Project: TMDB movie Data Analysis

Table of Contents

Introduction

Dataset Description

> In this project, we need to examine the TMDB movie dataset and share what we discover. We will use Python tools such as NumPy, pandas, and Matplotlib to simplify our analysis of the TMDB movie data. This dataset has details on 10,000 movies from the Movie Database (TMDb), covering aspects like user ratings and revenue. It includes information on various fields such as 'cast', 'genres', and 'characters'.

- **id**: Unique movie identifier.
- **imdb_id**: IMDB code specific to the movie.
- **popularity**: Metric indicating the movie's popularity.
- **budget**: Amount spent to produce the movie.
- **revenue**: Earnings from the movie.
- **original_title**: The movie's original name.
- **cast**: Leading actors involved in the movie.
- **homepage**: Official website of the movie.
- **director**: The movie's director.
- **tagline**: A catchy phrase representing the movie.
- **keywords**: Terms associated with the movie.
- **overview**: A brief summary of the movie.
- **runtime**: Total duration of the movie in minutes.
- **genres**: Categories describing the movie's style and content.
- **production_companies**: Firms that produced the movie.
- **release_date**: When the movie was first released.
- **vote_count**: Total votes received by the movie.
- **vote_average**: Average rating given to the movie.
- **release_year**: Year the movie was released.
- **budget_adj**: Movie's budget adjusted for inflation.
- **revenue_adj**: Movie's revenue adjusted for inflation.

Question(s) for Analysis

```
Q1. What are the most common movie genres?
Q2. Who are the most cast actors?
Q3. What production company produces the most movies?
Q4. What are the top movies in terms of profit?
Q5. What are the top movies based on popularity?
Q6. What are the top movies based on viewer rating?
Q7. What are the most common keywords?
Q8. Is the budget related to a higher average vote?
Q9. what's the correlation between runtime and vote average, budget and popularity?
Q10. Who are the most successful directors?
Q11. How did the runtime of movies change over the years? What Movie has the longest runtime? what movie has the shortest runtime? what's the aver
```

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling

- Loading the Data
- Exploring the Data
- Data Cleaning
  - Check if the data is clean, remove columns that are not needed, look for missing values (NAN values) and correct them, etc.

Loading the Data

```
# Load the dataset
# Importing the movie dataset
dataset_location = '/content/tmdb-movies.csv'
data = pd.read_csv(dataset_location)
```

Exploring the Data

```
# Show initial and final rows of the movie data
data.head() # Display the first few entries
```

	id	imdb_id	popularity	budget	revenue	original_title	cast
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...

5 rows × 21 columns

```
# Display the last few entries of the dataset
data.tail()
```

	id	imdb_id	popularity	budget	revenue	original_title	cas
8290	10497	tt0104779	0.529727	5000000	1862805	Bitter Moon	Hugh Grant|Kristi Sco Thomas|Emmanuell Sei.
8291	9278	tt0104299	0.524767	0	0	Freejack	Emilio Estevez|Mic Jagger|Ren Russo|Anthony .
8292	14002	tt0103767	0.521669	4000000	0	Baraka	Na
8293	10699	tt0104412	0.518807	42000000	0	Hero	Dusti Hoffman|Geen Davis|Andy Garcí a|Joan C.
8294	47821	tt0103976	0.499566	27000000	14683921	City of Joy	Patrick Swayze|Oı Puri|Paulin Collins|Shabana.

5 rows × 21 columns

```
# Output the total count of rows and columns in the dataset
print(f"There are {data.shape[0]:,} rows and {data.shape[1]:,} columns in the dataset.")
```

```
There are 8,295 rows and 21 columns in the dataset.
```

```
# Generate statistical summary for the dataset's numerical columns
data.describe()
```

	id	popularity	budget	revenue	runtime	vote_count	v
count	8295.000000	8295.000000	8.295000e+03	8.295000e+03	8294.000000	8294.000000	
mean	81362.538638	0.690335	1.619967e+07	4.303695e+07	100.914758	246.470822	
std	100212.107764	1.102887	3.364056e+07	1.274732e+08	32.998940	634.644182	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	11616.500000	0.209871	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	28512.000000	0.396280	0.000000e+00	0.000000e+00	98.000000	42.000000	
75%	119622.000000	0.762727	1.800000e+07	2.370166e+07	110.000000	171.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

1. Popularity:

Popularity scores in the dataset range widely from nearly 0 to 32.99. However, most movies have low popularity, with an average score of about 0.65.

2. Revenue:

Revenues range from 0 dollars many movies made no money to about 2.78 billion dollars. This shows that while some movies earn huge amounts, many do not make any revenue.

