TMDb Movie Data Analysis and Building a Movie Recommendation System

- # Part 2: Exploratory Data Analysis/Visualization

In this analysis, we will explore the now-cleaned dataset, and try to gain some knowledge/insights about the movie industry, the trends/changes in it, what makes movies popular, among other things.

- **id:** The ID of the movie (clear/unique identifier).
- **title:** The Official Title of the movie.
- **tagline:** The tagline of the movie.
- **release_date:** Theatrical Release Date of the movie.
- **genres:** Genres associated with the movie.
- **belongs_to_collection:** Gives information on the movie series/franchise the particular film belongs to.
- **original_language:** The language in which the movie was originally shot in.
- **budget_musd:** The budget of the movie in million dollars.
- **revenue_musd:** The total revenue of the movie in million dollars.
- **production_companies:** Production companies involved with the making of the movie.
- **production_countries:** Countries where the movie was shot/produced in.
- **vote_count:** The number of votes by users, as counted by TMDB.
- **vote_average:** The average rating of the movie.
- **popularity:** The Popularity Score assigned by TMDB.
- **runtime:** The runtime of the movie in minutes.
- **overview:** A brief blurb of the movie.
- **spoken_languages:** Spoken languages in the film.
- **poster_path:** The URL of the poster image.
- **cast:** (Main) Actors appearing in the movie.
- **cast_size:** number of Actors appearing in the movie.
- **director:** Director of the movie.
- **crew_size:** Size of the film crew (incl. director, excl. actors).

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from IPython.display import HTML
from statsmodels.graphics.boxplots import violinplot
```

```
pd.options.display.max_columns = 30
pd.options.display.float_format = '{:.2f}'.format
```

Loading the data

In [2]:
```
df = pd.read_csv('movies_complete.csv')
df.head()
```

Out[2]:

	index	belongs_to_collection	budget_musd	genres	id	original_language	overview
0	0	Blondie Collection	NaN	Comedy	3924	en	Blondie and Dagwood are about to celebrate the...
1	1	NaN	NaN	Adventure	6124	de	Der Mann ohne Namen is a German adventure movi...
2	2	NaN	NaN	Drama|Romance	8773	fr	Love at Twenty unites five directors from five...
3	3	New World Disorder	NaN	NaN	25449	en	Gee Atherton ripping the Worlds course the day...
4	4	NaN	NaN	Family	31975	en	Elmo is making a very, very super special surp...

In [3]:
```
df.columns
```

Out[3]:
```
Index(['index', 'belongs_to_collection', 'budget_musd', 'genres', 'id',
       'original_language', 'overview', 'popularity', 'poster_path',
       'production_companies', 'production_countries', 'release_date',
       'revenue_musd', 'runtime', 'spoken_languages', 'tagline', 'title',
       'vote_average', 'vote_count', 'year', 'html', 'cast_names',
       'crew_names', 'director'],
      dtype='object')
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 578044 entries, 0 to 578043
Data columns (total 24 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   index                   578044 non-null  int64  
 1   belongs_to_collection   15891 non-null   object 
 2   budget_musd             24897 non-null   float64
 3   genres                  425057 non-null  object 
 4   id                      578044 non-null  int64  
 5   original_language       578044 non-null  object 
 6   overview                496805 non-null  object 
 7   popularity              578044 non-null  float64
 8   poster_path             420708 non-null  object 
 9   production_companies    273625 non-null  object 
 10  production_countries    377206 non-null  object 
 11  release_date            556155 non-null  object 
 12  revenue_musd            13607 non-null   float64
 13  runtime                 474096 non-null  float64
 14  spoken_languages        370286 non-null  object 
 15  tagline                 90470 non-null   object 
 16  title                   578044 non-null  object 
 17  vote_average            256163 non-null  float64
 18  vote_count              578044 non-null  int64  
 19  year                    578044 non-null  object 
 20  html                    420708 non-null  object 
 21  cast_names              413841 non-null  object 
 22  crew_names              502382 non-null  object 
 23  director                494305 non-null  object 
dtypes: float64(5), int64(3), object(16)
memory usage: 105.8+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	index	budget_musd	id	popularity	revenue_musd	runtime	vote_average	v
count	578044.00	24897.00	578044.00	578044.00	13607.00	474096.00	256163.00	
mean	315812.41	10.34	453308.18	1.80	48.40	278.87	6.07	
std	187057.84	41.29	250687.94	15.98	132.40	103408.88	1.77	
min	0.00	0.00	2.00	0.60	0.00	1.00	0.00	
25%	150207.75	0.03	259484.75	0.60	0.60	19.00	5.00	
50%	314801.50	0.81	457508.50	0.60	5.90	75.00	6.00	
75%	475333.25	7.30	664758.25	1.20	34.26	95.00	7.00	
max	650254.00	5000.00	890932.00	6121.12	2847.25	50505050.00	10.00	

Distributions of the Numeric Attributes

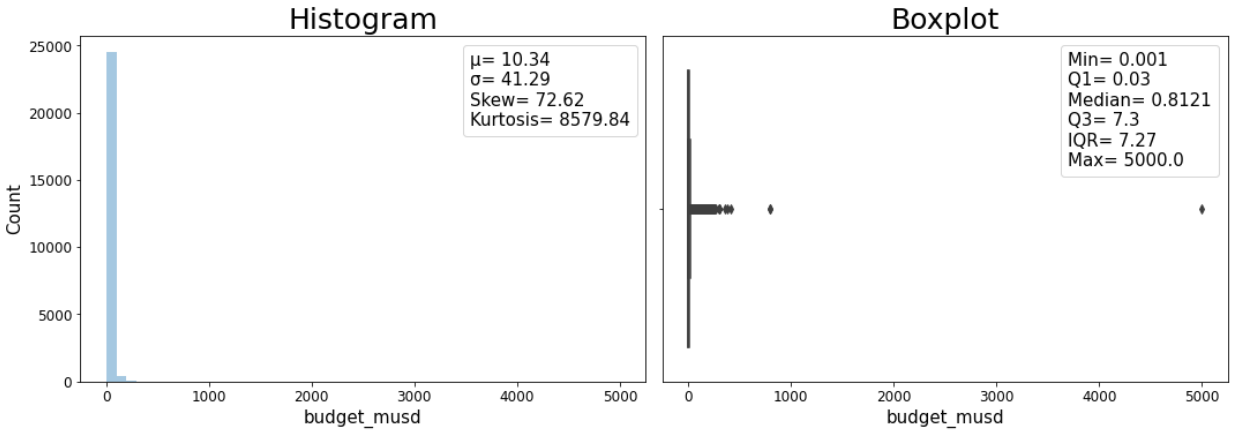
In this section, we will look at the histograms and boxplots for each appropriate numeric attribute, and remove any outliers if necessary.

In [6]:
```
# Creating a function to plot the histogram and boxplot distributions of a num
def hist_boxplot(column, title_insert=None):

    mean = np.mean(df[column])
    median = np.nanmedian(df[column])
    st_dev = np.std(df[column])
    min_ = np.min(df[column])
    max_ = np.max(df[column])
    q1 = np.nanquantile(df[column], 0.25)
    q3 = np.nanquantile(df[column], 0.75)
    iqr = q3 - q1
    skew = stats.skew(df[column], nan_policy='omit')
    kurtosis = stats.kurtosis(df[column], nan_policy='omit')


    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,6))
    if title_insert:
        fig.suptitle(f'Distribution of {title_insert}', fontsize=35, fontweigh
    else:
        fig.suptitle(f'Distribution of "{column}"', fontsize=35, fontweight='s

    sns.distplot(df[column], ax=ax1, kde=False)
    ax1.set_title('Histogram', fontsize=25)
    ax1.legend([f"μ= {np.round(mean,2)}\nσ= {np.round(st_dev,2)}\nSkew= {np.ro
    ax1.tick_params(labelsize=12)
    ax1.set_xlabel(column, fontsize=15)
    ax1.set_ylabel('Count', fontsize=15)

    sns.boxplot(df[column], ax=ax2)
    ax2.set_title('Boxplot', fontsize=25)
    ax2.legend([f"Min= {np.round(min_,4)}\nQ1= {np.round(q1,4)}\nMedian= {np.r
    ax2.tick_params(labelsize=12)
    ax2.set_xlabel(column, fontsize=15)
    fig.tight_layout()
    return plt.show()
```

In [10]:
```
# Creating a function to replace outliers in the dataset with NaN
def remove_outliers(column, limit):
    df.loc[df[column] >= limit, column] = np.nan


# Creating a function to display rows above a certain attribute threshold
def show_outliers(column, limit):
    return HTML(df.loc[df[column] > limit, ['html', 'id', 'title', column]].so
```

- budget_musd

In [11]:
```
hist_boxplot('budget_musd')
```



• The distribution of the "budget_musd" column is heavily right-skewed due to the presence of outliers.

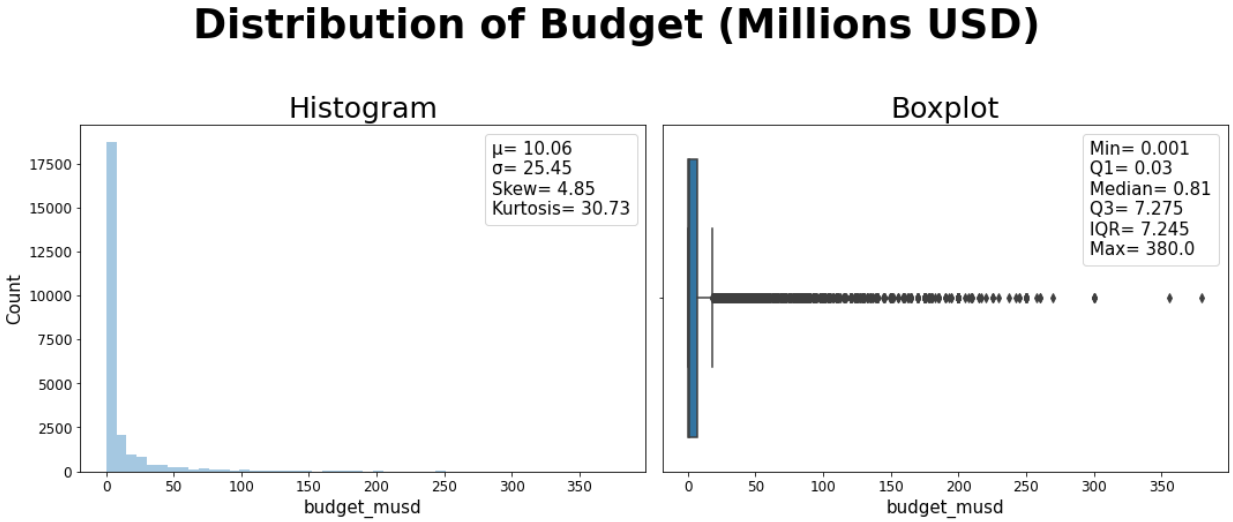
In [12]:
```
show_outliers('budget_musd', 300)
```

Out[12]:

	html	id	title	budget_musd
498231	NaN	761770	Humor Us! Presents: SOCIAL DISSOCIATION	5000.00
489360	NaN	747982	Smiles For Miles	800.00
495549	NaN	757658	Ex-Wife	800.00
574812		886156	Wake Me Up When September Ends	420.00
1243		1865	Pirates of the Caribbean: On Stranger Tides	380.00
175468		299534	Avengers: Endgame	356.00

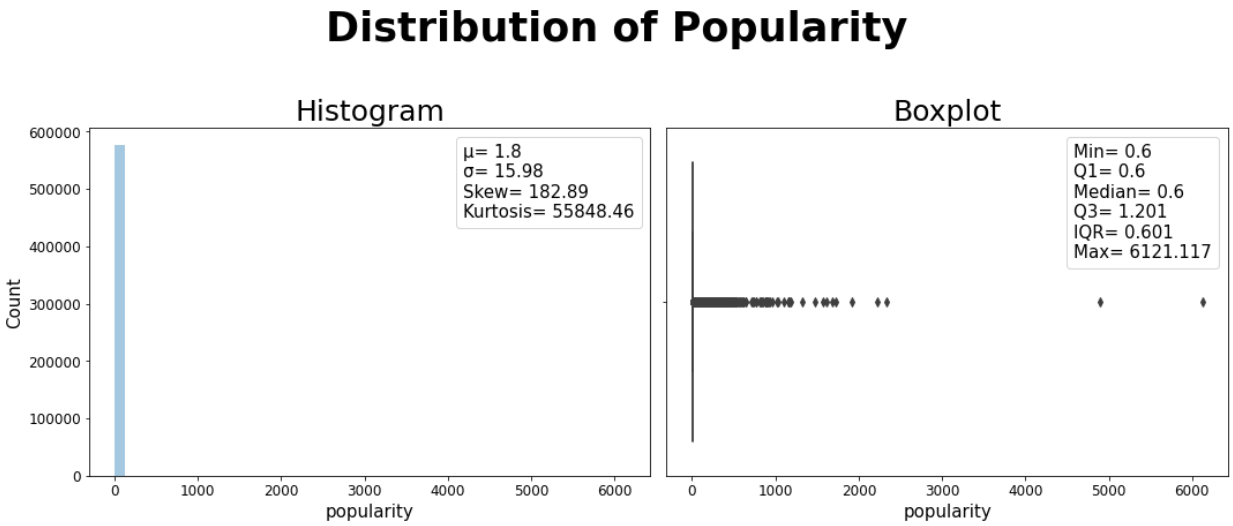
In [13]:
```
# Budgets above 420 million USD will be set to NaN
remove_outliers('budget_musd', 400)
```

In [14]:
```
hist_boxplot('budget_musd', 'Budget (Millions USD)')
```



- popularity

In [15]:
```
hist_boxplot('popularity', 'Popularity')
```



- The distribution of the "popularity" column is also heavily right-skewed due to the presence of outliers.

In [16]:
```
show_outliers('popularity', 2000)
```

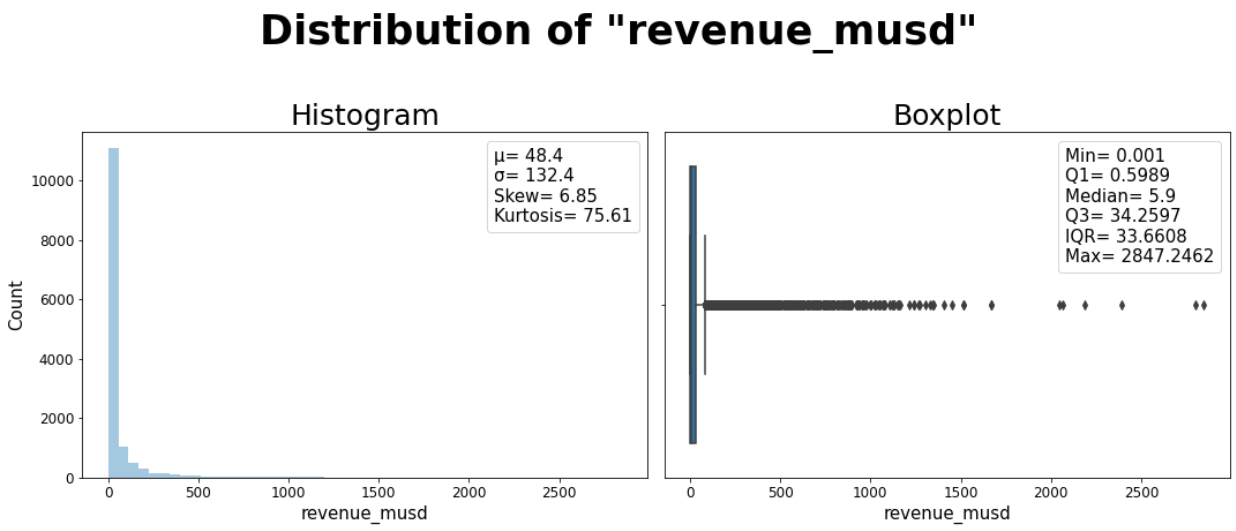
Out[16]:

	html	id	title	popularity
376183		580489	Venom: Let There Be Carnage	6121.12
275348		438631	Dune	4893.62
357053		550988	Free Guy	2334.62
396885		610253	Halloween Kills	2223.22

• All of the outliers seem to indicate really successful movies rather than bad data. Outliers will not be removed in this case.

• revenue_musd

In [17]:
```
hist_boxplot('revenue_musd')
```



• The distribution of the "revenue_musd" column is also heavily right-skewed due to the presence of outliers.

In [18]:
```
show_outliers('revenue_musd', 1500)
```

Out[18]:

	html	id	title	revenue_musd
12036		19995	Avatar	2847.25
175468		299534	Avengers: Endgame	2797.80
560181	NaN	860818	Duckava: The movie	2389.34
476		597	Titanic	2187.46
87078		140607	Star Wars: The Force Awakens	2068.22
175470		299536	Avengers: Infinity War	2046.24
84741		135397	Jurassic World	1671.71
262185		420818	The Lion King	1667.64
14385		24428	The Avengers	1518.82

	html	id	title	revenue_musd
99025		168259	Furious 7	1515.05

• Same as with the "popularity" attribute, outliers here seem appropriate.

In [28]:
```
df[df.title == 'Duckava: The movie']
```

Out[28]:

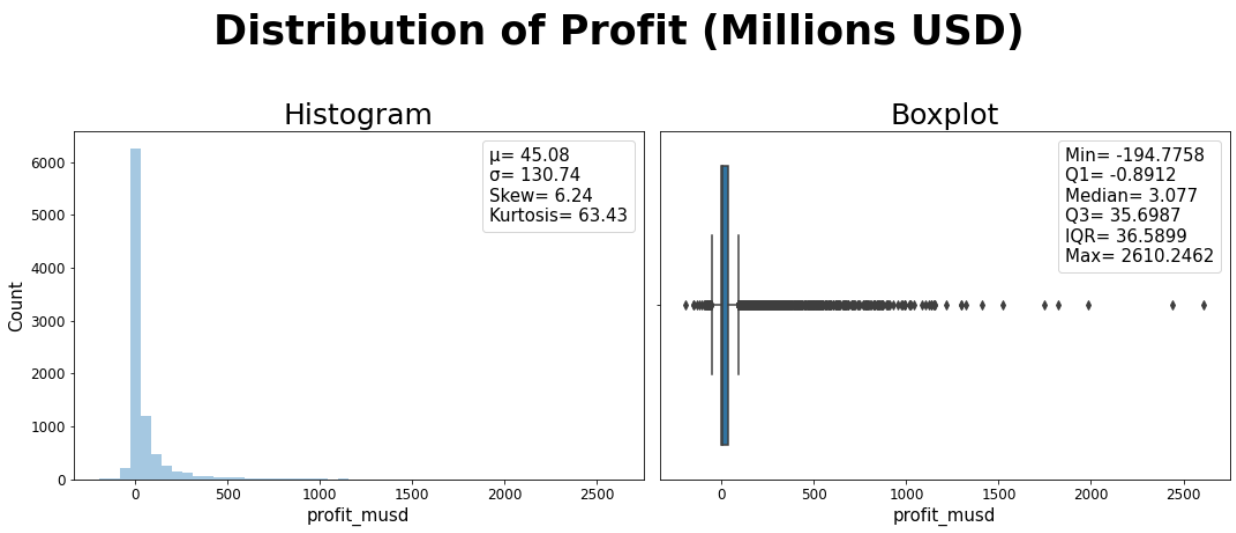
	index	belongs_to_collection	budget_musd	genres	id	original_language	overview
560181	627620	NaN	29.39	NaN	860818	en	She is the awesomest girl in the world and bet...

