Data Mining and Business Intelligence (ITA5007)

Digital Assignment 1

Lhingnunching L, 22MBS0007

TOPIC: Visualizing the 'IMDb Top 250 films' data

Importing the necessary libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.io as pio
pio.renderers.default = 'notebook'
import warnings

warnings.filterwarnings("ignore")
```

```
C:\Users\nunun\AppData\Roaming\Python\Python38\site-packages\pandas\core\computation\expressions.py:20: U
serWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

I. Loading the data

Dataset downloaded from: https://github.com/ExplorerMunchkin/web-scraping-imdb-top250-python/blob/main/IMDBTop250.csv (https://github.com/ExplorerMunchkin/web-scraping-imdb-top250-python/blob/main/IMDBTop250.csv)

In [2]:

```
data = pd.read_csv('IMDBTop250.csv')
```

Dropping Null values

In [3]:

```
df = data.dropna()
```

II. Exploratory the Data

In [4]:

```
df.columns
```

Out[4]:

```
Index(['ranking', 'movieTitle', 'movieYear', 'rating', 'voteCount',
       'censorRating', 'movieLength', 'runtime', 'genre', 'movieMonth',
       'releaseDate', 'summary', 'starList', 'writerList', 'director',
       'country', 'language', 'budget', 'gross_worldwide', 'production',
       'url'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 209 entries, 0 to 246
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ranking          209 non-null    int64
 1   movieTitle       209 non-null    object
 2   movieYear        209 non-null    int64
 3   rating           209 non-null    float64
 4   voteCount        209 non-null    object
 5   censorRating     209 non-null    object
 6   movieLength      209 non-null    object
 7   runtime          209 non-null    float64
 8   genre            209 non-null    object
 9   movieMonth       209 non-null    int64
 10  releaseDate      209 non-null    object
 11  summary          209 non-null    object
 12  starList         209 non-null    object
 13  writerList       209 non-null    object
 14  director         209 non-null    object
 15  country          209 non-null    object
 16  language         209 non-null    object
 17  budget           209 non-null    object
 18  gross_worldwide  209 non-null    object
 19  production       209 non-null    object
 20  url              209 non-null    object
dtypes: float64(2), int64(3), object(16)
memory usage: 35.9+ KB
```

In [6]:

```
df.head()
```

Out[6]:

	ranking	movieTitle	movieYear	rating	voteCount	censorRating	movieLength	runtime	genre	movieMonth	...	su
0	1	The Shawshank Redemption	1994	9.3	24,08,140	R	2h 22min	142.0	Drama	10	...	imp me nu
1	2	The Godfather	1972	9.2	16,66,445	R	2h 55min	175.0	Crime,Drama	3	...	or dy p
2	3	The Godfather: Part II	1974	9.0	11,57,841	R	3h 22min	202.0	Crime,Drama	12	...	Tl c C
3	4	The Dark Knight	2008	9.0	23,67,334	PG-13	2h 32min	152.0	Action,Crime,Drama	7	...	W r kr th
4	5	12 Angry Men	1957	9.0	7,09,744	Approved	1h 36min	96.0	Crime,Drama	4	...	a to mis

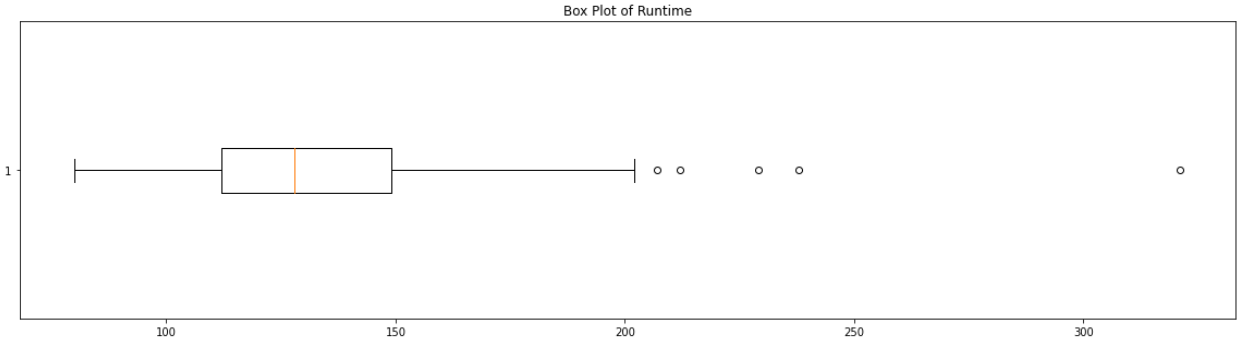
5 rows × 21 columns

III. Visualizing the data

1. 5-Point Summary of Runtime

In [7]:

```
# Create a box plot
plt.figure(figsize=(20, 5))
plt.boxplot(df['runtime'], vert=False)
plt.title('Box Plot of Runtime')

# Generate and display the 5-point summary table with space below
summary_table = df['runtime'].describe()

# Adjust the position of the table (change the y-coordinate in bbox)
table = plt.table(cellText=[summary_table.values],
                  colLabels=summary_table.index,
                  cellLoc='center',
                  loc='bottom',
                  bbox=[0, -0.3, 1, 0.2],
                 fontsize =50),

# [left, bottom, width, height]

plt.show()
```



count	mean	std	min	25%	50%	75%	max
209.0	132.14832535885168	32.00709070432944	80.0	112.0	128.0	149.0	321.0

2. Censor Rating: Count, Mean, and Median IMDb ratings

In [8]:

```
df['censor_rating_count'] = df['censorRating'].apply(lambda x: len(x.split(',')))
rating_dummies = df['censorRating'].str.get_dummies(sep=',')
df1 = pd.concat([df, rating_dummies], axis=1)

melted_df1 = df1.melt(id_vars=['censorRating'], value_vars=df1['censorRating'], var_name='censorRating', value_name='val

# filter rows where value is 1
melted_df1 = melted_df1[melted_df1['value'] == 1]

df1_reset = df1.reset_index()

# Now you can use the 'groupby' function
grouped_df1 = df1_reset.groupby('censorRating')['rating'].agg(['count', 'mean', 'median'])

# group by genre and calculate count, median, and mean of rating column
#grouped_df1 = melted_df1.groupby('censorRating')['rating'].agg(['count'])
grouped_df1.sort_values(by = 'count' )
```

