In [1]:
```python
import pandas as pd
import numpy as np
```

1. Loading Data

In [2]:
```python
df = pd.read_csv("IMDB_Movies.csv")
df.head()
```

Out[2]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Ac
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Actio
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Actic
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	

5 rows × 28 columns

2. Exploratory Data Analysis

In [3]:
```python
df.shape
```

Out[3]:
```
(5043, 28)
```

In [4]:
```python
df.describe()
```

Out[4]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_total_facebook
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+03	5.043000e+03	5043.0
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+07	8.366816e+04	9699.0
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+07	1.384853e+05	18163.7
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02	5.000000e+00	0.0
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+06	8.593500e+03	1411.0
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+07	3.435900e+04	3090.0
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+07	9.630900e+04	13756.5
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08	1.689764e+06	656730.0

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5023 non-null   object 
 19  language                   5031 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
dtypes: float64(12), int64(3), object(13)
memory usage: 1.1+ MB
```

In [6]:
```
df.isnull().sum() # missing values
```

```
Out[6]: color                         19
        director_name                104
        num_critic_for_reviews        50
        duration                      15
        director_facebook_likes      104
        actor_3_facebook_likes        23
        actor_2_name                  13
        actor_1_facebook_likes         7
        gross                        884
        genres                         0
        actor_1_name                   7
        movie_title                    0
        num_voted_users                0
        cast_total_facebook_likes      0
        actor_3_name                  23
        facenumber_in_poster          13
        plot_keywords                153
        movie_imdb_link                0
        num_user_for_reviews          20
        language                      12
        country                        5
        content_rating               303
        budget                       492
        title_year                   108
        actor_2_facebook_likes        13
        imdb_score                     0
        aspect_ratio                 329
        movie_facebook_likes           0
        dtype: int64
```

```
In [7]: df.isnull().sum().sum() # total missing values
```

```
Out[7]: 2697
```

```
In [8]: df['movie_title'].nunique()
```

```
Out[8]: 4917
```

There are 5043 total records and in "movie_title" column there are 4917 unique records present. So it means we have 126 duplicate records in dataset

3. Data Cleaning

Removing the duplicate records from column "movie_title"

In [9]:
```
df.drop_duplicates(subset="movie_title",keep='first', inplace=True)
```

In [10]:
```
df.shape
```

Out[10]:
```
(4917, 28)
```

Dropping unnecesssary columns from data

In [11]:
```
df1 = pd.DataFrame(df.drop(['color','director_facebook_likes','actor_3_facebook_likes', 'actor_2_name','actor_1_facebook_likes',
         'cast_total_facebook_likes', 'actor_3_name','facenumber_in_poster','plot_keywords','movie_imdb_link',
        'content_rating','actor_2_facebook_likes','aspect_ratio','movie_facebook_likes'], axis=1))
```

In [12]:
```
df1.shape
```

Out[12]:
```
(4917, 14)
```

In [13]:
```
df1.head()
```

Out[13]:

	director_name	num_critic_for_reviews	duration	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	langu
0	James Cameron	723.0	178.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	En
1	Gore Verbinski	302.0	169.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	En
2	Sam Mendes	602.0	148.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	En
3	Christopher Nolan	813.0	164.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	En
4	Doug Walker	NaN	NaN	NaN	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8		

```
In [14]: df1.isna().sum() # missing values
```

```
Out[14]: director_name             102
         num_critic_for_reviews     49
         duration                   15
         gross                     863
         genres                      0
         actor_1_name                7
         movie_title                 0
         num_voted_users             0
         num_user_for_reviews       20
         language                   12
         country                     5
         budget                    484
         title_year                106
         imdb_score                  0
         dtype: int64
```

Removing missing values

```
In [15]: df2=df1.dropna(subset=['director_name','num_critic_for_reviews','duration','gross','actor_1_name','num_user_for_reviews','language','countr
```

```
In [16]: df2.head()
```

Out[16]:

	director_name	num_critic_for_reviews	duration	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	langu
0	James Cameron	723.0	178.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	En
1	Gore Verbinski	302.0	169.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	En
2	Sam Mendes	602.0	148.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	En
3	Christopher Nolan	813.0	164.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	En
5	Andrew Stanton	462.0	132.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	738	En