In [29]:
```
df.drop(df[df.title == 'Duckava: The movie'].index[0], axis=0, inplace=True)
```

• Profit

In [30]:
```
# Creating profit and return on investment columns for the dataset
df["profit_musd"] = df.revenue_musd.sub(df.budget_musd)
df["return_musd"] = df.revenue_musd.div(df.budget_musd)
```

In [31]:
```
hist_boxplot('profit_musd', 'Profit (Millions USD)')
```



In [34]:
```
show_outliers('profit_musd', 1600)
```

Out[34]:

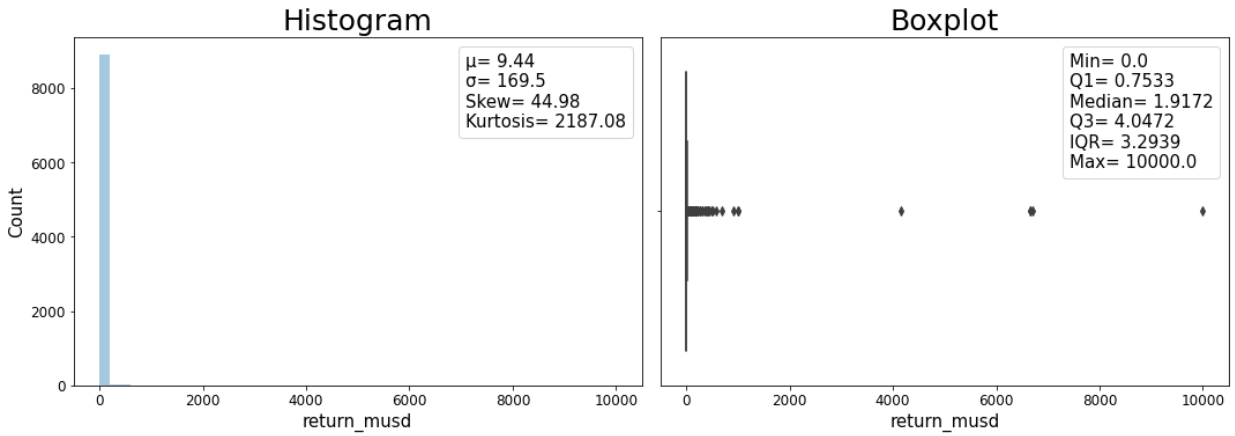
	html	id	title	profit_musd
12036		19995	Avatar	2610.25
175468		299534	Avengers: Endgame	2441.80
476		597	Titanic	1987.46
87078		140607	Star Wars: The Force Awakens	1823.22
175470		299536	Avengers: Infinity War	1746.24

- Nothing out of place with "profit_musd" as well.

- Return on Investment

In [35]:
```
hist_boxplot('return_musd', 'Return on Investment (Millions USD)')
```

Distribution of Return on Investment (Millions USD)



In [36]:
```
show_outliers('return_musd', 4000)
```

Out[36]:

	html	id	title	return_musd
391237	NaN	602324	Jeannette Sousa	10000.00
313477	NaN	491067	BURNER	6700.00
325062		506972	Khaltoor	6666.67
330141		513434	One Cut of the Dead	6666.67
1791		2667	The Blair Witch Project	4143.98

In [38]:
```
df[df.title.isin(['Jeannette Sousa', 'BURNER'])]
```

Out[38]:

	index	belongs_to_collection	budget_musd	genres	id	original_language	ov
313477	341089	NaN	0.00	NaN	491067	en	NEIGHBOF BRAWL L SAM \ BR
391237	428082	NaN	0.00	NaN	602324	en	Jeannette is an actre ex

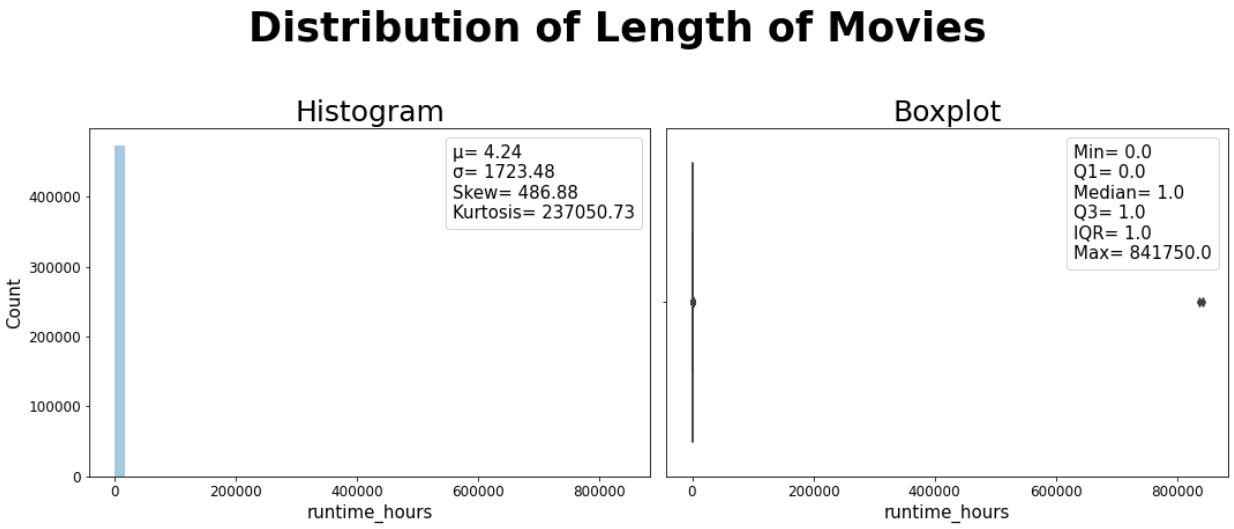
In [43]:
```
# Removing the invalid entries
df.drop(df[df.title.isin(['Jeannette Sousa', 'BURNER'])].index, axis=0, inplac
```

- runtime

In order to have a better understanding of the distribution of "runtime", we will create a new attribute by changing its scale from minutes to hours.

In [44]:
```
df['runtime_hours'] = np.floor(df.runtime / 60)
```

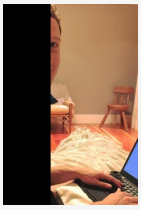
In [45]:
```
hist_boxplot('runtime_hours', 'Length of Movies')
```



- The distribution of the "runtime" column is also heavily right-skewed due to the presence of outliers.

In [46]:
```
show_outliers('runtime', 60000)
```

Out[46]:

	html	id	title	runtime
431184		660968	Mr Marcus FriendPop Facebook Adventures	50505050.00
431185		660969	Mr Marcus FriendBoop Facebook Adventures	50189181.00

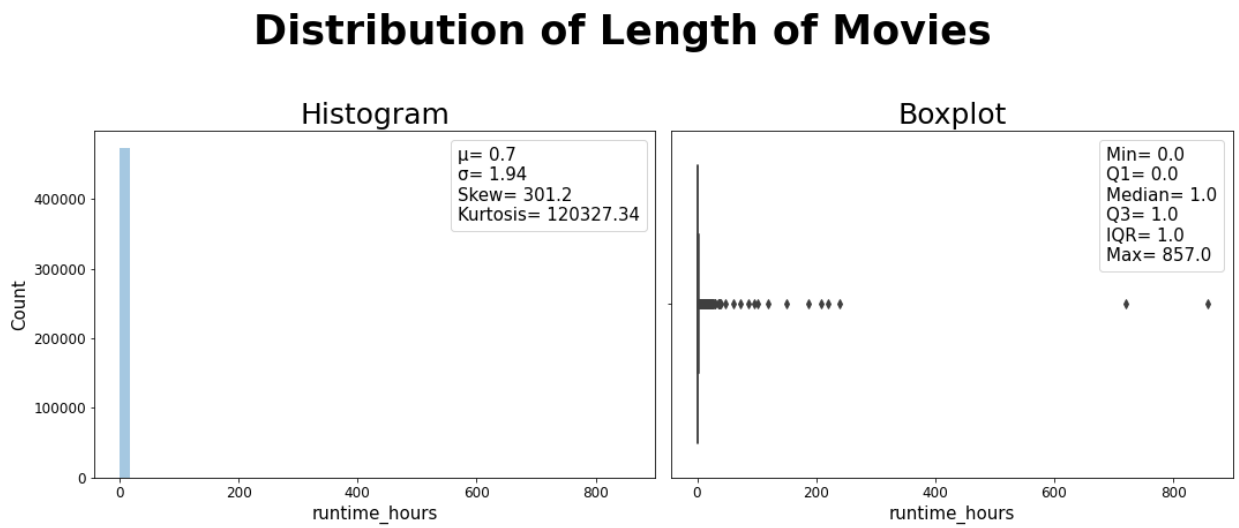
In [47]:
```
df[df.title.isin(['Mr Marcus FriendPop Facebook Adventures', 'Mr Marcus Friend
```

Out[47]:

	index	belongs_to_collection	budget_musd	genres	id	original_language	overview
431184	472657	NaN	NaN	NaN	660968	en	Part 1 of 3
431185	472658	NaN	NaN	NaN	660969	en	Part 2 of 3

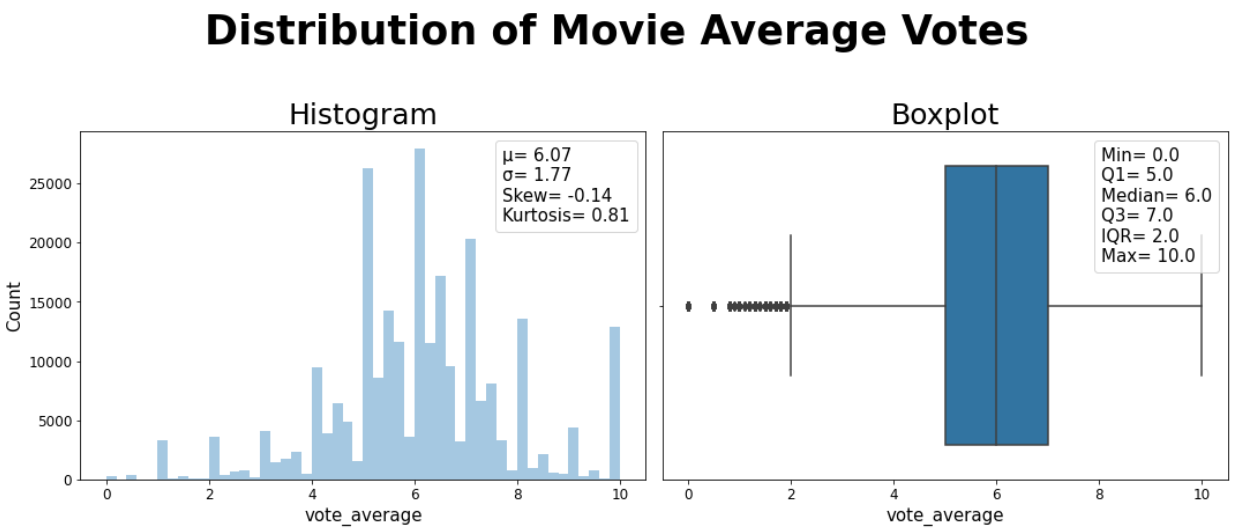
In [49]:
```
# Removing the invalid entries
df.drop(df[df.title.isin(['Mr Marcus FriendPop Facebook Adventures', 'Mr Marcu
```

In [50]:
```
df['runtime_hours'] = np.floor(df.runtime / 60)
hist_boxplot('runtime_hours', 'Length of Movies')
```



• vote_average

In [51]:
```
hist_boxplot('vote_average', 'Movie Average Votes')
```

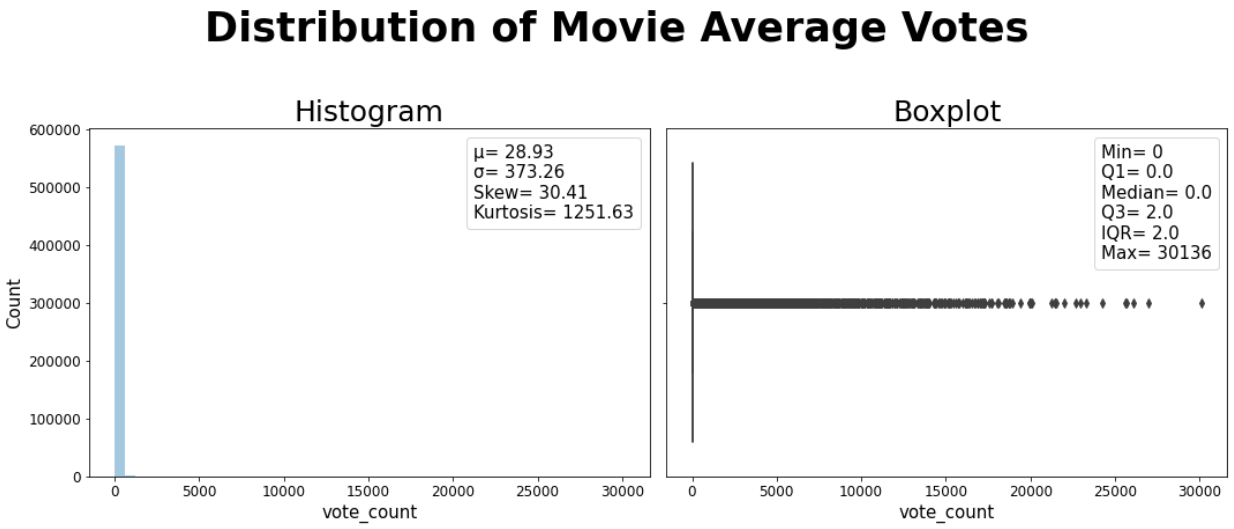


• "vote_average" has a slightly left-skewed Gaussian-like distribution.

• Since all values are within 0 to 10, none will be treated as outliers.

• vote_count

In [52]:
```
hist_boxplot('vote_count', 'Movie Average Votes')
```



• "vote_count" has a heavily right-skewed distribution.

• Although this distribution seems to have extreme outliers to the right of the distribution, popular movies are to be expected to have

an exponentially higher number of votes than other less-popular movies.

In [54]:
```
show_outliers('vote_count', 25000)
```

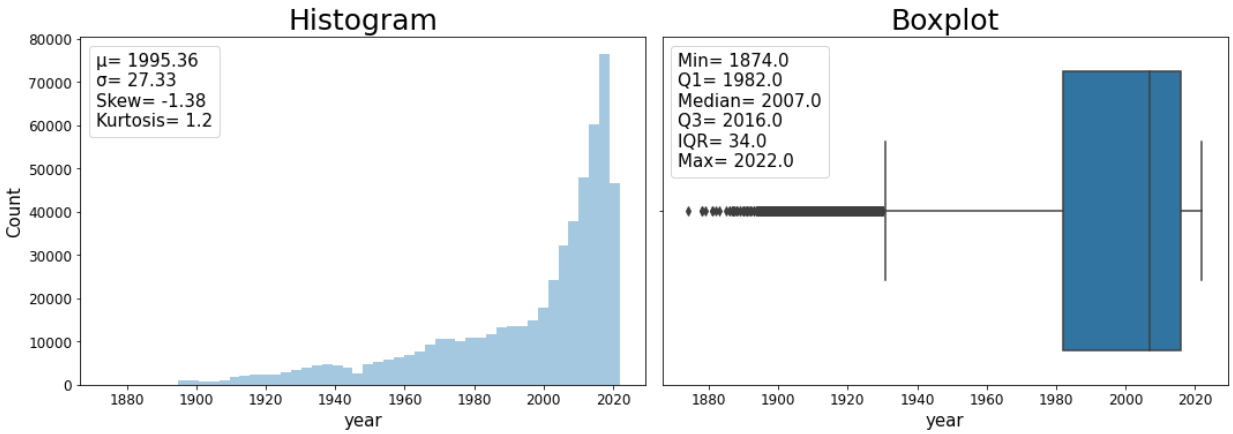
Out[54]:

	html	id	title	vote_count
16277		27205	Inception	30136
94263		157336	Interstellar	26950
121		155	The Dark Knight	26099
170974		293660	Deadpool	25697
14385		24428	The Avengers	25602

- year

In [55]:
```
df.loc[df.year == 'NaT', 'year'] = np.nan
df.year = df.year.astype(float)
```

In [56]:
```
hist_boxplot('year', 'Movie Production Year')
```



• The distribution of year of release is skewed to the left.

• As filming technology and barriers of entry have eased over the years/decades, the amount of movies produced year by year has increased.

