Exploratory data analysis of Movies extracted from TMDB API

1. Importing Libraries

In [37]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import plotly.offline as pyo
from plotly.offline import init_notebook_mode,iplot
init_notebook_mode(connected=True)
import warnings
warnings.filterwarnings("ignore")
import ast
import plotly.figure_factory as ff
import plotly.express as px
```

2. Reading data from csv files which were extracted from API (file - Movie Data Collection.ipynb)

In [38]:
```
movies = pd.read_csv('moviedata.csv')
cast = pd.read_csv('cast_crew.csv')
```

In [39]:
```
movies = movies.merge(cast, on='id')
```

In [40]:
```
movies.head(5)
```

Out[40]:

	Unnamed: 0_x	id	budget	genres	overview	popularity	production_companies	release_date	revenue	title	vote_average	original_language	vote_count	Unnamed: 0_y	cast	
0	0	675353	110000000	[{'id': 28, 'name': 'Action'}, {'id': 878, 'na...	After settling in Green Hills, Sonic is eager ...	9124.409	[{'id': 113750, 'logo_path': '/A3QVZ9Ah0yI2d2G...	2022-03-30	355200000	Sonic the Hedgehog 2	7.7	en	1395	0	[{'adult': False, 'gender': 2, 'id': 222121, '...	[{'a 'ge
1	1	335787	120000000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	A young street-smart, Nathan Drake and his wis...	4274.019	[{'id': 5, 'logo_path': '/71BqEFAF4V3qjjMPCpLu...	2022-02-10	395124202	Uncharted	7.2	en	1954	1	[{'adult': False, 'gender': 2, 'id': 1136406, ...	[{'a 'ge
2	2	414906	185000000	[{'id': 80, 'name': 'Crime'}, {'id': 9648, 'na...	In his second year of fighting crime, Batman u...	3923.239	[{'id': 101405, 'logo_path': None, 'name': '6t...	2022-03-01	764573021	The Batman	7.8	en	4579	2	[{'adult': False, 'gender': 2, 'id': 11288, 'k...	[{'a 'ge
3	3	629542	80000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	When the infamous Bad Guys are finally caught ...	3511.281	[{'id': 521, 'logo_path': '/kP7t6RwGz2AvvTkvnl...	2022-03-17	165558000	The Bad Guys	7.8	en	415	3	[{'adult': False, 'gender': 2, 'id': 6807, 'kn...	[{'a 'ge 5
4	4	453395	200000000	[{'id': 14, 'name': 'Fantasy'}, {'id': 28, 'na...	Doctor Strange, with the help of mystical alli...	3770.076	[{'id': 420, 'logo_path': '/hUzeosd33nzE5MCNsZ...	2022-05-04	688000000	Doctor Strange in the Multiverse of Madness	7.5	en	1577	4	[{'adult': False, 'gender': 2, 'id': 71580, 'k...	[{'a 'ge

3. Data Cleaning

In [41]:
```
del movies['Unnamed: 0_x']
del movies['Unnamed: 0_y']
del movies['overview']
```

In [42]:
```
movies.isnull().sum()
```

Out[42]:
```
id                       0
budget                   0
genres                   0
popularity               0
production_companies     0
release_date            11
revenue                  0
title                    0
vote_average             0
original_language        0
vote_count               0
cast                     0
crew                     0
dtype: int64
```

In [43]:
```
movies.dropna(inplace=True)
```

3.1 In the dataset genres & production company are list of dictionaries and for our analysis we just need "name" of both the fields.

In [44]:
```
def convert(obj):
    for i in ast.literal_eval(obj):