```
In [17]: df2.shape
```

```
Out[17]: (3781, 14)
```

```
In [18]: df2.isna().sum() # missing values
```

```
Out[18]:
director_name            0
num_critic_for_reviews   0
duration                 0
gross                    0
genres                   0
actor_1_name             0
movie_title              0
num_voted_users          0
num_user_for_reviews     0
language                 0
country                  0
budget                   0
title_year               0
imdb_score               0
dtype: int64
```

No missing or Null values so we have cleaned data ready for analysis

```
In [19]: df2.to_excel('IMDB_Movies(Cleaned data).xlsx')
```

A. Movie Genre Analysis

```
In [20]: from statistics import mode
         from tabulate import tabulate
```

```
In [21]: genre_counts = df2['genres'].str.split(', ').explode().value_counts()
         most_common_genres = genre_counts.head(20)
         print("Most common genres:")
         print(most_common_genres)
```

```
Most common genres:
Drama                                 152
Comedy|Drama|Romance                  149
Comedy|Drama                          147
Comedy                                145
Comedy|Romance                        135
Drama|Romance                         118
Crime|Drama|Thriller                   80
Action|Crime|Thriller                  54
Action|Crime|Drama|Thriller            48
Comedy|Crime                           45
Action|Adventure|Sci-Fi                45
Action|Adventure|Thriller              43
Horror                                 41
Crime|Drama                            41
Drama|Thriller                         40
Crime|Drama|Mystery|Thriller           40
Action|Adventure|Sci-Fi|Thriller       33
Horror|Thriller                        32
Horror|Mystery|Thriller                31
Biography|Drama                        30
Name: genres, dtype: int64
```

In [22]:
```
genre_statistics = {}

for genre in most_common_genres.index:
    genre_data = df2[df2['genres'].str.contains(genre, case=False, na=False)]

    mean = genre_data['imdb_score'].mean()
    median = genre_data['imdb_score'].median()
    mode_value = mode(genre_data['imdb_score'])
    range_value = genre_data['imdb_score'].max() - genre_data['imdb_score'].min()
    variance = genre_data['imdb_score'].var()
    std_deviation = genre_data['imdb_score'].std()

    genre_statistics[genre] = {
        'Mean': mean,
        'Median': median,
        'Mode': mode_value,
        'Range': range_value,
        'Variance': variance,
        'Std Deviation': std_deviation
    }


# Print the statistics for each genre
```

```
#for genre, stats in genre_statistics.items():
    #print(f"Statistics for {genre} genre:")
   # for stat, value in stats.items():
        #print(f"{stat}: {value}")


statistics = pd.DataFrame.from_dict(genre_statistics, orient='index')
statistics
```

Out[22]:

	Mean	Median	Mode	Range	Variance	Std Deviation
Drama	6.789005	6.9	6.7	7.2	0.794389	0.891285
Comedy|Drama|Romance	6.513204	6.6	6.7	7.4	1.066123	1.032532
Comedy|Drama	6.517128	6.6	6.7	7.4	1.062472	1.030763
Comedy	6.182763	6.3	6.3	6.9	1.081709	1.040053
Comedy|Romance	6.301441	6.4	6.7	6.9	1.076757	1.037669
Drama|Romance	6.673146	6.8	7.1	7.2	0.909898	0.953886
Crime|Drama|Thriller	6.616898	6.7	6.7	7.2	0.936684	0.967824
Action|Crime|Thriller	6.409516	6.5	6.6	7.2	1.104428	1.050917
Action|Crime|Drama|Thriller	6.578028	6.7	6.7	7.2	0.998129	0.999064
Comedy|Crime	6.311207	6.4	6.3	7.4	1.085044	1.041655
Action|Adventure|Sci-Fi	6.362007	6.4	6.6	7.1	1.173826	1.083432
Action|Adventure|Thriller	6.393534	6.5	6.6	6.9	1.092389	1.045174
Horror	5.901058	5.9	6.2	6.3	0.981537	0.990726
Crime|Drama	6.704191	6.8	6.7	7.2	0.861502	0.928171
Drama|Thriller	6.644390	6.7	6.7	7.2	0.901805	0.949634
Crime|Drama|Mystery|Thriller	6.607588	6.7	6.7	7.2	0.941476	0.970297
Action|Adventure|Sci-Fi|Thriller	6.391295	6.5	6.6	7.1	1.120266	1.058426
Horror|Thriller	6.329286	6.4	6.4	6.7	0.946203	0.972730
Horror|Mystery|Thriller	6.357871	6.4	6.4	6.7	0.954953	0.977217
Biography|Drama	6.787617	6.9	6.7	7.2	0.797494	0.893025