Analysis of Non-Numeric Attributes

In [58]:
```
df.describe(include='object')
```

Out[58]:

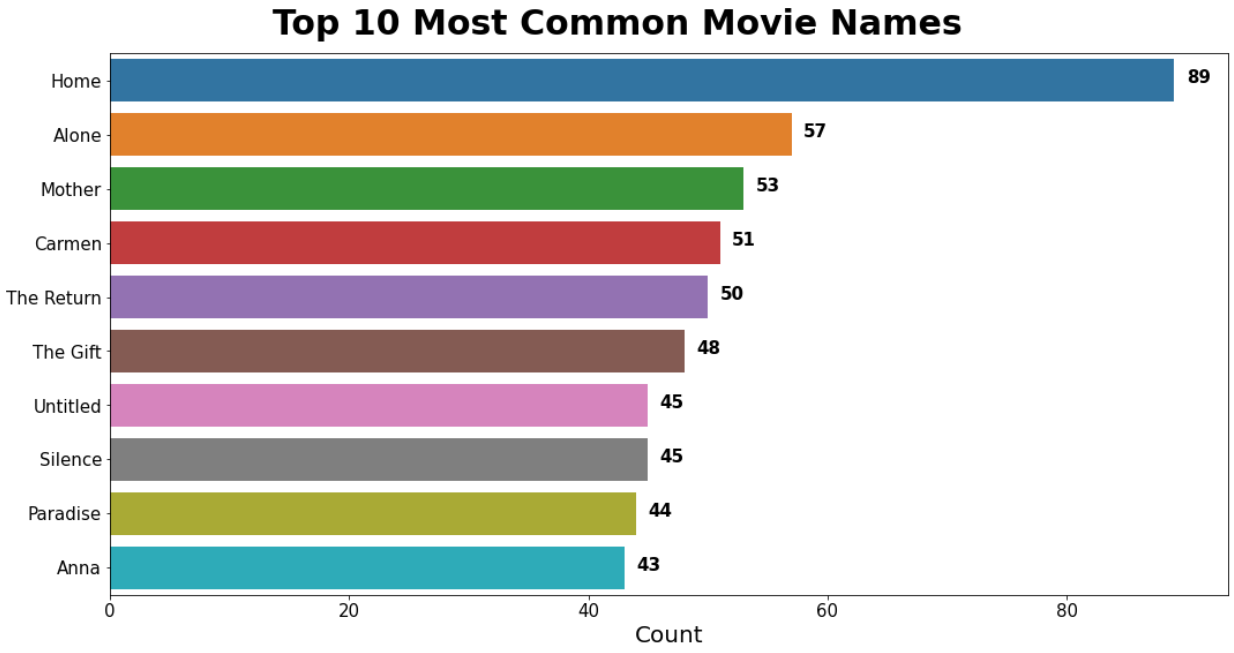
	belongs_to_collection	genres	original_language	overview	
count	15891	425057	578039	496800	
unique	4368	10483	160	488377	
top	Our Gang: The Roach/MGM talkies	Documentary	en	Mexican feature film	https://image.tmdb.org/t/p/w50
freq	80	77314	312717	799	

In [59]:
```
# Creating a function to plot the top 10 values by count
def plot_top_10(data, title='Top 10', explode=False, x_axis_add=0):
    if explode:
        data = data.apply(lambda x: x.split('|') if type(x) == str else x)
        data = data.explode()
    top_10 = data.value_counts(sort=True, ascending=False).head(10)

    fig, ax = plt.subplots(figsize=(15,8))
    sns.barplot(top_10.values, top_10.index, ax=ax, orient='h')
    for i in range(len(top_10)):
        ax.text(top_10.values[i]+1, i+0.05, format(top_10.values[i], ',d'), fo
    #ax.set_title(title, fontsize=30)
    fig.suptitle(title, fontsize=30, fontweight='semibold')
    ax.set_xlabel('Count', fontsize=20)
    ax.tick_params(labelsize=15)
    ax.set_xlim(0, ax.get_xlim()[1] + x_axis_add)
```

```
    fig.tight_layout()
    return plt.show()
```

Top 10 Most Common Movie Names

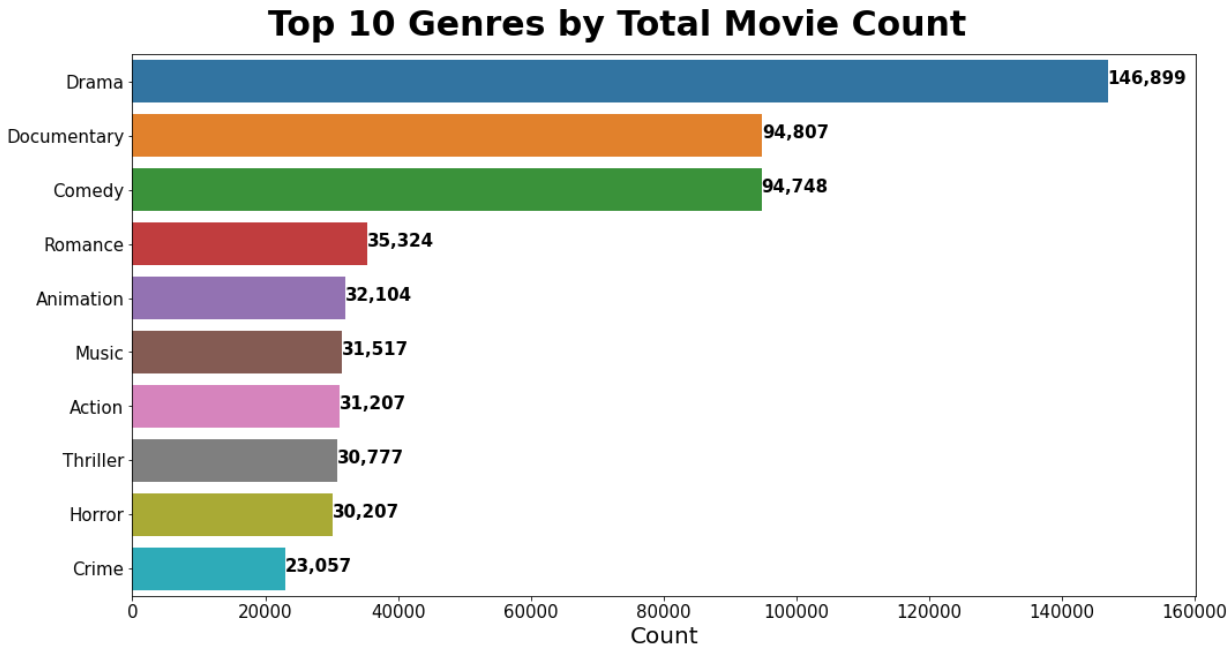
In [60]:
```
plot_top_10(df.title, 'Top 10 Most Common Movie Names')
```



• "Cinderella" is the most frequent title for a movie, appearing a total of 89 times in this dataset.

Top 10 Genres

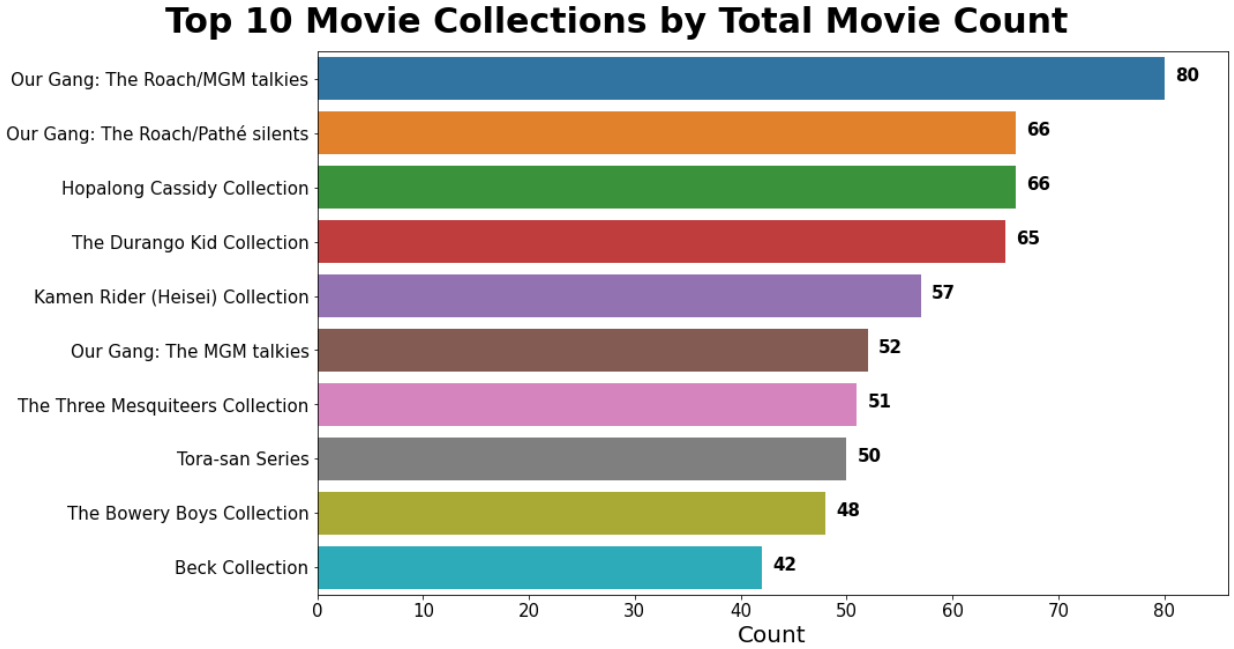
In [61]:
```
plot_top_10(df.genres, 'Top 10 Genres by Total Movie Count', explode=True, x_a
```



• "Drama" is the most frequent genre in the dataset, followed by "Documentary" and "Comedy".

Top 10 Movie Collections

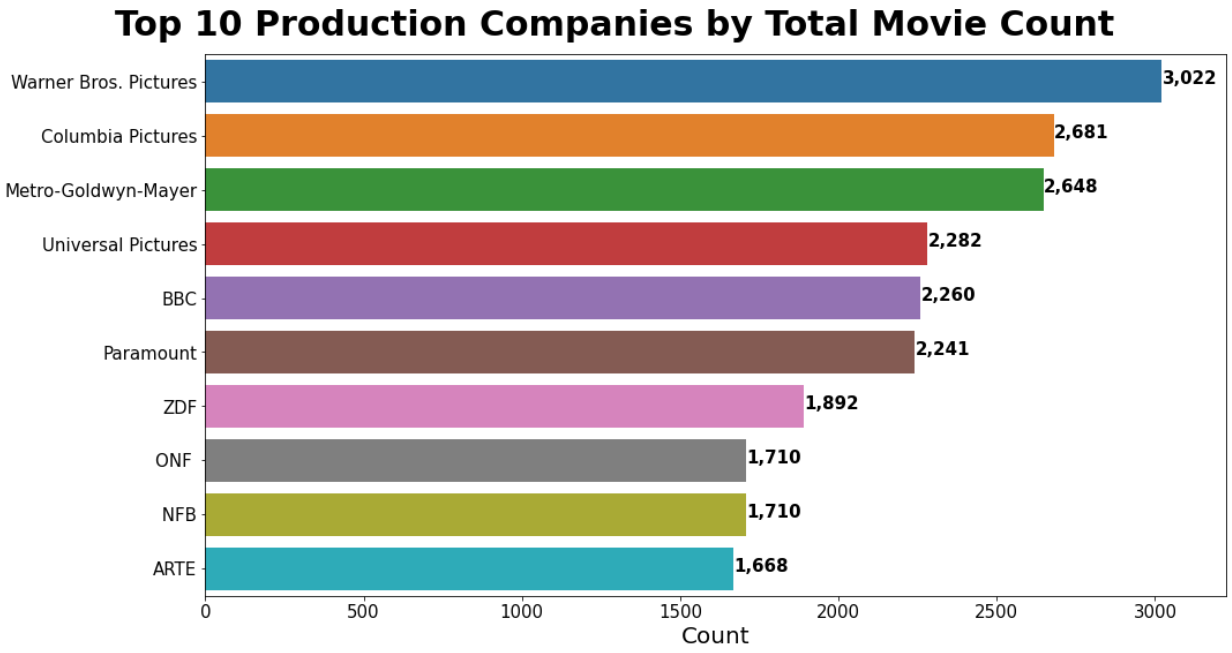
```
In [62]: plot_top_10(df.belongs_to_collection, 'Top 10 Movie Collections by Total Movie
```



• The biggest movie collections contain 1920s/1930s short films which were either silent films or "talkies".

Top 10 Production Companies

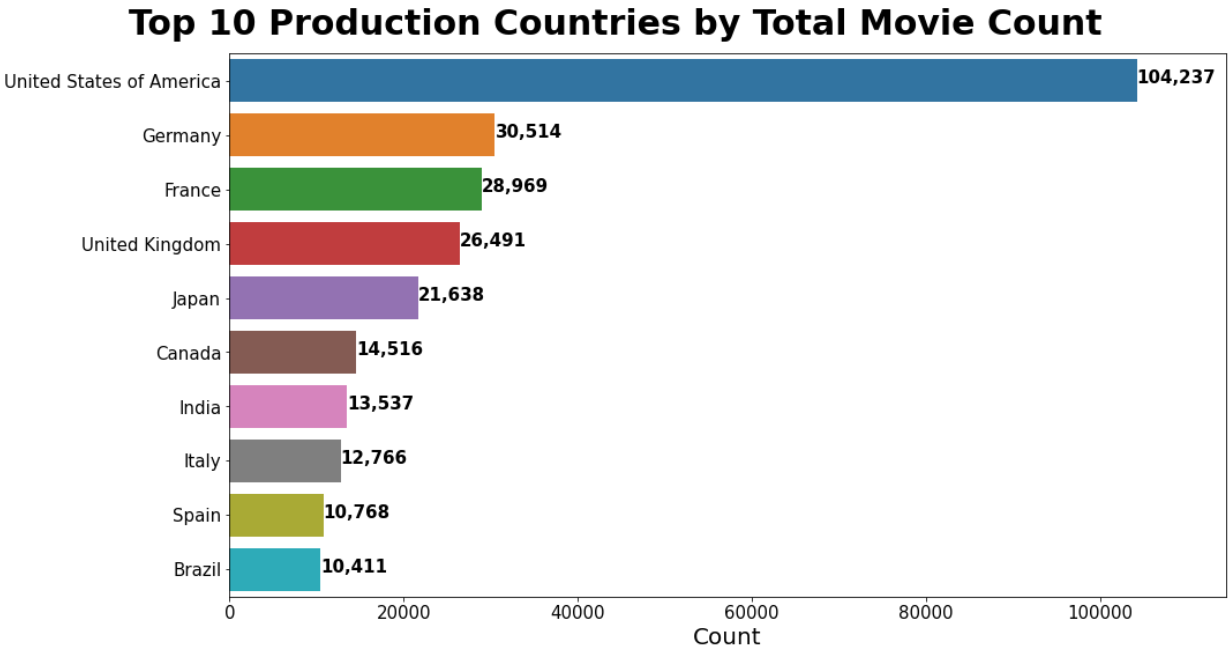
In [63]:
```
plot_top_10(df.production_companies, 'Top 10 Production Companies by Total Mov
```



• Metro-Goldwyn-Mayer is the production company with the highest count of movies produced, followed by Columbia Pictures and Warner Bros. Pictures.

Top 10 Production Countries

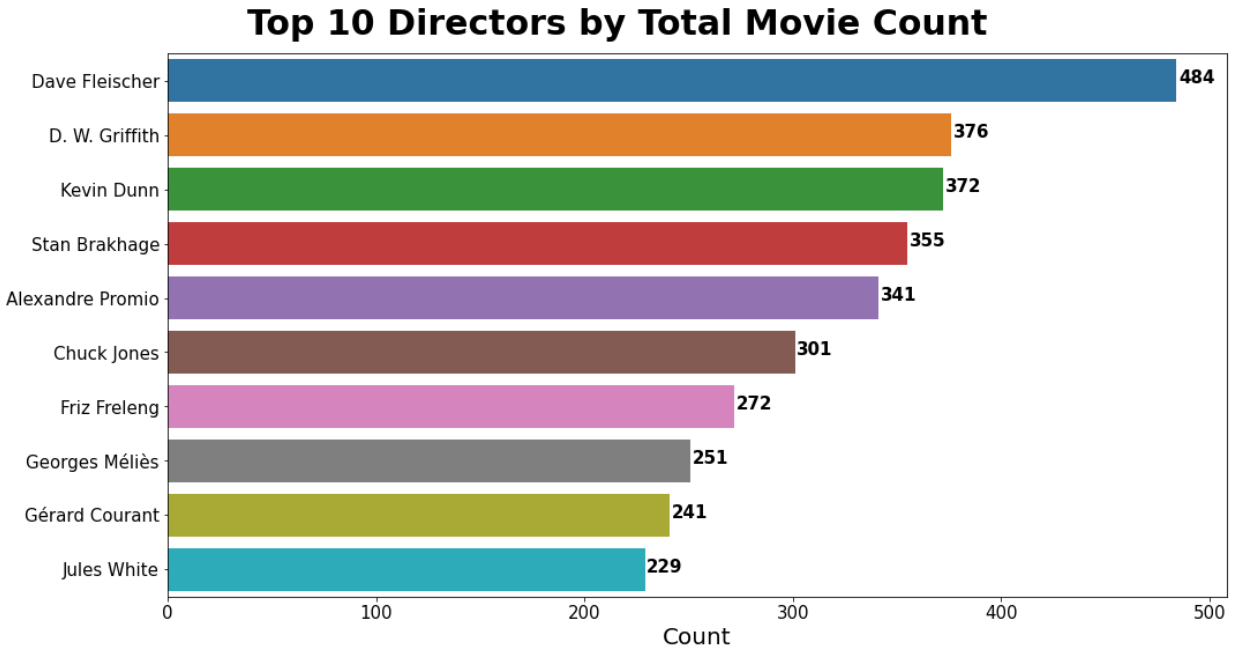
In [64]:
```
plot_top_10(df.production_countries, 'Top 10 Production Countries by Total Mov
```



• By far, the most frequent production company is the United States, as would be expected due to the influence of Hollywood on film-making.

Top 10 Directors

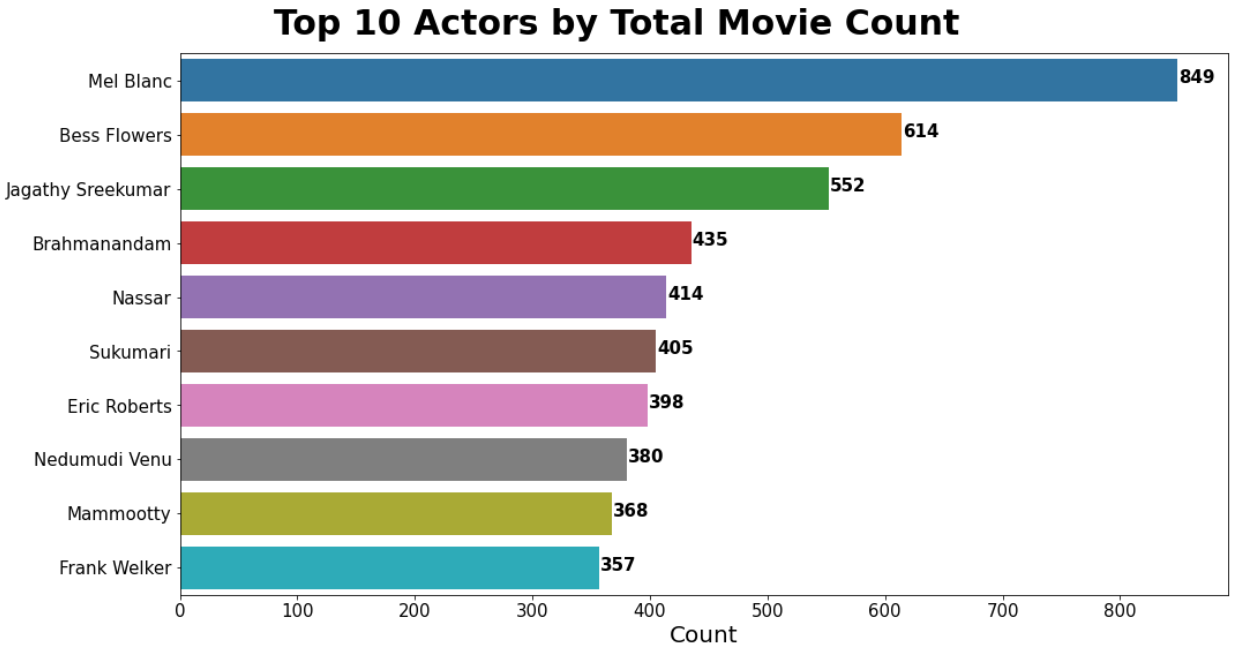
In [65]:
```
plot_top_10(df.director, 'Top 10 Directors by Total Movie Count')
```



• The most active director is "Dave Fleischer", followed by "D.W. Griffith" and "Kevin Dunn".

Top 10 Actors

In [66]:
```
plot_top_10(df.cast_names, 'Top 10 Actors by Total Movie Count', explode=True)
```



• The most active actor is "Mel Blanc", followed by "Bess Flowers" and "Jagathy Sreekumar"

What Are The Most Successful Movies?