        return (i['name'])
```

In [45]:
```
movies['genres']= movies['genres'].apply(convert)
movies['production_companies']= movies['production_companies'].apply(convert)
```

In [46]:
```
#movies['crew']=movies['crew'].map(lambda obj:[i['name'] for i in ast.literal_eval(obj) if i['job']=='Director'])
#movies['production_companies']= movies['production_companies'].map(lambda obj:[i['name'] for i in ast.literal_eval(obj)])
```

3.2 For "cast" column we will be doing analyis only on top two cast so fetching data accordingly using convertcast function.

In [47]:
```
def convertcast(obj):
    l=[]
    c=0
    for i in ast.literal_eval(obj):
        if c<2:
            l.append(i['name'])
            c+=1
        else:
            break
    return l
```

In [48]:
```
movies['cast'] = movies['cast'].apply(convertcast)
```

In [49]:
```
movies[['test','cast1','cast2']]=pd.DataFrame(movies['cast'].to_list(), index= movies.index) .reset_index()
```

In [50]:
```
del movies['test']
del movies['cast']
```

In [51]:
```
movies['genres'] = movies['genres'].fillna("None")
```

3.3 For "crew" column we will be doing analyis only where job is "director" so fetching data accordingly using convertcrew function.

In [52]:
```
def convertcrew(obj):
    for i in ast.literal_eval(obj):
        if i['job']=='Director':
            return (i['name'])
```

In [53]:
```
movies['crew']=movies['crew'].apply(convertcrew)
movies.rename(columns={'crew':'director'},inplace=True)
```

In [54]:
```
movies.head()
```

Out[54]:

	id	budget	genres	popularity	production_companies	release_date	revenue	title	vote_average	original_language	vote_count	director	cast1	cast2
0	675353	110000000	Action	9124.409	SEGA	2022-03-30	355200000	Sonic the Hedgehog 2	7.7	en	1395	Jeff Fowler	Ben Schwartz	Idris Elba
1	335787	120000000	Action	4274.019	Columbia Pictures	2022-02-10	395124202	Uncharted	7.2	en	1954	Ruben Fleischer	Tom Holland	Mark Wahlberg
2	414906	185000000	Crime	3923.239	6th & Idaho	2022-03-01	764573021	The Batman	7.8	en	4579	Matt Reeves	Robert Pattinson	Zoë Kravitz
3	629542	80000000	Animation	3511.281	DreamWorks Animation	2022-03-17	165558000	The Bad Guys	7.8	en	415	Pierre Perifel	Sam Rockwell	Marc Maron
4	453395	200000000	Fantasy	3770.076	Marvel Studios	2022-05-04	688000000	Doctor Strange in the Multiverse of Madness	7.5	en	1577	Sam Raimi	Benedict Cumberbatch	Elizabeth Olsen

In [55]:
```
movies['release_date'] = pd.to_datetime(movies.release_date)
```

In [56]:
```
movies['release_year']= pd.DatetimeIndex(movies['release_date']).year
```

4. Analysing dataset

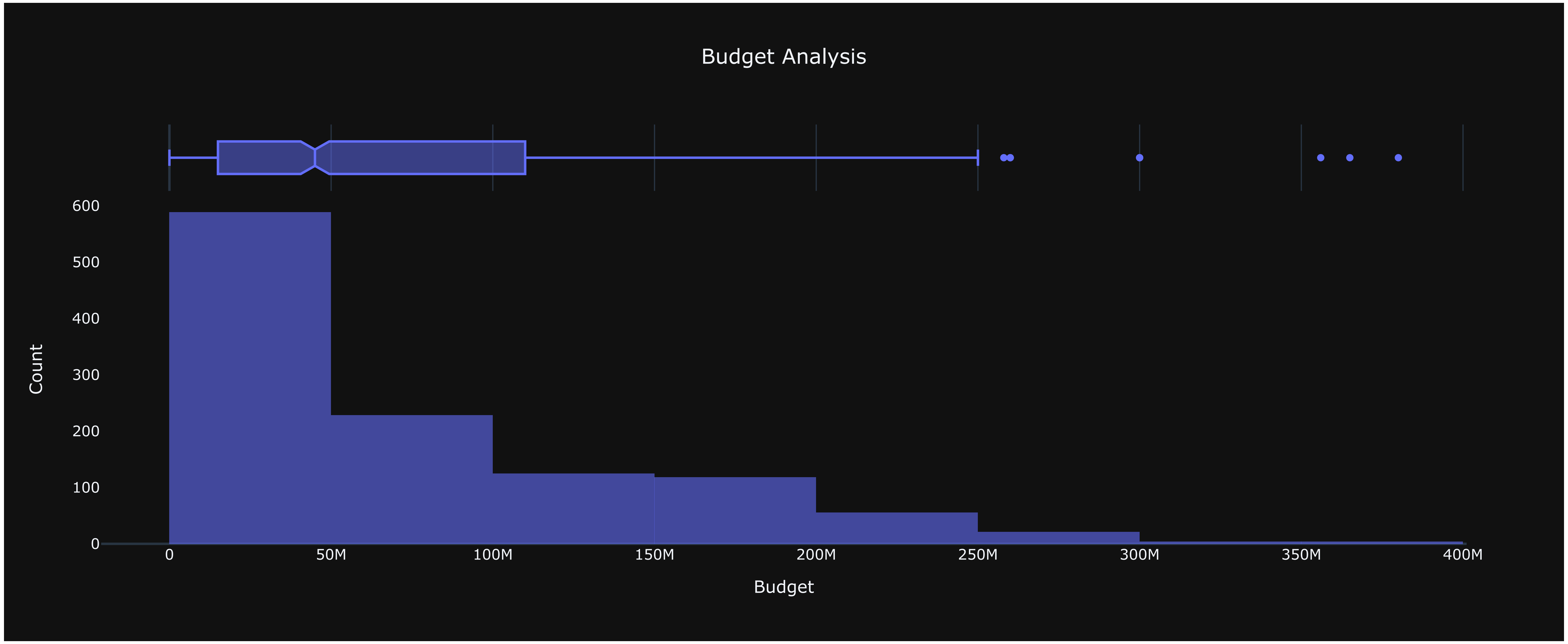
4.1 Budget Analysis and Revenue Analysis

In [57]:
```
movies[(movies['budget']==0) | (movies['revenue']==0) ]['id'].nunique()
```

Out[57]:
```
936
```

Many values for budget and revenue are zero so we will be excluding those in our analysis

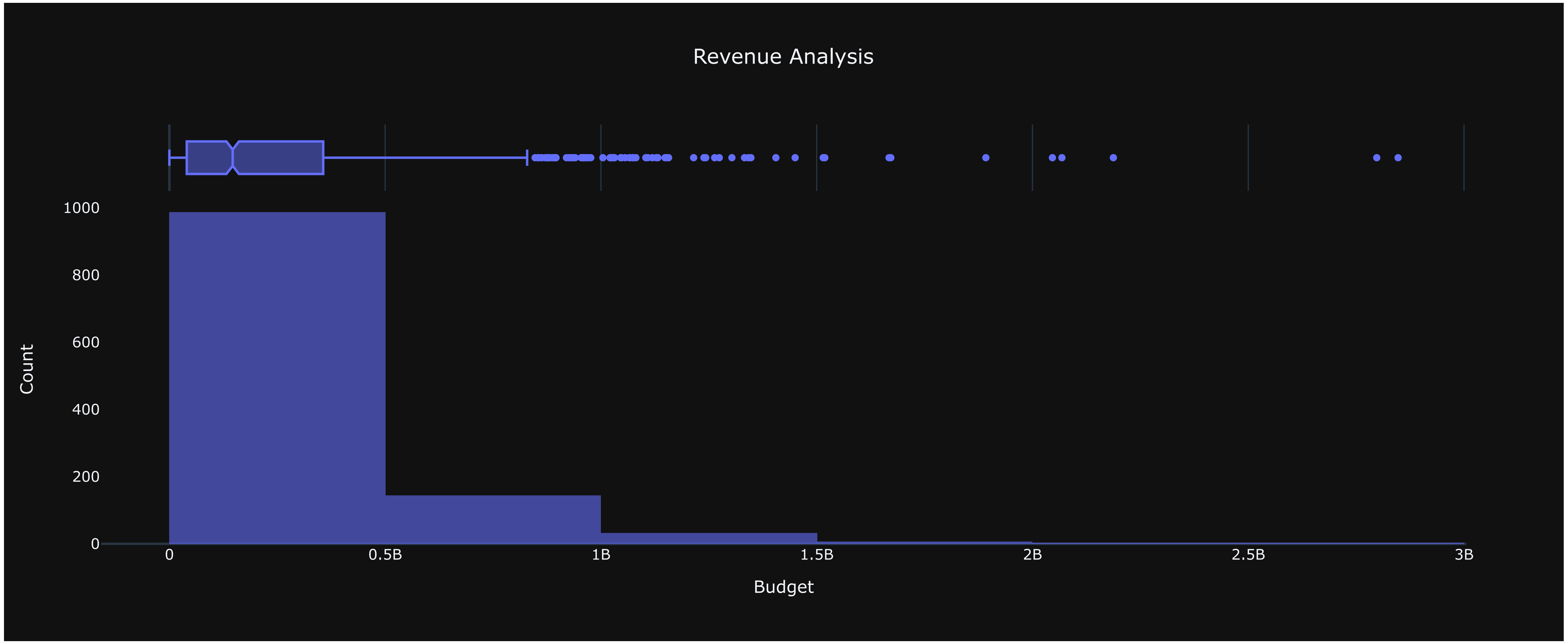
In [58]:
```
fig = px.histogram(movies,x=movies[movies['budget']!=0]['budget'],marginal='box',opacity=0.6,nbins=10,
                   title='Budget Analysis',template = 'plotly_dark')

