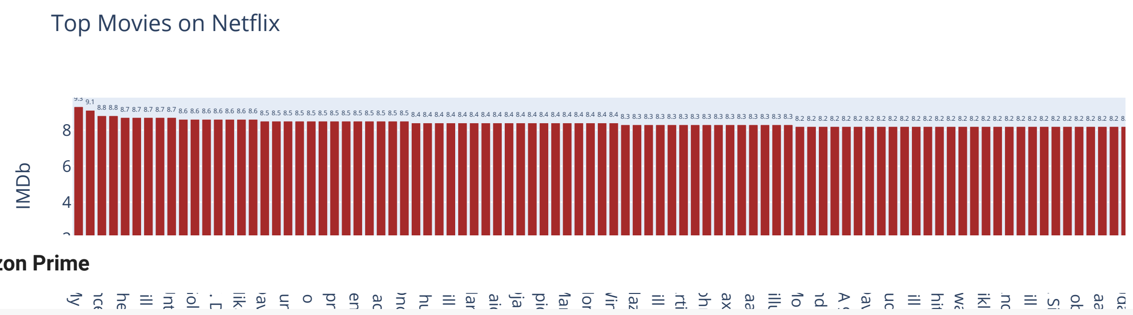
	Title	IMDb
1292	My Next Guest with David Letterman and Shah Ru...	9.3
947	Natsamrat	9.1
0	Inception	8.8
4	The Good, the Bad and the Ugly	8.8
1	The Matrix	8.7
...	...	...
1510	Uyare	8.1
133	Barfi!	8.1
123	Neon Genesis Evangelion: The End of Evangelion	8.1
1668	Sebastian Maniscalco: What's Wrong with People?	8.1
67	Blackfish	8.1

128 rows × 2 columns

```
data_netflix_top.shape
```

```
(128, 2)
```

```
fig = px.bar (data_netflix_top,
x=data_netflix_top ['Title'],
y=data_netflix_top ['IMDb'],
title="Top Movies on Netflix",
text=data_netflix_top [ 'IMDb' ],
height=600)
fig.update_traces (marker_color='brown', texttemplate='%{text: .2s}', textposition= 'outside')
HTML (fig.to_html())
```



On Amazon Prime

```
amz_top=prime_data[prime_data['IMDb']>8]
amz_top=amz_top[['Title', 'IMDb']].sort_values (ascending=False, by='IMDb')
amz_top
```

	Title	IMDb
7426	Bounty	9.3
5110	Love on a Leash	9.3
6566	Square One	9.3
6837	Steven Banks: Home Entertainment Center	9.3
7220	Down, But Not Out!	9.3
...	...	...
10119	Forgotten	8.1
7340	Alive Day Memories: Home from Iraq	8.1
2130	Tim Minchin: So F**king Rock Live	8.1
10016	Engrams	8.1
8482	Naya Daur	8.1

324 rows × 2 columns

```
amz_top.shape
```

```
(324, 2)
```

```
fig = px.bar (amz_top,
x=amz_top ['Title'],
y=amz_top ['IMDb'],
title="Top Movies on Prime",
text=amz_top [ 'IMDb' ],
height=600)
fig.update_traces (marker_color='yellow', texttemplate='%{text: .2s}', textposition= 'outside')
HTML (fig.to_html())
```

Top Movies on Prime

On Hotsar

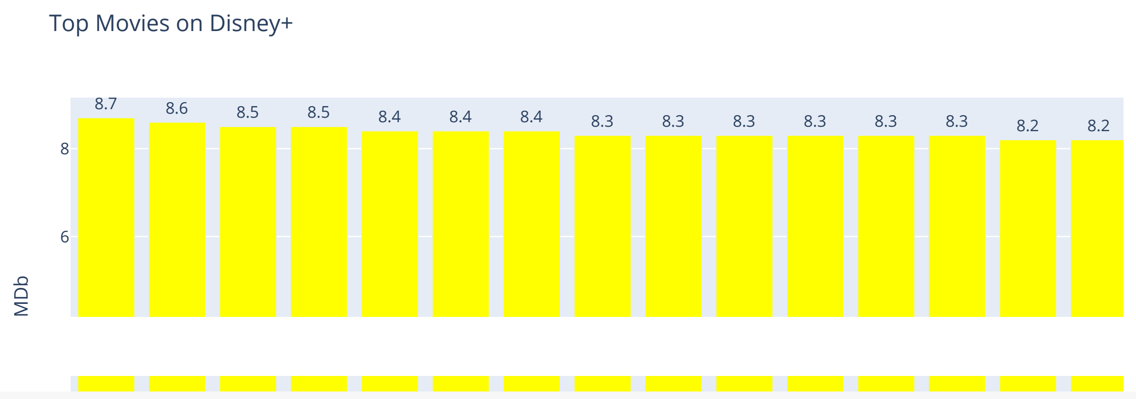
```
disney_top=Disney_data[Disney_data['IMDb']>8]
disney_top=disney_top[['Title', 'IMDb']].sort_values (ascending=False, by='IMDb')
disney_top
```