3. Vote Count:

Votes per movie vary greatly from 10 to 9,767, indicating that some movies are much more popular with viewers than others.

```
# Count the unique values in each column
unique_values_count = data.nunique()
unique_values_count
```

```
id                      8294
imdb_id                 8284
popularity              8266
budget                   473
revenue                 3596
original_title          8159
cast                    8176
homepage                2764
director                4468
tagline                 5888
keywords                6503
overview                8276
runtime                  237
genres                  1710
production_companies    6009
release_date            4119
vote_count              1209
vote_average              72
release_year              27
budget_adj              1856
revenue_adj             3645
dtype: int64
```

```
# Calculate the total count of missing values per column
missing_values_count = data.isnull().sum()
missing_values_count
```

```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      65
homepage                5497
director                  41
tagline                 2371
keywords                1320
overview                   5
runtime                    1
genres                    19
production_companies     882
release_date               1
vote_count                 1
vote_average               1
release_year               1
budget_adj                 1
revenue_adj                1
dtype: int64
```

Columns with many unique values compared to the number of rows are high-cardinality categorical features. Columns with few unique values are likely categorical.

```
# Display the data types and check for missing values in each column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8295 entries, 0 to 8294
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    8295 non-null   int64
 1   imdb_id               8285 non-null   object
 2   popularity            8295 non-null   float64
 3   budget                8295 non-null   int64
 4   revenue               8295 non-null   int64
 5   original_title        8295 non-null   object
 6   cast                  8230 non-null   object
 7   homepage              2798 non-null   object
 8   director              8254 non-null   object
 9   tagline               5924 non-null   object
 10  keywords              6975 non-null   object
 11  overview              8290 non-null   object
 12  runtime               8294 non-null   float64
 13  genres                8276 non-null   object
```

```
 14  production_companies  7413 non-null   object
 15  release_date          8294 non-null   object
 16  vote_count            8294 non-null   float64
 17  vote_average          8294 non-null   float64
 18  release_year          8294 non-null   float64
 19  budget_adj            8294 non-null   float64
 20  revenue_adj           8294 non-null   float64
dtypes: float64(7), int64(3), object(11)
memory usage: 1.3+ MB
```

Some columns have missing values: cast, homepage, director, tagline, keywords, overview, genres, and production_companies. All data types are correct except for release_date and release_year. I will keep the datatype of release_year as it is. The release_date will be dropped later in the analysis because it is not important for this study.

Data Cleaning

- Based on my observations from the first 5 rows, and from previous datasets we need to do the following:
    1. The columns id, imdb_id, homepage, budget_adj, revenue adj are useless, hence, we need to delete them
    2. remove dublicates
    3. check for NAN values
    4. replace null values with NAN

Dropping Unimportant Columns, duplicates, and null values.

I will drop the following columns because they are not important or needed for this analysis: 'id', 'imdb_id', 'homepage', 'tagline', 'overview', 'vote_count', 'budget_adj', and 'revenue_adj'.

```
# Remove columns that are not significant for our analysis
columns_to_drop = ['id', 'imdb_id', 'homepage', 'tagline', 'overview', 'vote_count', 'budget_adj', 'revenue_adj', 'release_date']
data.drop(columns=columns_to_drop, axis=1, inplace=True)

# Display the first few rows after dropping columns
data.head()
```

	popularity	budget	revenue	original_title	cast	director
0	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow
1	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller
2	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo	Robert

Next steps: Generate code with data View recommended plots

```
# Calculate the number of duplicate rows in the dataset
duplicate_entries_count = data.duplicated().sum()
duplicate_entries_count
```

```
1
```

```
# Remove duplicate rows from the dataset
data.drop_duplicates(inplace=True)

# Verify that all duplicate rows have been removed
remaining_duplicates = data.duplicated().sum()
remaining_duplicates
```

```
0
```

```
# Drop rows with missing values in critical columns
data = data.dropna(subset=['genres','director', 'cast'])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8177 entries, 0 to 8293
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   popularity            8177 non-null   float64
 1   budget                8177 non-null   int64
 2   revenue               8177 non-null   int64
 3   original_title        8177 non-null   object
 4   cast                  8177 non-null   object
 5   director              8177 non-null   object
 6   keywords              6918 non-null   object
 7   runtime               8177 non-null   float64
 8   genres                8177 non-null   object
 9   production_companies  7361 non-null   object
 10  vote_average          8177 non-null   float64
 11  release_year          8177 non-null   float64
dtypes: float64(4), int64(2), object(6)
memory usage: 830.5+ KB
```

```
# Convert specified object type columns to string type
cols_to_convert = ['production_companies', 'genres', 'keywords', 'director', 'original_title', 'cast']
data[cols_to_convert] = data[cols_to_convert].astype(str)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8177 entries, 0 to 8293
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   popularity            8177 non-null   float64
 1   budget                8177 non-null   int64
 2   revenue               8177 non-null   int64
 3   original_title        8177 non-null   object
 4   cast                  8177 non-null   object
 5   director              8177 non-null   object
 6   keywords              8177 non-null   object
 7   runtime               8177 non-null   float64
 8   genres                8177 non-null   object
 9   production_companies  8177 non-null   object
 10  vote_average          8177 non-null   float64
 11  release_year          8177 non-null   float64
dtypes: float64(4), int64(2), object(6)
memory usage: 830.5+ KB
```

Changing columns to string makes sure all entries, even numbers and NaN values, are treated as strings in those columns.