• In this section, we will compare movies based on revenue, budget, ROI, profit, average rating, vote count, and popularity.

In [67]:
```
# Creating a function to find best/worst movies
def best_worst(n, by, ascending=False, min_bud=0, min_votes=0):
    df2 = df.loc[(df['budget_musd'] >= min_bud) & (df['vote_count'] >= min_vot
                     ['title','html',by]].sort_values(by=by, ascending=ascendi
    return HTML(df2.to_html(escape=False))
```

Top 10 Movies with Highest Revenue

In [68]:
```
best_worst(10,'revenue_musd', min_bud=5)
```

Out[68]:

title	html	revenue_musd
Avatar		2847.25
Avengers: Endgame		2797.80
Titanic		2187.46
Star Wars: The Force Awakens		2068.22
Avengers: Infinity War		2046.24
Jurassic World		1671.71
The Lion King		1667.64
The Avengers		1518.82

title	html	revenue_musd
Furious 7		1515.05
Frozen II		1450.03

Top 10 Movies with Highest Budget

In [70]:
```
best_worst(10,'budget_musd')
```

Out[70]:

title	html	budget_musd
Pirates of the Caribbean: On Stranger Tides		380.00
Avengers: Endgame		356.00
Avengers: Infinity War		300.00
Pirates of the Caribbean: At World's End		300.00
Mega Man X: The Day of Sigma		300.00
Justice League		300.00
Superman Returns		270.00
The Lion King		260.00

title	html	budget_musd
Tangled		260.00
Spider-Man 3		258.00

Top 10 Movies with Highest Profit

In [71]:
```
best_worst(10,'profit_musd', min_bud=5)
```

Out[71]:

title	html	profit_musd
Avatar		2610.25
Avengers: Endgame		2441.80
Titanic		1987.46
Star Wars: The Force Awakens		1823.22
Avengers: Infinity War		1746.24
Jurassic World		1521.71
The Lion King		1407.64
Furious 7		1325.05

title	html	profit_musd
Frozen II		1300.03
The Avengers		1298.82

Top 10 Movies with Lowest Profit

In [74]:
```
best_worst(10,'profit_musd',True, min_bud=5)
```

Out[74]:

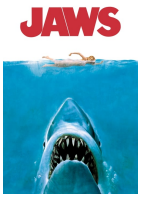
title	html	profit_musd
Messengers		-194.78
The Irishman		-151.00
Luca		-150.99
Mulan		-133.20
The Alamo		-119.18
Mars Needs Moms		-111.01
The 13th Warrior		-98.30
The Adventures of Pluto Nash		-92.90

title	html	profit_musd
Cutthroat Island		-87.98
Truth and Lies		-86.00

Top 10 Movies with Highest Return on Investment

In [75]:
```
best_worst(10,'return_musd',min_bud=5)
```

Out[75]:

title	html	return_musd
Operation Thunderbolt		207.40
Hamood Habibi		100.00
aflam Movie Title	NaN	100.00
E.T. the Extra-Terrestrial		75.52
My Big Fat Greek Wedding		73.75
Star Wars		70.49
Jaws		67.24
Crocodile Dundee		65.64

title	html	return_musd
The Exorcist		55.16
Paranormal Activity 3		41.41

In [76]:
```
df.drop(df[df.title == 'aflam Movie Title'].index[0], axis=0, inplace=True)
```

Top 10 Movies with Most Votes

In [77]:
```
best_worst(10,'vote_count')
```

Out[77]:

title	html	vote_count
Inception		30136
Interstellar		26950
The Dark Knight		26099
Deadpool		25697
The Avengers		25602
Avatar		24229
Guardians of the Galaxy		23290
Avengers: Infinity War		22979

title	html	vote_count
Fight Club		22711
Pulp Fiction		21996

Top 10 Movies with Highest Rating (Min Votes = 5000)

In [78]:
```
best_worst(10,'vote_average',min_votes=5000)
```

Out[78]:

title	html	vote_average
The Godfather		8.70
The Shawshank Redemption		8.70
The Godfather: Part II		8.60
Schindler's List		8.60
The Green Mile		8.50
Life Is Beautiful		8.50
The Dark Knight		8.50
The Lord of the Rings: The Return of the King		8.50

title	html	vote_average
GoodFellas		8.50
The Good, the Bad and the Ugly		8.50

Top 10 Movies with Lowest Rating(Min Votes = 5000)

In [79]:
```
best_worst(10,'vote_average',ascending=True,min_votes=5000)
```

Out[79]:

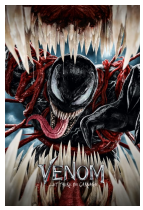
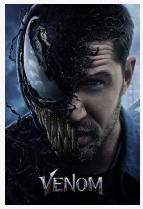
title	html	vote_average
Green Lantern		5.20
After Earth		5.20
Independence Day: Resurgence		5.20
A Good Day to Die Hard		5.30
Jupiter Ascending		5.40
Ghostbusters		5.40
Assassin's Creed		5.40
The Mummy		5.50

title	html	vote_average
Noah		5.60
Fantastic Four: Rise of the Silver Surfer		5.60

Top 10 Movies by Popularity

In [80]:
```
best_worst(10,'popularity', min_votes=1000)
```

Out[80]:

title	html	popularity
Venom: Let There Be Carnage		6121.12
Dune		4893.62
Free Guy		2334.62
Halloween Kills		2223.22
Coco		1916.33
Venom		1568.26
Shang-Chi and the Legend of the Ten Rings		1158.57
The Suicide Squad		941.32

title	html	popularity
The Boss Baby: Family Business		914.18
F9		895.27

Are Franchises More Successful than Stand-alone Movies?

• In this section, we will compare franchises and stand-alone movies based on revenue, ROI, budget, popularity, and vote average.

In [83]:
```
df['Franchise'] = df.belongs_to_collection.notna()
df['Franchise'] = df.Franchise.map({True: 'Franchise', False: 'Stand-alone'})
df['Franchise']
```

Out[83]:
```
0           Franchise
1         Stand-alone
2         Stand-alone
3           Franchise
4         Stand-alone
             ...
578039    Stand-alone
578040    Stand-alone
578041    Stand-alone
578042    Stand-alone
578043    Stand-alone
Name: Franchise, Length: 578038, dtype: object
```

In [84]:
```
df.Franchise.value_counts()
```

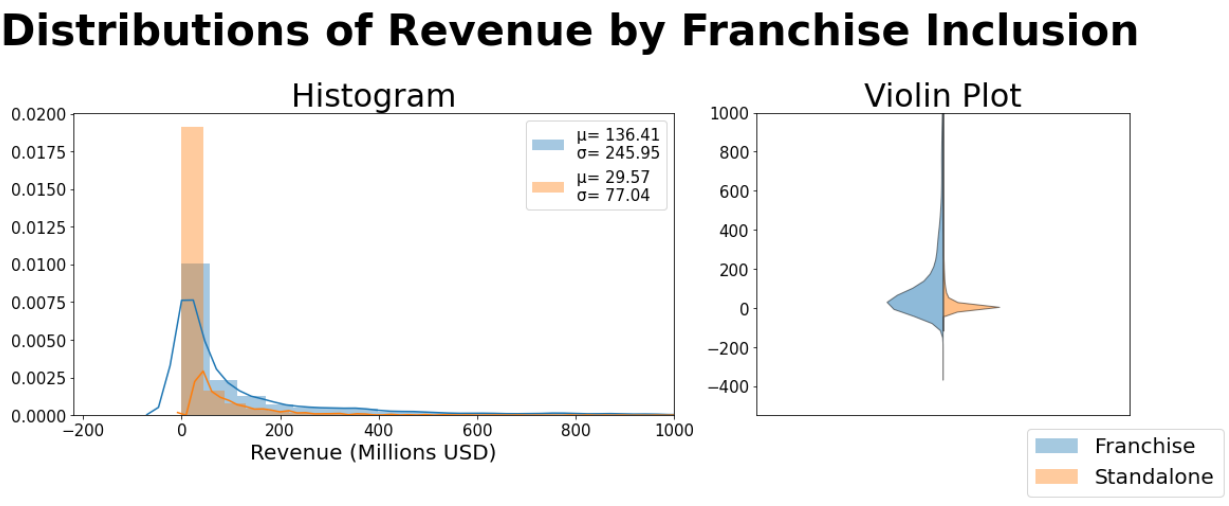
Out[84]:
```
Stand-alone    562147
Franchise       15891
Name: Franchise, dtype: int64
```

In [85]:
```
# Creating a function to create histogram and violin plot figures.
def hist_violin(column, title, xlabel, bins=40, lim_max=None):
    data1 = df.loc[df.Franchise == 'Franchise', column]
    data2 = df.loc[df.Franchise == 'Stand-alone', column]
    data1_mean = np.round(data1.mean(),2)
    data2_mean = np.round(data2.mean(),2)
    data1_std = np.round(data1.std(),2)
    data2_std = np.round(data2.std(),2)
```

```
    pie_group = df.groupby('Franchise')[column].mean()
    fig = plt.figure(figsize=(15,6))
    fig.suptitle(title, fontsize=40, fontweight='semibold', y=1.01)
    ax1 = plt.subplot2grid((1,5), (0,0), colspan=3)
    ax2 = plt.subplot2grid((1,5), (0,3), colspan=2)



    sns.distplot(data1, ax=ax1, bins=bins, label=f"μ= {data1_mean}\nσ= {data1_
    sns.distplot(data2, ax=ax1, bins=bins, label=f"μ= {data2_mean}\nσ= {data2_
    ax1.set_title('Histogram', fontsize=30)
    ax1.tick_params(labelsize=15)
    ax1.set_xlabel(xlabel, fontsize=20)
    ax1.legend(fontsize=15)

    violinplot([data1.dropna()], ax=ax2, positions=[0], show_boxplot=False, si
    violinplot([data2.dropna()], ax=ax2, positions=[0], show_boxplot=False, si
    ax2.set_xticks([])
    ax2.tick_params(labelsize=15)
    ax2.set_title('Violin Plot', fontsize=30)

    if lim_max:
        ax1.set_xlim(ax1.get_xlim()[0], lim_max)
        ax2.set_ylim(ax2.get_ylim()[0], lim_max)


    fig.tight_layout()
    #fig.legend(['Franchise', 'Standalone'], loc='lower right', fontsize=20)
    fig.legend(['Franchise', 'Standalone'], bbox_to_anchor=(.9, .1), loc=2, bo
    plt.show()
```

Franchise vs. Stand-alone: Movie Revenue

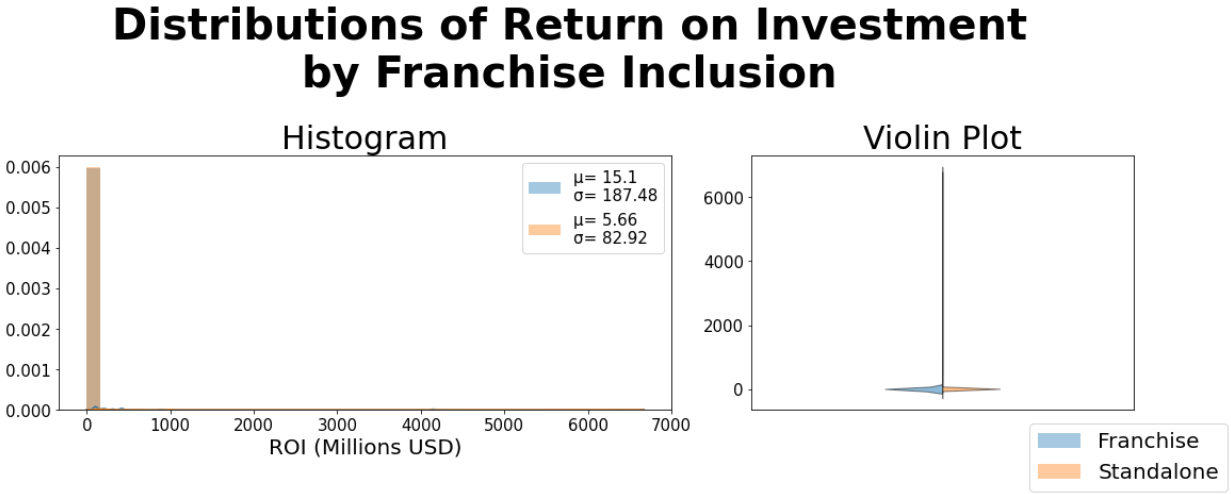
In [86]:
```
hist_violin('revenue_musd', 'Distributions of Revenue by Franchise Inclusion',
```



- Franchises have a higher average revenue than stand-alone movies.

Franchise vs. Stand-alone: Return on Investment/Profitability

In [87]:
```
hist_violin('return_musd', 'Distributions of Return on Investment\nby Franchis
```

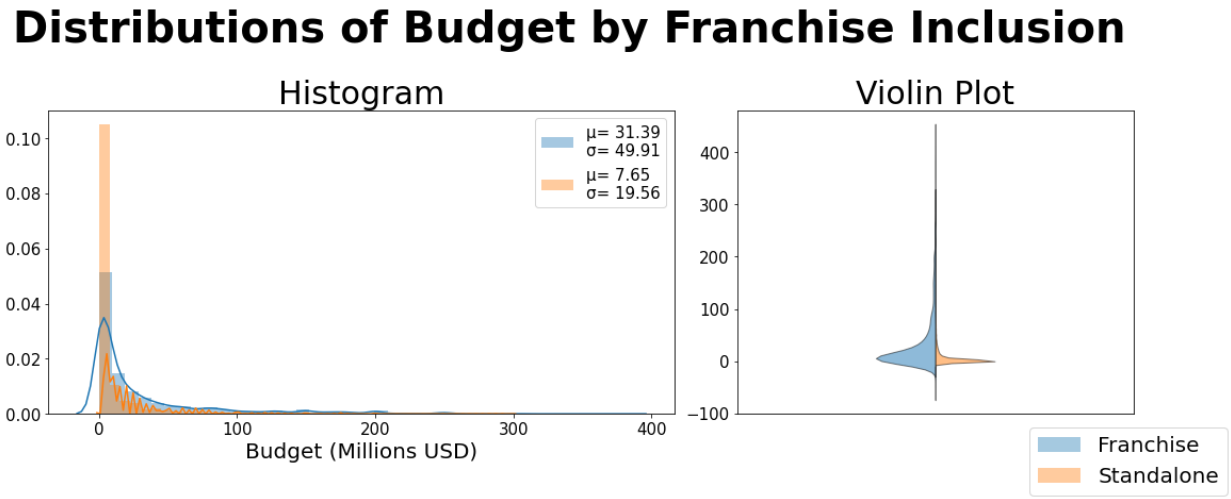


• Franchises have a higher average return on investment than stand-alone movies.

• Franchises distribution has a "fatter" tail than stand-alone movies distribution.

Franchise vs. Stand-alone: Movie Budget

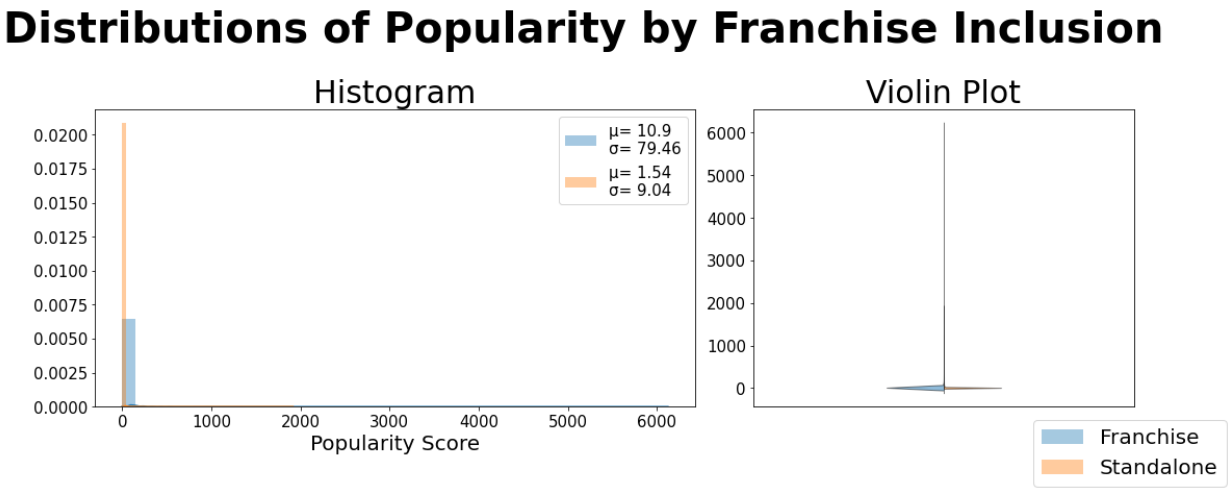
In [88]:
```
hist_violin('budget_musd', 'Distributions of Budget by Franchise Inclusion', '
```



• Franchises have a higher budget on average than stand-alone movies, which is to be expected due to previous movie successes in a franchise.

Franchise vs. Stand-alone: Movie Popularity

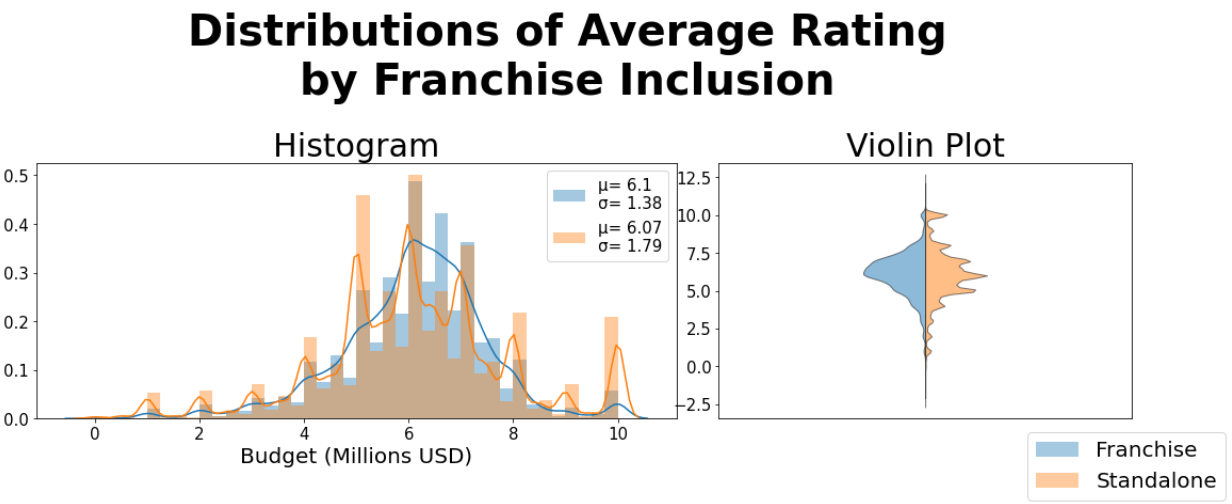
In [89]:
```
hist_violin('popularity', 'Distributions of Popularity by Franchise Inclusion'
```



• Franchises have a higher popularity score on average than stand-alone movies.