fig.update_layout(hovermode="x unified",xaxis_tickangle=360,xaxis_title='Budget',title_x=0.5,
                  yaxis={'title':'Count','showgrid':False})
pyo.iplot(fig)
```



1. Graph is right-skewed , as most of the movies are in budget 0-49 M having median value 45M.

2. Few of the outliers are having budget more than 300M.

```
In [59]: fig = px.histogram(movies,x=movies[movies['revenue']!=0]['revenue'],marginal='box',opacity=0.6,nbins=10,
                          title='Revenue Analysis',template = 'plotly_dark')

         fig.update_layout(hovermode="x unified",xaxis_tickangle=360,
                          xaxis_title='Budget',title_x=0.5,yaxis={'title':'Count','showgrid':False})
         iplot(fig)
```

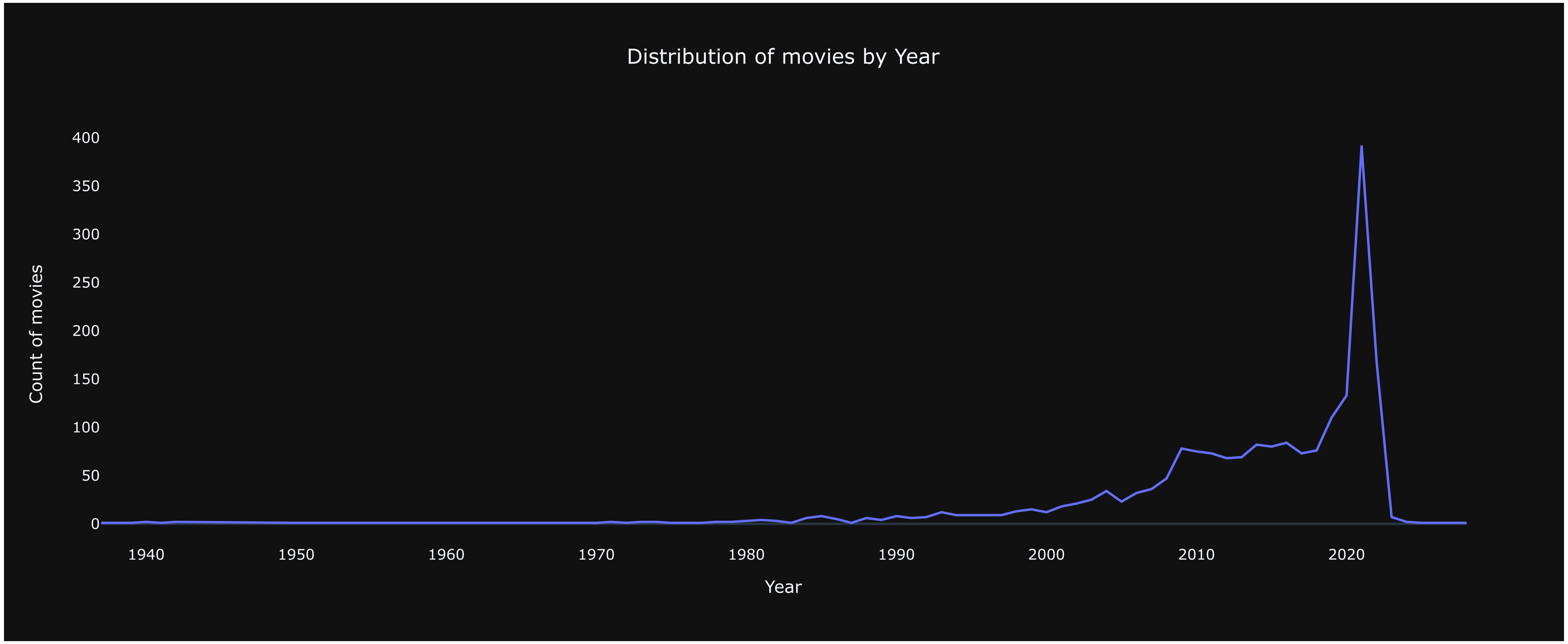


1. Graph is right-skewed , as most of the movies earn revenue 0-499 M having median value 146M.

2. Few of the outliers are having budget more than 2B.

4.2 Yearly distribution of movies

In [60]:
```
moviebyyear = movies.release_year.value_counts().reset_index()
moviebyyear.columns=['release_year','count']
moviebyyear = moviebyyear.sort_values(by='release_year',ascending=False)
data = go.Scatter(x=moviebyyear['release_year'],y=moviebyyear['count'],connectgaps=False)
layout = go.Layout(title='Distribution of movies by Year',title_x=0.5,
                  xaxis={'title':'Year'},
                  yaxis ={'title':'Count of movies'})
fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_xaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```