The keywords and production_companies columns have null values, but since I might not need these columns, I'll leave them as they are (impute them).

```
# Save the data before removing 'production_companies' and 'keywords' columns
# These columns will be used only in two specific research questions

# Create a copy of the data for company-related analysis
company_data = data.copy()

# Drop the 'keywords' column from the company_data
company_data.drop('keywords', axis=1, inplace=True)

# Remove rows with missing values in the 'production_companies' column
company_data.dropna(subset=['production_companies'], inplace=True)

# Remove NaN values from company_data
company_data = company_data[company_data != 'nan']

##########################################################################

# Create a copy of the data
keywords_data = data.copy()

# Drop the 'production_companies' column from the keywords_data
keywords_data.drop('production_companies', axis=1, inplace=True)

# Remove rows with missing values in the 'keywords' column
keywords_data.dropna(subset=['keywords'], inplace=True)
# Remove NaN values from keywords_data
keywords_data = keywords_data[keywords_data != 'nan']
```

```
# Verify the changes by checking for missing values in the modified datasets
print("Missing values in company_data:")
print(company_data.isnull().sum())

print("\nMissing values in keywords_data:")
print(keywords_data.isnull().sum())
```

```
Missing values in company_data:
popularity                0
budget                    0
revenue                   0
original_title            0
cast                      0
director                  0
runtime                   0
genres                    0
production_companies    816
vote_average              0
release_year              0
dtype: int64

Missing values in keywords_data:
popularity           0
budget               0
revenue              0
original_title       0
cast                 0
director             0
keywords          1259
runtime              0
genres               0
vote_average         0
release_year         0
dtype: int64
```

We save the data before removing the 'production_companies' and 'keywords' columns. These columns will be used only for two research questions. They are removed because they have many missing values. Dropping these rows would affect other columns, so we will handle the questions with the missing rows imputed.

```
data.drop(['production_companies','keywords'], axis = 1, inplace=True)
```

```
#Number of missing values in each column.
data.isnull().sum()
```

```
popularity        0
budget            0
revenue           0
original_title    0
cast              0
director          0
runtime           0
genres            0
vote_average      0
release_year      0
dtype: int64
```

```
# Remove NaN values
data = data[data != 'nan']
```

Review dataset

```
# Display the first 10 rows of the dataset to get an overview
data.head(10)
```

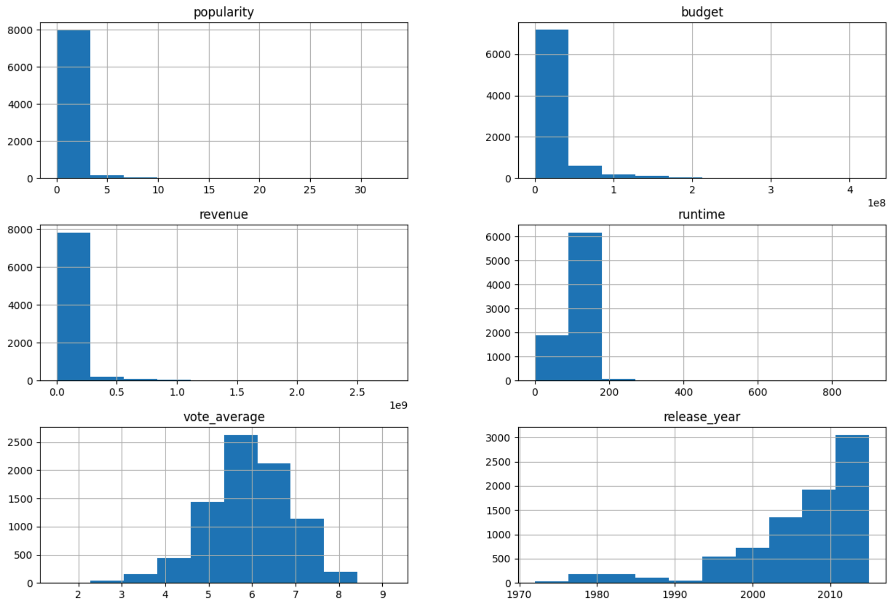
	popularity	budget	revenue	original_title	cast	director
0	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow
1	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller
2	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke
3	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams
					Vin Diesel|Paul	

Next steps: Generate code with data | View recommended plots

```
# Get the dimensions of the dataset (number of rows and columns)
dataset_dimensions = data.shape

# Display the dimensions
dataset_dimensions
```

```
(8177, 10)
```

```
# Plot the distribution of the numerical features in the dataset
data.hist(figsize=(15,10));
```



The popularity column's distribution, seen in the summary statistics, is skewed to the right. This is also true for the budget, revenue, vote_count, vote_average, and runtime columns. The release_year column is skewed to the left, showing that more movies were made or released from the 2000s to 2015 compared to earlier years.

Exploratory Data Analysis

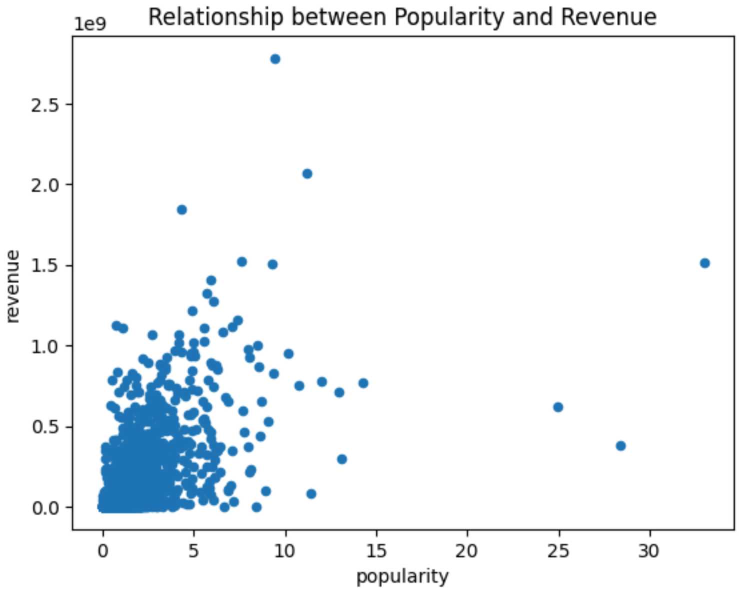
- Relationship Between Independent Features
- Total Number of Movies Produced by Year
- Total Number of Movies Produced by Genre

> Now that we've cleaned the data, we can start exploring it. In this section, we'll calculate statistics and make visualizations to answer the research questions from the Introduction.