```
In [ ]:
```

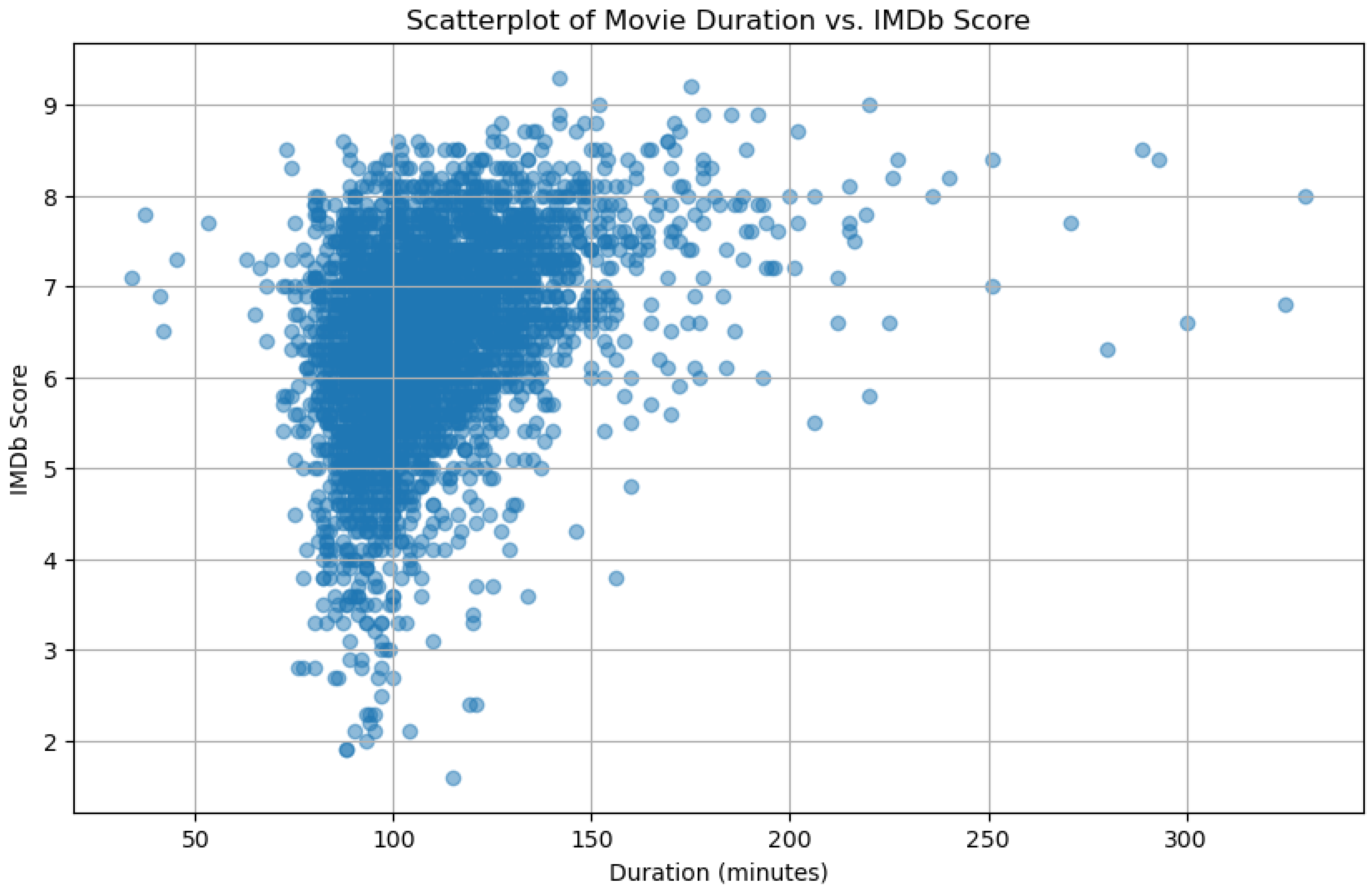
B. Movie Duration Analysis