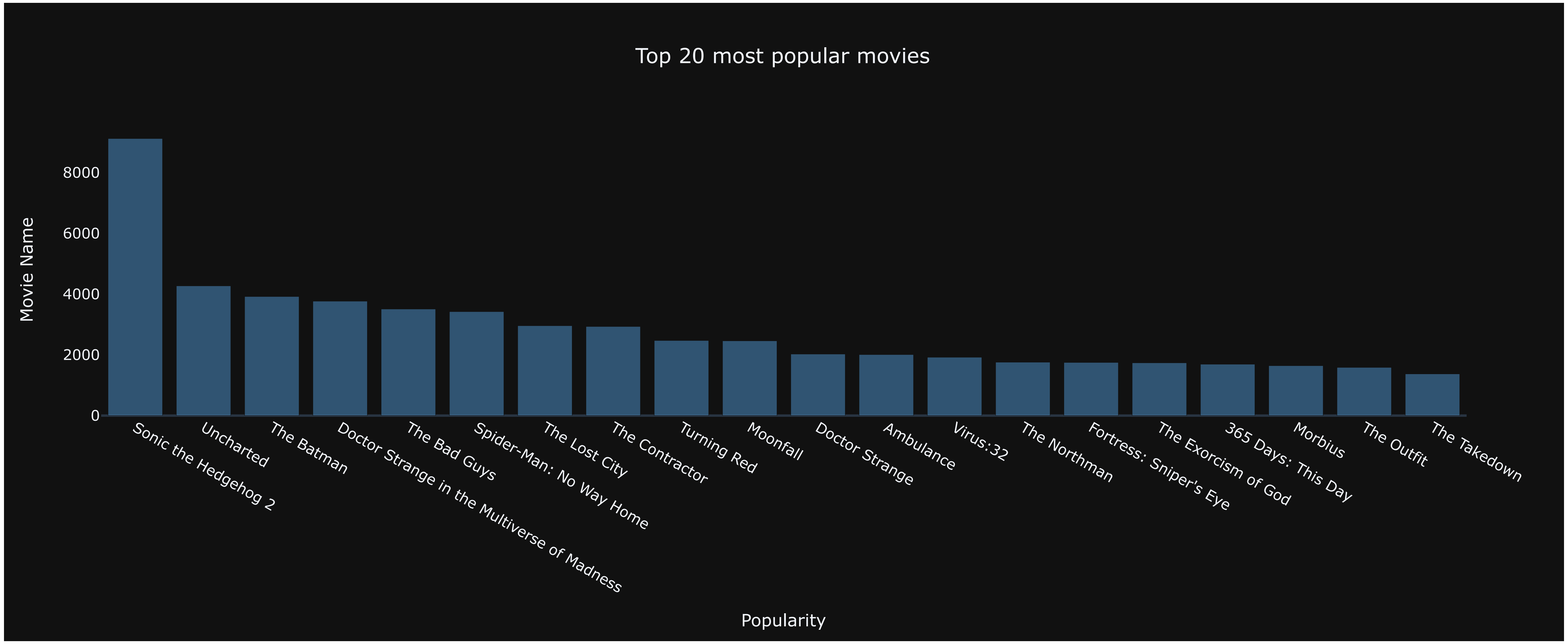


1. Dataset mostly contains movies for past 12 years having only few movies before 2000.

2. Most the movies data is from 2021.

4.3 Most popular movies.

In [61]:
```
popularmovies = movies.sort_values(by='popularity',ascending=False)[['title','popularity','release_year']].head(20)

data = go.Bar(y= popularmovies['popularity'],x=popularmovies['title'],orientation='v',marker={'color':'steelblue'},
             opacity=0.6,hovertext=popularmovies['release_year'],
             hovertemplate=" Popularity : %{x}" " <br>Movie Name : %{y}" "<br> Release Year :%{hovertext}</b><extra></extra>")

layout = go.Layout(title='Top 20 most popular movies',title_x=0.5,
               xaxis={'title':'Popularity'},
               yaxis={'title':'Movie Name'})


fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```

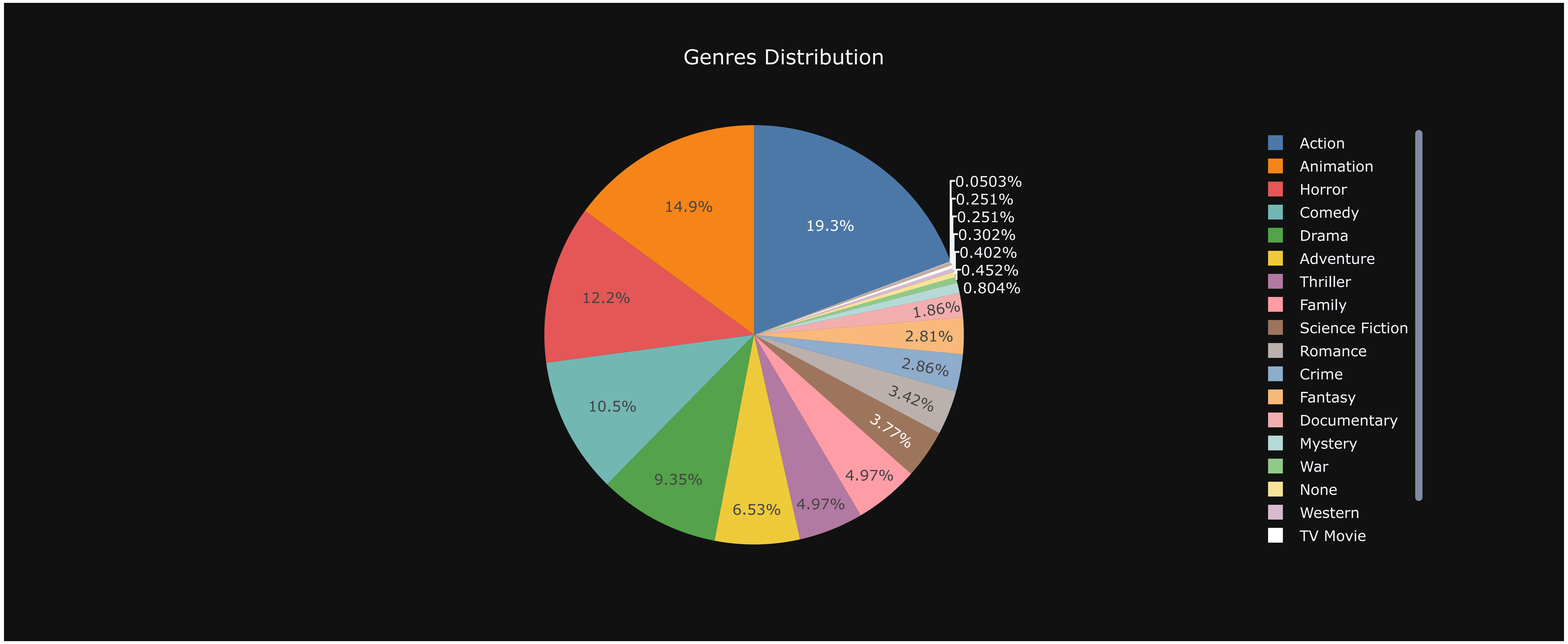


1. Most popular movie is "Sonic the hedgehog2" followed by "Uncharted and the Bataman".