General Questions

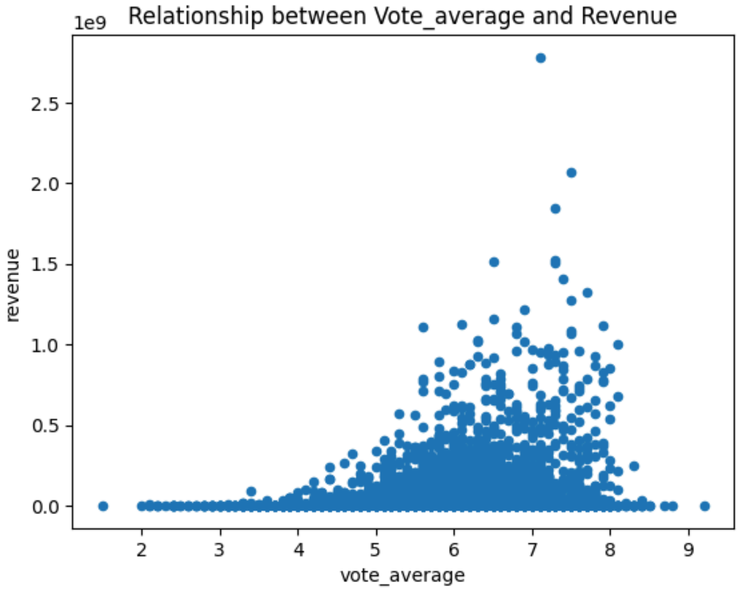
Q1. What is the relationship Between Independent Features

```
# Plots the relationship between two variables as a scatter plot.

#     Args:
#     data : DataFrame containing the data.
#     x_axis : Column name for the x-axis.
#     y_axis : Column name for the y-axis (default is 'revenue').
#     plot_kind : Type of plot (default is 'scatter').
#     title_prefix : Prefix for the title (default is 'Relationship between').

def plot_relationship(data, x_axis, y_axis='revenue', plot_kind='scatter', title_prefix='Relationship between'):
    data.plot(x=x_axis, y=y_axis, kind=plot_kind)
    plt.title(f'{title_prefix} {x_axis.capitalize()} and {y_axis.capitalize()}')
    plt.show()


# #Relationship between popularity and revenue.


plot_relationship(data, 'popularity')
```



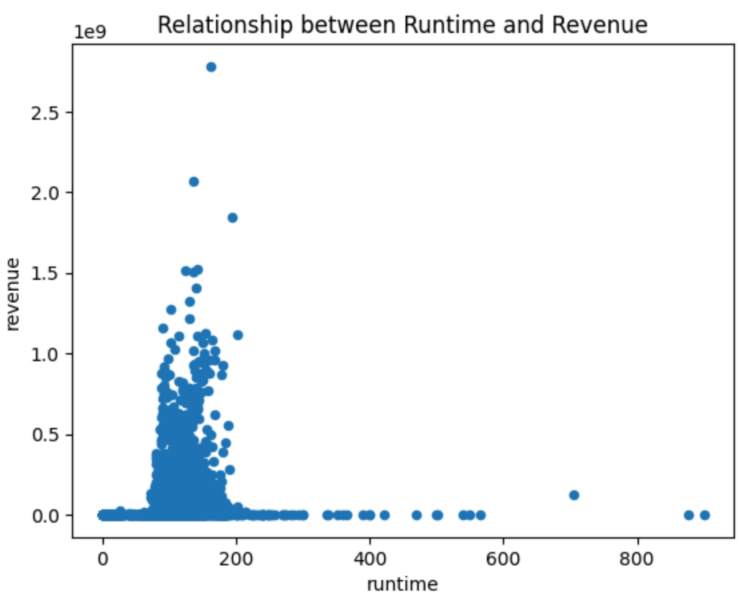
1. There is some positive correlation between popularity and revenue as this plot shows that the correlation is not that strong. This will be investigated further later in the analysis.
2. X-Axis (Popularity): Represents the independent variable, popularity. This axis measures how popular the instances (e.g., movies) are, with values ranging from 0 to above 30.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

```
# #Relationship between vote_average and revenue.

plot_relationship(data, 'vote_average')
```



1. There is some positive correlation between vote_average and revenue. This will be investigated further later in the analysis.
2. X-Axis (Vote Average): Represents the independent variable, vote average. This axis measures the average rating given to the instances (e.g., movies), with values ranging from 2 to 9.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