```
In [23]: import matplotlib.pyplot as plt
```

```
In [24]: correlation = df2['duration'].corr(df2['imdb_score'])
         print(f'Correlation coefficient: {correlation}')
```

```
Correlation coefficient: 0.3621800423106813
```

```
In [25]: plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
         plt.scatter(df2['duration'], df2['imdb_score'], alpha=0.5)
         plt.title('Scatterplot of Movie Duration vs. IMDb Score')
         plt.xlabel('Duration (minutes)')
         plt.ylabel('IMDb Score')
         plt.grid(True)

         # Show the plot
         plt.show()
```



In [ ]:

```

```

C. Language Analysis:

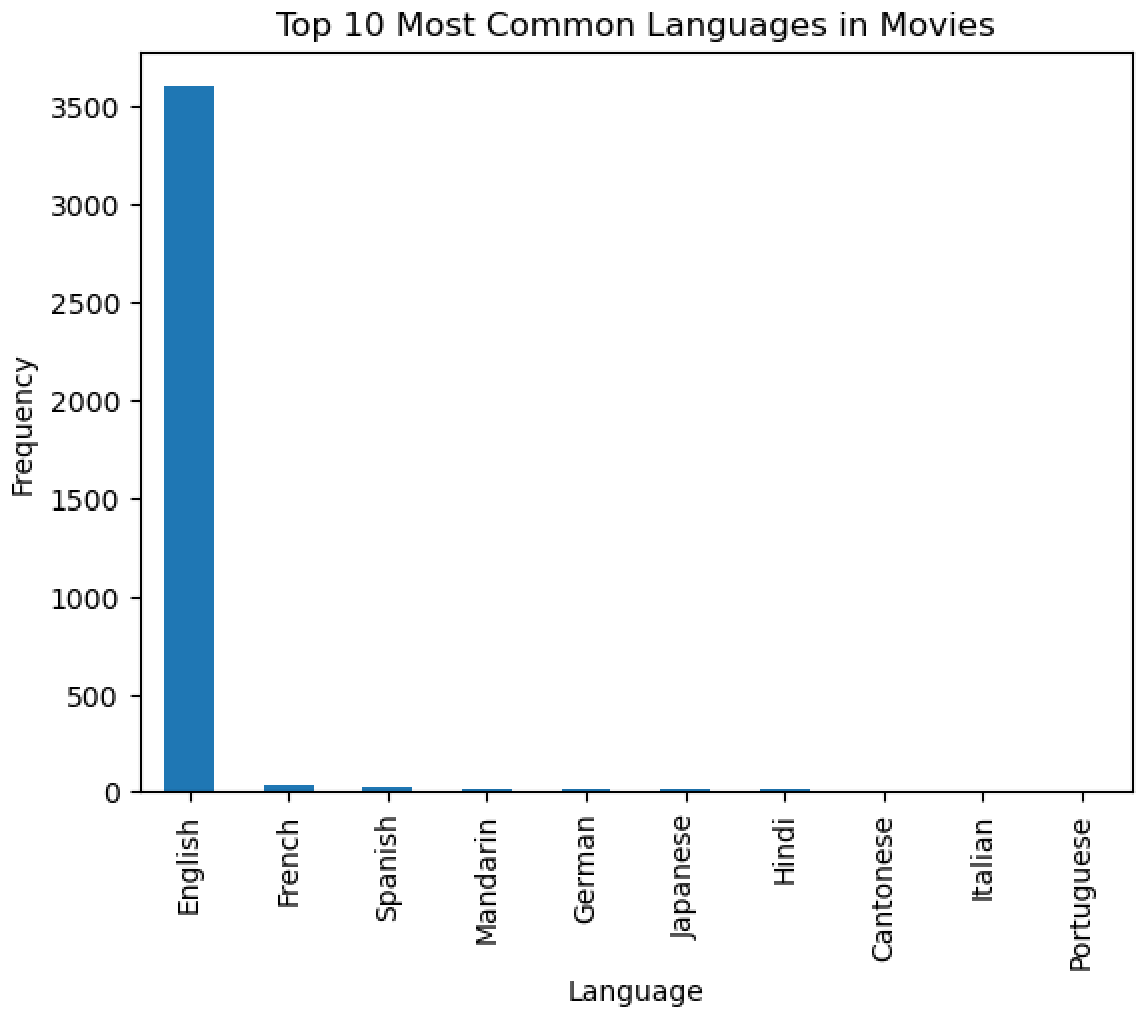
In [27]:

```
# Calculate IMDb score statistics for each language
language_stats = df2.groupby('language')['imdb_score'].describe()

# Print the summary statistics for IMDb scores
print(language_stats)
```

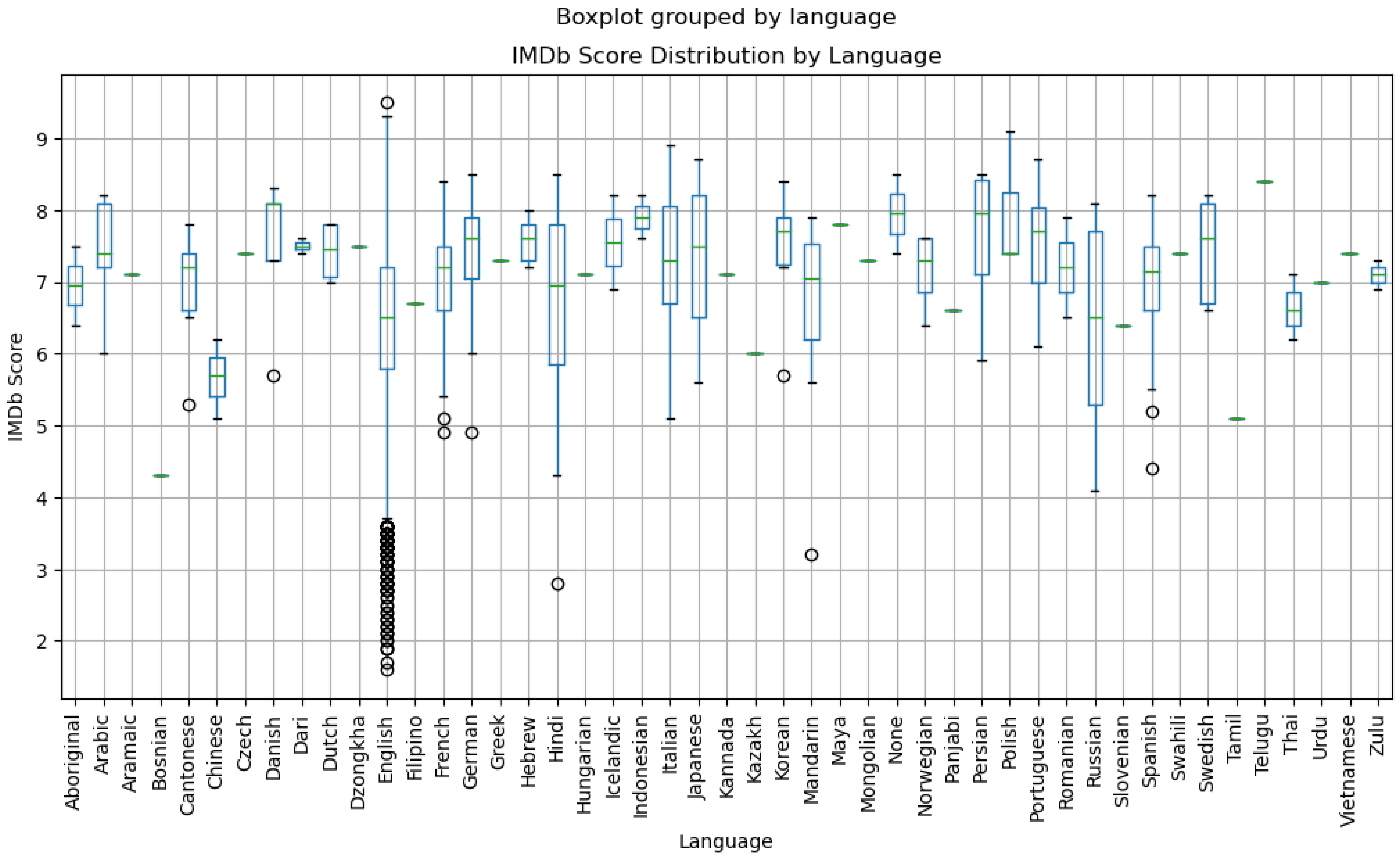
```
             count      mean       std  min    25%   50%    75%  max
language
Aboriginal     2.0  6.950000  0.777817  6.4  6.675  6.95  7.225  7.5
Arabic         1.0  7.200000       NaN  7.2  7.200  7.20  7.200  7.2
Aramaic        1.0  7.100000       NaN  7.1  7.100  7.10  7.100  7.1
Bosnian        1.0  4.300000       NaN  4.3  4.300  4.30  4.300  4.3
Cantonese      8.0  7.237500  0.440576  6.5  7.075  7.30  7.525  7.8
Czech          1.0  7.400000       NaN  7.4  7.400  7.40  7.400  7.4
Danish         3.0  7.900000  0.529150  7.3  7.700  8.10  8.200  8.3
Dari           2.0  7.500000  0.141421  7.4  7.450  7.50  7.550  7.6
Dutch          3.0  7.566667  0.404145  7.1  7.450  7.80  7.800  7.8
Dzongkha       1.0  7.500000       NaN  7.5  7.500  7.50  7.500  7.5
English     3602.0  6.420850  1.052605  1.6  5.800  6.50  7.100  9.3
Filipino       1.0  6.700000       NaN  6.7  6.700  6.70  6.700  6.7
French        37.0  7.286486  0.561329  5.8  6.900  7.20  7.700  8.4
German        13.0  7.692308  0.640913  6.1  7.400  7.70  8.300  8.5
Hebrew         3.0  7.500000  0.435890  7.2  7.250  7.30  7.650  8.0
Hindi         10.0  6.760000  1.111755  4.8  6.050  7.05  7.700  8.0
Hungarian      1.0  7.100000       NaN  7.1  7.100  7.10  7.100  7.1