Franchise vs. Stand-alone: Average Rating

In [90]:
```
hist_violin('vote_average', 'Distributions of Average Rating\nby Franchise Inc
```



• The average rating for both franchises and stand-alone movies is very similar.

What Are the Most Successful Franchises?

• In this section, we will compare franchises based on movie count, revenue, vote average, and return on investment.

In [91]:
```
franchises = df.groupby("belongs_to_collection").agg({"title":"count", "budget
                                                      "revenue_musd": ["sum",
                                                      "vote_average": "mean",
                                                      "return_musd":"median",
                                                      "vote_count":"sum"})

franchises.head()
```

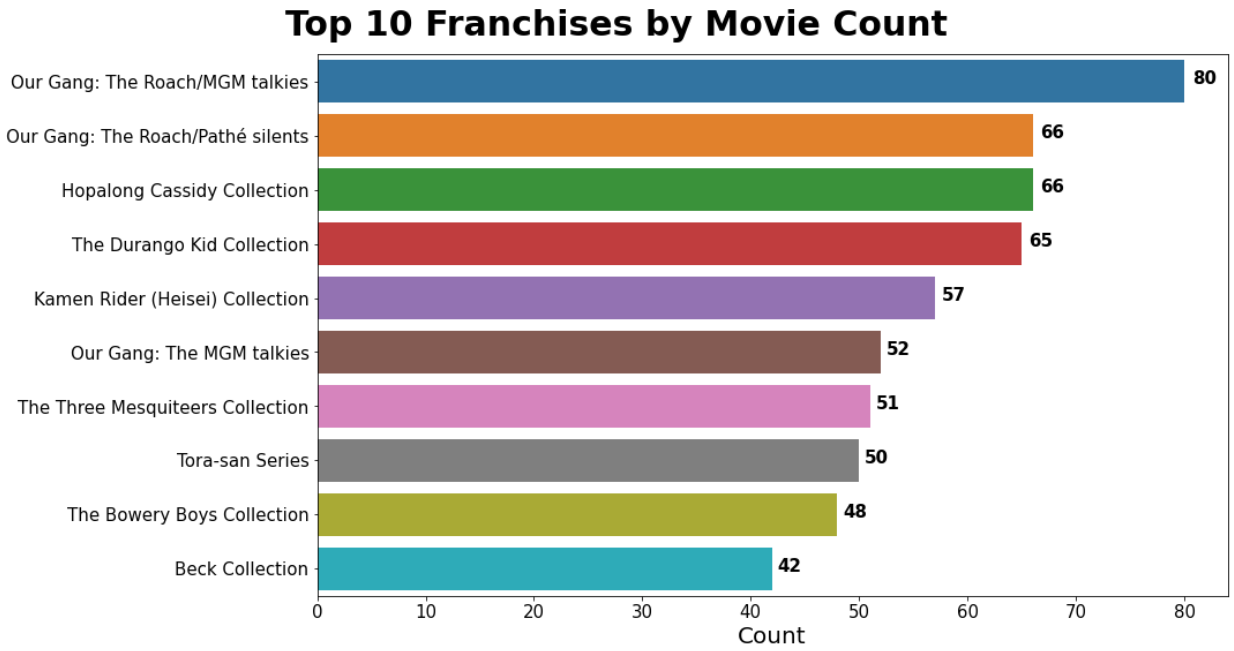
Out[91]:

	title	budget_musd		revenue_musd		vote_average	popularity	return_musd
	count	sum	mean	sum	mean	mean	mean	median
belongs_to_collection								
#TemanTapiMenikah	2	0.00	NaN	0.00	NaN	7.15	1.63	NaN
#TubeClash	3	0.00	NaN	0.00	NaN	7.33	0.85	NaN
'Missing' Collection	6	0.00	NaN	0.00	NaN	3.25	0.90	NaN
'Saturn' Collection	3	0.00	NaN	0.00	NaN	6.50	0.87	NaN
... Blood Collection	2	0.00	NaN	0.00	NaN	5.15	2.05	NaN

In [92]:
```
# Creating a function to plot the grouped and aggregated dataframe
def plot_groupby(df, level1, level2, title='Top 10', min_votes=None, agg='mean
    if min_votes:
        data = df.loc[df['vote_count',agg] > min_votes][level1,level2].sort_va
    else:
        data = df[level1, level2].sort_values(ascending=False).head(10)

    fig, ax = plt.subplots(figsize=(15,8))
    fig.suptitle(title, fontsize=30, fontweight='semibold')

    sns.barplot(data.values, data.index, orient='h')
    for i in range(len(data)):
        ax.text(data.values[i]* 1.01, i+0.05, format(np.round(data.values[i],2
    ax.set_xlabel('Count', fontsize=20)
    ax.set_ylabel(None)
    ax.tick_params(labelsize=15)
    ax.set_xlim(0, ax.get_xlim()[1] + x_axis_add)
    fig.tight_layout()
```

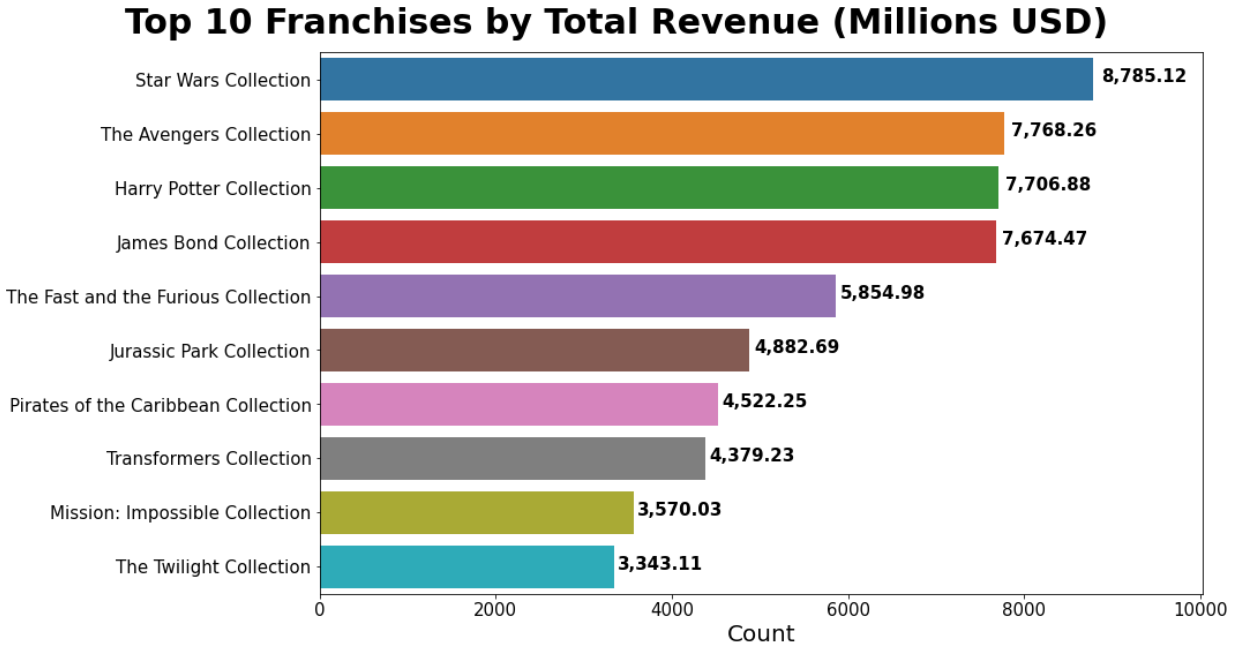
Top 10 Franchises by Movie Count

In [93]:
```
plot_groupby(franchises, 'title', 'count', 'Top 10 Franchises by Movie Count')
```

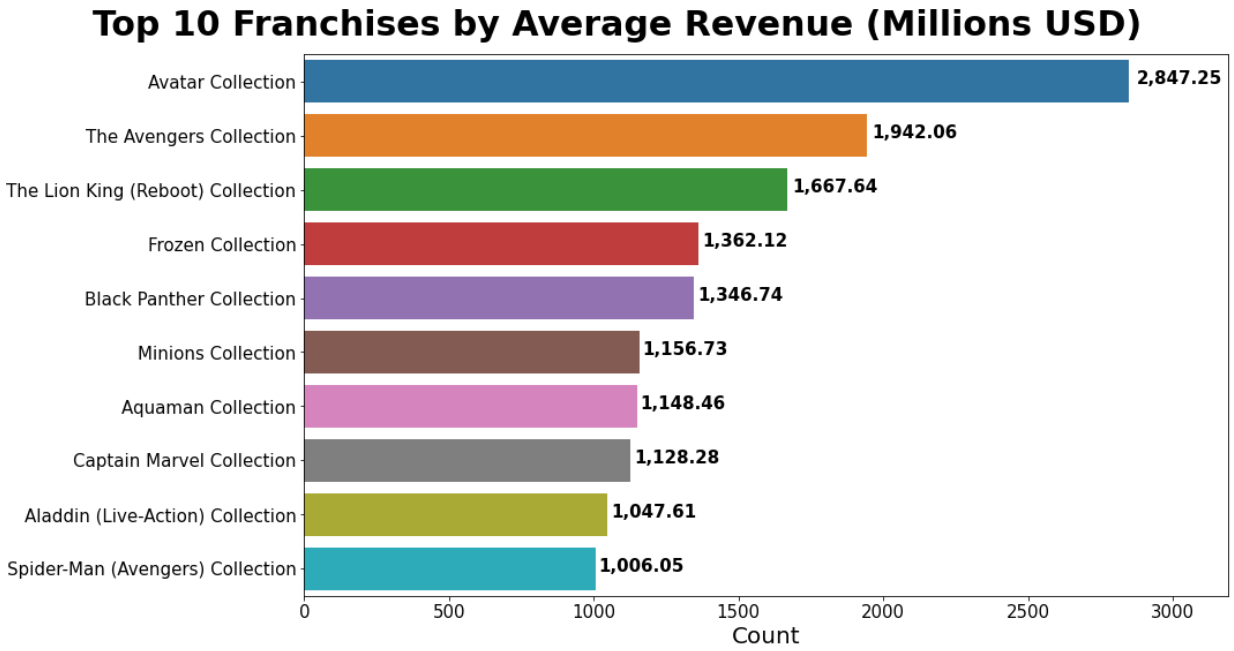


Top 10 Franchises by Revenue Sum

In [94]:
```
plot_groupby(franchises, 'revenue_musd', 'sum', 'Top 10 Franchises by Total Re
```

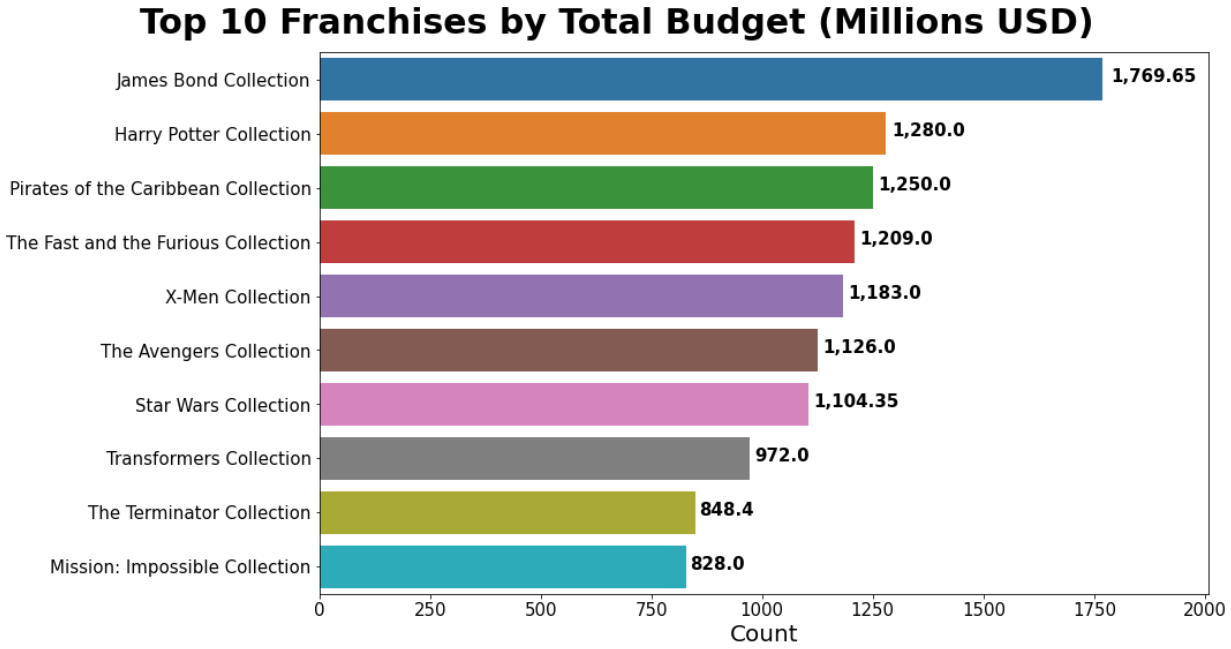


In [95]:
```
plot_groupby(franchises, 'revenue_musd', 'mean', 'Top 10 Franchises by Average
```

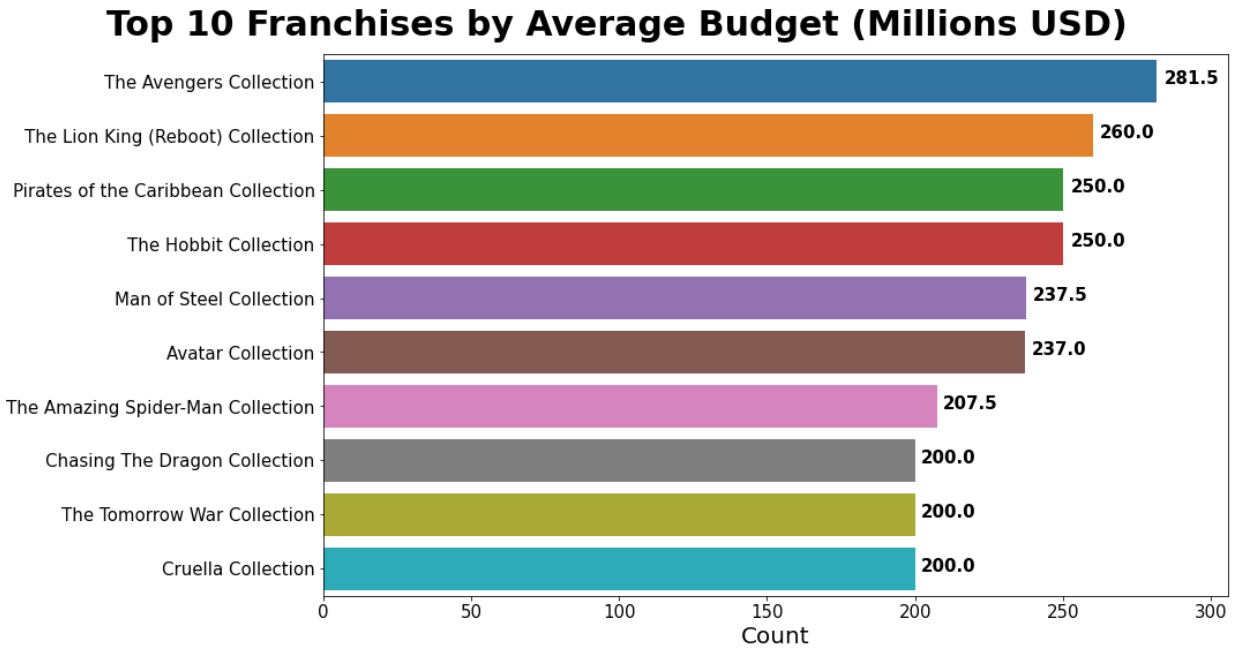


Top 10 Franchises by Budget Sum

In [96]:
```
plot_groupby(franchises, 'budget_musd', 'sum', 'Top 10 Franchises by Total Bud
```

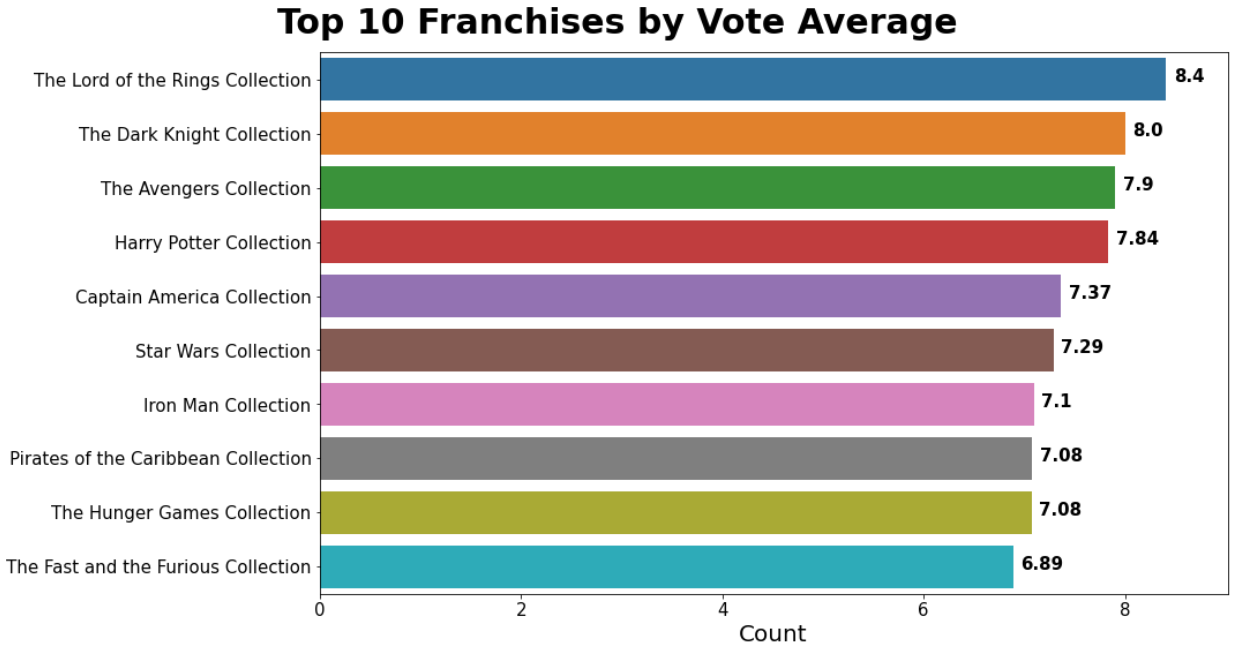


In [99]:
```
plot_groupby(franchises, 'budget_musd', 'mean', 'Top 10 Franchises by Average
```



Top 10 Franchises by Vote Average

In [100…
```
plot_groupby(franchises, 'vote_average', 'mean', 'Top 10 Franchises by Vote Av
```



Who Are The Most Successful Directors?

• In this section, we will compare directors based on movie count, vote average, profit, and revenue.