2. We can also see all top 20 movies are from 2022 or 2021, so this might be because people started voting more on recent movies.

4.4 Genre Distribution

In [62]:
```
genres = movies.genres.value_counts().reset_index()
genres.columns=['genres','count']
genres = genres.sort_values(by='count',ascending=False)

fig = px.pie(genres,values='count',names='genres',title='Genres Distribution',
             color_discrete_sequence=px.colors.qualitative.T10)
fig.update_layout(template = 'plotly_dark',title_x=0.5)
fig.show()
```

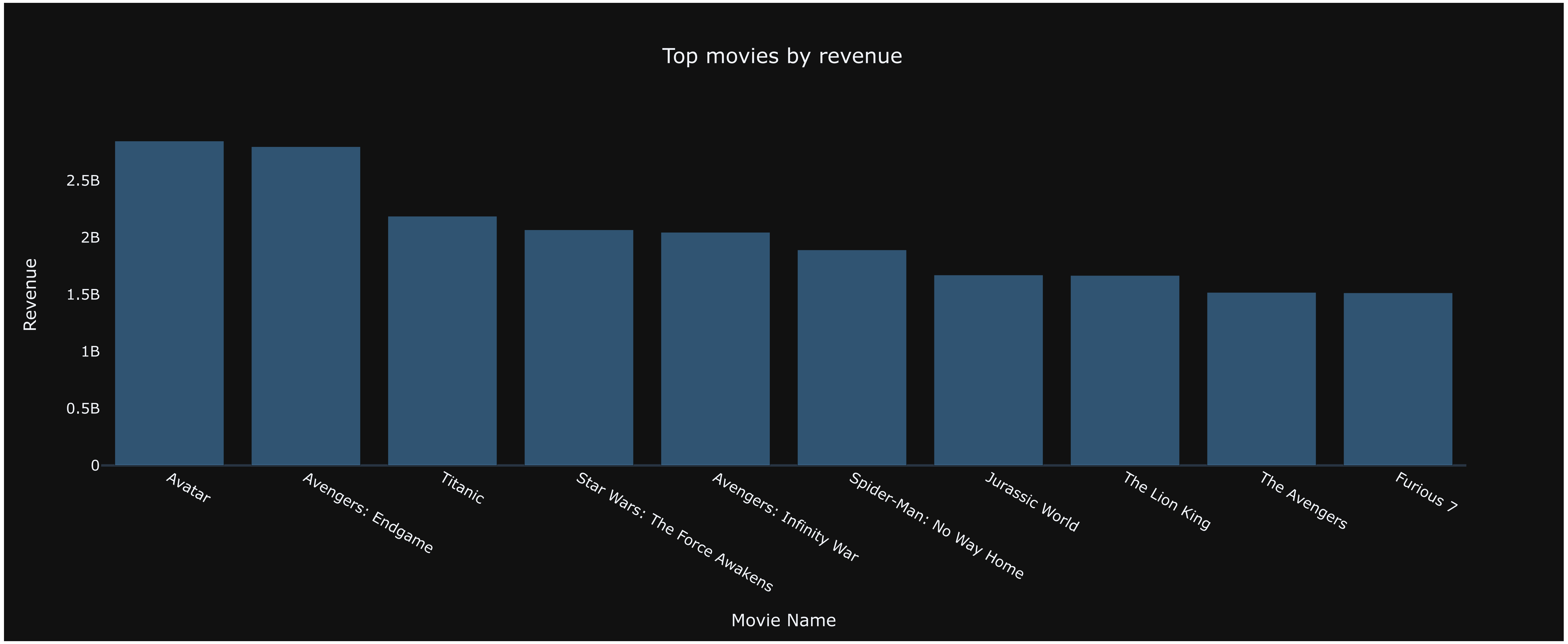


1. Mostly movies are made of "Action" genre having around 20% of all movies, maybe because people prefer watching Action movies.

2. Animated movies and comedy also prefferred among individuals.

4.5 Top movies by revenue

In [63]:
```
revenue = movies.sort_values(by='revenue',ascending=False)[['title','revenue','production_companies']].head(10)