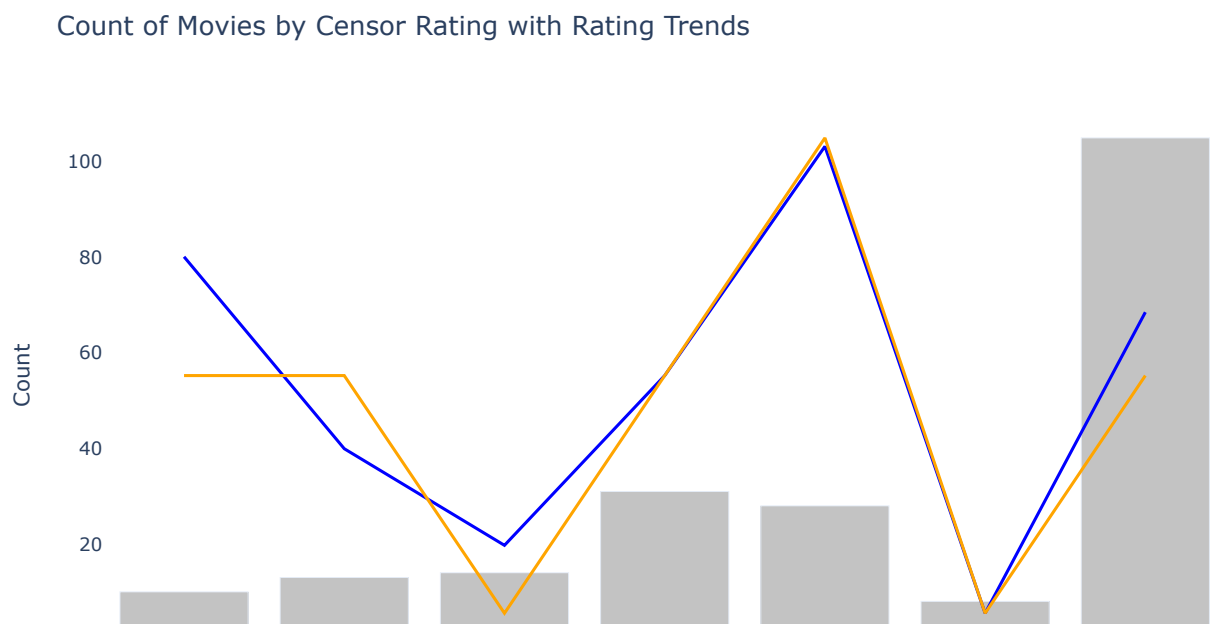
Out[8]:

censorRating	count	mean	median
Passed	8	8.200000	8.2
Approved	10	8.350000	8.3
G	13	8.269231	8.3
Not Rated	14	8.228571	8.2
PG-13	28	8.396429	8.4
PG	31	8.300000	8.3
R	105	8.326667	8.3

Censor-rating-wise IMDb Ratings

In [9]:

```
import plotly.graph_objects as go

bar_tracec = go.Bar(x=grouped_df1.index, y=grouped_df1['count'], name='Count of Movies', yaxis='y1', marker_color='#C3C3

# Create a scatter trace for the 'mean' of rating trend line on the right y-axis
mean_tracec = go.Scatter(x=grouped_df1.index, y=grouped_df1['mean'], mode='lines', name='Mean Rating', line=dict(color='

# Create a scatter trace for the 'median' of rating trend line on the right y-axis
median_tracec = go.Scatter(x=grouped_df1.index, y=grouped_df1['median'], mode='lines', name='Median Rating', line=dict(c

# Create a figure
figc = go.Figure(data=[bar_tracec, mean_tracec, median_tracec])

# Update the layout to have two y-axes
figc.update_layout(title='Count of Movies by Censor Rating with Rating Trends',
                   xaxis_title='Censor Rating',
                   yaxis_title='Count',
                   yaxis2=dict(title='Rating', overlaying='y', side='right'),
                  plot_bgcolor= 'white'
                   )

figc.show()
```



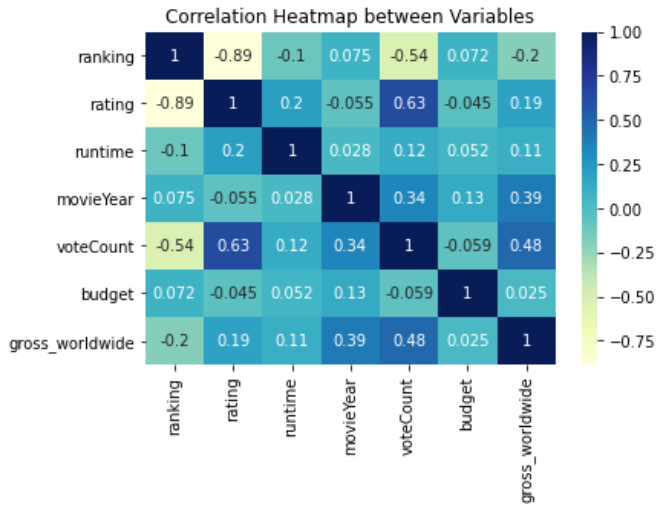
3. Correlation between the parameters

In [10]:

```
df['runtime'] = df['runtime'].astype(int)
regex = r'\d+(?:\.\d+)?"?(?:\s*[Xx/.]\s*\d+(?:\.\d+)?"?)?(?:\s*[A-Z]+\b)?'
df['budget'] = df['budget'].str.findall(regex).str.join('')
df['budget'] = df['budget'].astype(float)
df['voteCount'] = df['voteCount'].str.replace(',', '')
df['voteCount'] = df['voteCount'].astype(float)
df['gross_worldwide'] = df['gross_worldwide'].str.findall(regex).str.join('')
df['gross_worldwide'] = df['gross_worldwide'].astype(float)

corrdata = df[['ranking','rating', 'runtime', 'movieYear', 'voteCount', 'budget', 'gross_worldwide']]
dataplot = sns.heatmap(corrdata.corr(), cmap="YlGnBu", annot=True) #correlation between runtime, rating, budget, gross
plt.title( "Correlation Heatmap between Variables")
```

Out[10]:

```
Text(0.5, 1.0, 'Correlation Heatmap between Variables')
```



4. Director with the most number of movies

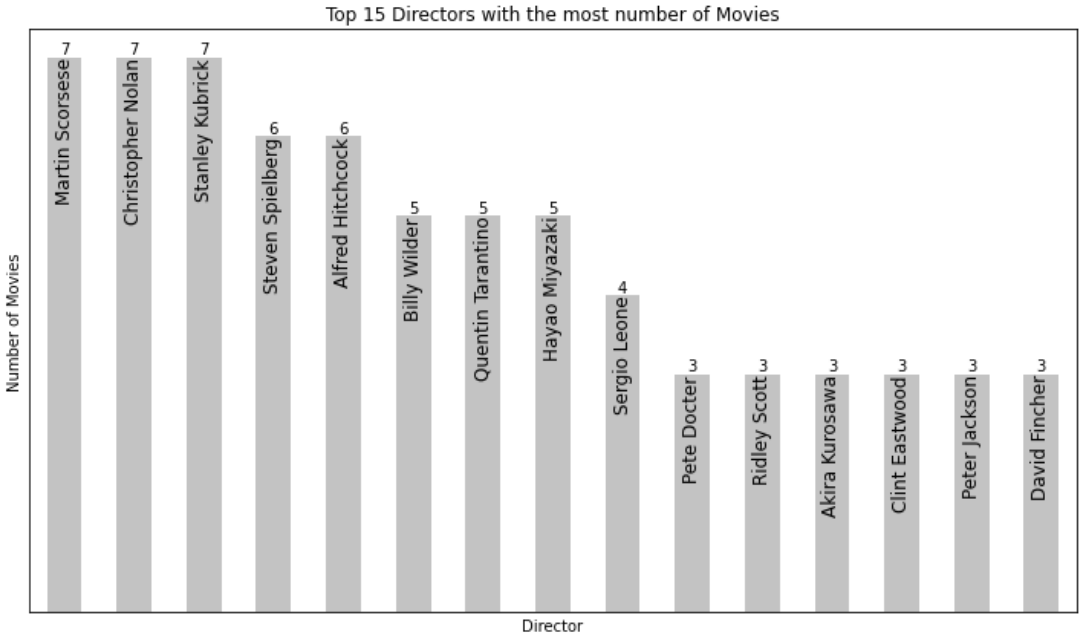
In [11]:

```
direc = df['director'].value_counts().head(15)
```