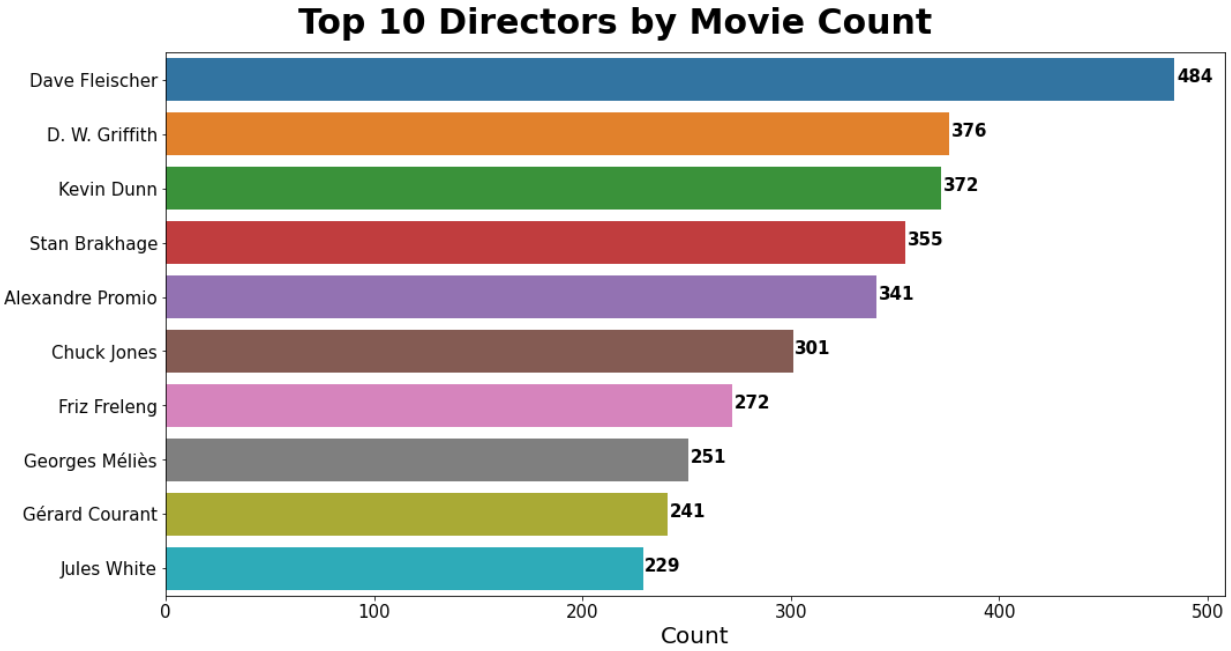
In [101…
```
directors = df.groupby("director").agg({"title": "count", "vote_average" :"mea
directors.head()
```

Out[101]:

	title	vote_average	vote_count	profit_musd		revenue_musd	
	count	mean	sum	mean	sum	mean	sum
director							
\tZhang Jing	3	4.00	1	NaN	0.00	NaN	0.00
Faris Kermani	3	6.90	4	NaN	0.00	NaN	0.00
Anna van Keimpema	3	4.80	2	NaN	0.00	NaN	0.00
Ariel Hassan	1	5.70	3	NaN	0.00	NaN	0.00
Barnaby Southcombe	2	6.20	34	NaN	0.00	NaN	0.00

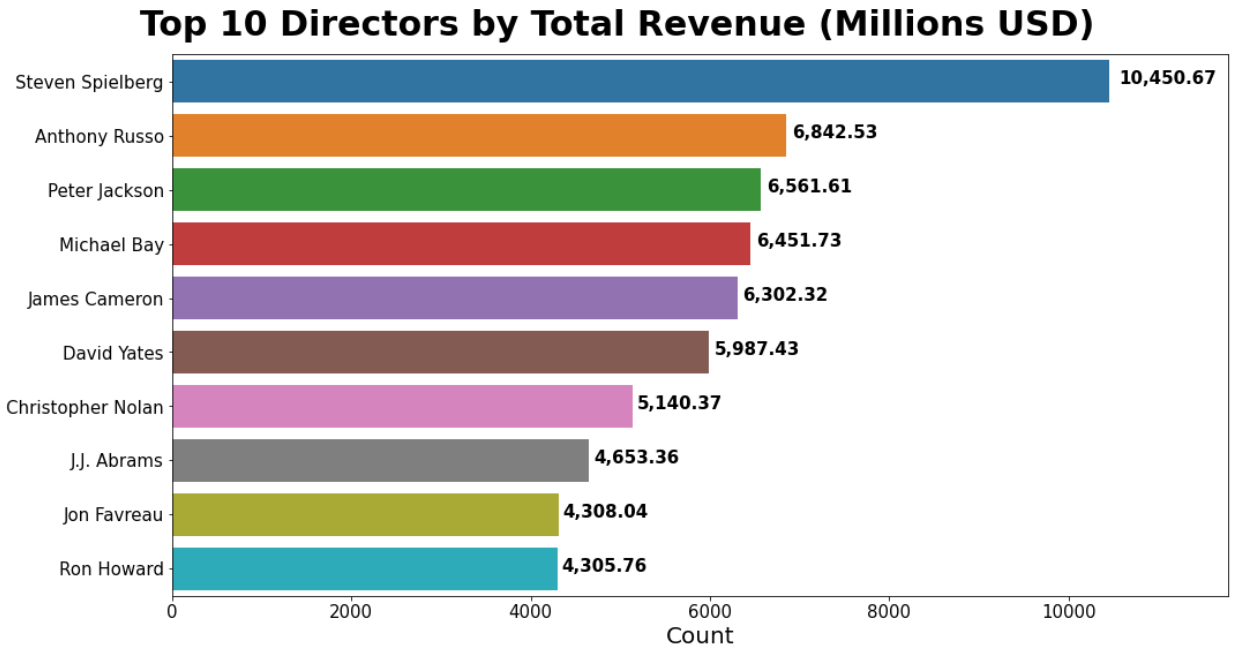
Top 10 Directors by Movie Count

In [102…
```
plot_top_10(df.director, 'Top 10 Directors by Movie Count')
```

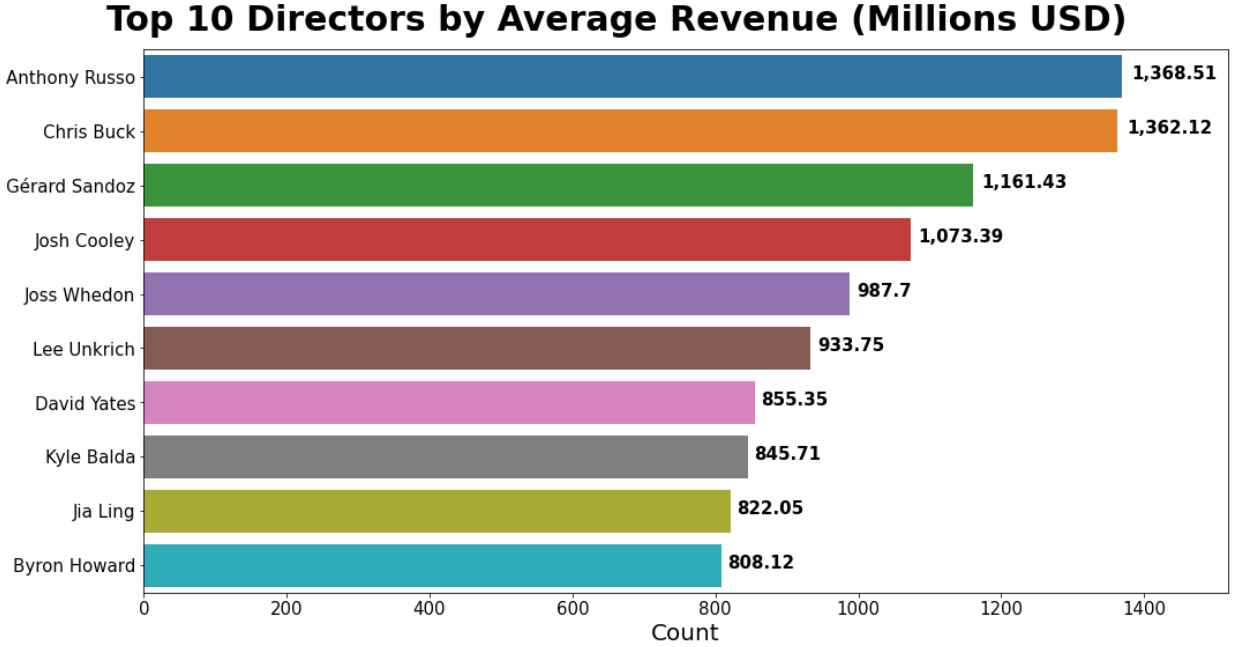


Top 10 Directors by Revenue

In [103…
```
plot_groupby(directors, 'revenue_musd', 'sum', title='Top 10 Directors by Tota
```

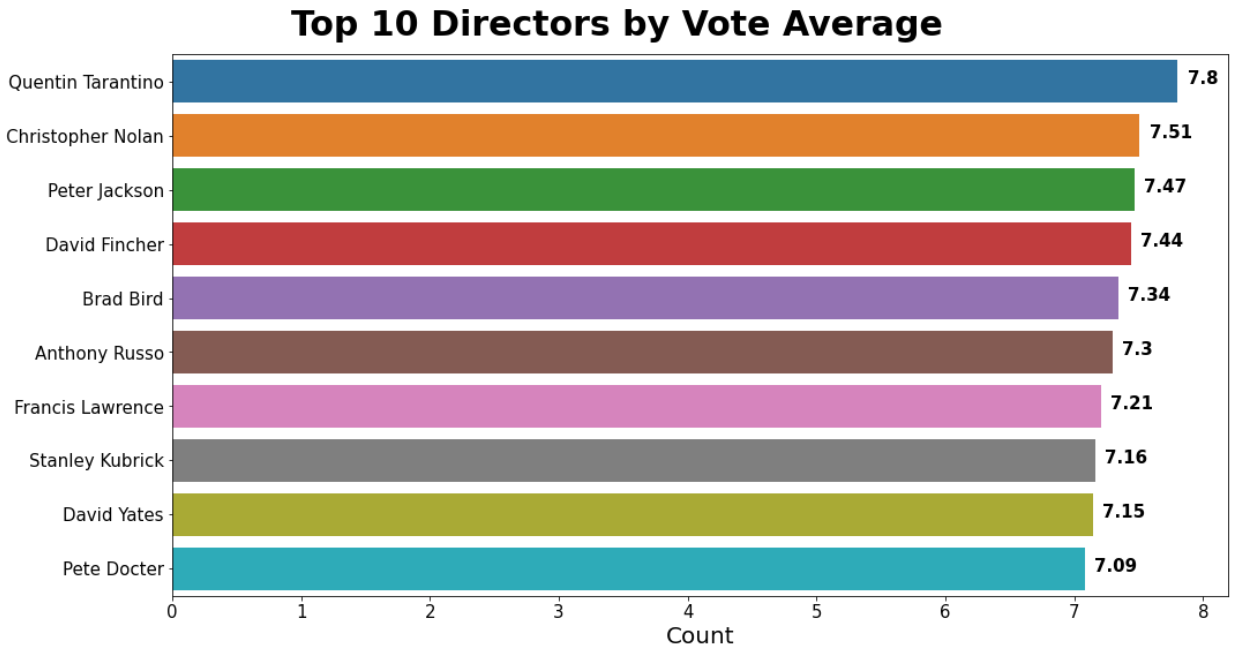


In [104…
```
plot_groupby(directors, 'revenue_musd', 'mean', title='Top 10 Directors by Ave
```



Top 10 Directors by Vote Average

In [106…
```
plot_groupby(directors, 'vote_average', 'mean', 'Top 10 Directors by Vote Aver
```



What Are The Most Successful Genres?

In this section, we will compare all the genres on vote average, popularity, profit, and revenue.

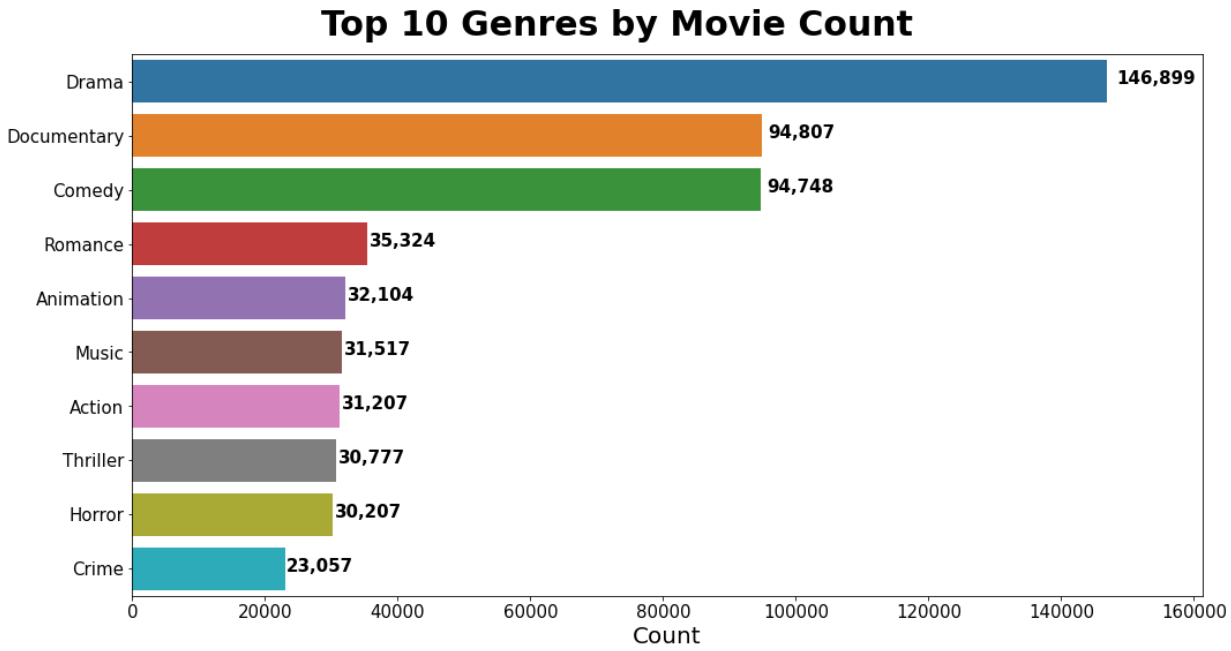
In [107…
```
genres_df = df.copy()
genres_df['genres'] = genres_df['genres'].apply(lambda x: x.split('|') if type
genres_df = genres_df.explode('genres').groupby('genres').agg({"title": "count
genres_df.head()
```

Out[107]:

	title	vote_average	vote_count	popularity	profit_musd		revenue_musd	
	count	mean	sum	mean	mean	sum	mean	sum
genres								
Action	31207	5.80	5094608	6.19	77.07	168093.43	94.04	267266.17
Adventure	16703	6.01	4375508	7.31	128.49	179505.72	147.40	266347.06
Animation	32104	6.35	1626483	3.36	127.42	58614.13	112.61	89524.45
Comedy	94748	5.95	5112056	2.63	44.22	135352.19	47.22	221183.92
Crime	23057	5.99	2333957	3.65	32.84	42031.87	44.07	75838.47

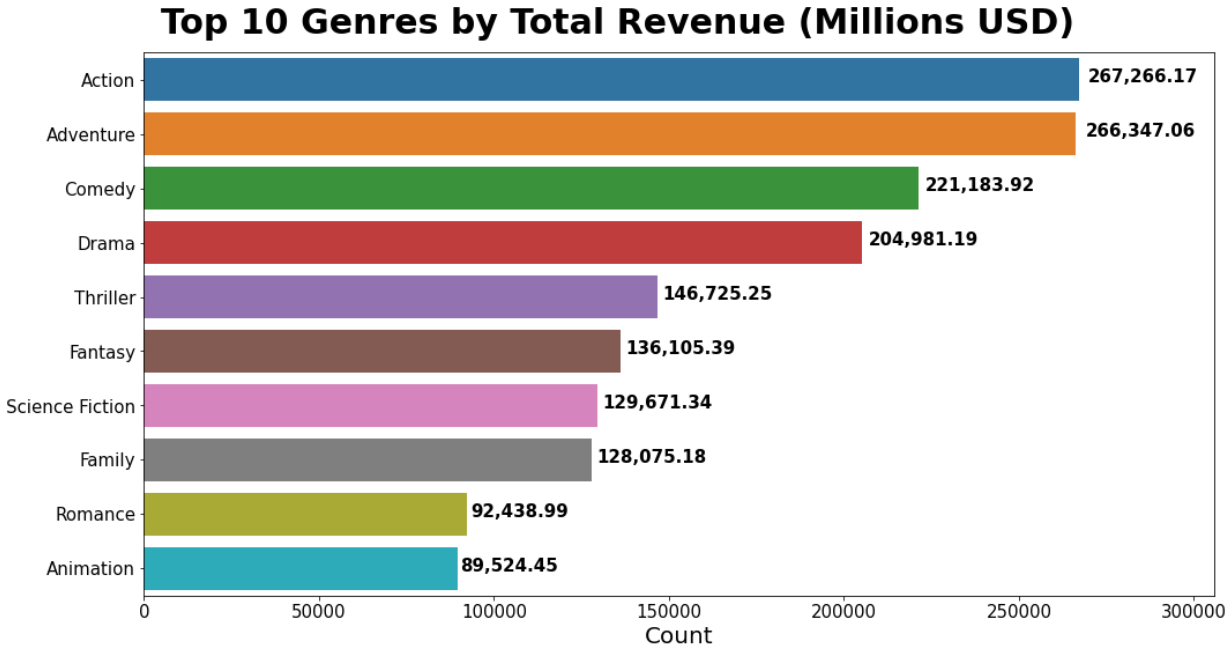
Top 10 Genres by Movie Count

In [108…
```
plot_groupby(genres_df, 'title', 'count', title='Top 10 Genres by Movie Count'
```