data = go.Bar(x= revenue['title'],y=revenue['revenue'],orientation='v',marker={'color':'steelblue'},
             opacity=0.6,hovertext=revenue['production_companies'],
             hovertemplate=" Revenue : %{y}" " <br>Movie Name : %{x}" "<br> Production Company : %{hovertext}</b><extra></extra>")

layout = go.Layout(title='Top movies by revenue',title_x=0.5,
               yaxis={'title':'Revenue'},
               xaxis={'title':'Movie Name'})


fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```

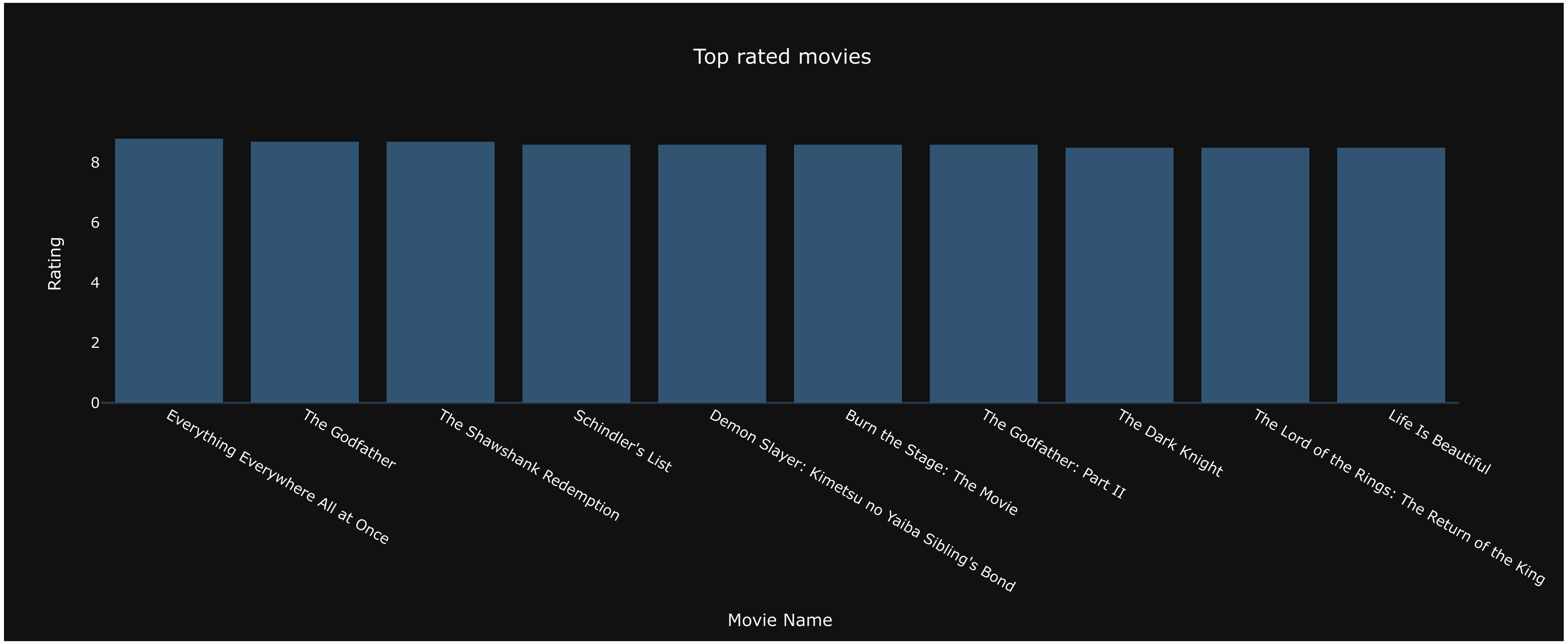


1. "Avtaar" movie of "Dune Entertainment" production company has generated most revenue of around 2.4B, followed by "Avengers: Endgame" of "Marvel Studios".