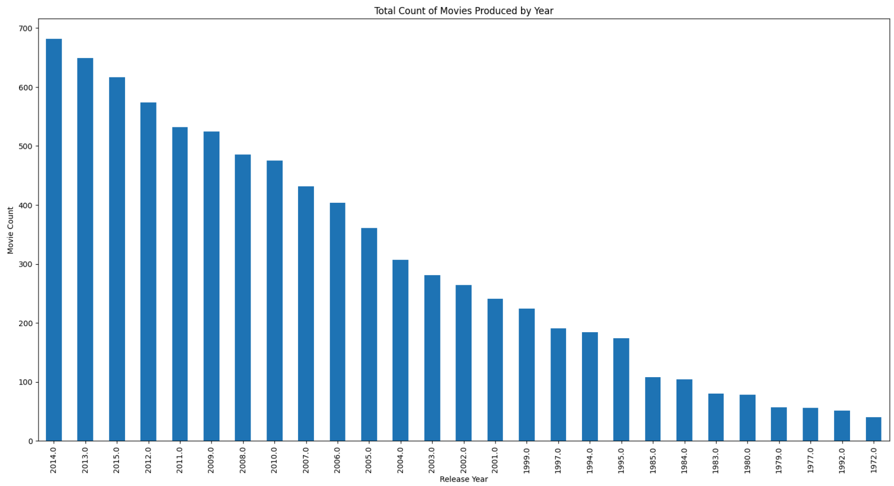
```
# #Relationship between runtime and revenue.

plot_relationship(data, 'runtime')
```



1. There is no correlation between runtime and revenue. This will be investigated further later in the analysis.
2. X-Axis (Runtime): Represents the independent variable, runtime. This axis measures the duration of the instances (e.g., movies) in minutes, with values ranging from 0 to over 800 minutes.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

Q2. What is the total Number of Movies Produced by Year?

```
#Value counts of movies for each year.
data.release_year.value_counts().plot(kind = 'bar', figsize = (20, 10));
plt.xlabel('Release Year');
plt.ylabel('Movie Count')
plt.title('Total Count of Movies Produced by Year');
```



The total amount of movies produced by year has been increasing steadily over the years. The year 2014 recorded the most number of movies released/produced.

Research Questions

Research Question 1: What are the most common movie genres?

> First thing we notice how the genres column is pipe separated and when we tried to check for unique values in returned all genres at once so we need to separate them

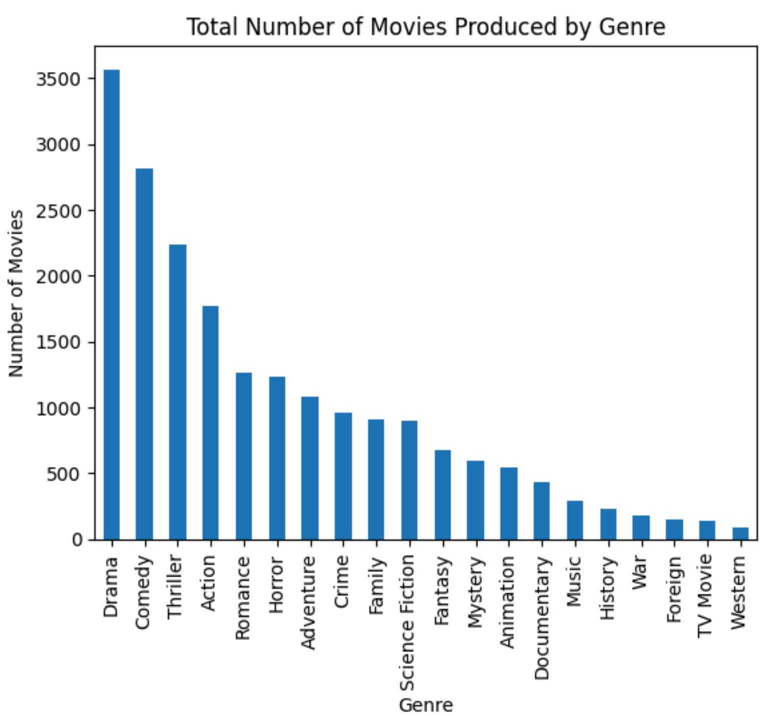
```
genres = pd.Series(data['genres'].str.split('|', expand=True).stack())
genre_counts = genres.value_counts()
genre_counts
```

```
Drama              3568
Comedy             2811
Thriller           2234
Action             1765
Romance            1259
Horror             1230
Adventure          1081
Crime               962
Family              908
Science Fiction     902
Fantasy             672
Mystery             593
Animation           548
Documentary         435
Music               289
History             232
War                 174
Foreign             145
TV Movie            139
Western              85
Name: count, dtype: int64
```

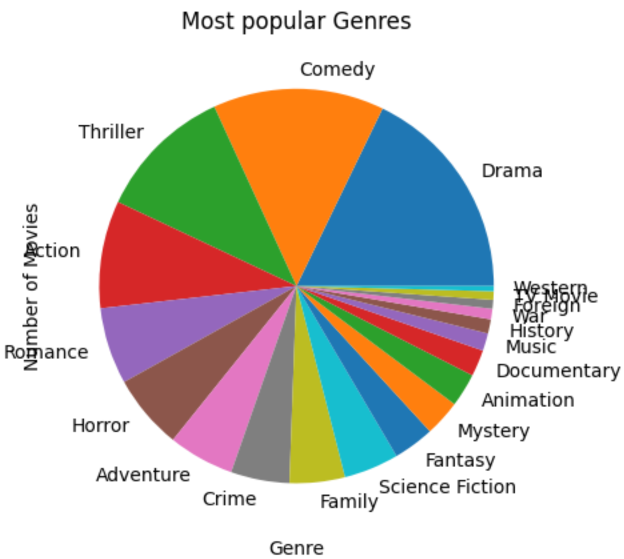
Conclusion:

> Drama movies are the most common genre followed by comedy and Thriller, Tv movies and Western movies are the least common genres

```
#get the visualization of this conclusion:
genres =  pd.Series(data['genres'].str.split('|', expand=True).stack()).value_counts()
genre_counts = genres
genre_counts.plot(kind='bar', title='Total Number of Movies Produced by Genre')

# Show the plot
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.show()
```




```
plotpie= genre_counts.plot.pie()
plotpie.set(title = 'Most popular Genres')
plotpie.set_xlabel('Genre')
plotpie.set_ylabel('Number of Movies')
plt.show()
```



Research Question 2: Who are the most cast actors?