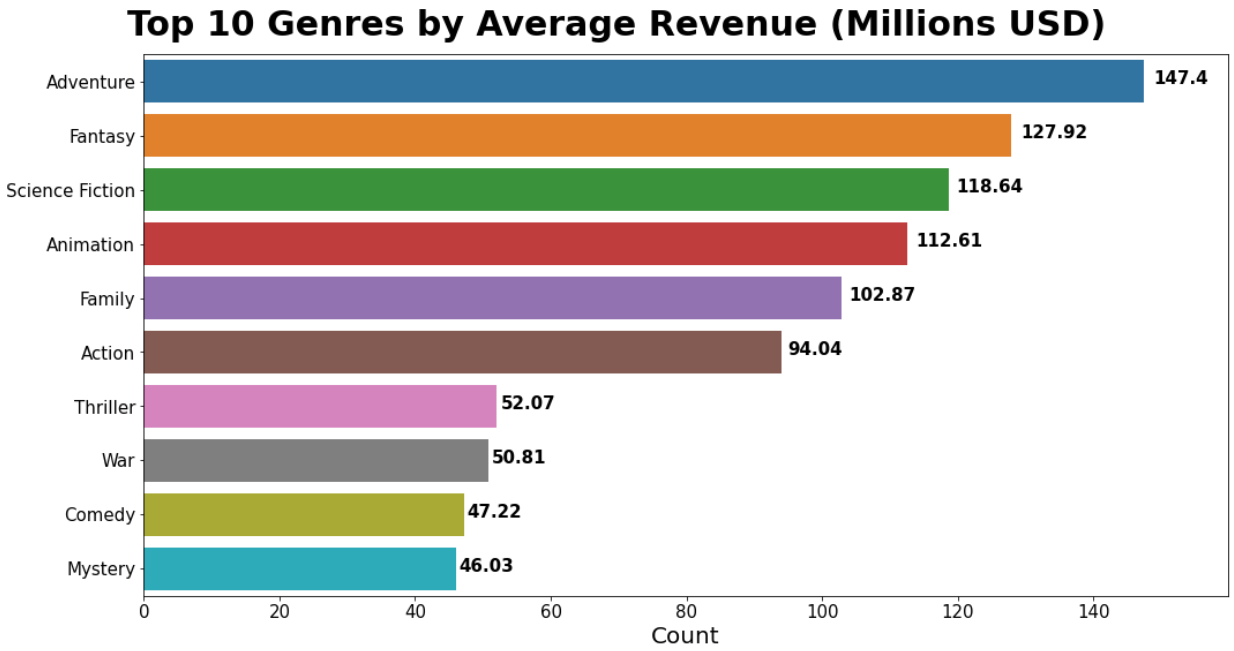


Top 10 Genres by Revenue

In [109…
```
plot_groupby(genres_df, 'revenue_musd', 'sum', title='Top 10 Genres by Total R
```

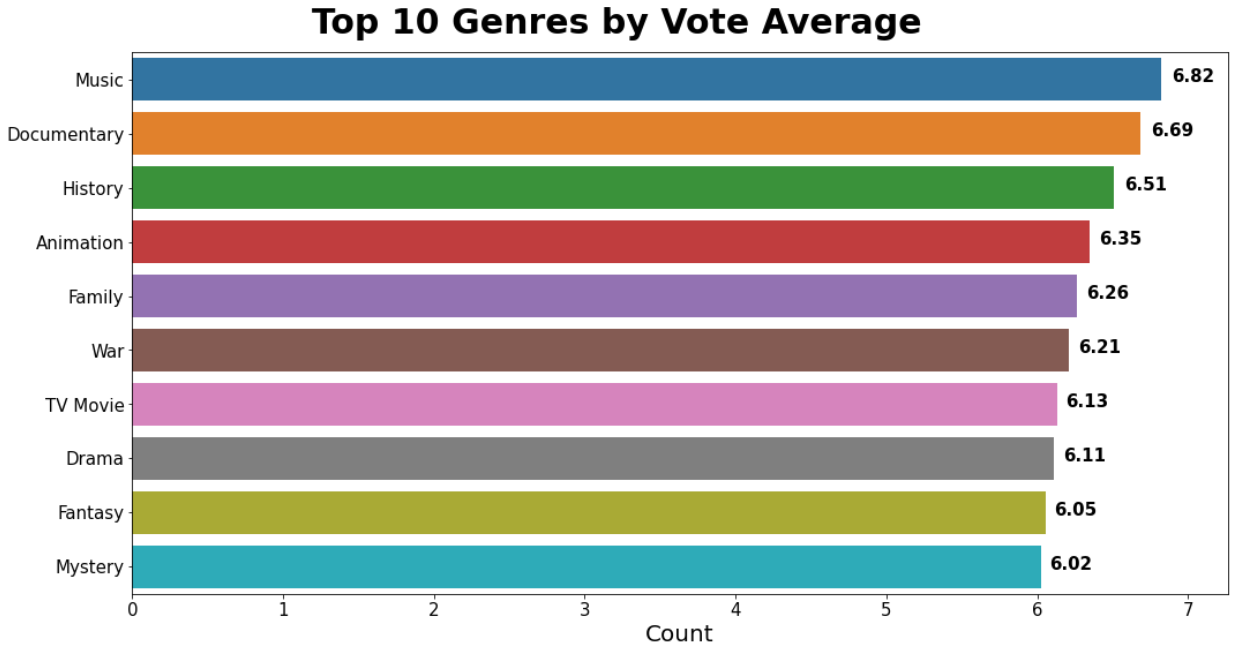


In [110…
```
plot_groupby(genres_df, 'revenue_musd', 'mean', title='Top 10 Genres by Averag
```



Top 10 Genres by Vote Average

In [111…
```
plot_groupby(genres_df, 'vote_average', 'mean', title='Top 10 Genres by Vote A
```



What Are The Most Successful Production Companies?

• In this section, we will compare production companies based on movie count, revenue, and popularity.

In [112…
```
production_df = df.copy()
production_df['company'] = production_df['production_companies'].apply(lambda
```

```
production_df = production_df.explode('company').groupby('company').agg({"titl
production_df.head()
```

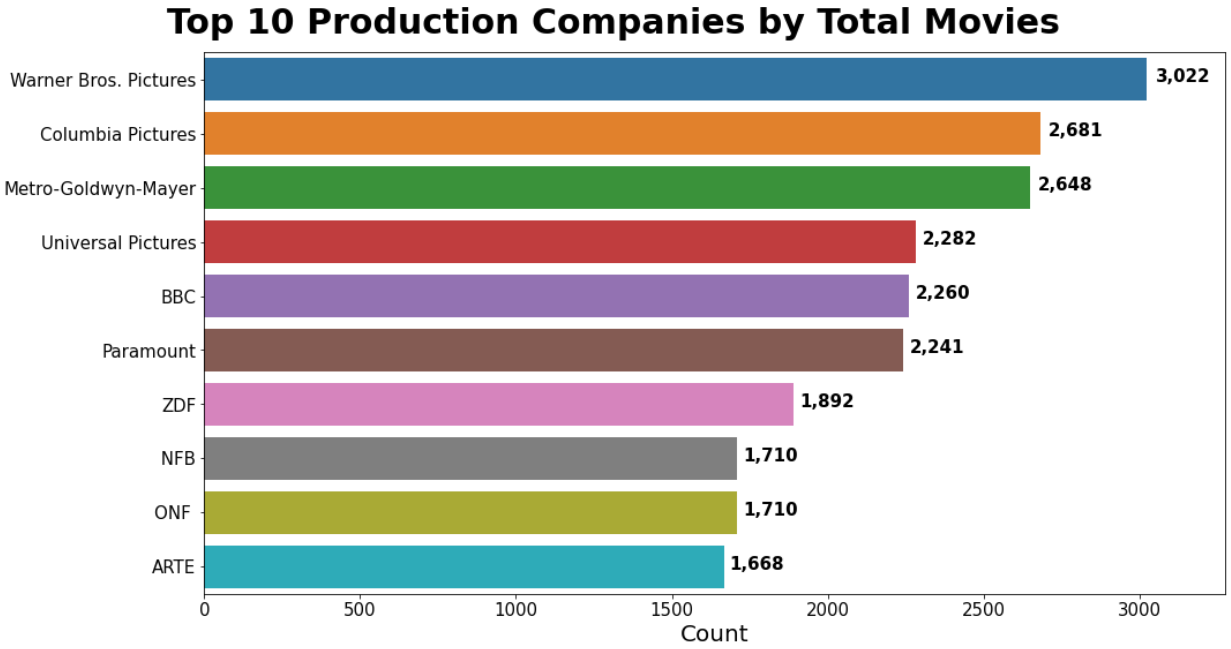
Out[112]:

company	title	vote_average	vote_count	popularity	profit_musd		revenue_musd	
	count	mean	sum	mean	mean	sum	mean	sum
Fonds Bell	1	NaN	0	0.60	NaN	0.00	NaN	0.00
Fonds des médias du Canada	12	6.49	61	1.75	NaN	0.00	NaN	0.00
Movie Studios	2	5.50	4	1.97	NaN	0.00	NaN	0.00
NFB	1710	6.20	3626	0.80	-0.51	-1.54	0.24	1.18
Sat1	1	6.00	2	0.75	NaN	0.00	NaN	0.00

Top 10 Production Companies by Total Movies

In [113…

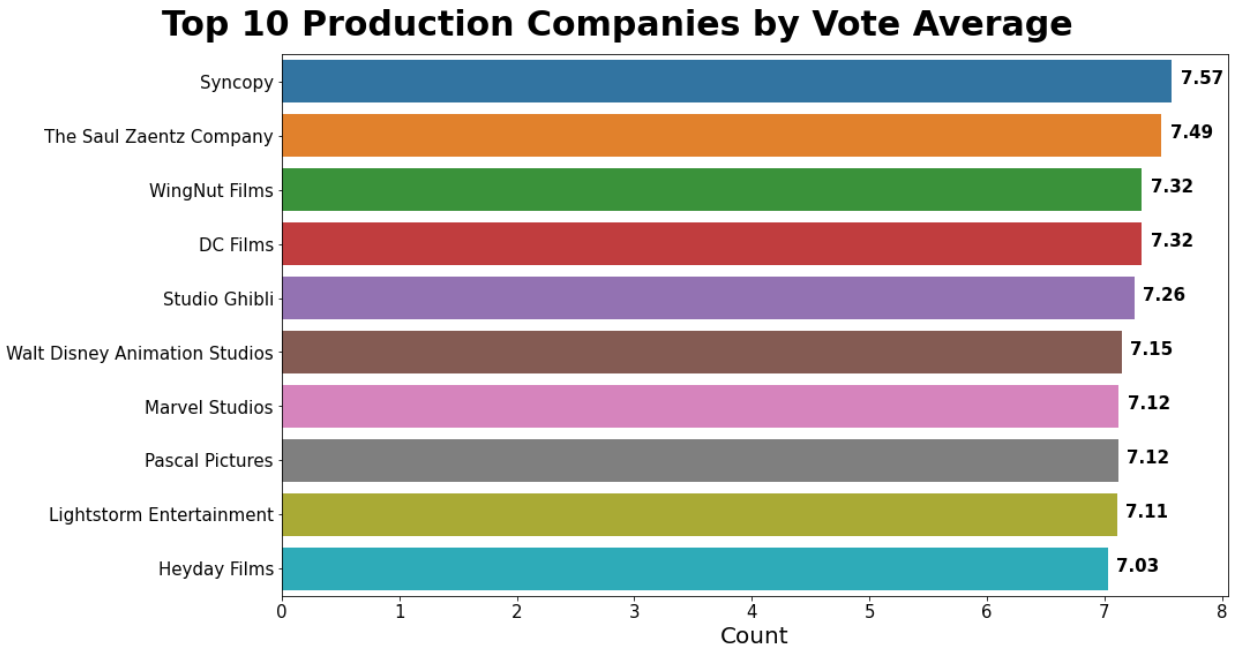
```
plot_groupby(production_df, 'title', 'count', title='Top 10 Production Compani
```



Top 10 Production Companies by Vote Average

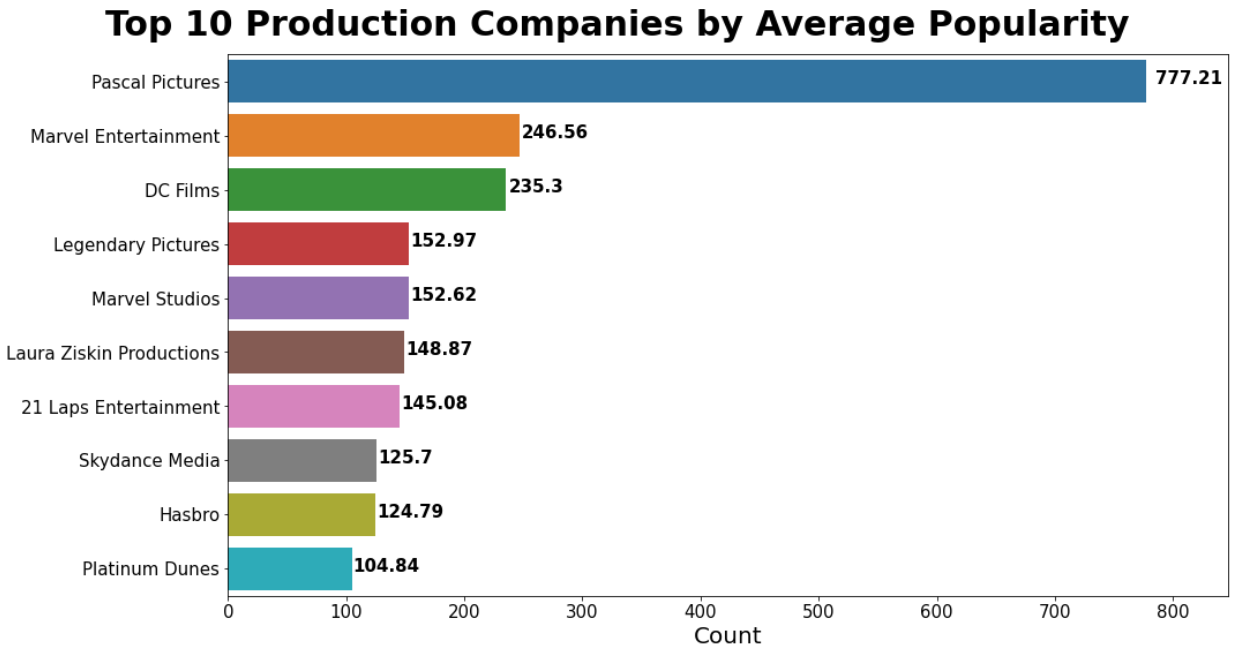
In [114…

```
plot_groupby(production_df, 'vote_average', 'mean', title='Top 10 Production C
```



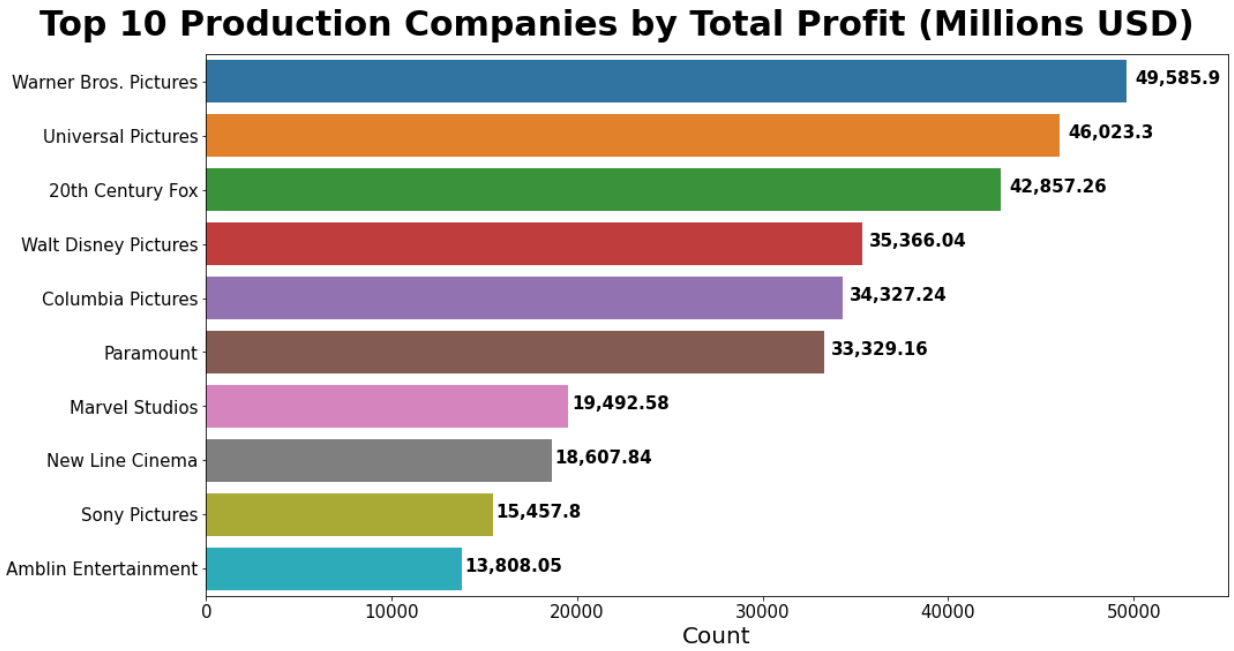
Top 10 Production Companies by Average Popularity

In [116…
```
plot_groupby(production_df, 'popularity', 'mean', title='Top 10 Production Com
```

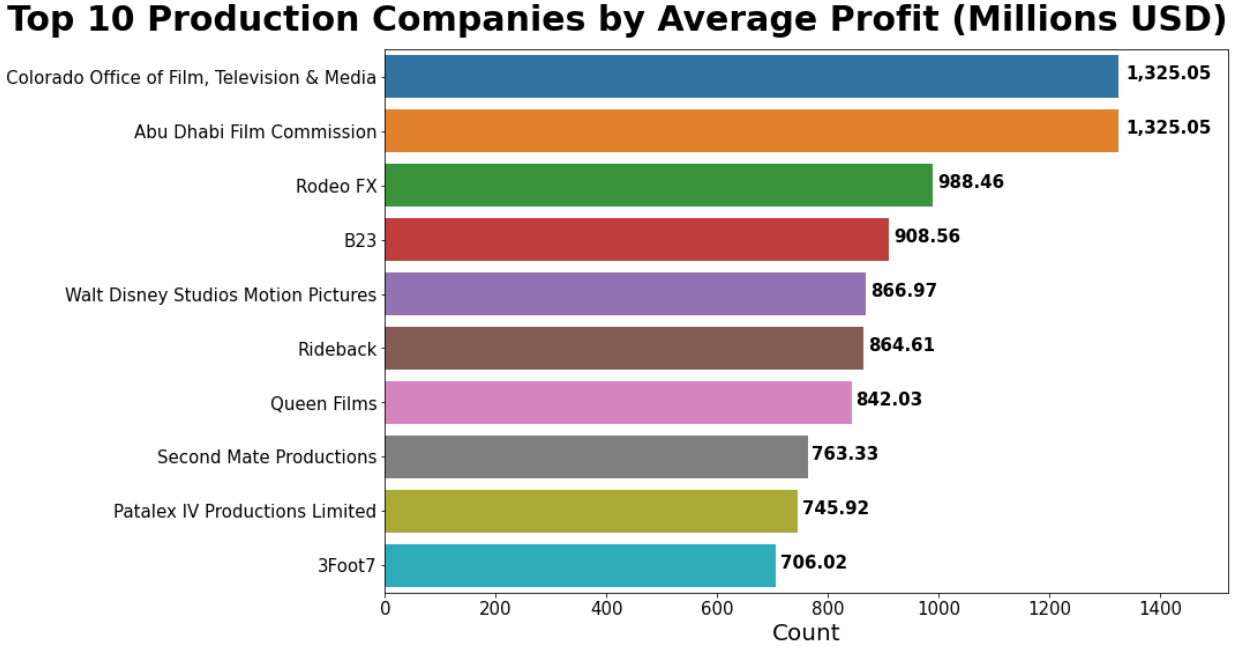


Top 10 Production Companies by Profit

In [117…
```
plot_groupby(production_df, 'profit_musd', 'sum', title='Top 10 Production Com
```