	Title	IMDb
16213	Star Wars: The Empire Strikes Back	8.7
16212	Star Wars: A New Hope	8.6
16214	The Lion King	8.5
16441	Newsies: The Broadway Musical	8.5
16221	Coco	8.4
16216	Avengers: Endgame	8.4
16217	WALL·E	8.4
16309	Before the Flood	8.3
16215	Toy Story	8.3
16582	Phineas and Ferb: Mission Marvel	8.3
5401	Empire of Dreams: The Story of the Star Wars T...	8.3
16222	Toy Story 3	8.3
16224	Star Wars: Return of the Jedi	8.3
16564	Phineas and Ferb: Star Wars	8.2
3580	Free Solo	8.2
16218	Up	8.2
16262	Togo	8.1
16229	The Princess Bride	8.1
16548	The Disney Family Singalong	8.1
16220	Finding Nemo	8.1
16693	The Flood	8.1

```
disney_top.shape
```

```
(21, 2)
```

```
fig = px.bar (disney_top,
x=disney_top ['Title'],
y=disney_top ['IMDb'],
title="Top Movies on Disney+",
text=disney_top [ 'IMDb' ],
height=600)
fig.update_traces (marker_color='yellow', texttemplate='%{text: .2s}', textposition= 'outside')
HTML (fig.to_html())
```



On Hulu

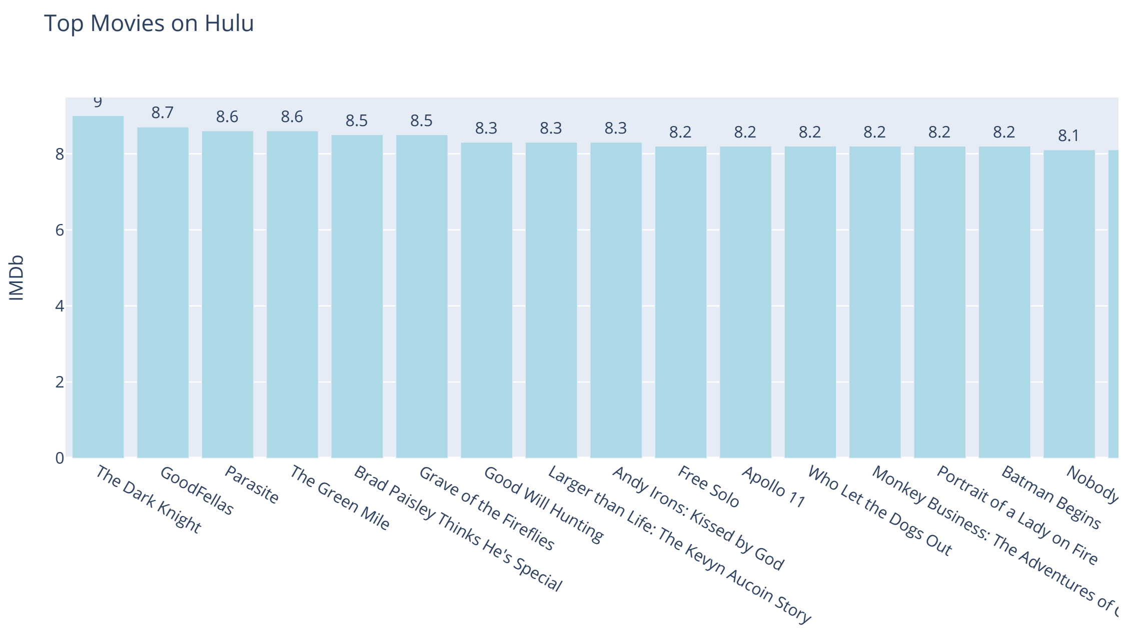
```
Hulu_top=Hulu_data[Hulu_data['IMDb']>8]
Hulu_top=Hulu_top[['Title', 'IMDb']].sort_values (ascending=False, by='IMDb')
Hulu_top
```

	Title	IMDb
3560	The Dark Knight	9.0
3561	GoodFellas	8.7
3562	Parasite	8.6
3564	The Green Mile	8.6
4283	Brad Paisley Thinks He's Special	8.5
3566	Grave of the Fireflies	8.5
3563	Good Will Hunting	8.3
3890	Larger than Life: The Kevyn Aucoin Story	8.3
3742	Andy Irons: Kissed by God	8.3
3580	Free Solo	8.2
3590	Apollo 11	8.2
4077	Who Let the Dogs Out	8.2
4325	Monkey Business: The Adventures of Curious Geo...	8.2
3577	Portrait of a Lady on Fire	8.2
3565	Batman Begins	8.2
3618	Nobody Knows	8.1
3625	Minding the Gap	8.1
3637	The Biggest Little Farm	8.1
3648	Turtles Can Fly	8.1
148	The Square	8.1
3567	Kill Bill: Vol. 1	8.1
4230	Beers of Joy	8.1
67	Blackfish	8.1

```
Hulu_top.shape
```

```
(23, 2)
```

```
fig = px.bar (Hulu_top,
x=Hulu_top ['Title'],
y=Hulu_top ['IMDb'],
title="Top Movies on Hulu",
text=Hulu_top [ 'IMDb' ],
height=600)
fig.update_traces (marker_color='lightblue', texttemplate='%{text: .2s}', textposition= 'outside')
HTML (fig.to_html())
```



Best Streaming Service According to our analysis :

```
#No.of Movies with more than 8.0 Rating in IMDB


#Netflix = 128
#Amazon = 324
#Disney+ = 21
#Hulu = 23

#Since Amazon Has Highest Number of movies with higher rating. Amazon is the best streaming service
```