2. There are 4 movies of "Marvel Studios" in top 10 movies list by revenue.

4.6 Top rated movies

In [64]:
```
vote = movies.sort_values(by='vote_average',ascending=False)[['title','vote_average','director']].head(10)

data = go.Bar(x= vote['title'],y=vote['vote_average'],orientation='v',marker={'color':'steelblue'},
             opacity=0.6,hovertext=vote['director'],
             hovertemplate=" Rating : %{y}" " <br>Movie Name : %{x}" "<br>  Director : %{hovertext}</b><extra></extra>")

layout = go.Layout(title='Top rated movies',title_x=0.5,
               yaxis={'title':'Rating'},
               xaxis={'title':'Movie Name'})


fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```

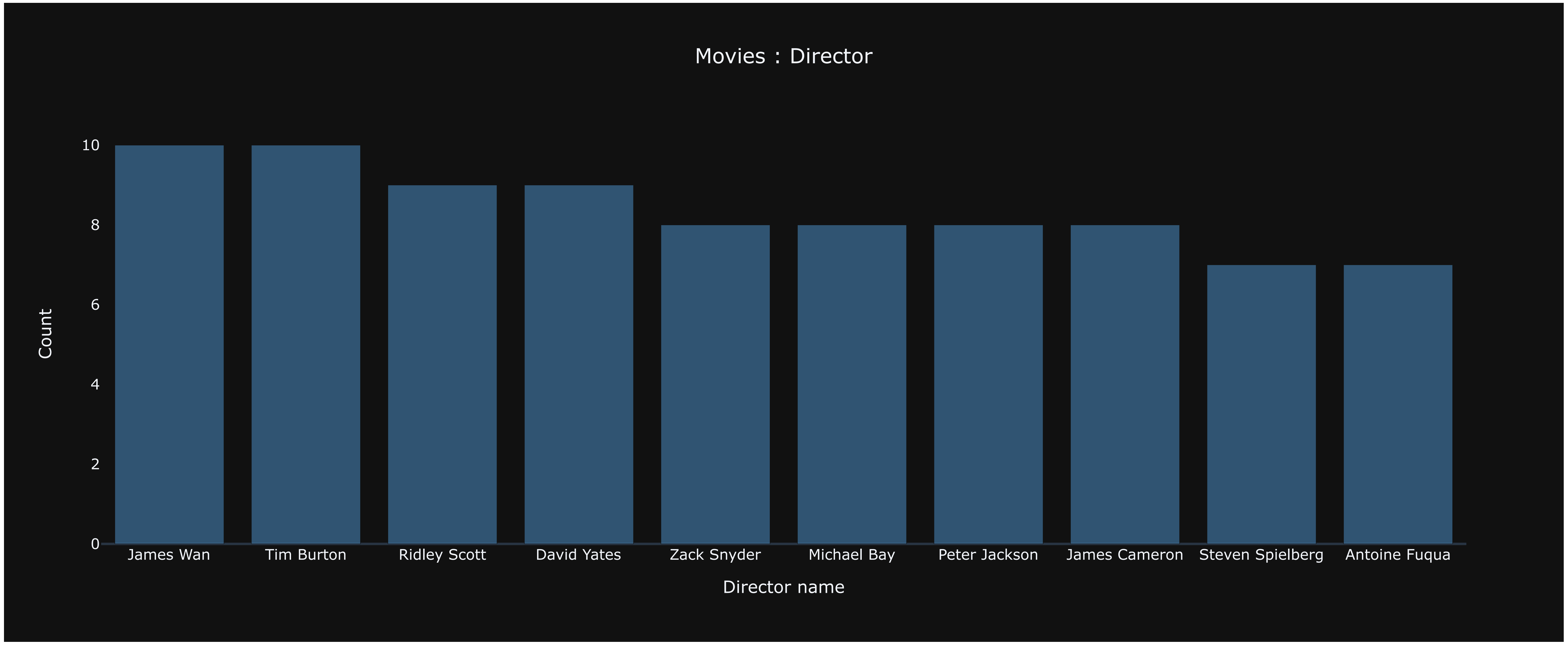


1. Almost all top movies have rating on the range of 8.7-8.5 .

2. "Daniel Scheinert's Everything Everywhere All at once" is most favourite with rating of 8.8

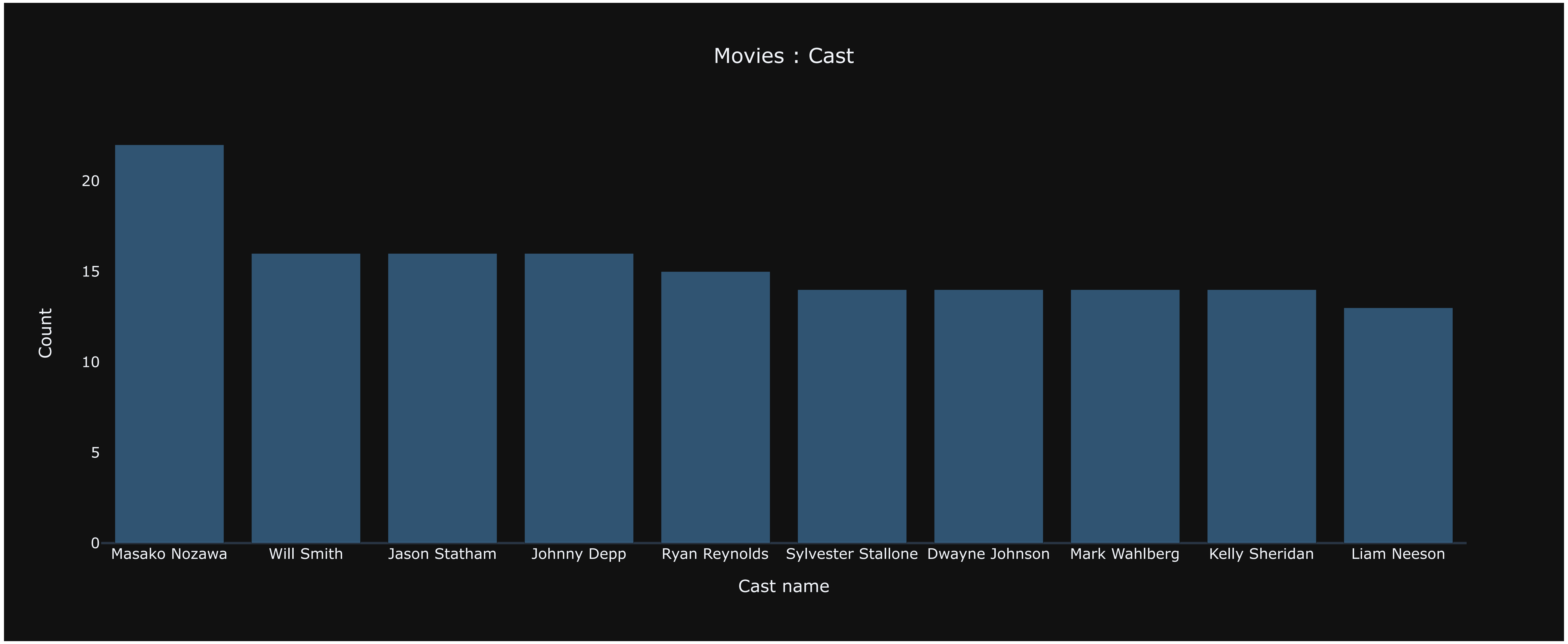
4.7 Movies: Director

In [65]:
```
director = movies.director.value_counts().reset_index().head(10)
director.columns=['director','count']
director = director.sort_values(by='count',ascending=False)