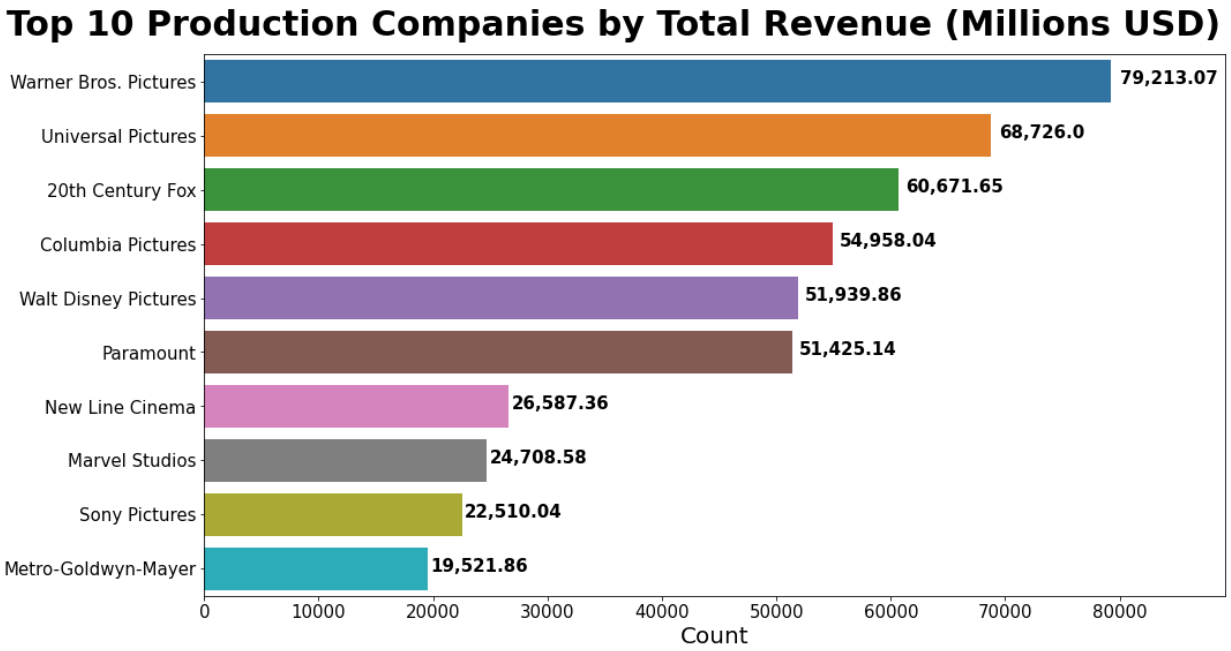


In [118…
```
plot_groupby(production_df, 'profit_musd', 'mean', title='Top 10 Production Co
```

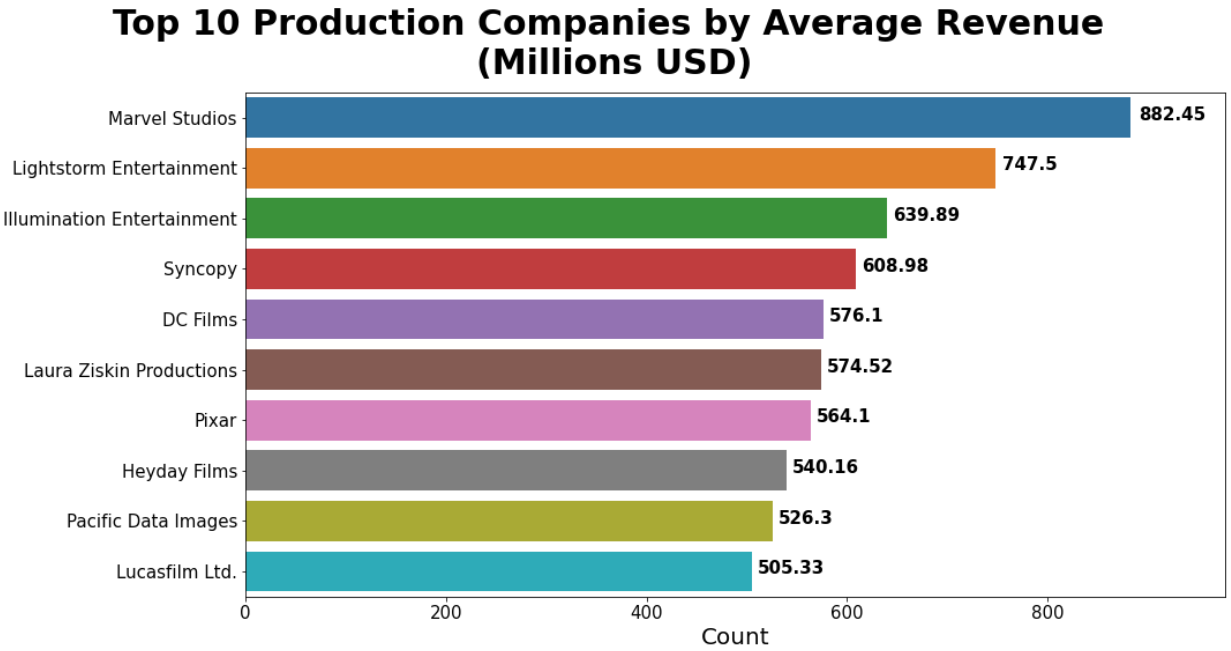


Top 10 Production Companies by Revenue

In [123…
```
plot_groupby(production_df, 'revenue_musd', 'sum', title='Top 10 Production Co
```



In [124…
```
plot_groupby(production_df, 'revenue_musd', 'mean', title='Top 10 Production C
```



Who Are The Most Successful Actors?

• In this section, we will compare actors based on total movie count, revenue, popularity, and vote average.

In [126…
```
actor = df.set_index('id').cast_names.str.split('|', expand=True)
actor = actor.stack().reset_index(level=1, drop=True).to_frame()
actor.columns = ['Actor']
actor.head()
```

Out[126]:

id	Actor
3924	Penny Singleton
3924	Arthur Lake
3924	Larry Simms
3924	Daisy
3924	Ann Doran

In [127…
```
actor = actor.merge(df[['title','revenue_musd','vote_average', 'vote_count', '
actor.head()
```

Out[127]:

	Actor	title	revenue_musd	vote_average	vote_count	popularity
2	Turo Pajala	Love at Twenty	NaN	6.80	36.00	4.99
2	Susanna Haavisto	Love at Twenty	NaN	6.80	36.00	4.99
2	Matti Pellonpää	Love at Twenty	NaN	6.80	36.00	4.99
2	Eetu Hilkamo	Love at Twenty	NaN	6.80	36.00	4.99
2	Erkki Pajala	Love at Twenty	NaN	6.80	36.00	4.99

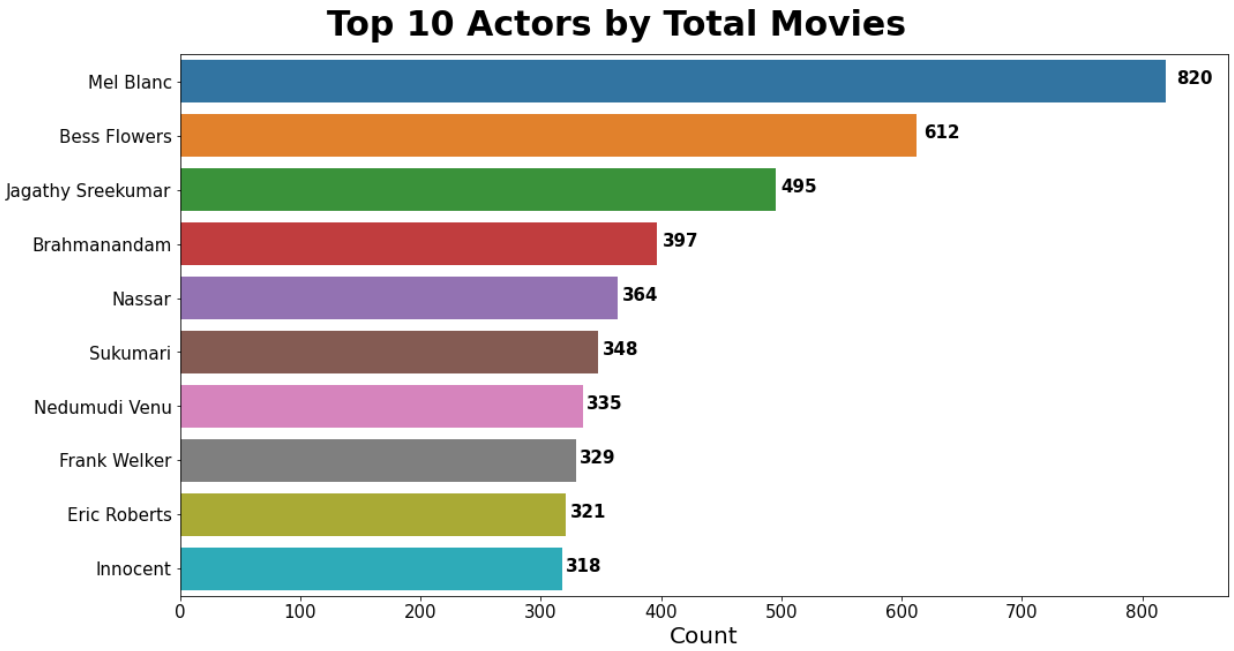
In [129…
```
agg = actor.groupby('Actor').agg({
    'revenue_musd': ['sum', 'mean'],
    'vote_average': 'mean',
    'vote_count': 'sum',
    'popularity': 'mean',
    'title': 'count'
})

agg.head()
```

Out[129]:

	revenue_musd		vote_average	vote_count	popularity	title
	sum	mean	mean	sum	mean	count
Actor						
	0.00	NaN	NaN	0.00	NaN	0
\t Corrado Fortuna	0.00	NaN	NaN	0.00	0.60	1
\tBenoît B. Mandelbrot	0.00	NaN	4.60	11.00	1.79	3
\tCirilo Fernández	0.00	NaN	3.80	2.00	1.23	1
\tDouglas Hegdahl	1.30	1.30	6.60	5.00	3.45	1

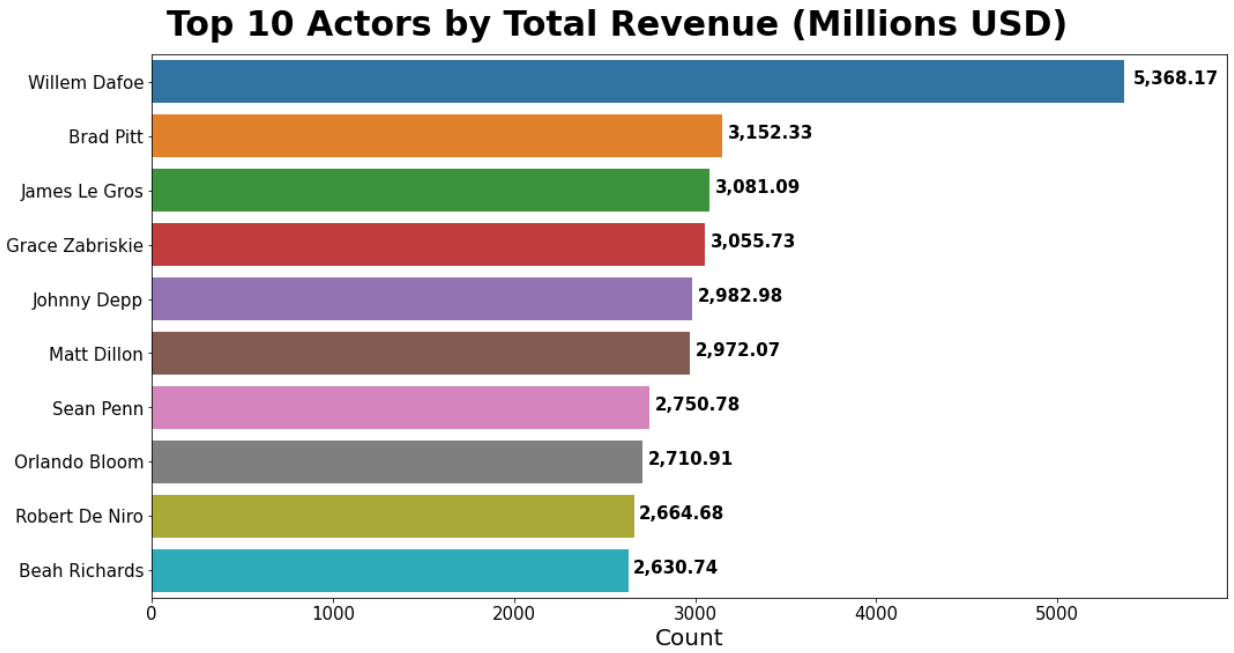
Top 10 Actors by Total Movies

In [130…
```
plot_groupby(agg, 'title', 'count', 'Top 10 Actors by Total Movies', min_votes
```

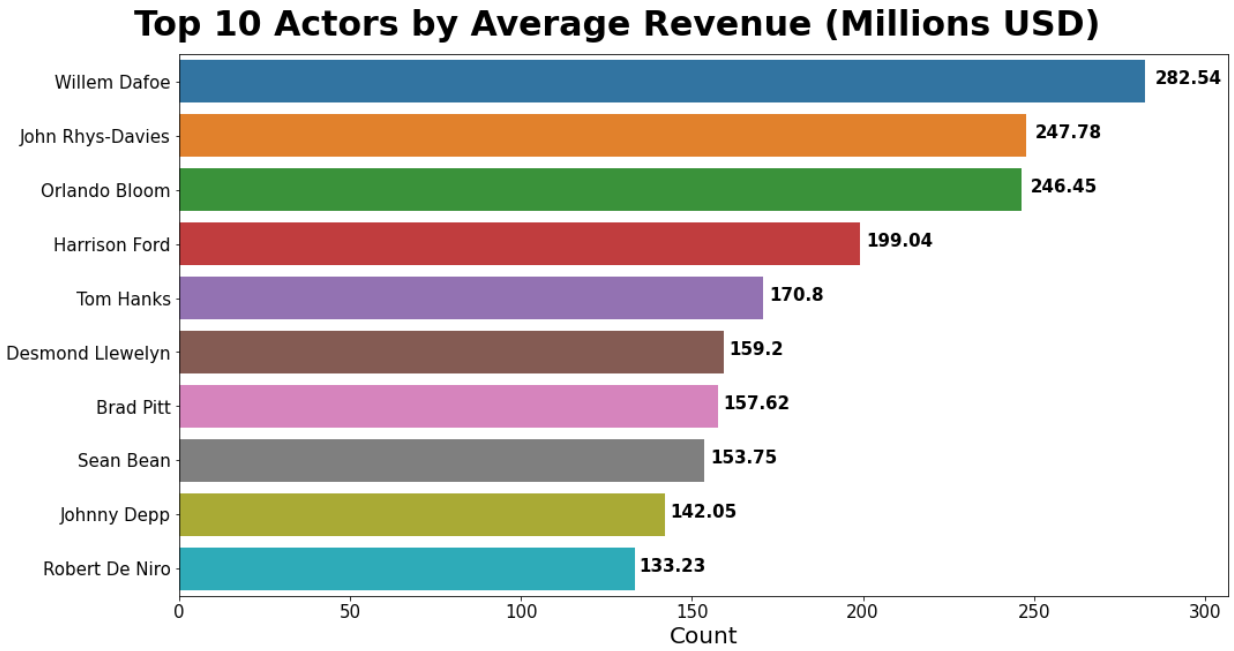


Top 10 Actors by Revenue

In [282…
```
plot_groupby(agg, 'revenue_musd', 'sum', 'Top 10 Actors by Total Revenue (Mill
```

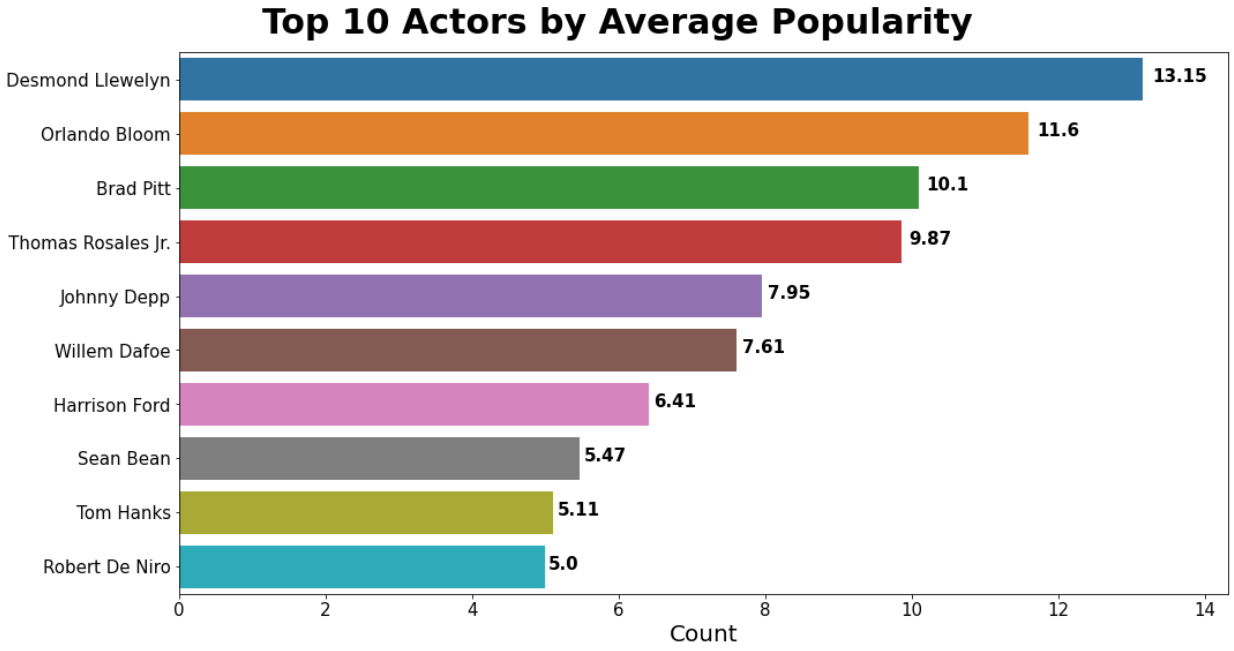


In [285…
```
plot_groupby(agg, 'revenue_musd', 'mean', 'Top 10 Actors by Average Revenue (M
```



Top 10 Actors by Popularity

In [294…
```
plot_groupby(agg, 'popularity', 'mean', 'Top 10 Actors by Average Popularity',
```



In [132…
```
# Saving the updated dataframe to a csv file.
df.reset_index(drop=True, inplace=True)
df.drop('index', axis=1, inplace=True)
df.to_csv('movies_complete.csv', index=False)
```

This concludes the Exploratory Data Analysis part of the project.

In the next session, we will create our model based on movie attributes using Natural Language Processing.