Icelandic      1.0  6.900000       NaN  6.9  6.900  6.90  6.900  6.9
Indonesian     2.0  7.900000  0.424264  7.6  7.750  7.90  8.050  8.2
Italian        7.0  7.185714  1.155319  5.3  6.700  7.00  7.850  8.9
Japanese      12.0  7.625000  0.899621  6.0  7.275  7.80  8.250  8.7
Kazakh         1.0  6.000000       NaN  6.0  6.000  6.00  6.000  6.0
Korean         4.0  7.875000  0.478714  7.3  7.600  7.90  8.175  8.4
Mandarin      14.0  7.021429  0.765786  5.6  6.425  7.25  7.600  7.9
Maya           1.0  7.800000       NaN  7.8  7.800  7.80  7.800  7.8
Mongolian      1.0  7.300000       NaN  7.3  7.300  7.30  7.300  7.3
None           1.0  8.500000       NaN  8.5  8.500  8.50  8.500  8.5
Norwegian      4.0  7.150000  0.574456  6.4  6.850  7.30  7.600  7.6
Persian        3.0  8.133333  0.550757  7.5  7.950  8.40  8.450  8.5
Portuguese     5.0  7.760000  0.978775  6.1  7.900  8.00  8.100  8.7
Romanian       1.0  7.900000       NaN  7.9  7.900  7.90  7.900  7.9
Russian        1.0  6.500000       NaN  6.5  6.500  6.50  6.500  6.5
Spanish       26.0  7.050000  0.826196  5.2  6.625  7.15  7.675  8.2
Swedish        1.0  7.600000       NaN  7.6  7.600  7.60  7.600  7.6
Telugu         1.0  8.400000       NaN  8.4  8.400  8.40  8.400  8.4
Thai           3.0  6.633333  0.450925  6.2  6.400  6.60  6.850  7.1
Vietnamese     1.0  7.400000       NaN  7.4  7.400  7.40  7.400  7.4
Zulu           1.0  7.300000       NaN  7.3  7.300  7.30  7.300  7.3
```

In [28]:
```
language_counts = df2['language'].value_counts()