Distribution of directors with the most movies

In [12]:

```
# Create a bar graph
plt.figure(figsize=(10, 6))
ax = direc.plot(kind='bar', color='#C3C3C3')
plt.title('Top 15 Directors with the most number of Movies')
plt.xlabel('Director')
plt.ylabel('Number of Movies')

# Remove x and y axis scales
ax.set_xticks([])  # Remove x-axis ticks
ax.set_yticks([])  # Remove y-axis ticks


# Write director names inside the bars
for i, v in enumerate(direc):
    ax.text(i, v, str(v), ha='center', va='bottom')
    ax.text(i, v, direc.index[i], ha='center', va='top', fontsize=12, rotation = 90)

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



5. Actor appearing in most number of movies

In [13]:

```
casts_df = df['starList'].str.split(',', expand=True)

# Create a new dataframe with the counts of each actor
actor_counts = casts_df.stack().value_counts()

# Get the name of the actor with the highest count
top_actor = actor_counts.index[0]

print("The actor who appeared in the most movies in the dataset is:", top_actor)
```

```
The actor who appeared in the most movies in the dataset is: Robert De Niro
```

6. Writer with most number of movies

In [14]:

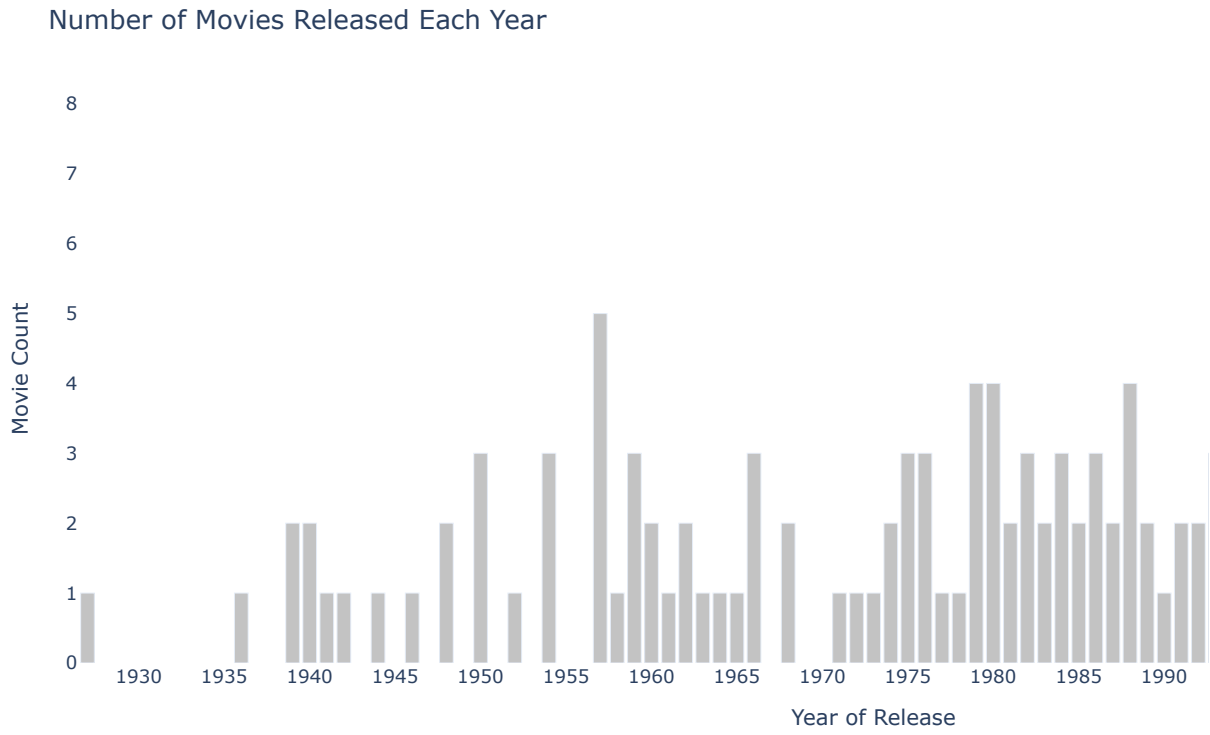
```
top_writer = df['writerList'].str.split(',', expand=True).stack().value_counts().index[0]
print("The writer who worked on the most movies in the dataset is:", top_writer)
```

```
The writer who worked on the most movies in the dataset is: Quentin Tarantino
```

7. Movies released each year

In [15]:

```
import plotly.express as px
# Group the data by 'movieYear' and count the number of movies released each year
yearly_movie_counts = df['movieYear'].value_counts().reset_index().dropna()
yearly_movie_counts.columns = ['Year', 'Movie Count']

# Create a bar graph using Plotly Express with increased bar size and color change
fig = px.bar(
    yearly_movie_counts,
    x='Year',
    y='Movie Count',
    width=1200,  # Adjust the bar width as desired (default is 0.8)
    color_discrete_sequence=['#C3C3C3']  # Change the bar color to #C3C3C3
)

# Update the layout with labels, title, and x-axis settings
fig.update_layout(
    xaxis_title='Year of Release',
    yaxis_title='Movie Count',
    title='Number of Movies Released Each Year',
    xaxis=dict(tickmode='linear', tick0=1930, dtick=5), plot_bgcolor= 'white'
)

# Display the plot
fig.show()
```



8. Genre: Count, Ratings, and Profits

In [16]:

```
df['genre_count'] = df['genre'].apply(lambda x: len(x.split(',')))
genre_dummies = df['genre'].str.get_dummies(sep=',')
df = pd.concat([df, genre_dummies], axis=1)

melted_df = df.melt(id_vars=['rating'], value_vars=df.columns[14:35], var_name='genre', value_name='value')

# filter rows where value is 1
melted_df = melted_df[melted_df['value'] == 1]

# group by genre and calculate count, median, and mean of rating column
grouped_df = melted_df.groupby('genre')['rating'].agg(['count', 'median', 'mean'])
#grouped_df.sort_values(by = "count", ascending = False)
```

In [17]:

```
df['gross_worldwide'] = pd.to_numeric(df['gross_worldwide'], errors='coerce')
df['budget'] = pd.to_numeric(df['budget'], errors='coerce')

df['gross_worldwide'] = df['gross_worldwide'].fillna(0).astype(int)
df['budget'] = df['budget'].fillna(0).astype(int)