data = go.Bar(x= director['director'],y=director['count'],orientation='v',marker={'color':'steelblue'},
             opacity=0.6)

layout = go.Layout(title='Movies : Director',title_x=0.5,
               xaxis={'title':'Director name'},
               yaxis={'title':'Count'})


fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```



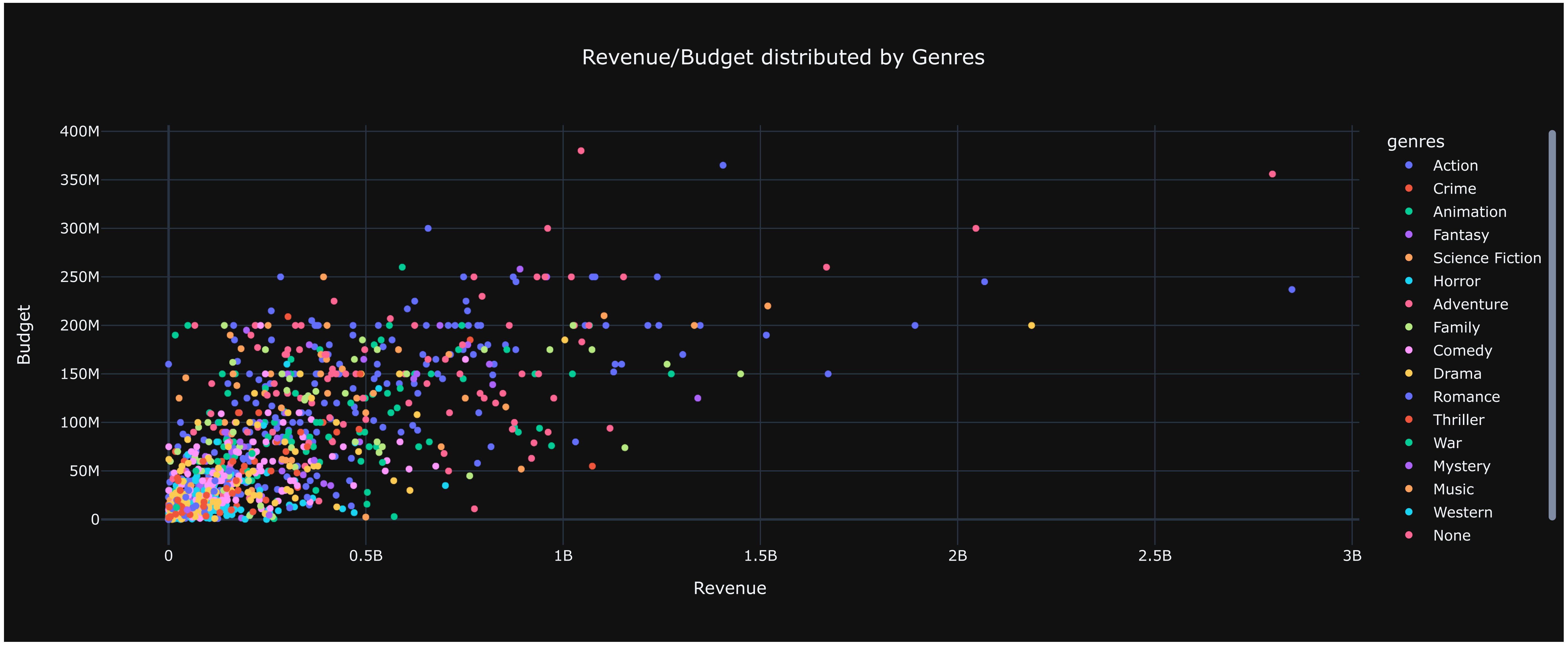
4.8 Movies: Cast

In [66]:
```
cast = pd.concat([movies['cast1'],movies['cast2']]).reset_index()
cast.dropna(inplace=True)
cast.columns=['index','castname']
cast = cast.castname.value_counts().reset_index().head(10)
cast.columns=['castname','count']
cast = cast.sort_values(by='count',ascending=False)


data = go.Bar(x= cast['castname'],y=cast['count'],orientation='v',marker={'color':'steelblue'},
             opacity=0.6)

layout = go.Layout(title='Movies : Cast',title_x=0.5,
               xaxis={'title':'Cast name'},
               yaxis={'title':'Count'})


fig = go.Figure(data=data,layout=layout)
fig.update_yaxes(showgrid=False)
fig.update_layout(template = 'plotly_dark')

iplot(fig)
```



4.9 Correlation between Revenue and Budget distributed by Genres

```
In [67]: df = movies[(movies['revenue']!=0) & (movies['budget']!=0)][['budget','revenue','genres','title']]
         fig = px.scatter(df , y='budget', x='revenue', hover_data=['title'],
                         color='genres',title ='Revenue/Budget distributed by Genres')

         #fig.update_xaxes(showgrid=False)
         #fig.update_yaxes(showgrid=False)
         fig.update_layout(template = 'plotly_dark')
         fig.update_layout(
                           xaxis_title='Revenue',title_x=0.5,yaxis_title='Budget')
         iplot(fig)
```



1. Graph is scattered close to origin as majority of movies have less revenue and budget.

2. There are lot movies having same budget of 100M , 150 M and 200 M .

3. "Titanic" movie can be seen having revenue of 2.2 B with only 200 M budget making it one of most successful movies.

4. Coorelation between Revenue and Budget is kind of linear when Budget increases revenue tends to increase.