# Plot the top N most common languages
top_languages = language_counts.head(10)
```

```
top_languages.plot(kind='bar')
plt.xlabel('Language')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Languages in Movies')
plt.show()
```



In [29]:
```
# Plot box plots to visualize the distribution of IMDb scores for each language
df.boxplot(column='imdb_score', by='language', figsize=(12, 6))
plt.xlabel('Language')
plt.ylabel('IMDb Score')
plt.title('IMDb Score Distribution by Language')
plt.xticks(rotation=90)
plt.show()
```



```
In [ ]:
```

D. Director Analysis:

In [30]:
```
# Calculate average IMDb scores per director
director_avg_scores = df2.groupby('director_name')['imdb_score'].mean().reset_index()

# Rank directors based on average IMDb scores
director_avg_scores = director_avg_scores.sort_values(by='imdb_score', ascending=False)

# Calculate percentiles
director_avg_scores['Percentile'] = pd.qcut(director_avg_scores['imdb_score'], q=10, labels=False)

# Display the top directors
top_directors = director_avg_scores.head(10)
print("Top Directors based on Average IMDb Score:")
print(top_directors)

# Analyze their contribution to success
percentile_counts = director_avg_scores['Percentile'].value_counts().sort_index()
```

```
Top Directors based on Average IMDb Score:
              director_name  imdb_score  Percentile
216         Charles Chaplin    8.600000           9
1670              Tony Kaye    8.600000           9
45          Alfred Hitchcock    8.500000           9
1435             Ron Fricke    8.500000           9
1014           Majid Majidi    8.500000           9
302          Damien Chazelle    8.500000           9
1493           Sergio Leone    8.433333           9
260       Christopher Nolan    8.425000           9
1032   Marius A. Markevicius    8.400000           9
1462           S.S. Rajamouli    8.400000           9
```

In [31]:
```
print("\nPercentile Counts:")
print(percentile_counts)
```