# calculate profoit
df['profit'] = df['gross_worldwide'] - df['budget']


# assuming your dataframe is stored in a variable called df
genres = ['Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Drama', 'Family', 'Fantasy', 'Film-Noir',
          'History', 'Horror', 'Music', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Sport', 'Thriller', 'War', 'Western']

result = pd.DataFrame(index=genres, columns=['count', 'Median Rating', 'Mean Rating'])

for genre in genres:
    genre_data = df.loc[df[genre] == 1, 'profit']
    result.loc[genre, 'count'] = len(genre_data)
    result.loc[genre, 'Median Profit'] = genre_data.median()
    result.loc[genre, 'Mean Profit'] = genre_data.mean()

for genre in genres:
    genre_data = df.loc[df[genre] == 1, 'rating']
    result.loc[genre, 'count'] = len(genre_data)
    result.loc[genre, 'Median Rating'] = genre_data.median()
    result.loc[genre, 'Mean Rating'] = genre_data.mean()

result[['Median Profit', 'Mean Profit']] = result[['Median Profit', 'Mean Profit']].applymap(lambda x: f'{x:,.0f}')
result = result.reset_index()
result = result.rename(columns={result.columns[0]: 'genre'})
result
```

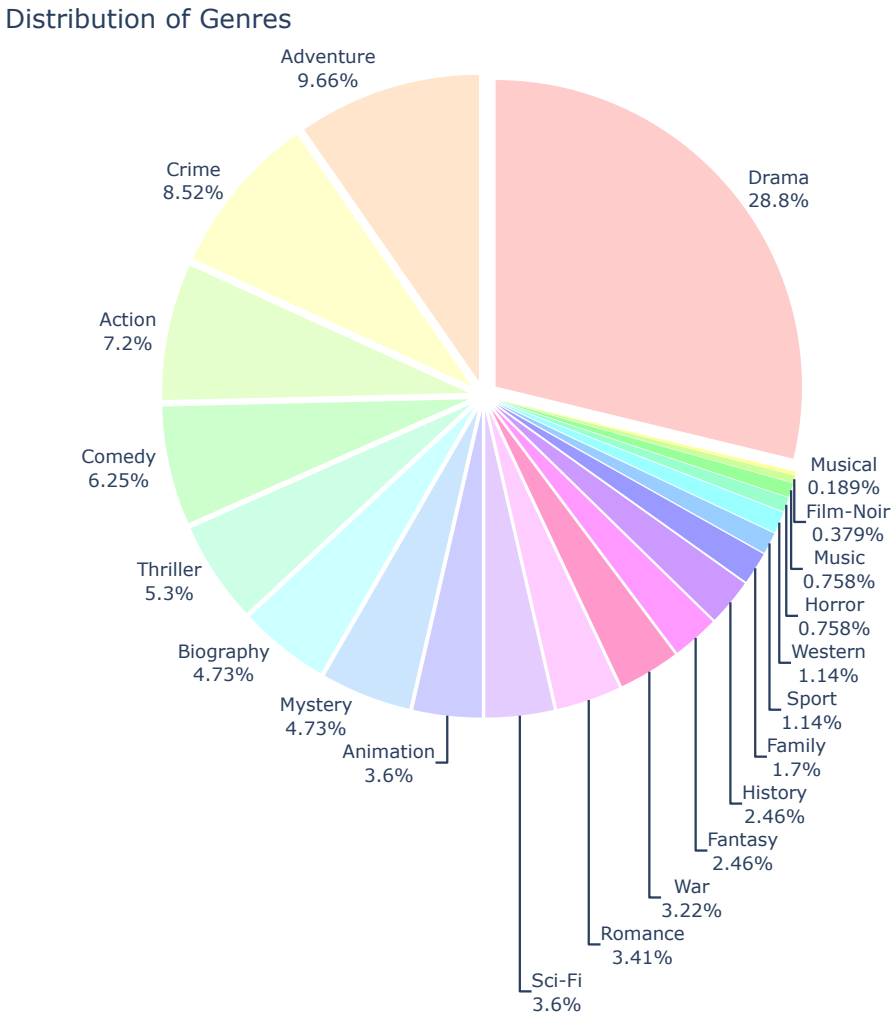
Out[17]:

	genre	count	Median Rating	Mean Rating	Median Profit	Mean Profit
0	Action	38	8.4	8.376316	307,710,151	334,597,726
1	Adventure	51	8.3	8.323529	341,311,890	387,801,353
2	Animation	19	8.2	8.278947	336,475,245	263,094,367
3	Biography	25	8.2	8.244	78,681,574	104,136,123
4	Comedy	33	8.2	8.251515	145,192,267	177,341,366
5	Crime	45	8.3	8.362222	53,611,975	115,455,241
6	Drama	152	8.3	8.321711	39,547,202	129,212,483
7	Family	9	8.3	8.333333	26,850,727	107,246,724
8	Fantasy	13	8.2	8.330769	26,850,727	252,219,183
9	Film-Noir	2	8.35	8.35	-1,182,500	-1,182,500
10	History	13	8.1	8.215385	59,427,638	101,635,489
11	Horror	4	8.4	8.35	29,616,704	39,787,746
12	Music	4	8.35	8.35	52,036,044	46,613,976
13	Musical	1	8.3	8.3	-675,744	-675,744
14	Mystery	25	8.3	8.296	31,047,078	80,372,850
15	Romance	18	8.3	8.288889	10,627,785	-20,357,452
16	Sci-Fi	19	8.3	8.331579	112,560,248	334,024,951
17	Sport	6	8.1	8.15	18,575,237	56,544,508
18	Thriller	28	8.25	8.278571	33,634,574	63,496,000
19	War	17	8.3	8.294118	16,357,676	74,973,256
20	Western	6	8.3	8.383333	19,226,876	85,289,066

8.1. Distribution of Genres

In [18]:

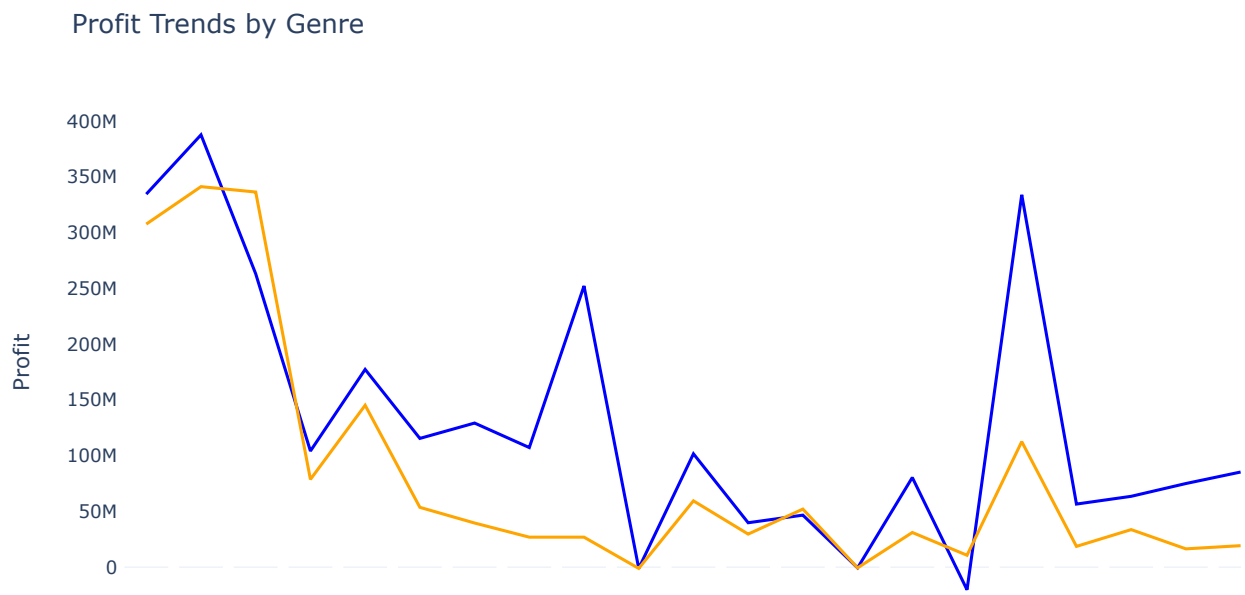
```
# Create a DataFrame
dfp = pd.DataFrame({'genre': result['genre'], 'counts': result['count']})