> Notice that the cast column is separated the same way as the genres column so it's basically the same process

```
# Split the cast column and count occurrences

cast_split = pd.Series(data['cast'].str.split('|', expand=True).stack())

# Count occurrences
cast_count = cast_split.value_counts()
print(cast_count.head(10))
```

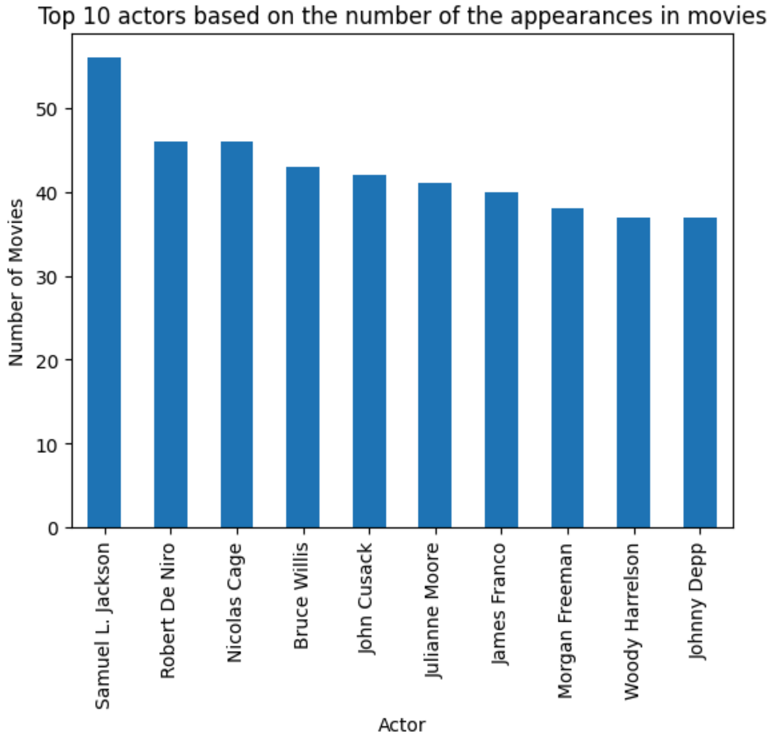
```
Samuel L. Jackson    56
Robert De Niro       46
Nicolas Cage         46
Bruce Willis         43
John Cusack          42
Julianne Moore       41
James Franco         40
Morgan Freeman       38
Woody Harrelson      37
Johnny Depp          37
Name: count, dtype: int64
```

conclusion:

> Robert De Niro is the most cast actor with 72 movies followed by Samuel L. Jackson with 71 movies, Bruce Willis comes third with 62 movies... Now we find the most common genre for each of these actors as well as the average rate for the movies they appeared in

> Here we can get the top 10 actors based on number of appearances in movies:

```
top_cast_counts = cast_count.head(10)
top_cast_counts.plot(kind='bar', title='Top 10 actors based on the number of the appearances in movies')
plt.xlabel('Actor')
plt.ylabel('Number of Movies')
plt.show()
```

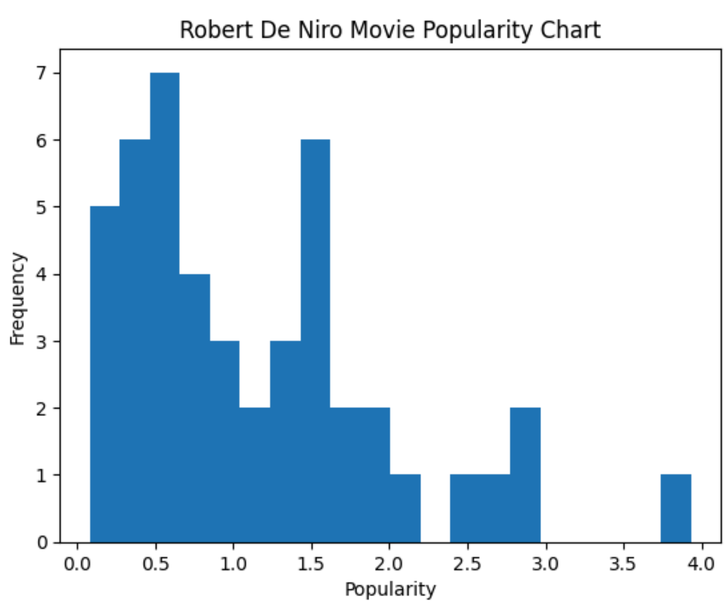


Conclusion:

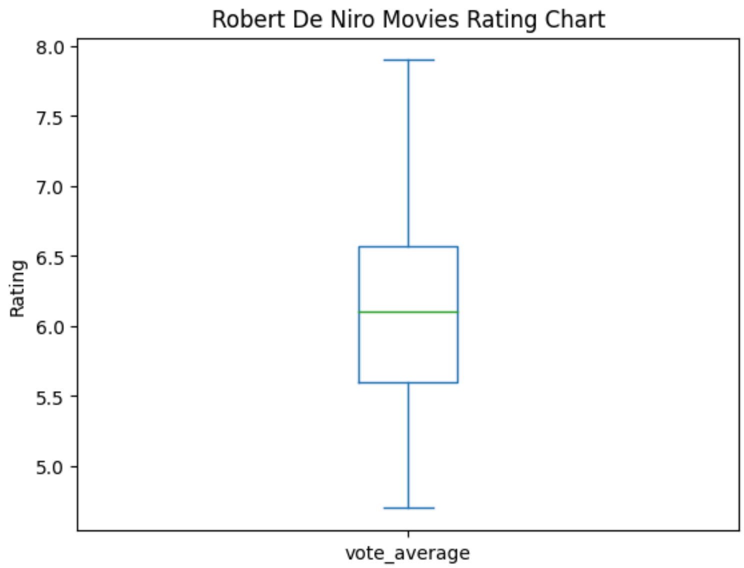
> Top 3 actors based on movie appearance are Robert De Niro, Samuel L.Jackson and Bruce Willis, we can now see a detailed visualization of the movies of the most popular actor

visualization of Robert De Niro movies:

```
# Filter the dataset for movies featuring Robert De Niro
filter1 = data[data['cast'].str.contains('Robert De Niro', na=False)]

# 1. Histogram of Robert De Niro movie popularity
filter1['popularity'].plot(kind='hist', title='Robert De Niro Movie Popularity Chart', bins=20)
plt.xlabel('Popularity')
plt.ylabel('Frequency')
plt.show()
```



```
# 2. Box plot of Robert De Niro movie ratings
filter1['vote_average'].plot(kind='box', title='Robert De Niro Movies Rating Chart')
plt.ylabel('Rating')
plt.show()
```




```
# Convert vote_average and runtime to Numpy arrays
vote_average_array = filter1['vote_average'].to_numpy()
runtime_array = filter1['runtime'].to_numpy()
```

```
# 3. Mean vote average of Robert De Niro movies using Numpy
mean_vote_average = np.mean(vote_average_array)
print(f"Mean vote average of Robert De Niro movies: {mean_vote_average}")
```

```
Mean vote average of Robert De Niro movies: 6.121739130434783
```

```
# 4. Mean runtime of Robert De Niro movies using Numpy
mean_runtime = np.mean(runtime_array)
print(f"Mean runtime of Robert De Niro movies: {mean_runtime} minutes")
```

```
Mean runtime of Robert De Niro movies: 110.6304347826087 minutes
```

Conclusion:

> from the previous examples we find that the average runtime of the most popular actor movies is 115 minutes, average rating is 6.33 and the popularity seems over 12

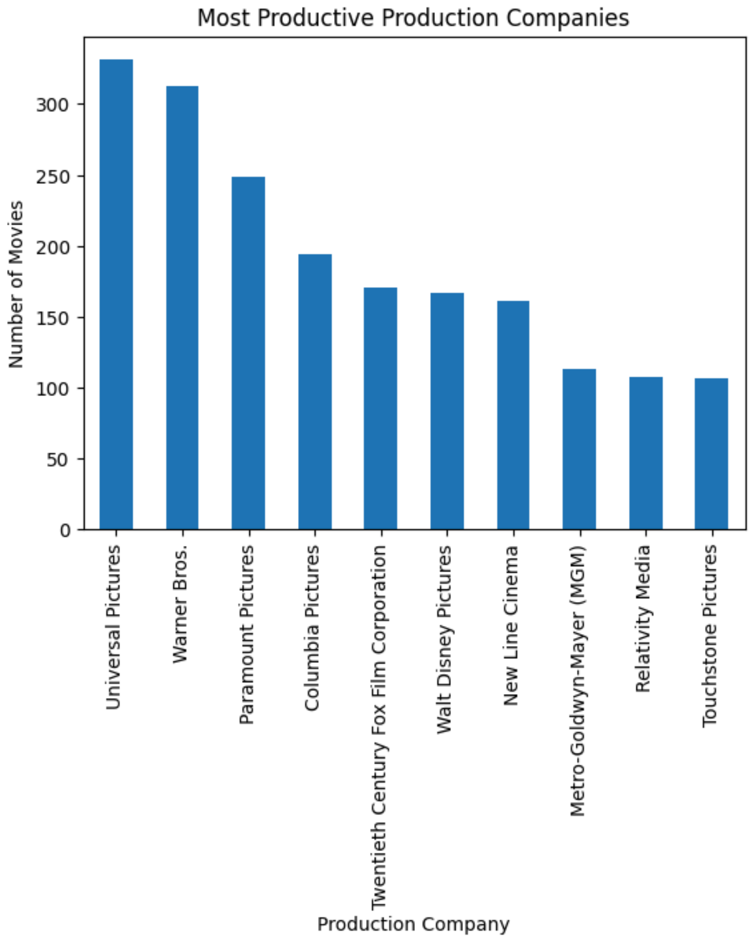
Research Question 3: What production company produces the most movies?

```
# Split the cast column and count occurrences

company_split = pd.Series(company_data['production_companies'].str.split('|', expand=True).stack())

# Count occurrences
company_count  = company_split.value_counts()
print(company_count.head(10))
```

```
Universal Pictures                        331
Warner Bros.                              313
Paramount Pictures                        249
Columbia Pictures                         194
Twentieth Century Fox Film Corporation    171
Walt Disney Pictures                      167
New Line Cinema                           161
Metro-Goldwyn-Mayer (MGM)                 113
Relativity Media                          108
Touchstone Pictures                       107
```

```
Name: count, dtype: int64
```

```
# Plot the most common production companies
top_company_counts = company_count.head(10)
top_company_counts.plot(kind='bar', title='Most Productive Production Companies')
plt.xlabel('Production Company')
plt.ylabel('Number of Movies')
plt.show()
```