```
Percentile Counts:
0    193
1    190
2    142
3    202
4    147
5    174
6    189
7    160
8    189
9    161
Name: Percentile, dtype: int64
```

E. Budget Analysis:

In [32]:
```
correlation = df2['budget'].corr(df2['gross'])
print(f"Correlation between Budget and Gross: {correlation}")
```

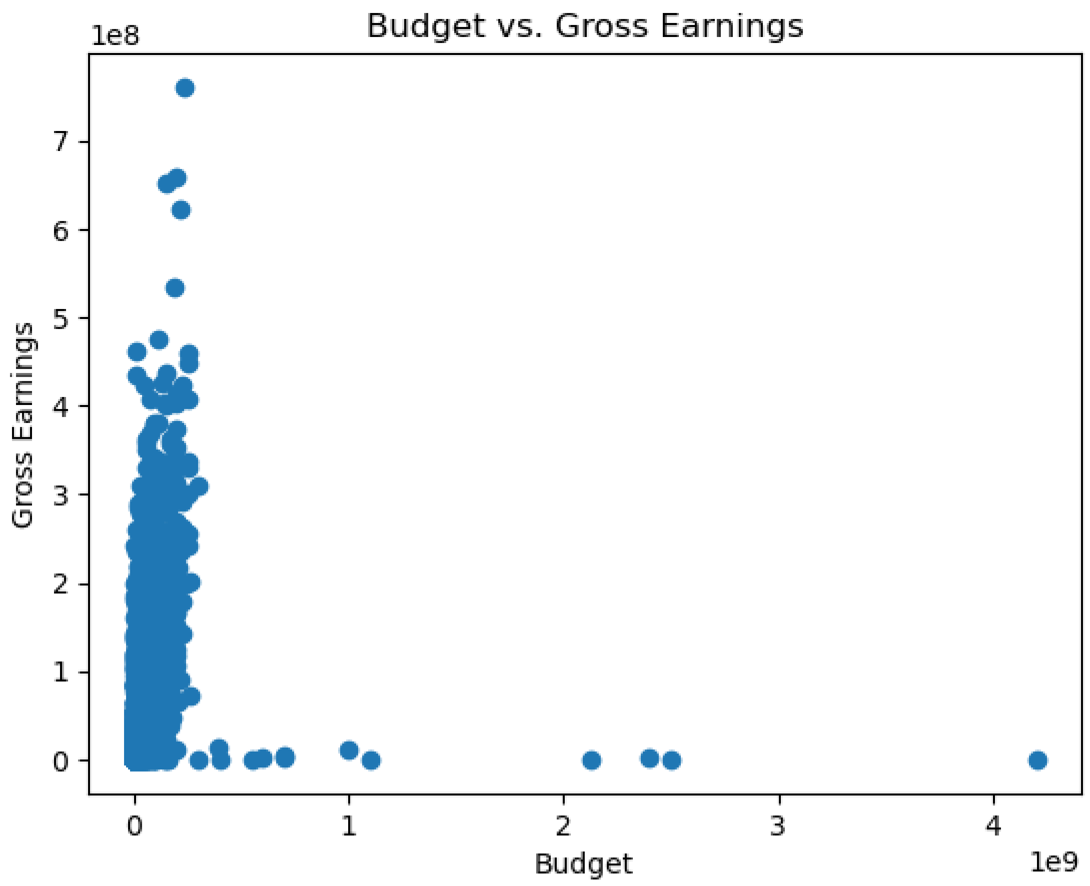
```
Correlation between Budget and Gross: 0.2229017828676018
```

In [46]:
```
# Calculate profit margin (Profit Margin = (Gross - Budget) / Gross)
df2['Profit Margin'] = ((df2['gross'] - df2['budget']) / df2['gross']) * 100

# Sort the DataFrame by profit margin in descending order
df_sorted = df2.sort_values(by='Profit Margin', ascending=False)

# Print the top movies with the highest profit margin
print("Top 10 Movies with the Highest Profit Margin:")
print(df_sorted[['movie_title', 'Profit Margin']].head(20))
```

```
Top 10 Movies with the Highest Profit Margin:
                           movie_title  Profit Margin
4793                Paranormal Activity      99.986100
4799                          Tarnation      99.963177
4707            The Blair Witch Project      99.957305
4984              The Brothers McMullen      99.756017
3278         The Texas Chain Saw Massacre      99.729311
5035                        El Mariachi      99.657017
4956                        The Gallows      99.560591
4977                      Super Size Me      99.436222
2492                          Halloween      99.361702
4674                   American Graffiti      99.324348
4530                              Rocky      99.181134
5011              In the Company of Men      99.124840
4791                   Napoleon Dynamite      99.101950
4955                   Facing the Giants      99.017166
4449   Snow White and the Seven Dwarfs      98.918483
4725                              Benji      98.735861
5042                   My Date with Drew      98.709253
5027                         The Circle      98.515836
4723                          Fireproof      98.505298
4726                         Open Water      98.360703
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_20004\702939183.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['Profit Margin'] = ((df2['gross'] - df2['budget']) / df2['gross']) * 100
```

In [49]:
```
plt.scatter(df['budget'], df['gross'])
plt.xlabel('Budget')
plt.ylabel('Gross Earnings')
plt.title('Budget vs. Gross Earnings')
plt.show()
```


Budget vs. Gross Earnings
1e8
Gross Earnings
1e9
Budget