# Generate a color gradient from grey to black
color_scale = ['#FFCCCC','#FFE5CC','#FFFFCC', '#E5FFCC', '#CCFFCC', '#CCFFE5','#CCFFFF', '#CCE5FF','#CCCCFF','#E5CCFF'
               ,'#FFCCFF','#FF99CC','#FF99FF','#CC99FF','#9999FF','#99CCFF','#99FFFF','#99FFCC','#99FF99'
               ,'#CCFF99','#FFFF99']  # Customize the shades as needed

# Create a pie chart with Plotly
fig = px.pie(
    dfp,
    names='genre',
    values='counts',
    color_discrete_sequence=color_scale,
)

fig.update_traces(textposition='outside', textinfo='percent+label', pull=[0.05] * len(result))

# Hide the legend
fig.update_layout(showlegend=False)

# Increase the size of the pie chart
fig.update_layout(height=700, width=700)

fig.update_layout(title='Distribution of Genres')

# Show the interactive pie chart
fig.show()
```



8.2. Profit Trends by Genre

In [19]:

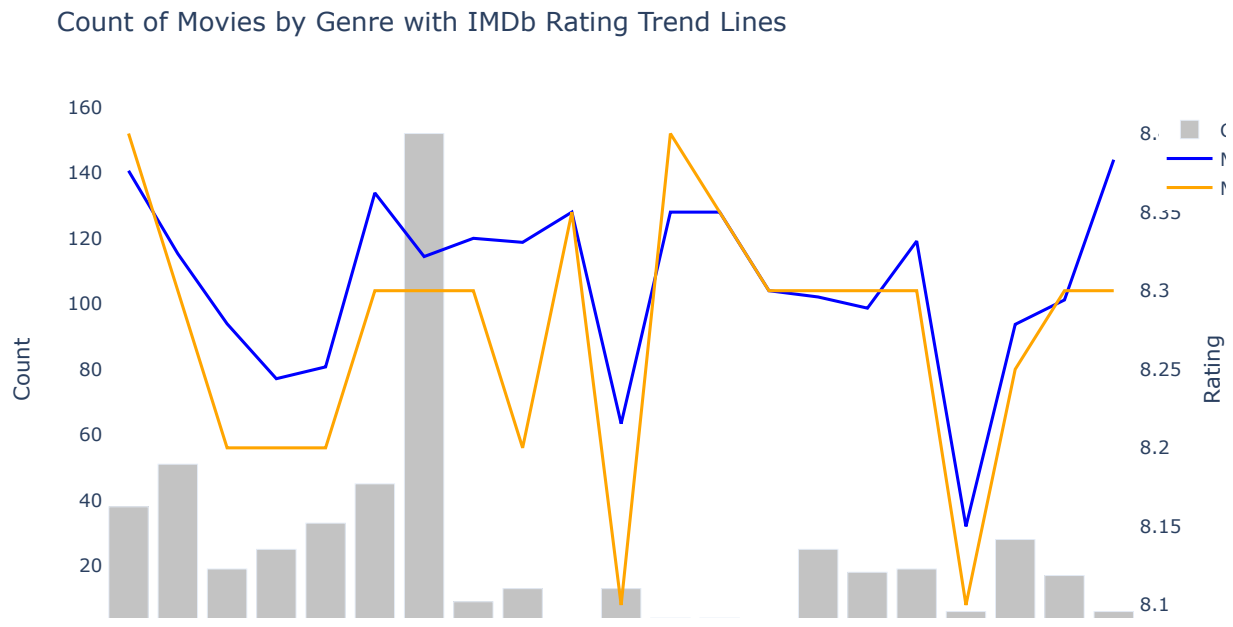
```
# Create a bar chart for the count of 'genre'
fig = px.bar(result, x='genre', title='Profit Trends by Genre')

# Add trend lines for 'mean profit' and 'median profit'
fig.add_trace(go.Scatter(x=result['genre'], y=result['Mean Profit'], mode='lines', name='Mean Profit', line=dict(color='
fig.add_trace(go.Scatter(x=result['genre'], y=result['Median Profit'], mode='lines', name='Median Profit', line=dict(col

# Update the layout
fig.update_layout(xaxis_title='Genre', yaxis_title='Profit', xaxis_tickangle=-45,plot_bgcolor= 'white' )
fig.show()
```



8.3. Genre-wise IMDb Ratings trend

In [20]:

```
import plotly.graph_objects as go

# Create a bar chart for the count of 'genre' on the left y-axis
bar_trace = go.Bar(x=result['genre'], y=result['count'], name='Count of Movies by Genre', yaxis='y1', marker_color='#C3C

# Create a scatter trace for 'mean rating' trend line on the right y-axis
mean_rating_trace = go.Scatter(x=result['genre'], y=result['Mean Rating'], mode='lines', name='Mean Rating', line=dict(c

# Create a scatter trace for 'median rating' trend line on the right y-axis
median_rating_trace = go.Scatter(x=result['genre'], y=result['Median Rating'], mode='lines', name='Median Rating', line=

# Create a figure
fig = go.Figure(data=[bar_trace, mean_rating_trace, median_rating_trace])

# Update the layout to have two y-axes
fig.update_layout(title='Count of Movies by Genre with IMDb Rating Trend Lines',
                  xaxis_title='Genre',
                  yaxis_title='Count',
                  yaxis2=dict(title='Rating', overlaying='y', side='right'),
                  plot_bgcolor= 'white'
                 )

fig.show()
```



9. Top Rated Directors: Number of movies, Mean IMDb Rating

In [21]:

```
directors = df[["director", "rating", 'language']]
directors = directors.explode('director')
director_avg_rating = directors.groupby('director')['rating'].agg(['count', 'mean'])
director_avg_rating = director_avg_rating.reset_index()
top_rated_directors = director_avg_rating.sort_values(by = 'mean', ascending = False)[:10]
top_rated_directors
```

Out[21]:

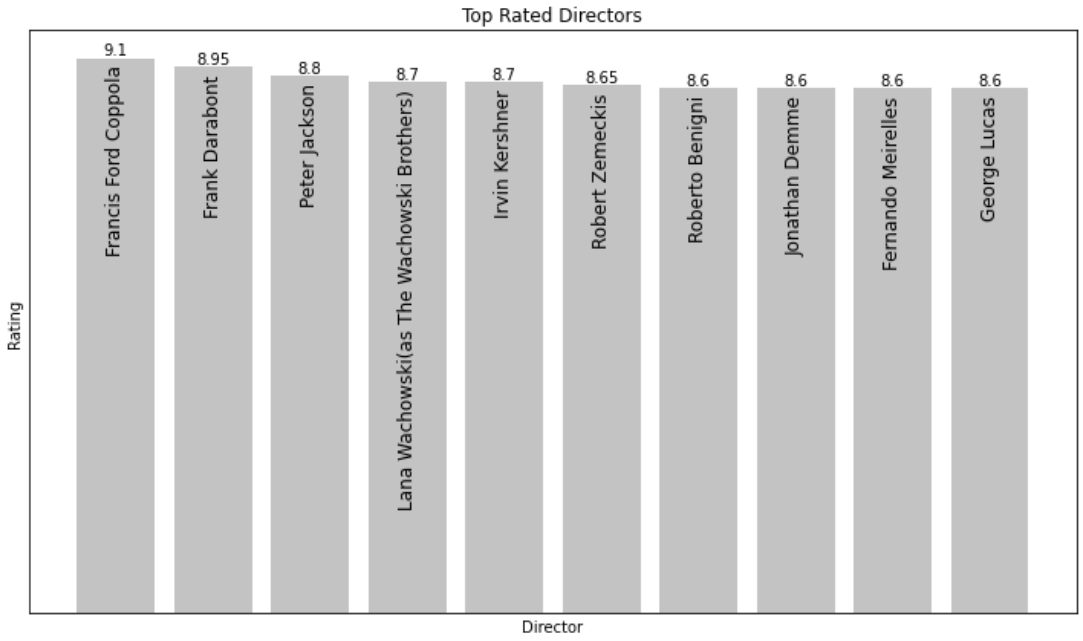
	director	count	mean
34	Francis Ford Coppola	2	9.10
37	Frank Darabont	2	8.95
93	Peter Jackson	3	8.80
66	Lana Wachowski(as The Wachowski Brothers)	1	8.70
49	Irvin Kershner	1	8.70
102	Robert Zemeckis	2	8.65
103	Roberto Benigni	1	8.60
63	Jonathan Demme	1	8.60
32	Fernando Meirelles	1	8.60
39	George Lucas	1	8.60

In [22]:

```
direc2 = top_rated_directors[['director', 'mean']]
direc2['mean'] = direc2['mean'].astype(float).round(2)
```

In [23]:

```
# Extract the 'director' and 'rating' columns
directors2 = direc2['director']
avgratings = direc2['mean']

# Create a bar graph
plt.figure(figsize=(10,6))  # Adjust the figure size as needed
bars = plt.bar(directors2, avgratings, color='#C3C3C3')
plt.xlabel('Director')
plt.ylabel('Rating')
plt.title('Top Rated Directors')

# Add director names and ratings inside the bars
for bar, director, rating in zip(bars, directors2, avgratings):
    plt.text(bar.get_x() + bar.get_width() / 2, rating, str(rating), ha='center', va='bottom')
    plt.text(bar.get_x() + bar.get_width() / 2, rating - 0.1, director, ha='center', va='top', rotation=90, fontsize = 1

plt.xticks([])
plt.yticks([])

plt.tight_layout()  # Ensure labels are not cut off
plt.show()
```



10. Top Produciton Houses: Number of Movies, Mean IMDb Rating

In [24]:

```
prodhs = df[["production", "rating"]]
prodhs = prodhs.explode('production')
pro_avg_rating = prodhs.groupby('production')['rating'].agg(['count', 'mean'])
pro_avg_rating = pro_avg_rating.reset_index()
top_rated_prodhs = pro_avg_rating.sort_values(by = "mean", ascending = False)[:10]
top_rated_prodhs
```

Out[24]:

	production	count	mean
80	Orion-Nova Productions	1	9.000000
45	Fox 2000 Pictures	1	8.800000
75	New Line Cinema	4	8.725000
42	Fantasy Films	1	8.700000
21	Castle Rock Entertainment	3	8.666667
61	Liberty Films (II)	1	8.600000
104	Strong Heart/Demme Production	1	8.600000
111	Toho Company	1	8.600000
113	Tokuma Shoten	1	8.600000
11	Barunson E&A	1	8.600000

11. Top Countries: Number of Movies, Mean IMDb Rating

In [25]:

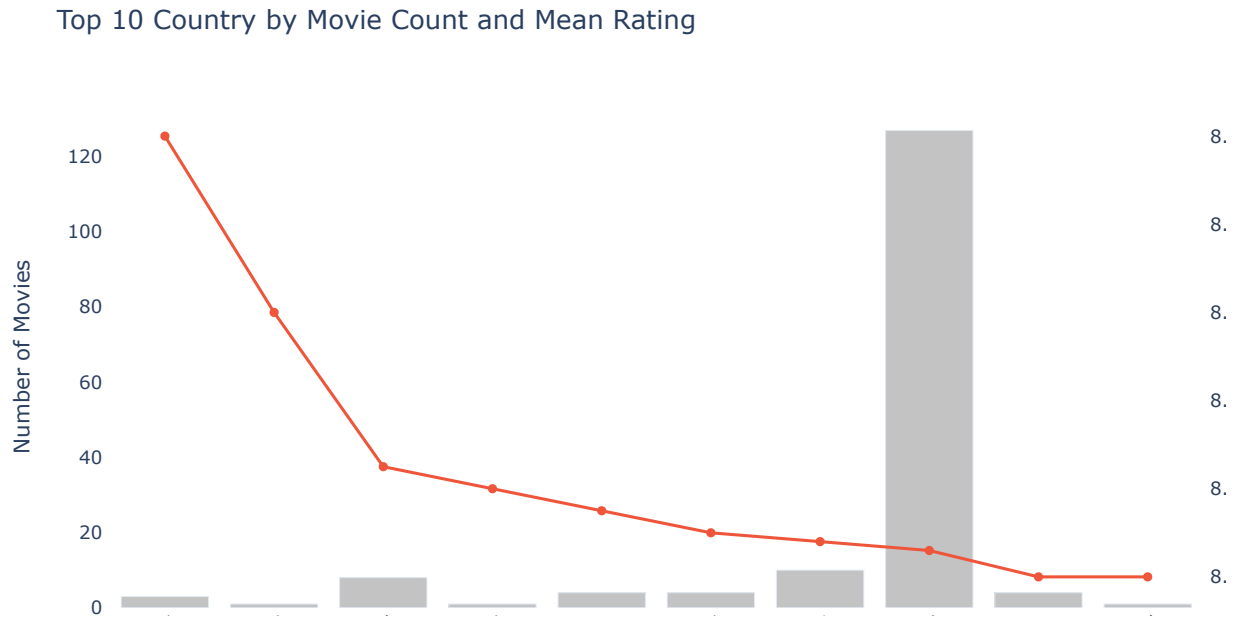
```
countries = df[["country", "rating"]]
countries = countries.explode('country')
countries_avg_rating = countries.groupby('country')['rating'].agg(['count', 'mean'])
countries_avg_rating = countries_avg_rating.reset_index()
top_rated_countries= countries_avg_rating.sort_values(by = "mean", ascending = False)[:10]
top_rated_countries
```

Out[25]:

	country	count	mean
14	New Zealand	3	8.800000
2	Brazil	1	8.600000
10	Italy	8	8.425000
12	Lebanon	1	8.400000
6	Germany	4	8.375000
5	France	4	8.350000
11	Japan	10	8.340000
19	USA	127	8.329921
15	South Korea	4	8.300000
20	West Germany	1	8.300000

In [26]:

```
# Extract 'language' and 'rating' columns
countr = df[["country", "rating"]]

# Explode the 'language' column to separate multiple languages in a single row
countr = countr.explode('country')

# Group by 'language' and calculate count and mean of 'rating'
countr_avg_rating = countr.groupby('country')['rating'].agg(['count', 'mean'])

# Reset the index to have 'language' as a regular column
countr_avg_rating = countr_avg_rating.reset_index()

# Sort by mean rating in descending order and select the top 10
top_countr = countr_avg_rating.sort_values(by="mean", ascending=False)[:10]

# Define custom colors for the bars

# Create a bar graph using go.Bar with custom colors
bar_trace7 = go.Bar(
    x=top_countr['country'],
    y=top_countr['count'],
    marker=dict(color='#c3c3c3'),
    name='Number of Movies'
)

# Create a trend line using go.Scatter for the mean rating
scatter_trace7 = go.Scatter(
    x=top_countr['country'],
    y=top_countr['mean'],
    mode='lines+markers',
    name='Mean Rating',
    yaxis='y2'  # Use the right-side y-axis for the rating scale
)

# Customize the layout to show the rating scale on the right side
layout7 = go.Layout(
    xaxis=dict(title='Country'),
    yaxis=dict(title='Number of Movies'),
    yaxis2=dict(title='IMDb Rating', overlaying='y', side='right'),
    plot_bgcolor='white',
    title='Top 10 Country by Movie Count and Mean Rating'
)

# Create a figure with both traces and layout
fig7 = go.Figure(data=[bar_trace7, scatter_trace7], layout=layout7)

# Show the graph
fig7.show()
```



12. Top Languages: Number of Movies, Mean IMDb Rating

In [27]:

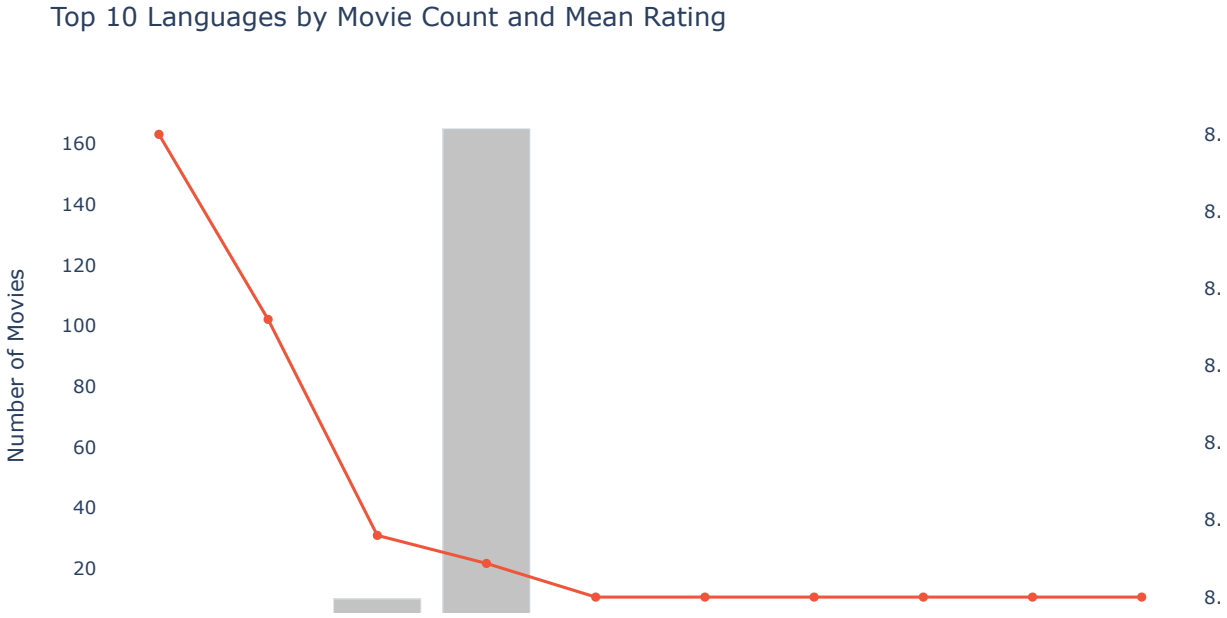
```
langs = df[["language", "rating"]]
langs = langs.explode('language')
langs_avg_rating = langs.groupby('language')['rating'].agg(['count', 'mean'])
langs_avg_rating = langs_avg_rating.reset_index()
top_rated_lang= langs_avg_rating.sort_values(by = "mean", ascending = False)[:10]
top_rated_lang
```

Out[27]:

	language	count	mean
10	Portuguese	1	8.600000
6	Italian	5	8.480000
7	Japanese	10	8.340000
2	English	165	8.321818
1	Danish	1	8.300000
3	French	4	8.300000
5	Hindi	4	8.300000
8	Korean	4	8.300000
9	Persian	2	8.300000
4	German	3	8.300000

In [28]:

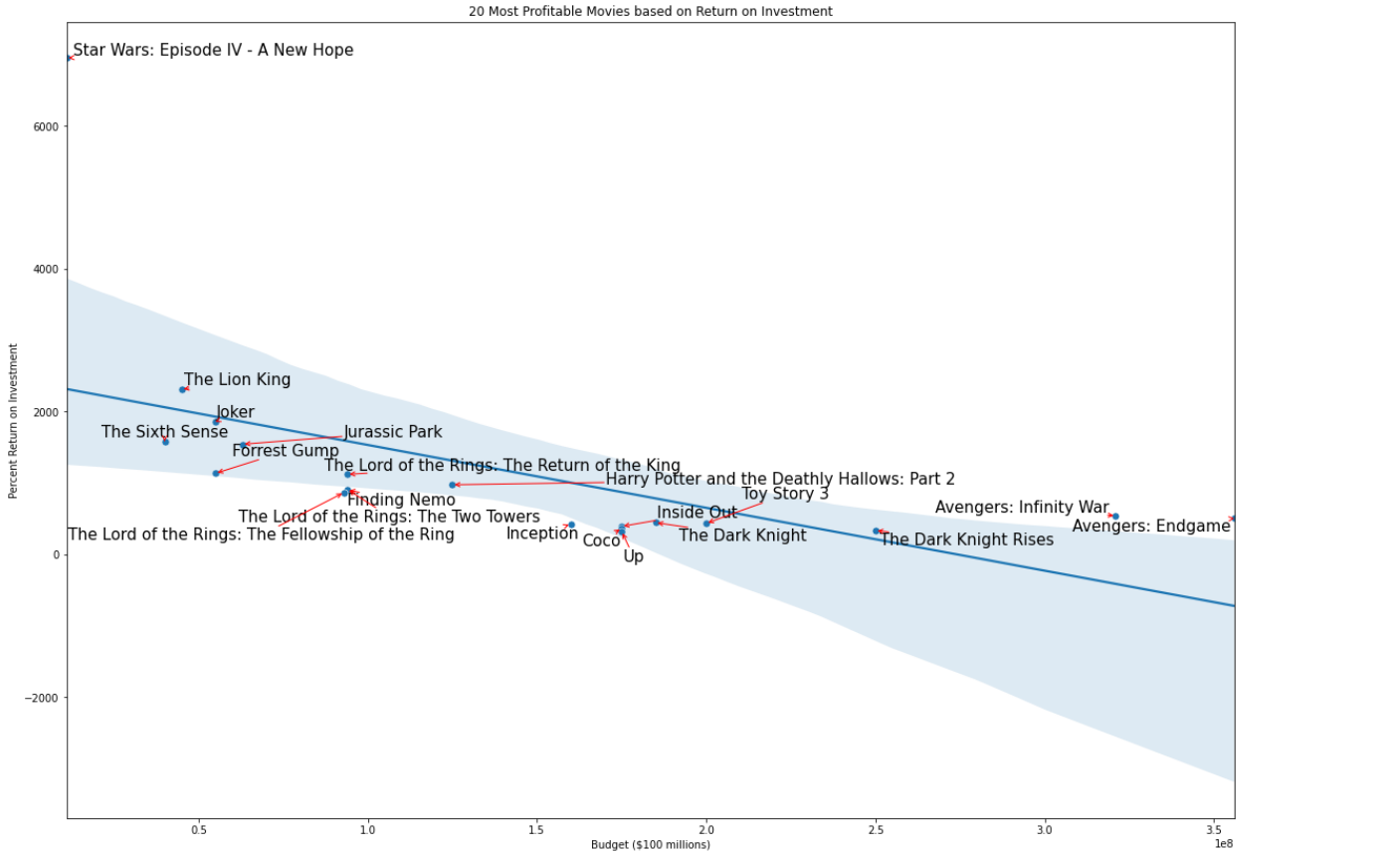
```
# Extract 'language' and 'rating' columns
langs = df[["language", "rating"]]

# Explode the 'language' column to separate multiple languages in a single row
langs = langs.explode('language')

# Group by 'language' and calculate count and mean of 'rating'
langs_avg_rating = langs.groupby('language')['rating'].agg(['count', 'mean'])

# Reset the index to have 'language' as a regular column
langs_avg_rating = langs_avg_rating.reset_index()

# Sort by mean rating in descending order and select the top 10
top_rated_lang = langs_avg_rating.sort_values(by="mean", ascending=False)[:10]

# Define custom colors for the bars

# Create a bar graph using go.Bar with custom colors
bar_trace5 = go.Bar(
    x=top_rated_lang['language'],
    y=top_rated_lang['count'],
    marker=dict(color='#c3c3c3'),
    name='Number of Movies'
)

# Create a trend line using go.Scatter for the mean rating
scatter_trace5 = go.Scatter(
    x=top_rated_lang['language'],
    y=top_rated_lang['mean'],
    mode='lines+markers',
    name='Mean Rating',
    yaxis='y2'  # Use the right-side y-axis for the rating scale
)

# Customize the layout to show the rating scale on the right side
layout5 = go.Layout(
    xaxis=dict(title='Language'),
    yaxis=dict(title='Number of Movies'),
    yaxis2=dict(title='IMDb Rating', overlaying='y', side='right'),
    plot_bgcolor='white',
    title='Top 10 Languages by Movie Count and Mean Rating'
)

# Create a figure with both traces and layout
fig5 = go.Figure(data=[bar_trace5, scatter_trace5], layout=layout5)

# Show the graph
fig5.show()
```


Top 10 Languages by Movie Count and Mean Rating
Number of Movies
160
140
120
100
80
60
40
20
8.
8.
8.
8.
8.
8.
8.

13. 20 Most Profitable Movies

In [30]:

```
from adjustText import adjust_text
sorted_data = df.sort_values(by='profit', ascending=False)
df['profit'] = (df['gross_worldwide'] - df['budget'])
df['roi'] = ((df['profit']/ df['budget'])*100)

top_movies = sorted_data.head(20)

# Visualization
plt.figure(figsize=(20, 14))
sns.scatterplot(data=top_movies, x='budget', y='roi', s=50)
sns.regplot(data=top_movies, x='budget', y='roi', scatter=False)


# Create a list of text labels
text_labels = [plt.text(row['budget'], row['roi'], row['movieTitle'], fontsize=15, ha='right') for i, row in top_movies.

# Adjust text labels to prevent overlap
adjust_text(text_labels, arrowprops=dict(arrowstyle='->', color='red'))

plt.xlabel('Budget ($100 millions)')
plt.ylabel('Percent Return on Investment')
plt.title('20 Most Profitable Movies based on Return on Investment')
plt.show()
```



In [31]:

```
df['roi'].mean()
```

Out[31]:

```
768.9796955725145
```

In [32]:

```
df['roi'].std()
```

Out[32]:

```
1419.242482511662
```

IV. Inferences

Crucial Insights of the analysis

Year Trends: The year 1995 stands out with the highest movie count. An upward trend in the number of IMDb Top 250 movies over time is evident. Some years have no movies in the list, underscoring IMDb's high standards.

Correlation Insights: Strong negative correlation between rating and ranking (-0.89) indicates highly-rated movies secure top positions. Other correlations are less significant.

Genre Diversity: Drama dominates at approximately 29%, while Musical has the smallest share at 0.2%.

Financial Success: Adventure, Sci-Fi, and Fantasy genres lead in mean profits. Some genres like Romance and Musical face negative mean profits.

Genres with High Ratings: Western, Action, and Crime genres exhibit the highest mean ratings. All genres fall within a narrow rating range of 8.1 to 8.4.

Movie Runtime: Median runtime is 128 minutes. Runtime varies across movies. **Prolific Actor:** Robert De Niro has the most movie appearances.

Prolific Writer: Quentin Tarantino is the most prolific writer.

Prolific Directors: Directors like Scorsese, Nolan, and Kubrick have seven movies each. Spielberg and Hitchcock follow with six each.

Top Rated Directors: Francis Ford Coppola and Frank Darabont lead with exceptional ratings of 9.1 and 8.95 respectively.

Censor Rating Impact: 'PG-13' has the highest peak, 'R' has the most movies. 'PG-13' tends to have the highest mean and median IMDb ratings.

Language Influence: 'English' dominates with 165 movies. 'Portuguese' and 'Italian' stand out with high mean ratings despite low counts.

Global Cinema: 'New Zealand' impresses with an outstanding mean rating of 8.8. 'Brazil' also excels (8.6) despite low representation. Other countries maintain ratings between 8.3 to 8.425.

Profit Disparity: "Star Wars: Episode IV - A New Hope" exhibits an exceptionally high ROI of ~7000% (mean ROI is just ~769%), emphasizing that budget doesn't always correlate with ROI.

These observations showcase the IMDb Top 250's dynamic nature, highlighting trends in movies, genres, ratings, and the diverse contributions of actors, writers, and directors.

$ThankYou$