Conclusion:

```
Universal Pictures is the company that produces most movies followed by Warner Bros. in 2nd place and Paramount pictures in 3rd
```

Research Question 4: What are the top movies in terms of profit?

Convert budget and revenue to numeric values (if not already)

```
# # Convert budget and revenue to numeric values (if not already)
# data['budget'] = pd.to_numeric(data['budget'], errors='coerce')
# data['revenue'] = pd.to_numeric(data['revenue'], errors='coerce')


def convert_to_numeric(data, columns):
    # Converts specified columns of a DataFrame to numeric types.
    for column in columns:
        data[column] = pd.to_numeric(data[column], errors='coerce')
    print(f"Columns converted to numeric: {columns}")

convert_to_numeric(data, ['budget', 'revenue'])
```

```
Columns converted to numeric: ['budget', 'revenue']
```

Identify the top 10 movies by revenue

```
# Identify the top 10 movies by revenue
top_revenue_movies = data[['original_title', 'revenue']].sort_values(by='revenue', ascending=False).head(10)

# Display the top 10 movies by revenue
print("Top 10 movies in terms of revenue:")
print(top_revenue_movies)
```

```
Top 10 movies in terms of revenue:
                                original_title     revenue
1386                                    Avatar  2781505847
3                 Star Wars: The Force Awakens  2068178225
5231                                   Titanic  1845034188
4361                              The Avengers  1519557910
0                               Jurassic World  1513528810
4                                    Furious 7  1506249360
14                     Avengers: Age of Ultron  1405035767
3374  Harry Potter and the Deathly Hallows: Part 2  1327817822
5422                                    Frozen  1274219009
5425                                Iron Man 3  1215439994
```

Identify the top 10 movies by budget

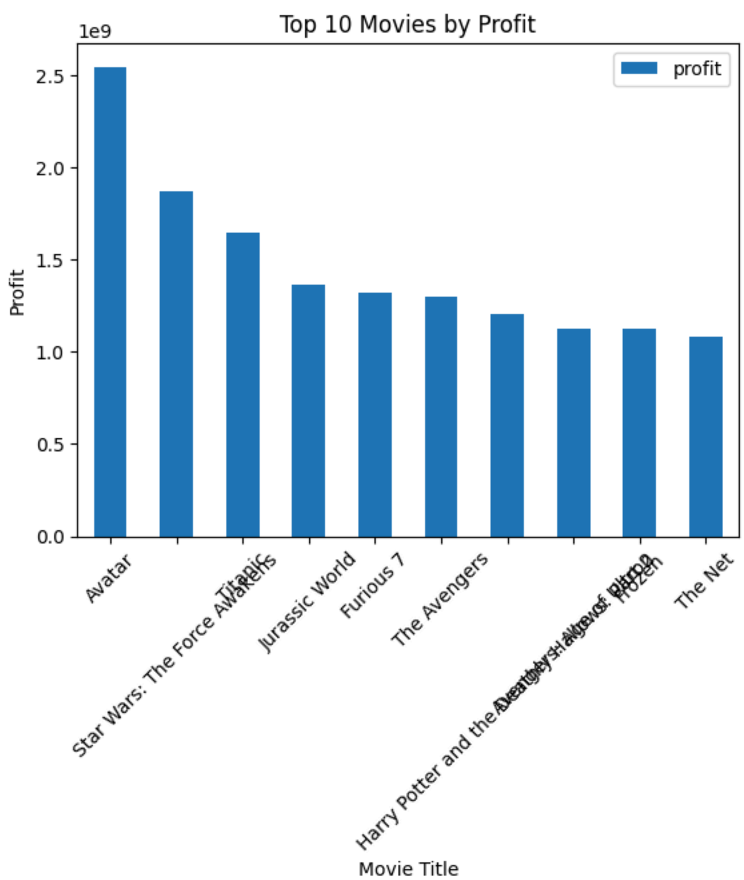
```
def display_top_movies_by_column(data, column):

    # Displays the top 10 movies sorted by a specified column.

    top_movies = data[['original_title', column]].sort_values(by=column, ascending=False).head(10)
    print(f"Top 10 movies in terms of {column}:")
    print(top_movies)

display_top_movies_by_column(data, 'budget')
```

```
Top 10 movies in terms of budget:
                               original_title     budget
2244                          The Warrior's Way  425000000
3375  Pirates of the Caribbean: On Stranger Tides  380000000
7387     Pirates of the Caribbean: At World's End  300000000
14                      Avengers: Age of Ultron  280000000
6570                           Superman Returns  270000000
4411                                John Carter  260000000
1929                                    Tangled  260000000
7394                               Spider-Man 3  258000000
5508                            The Lone Ranger  255000000
4363                      The Dark Knight Rises  250000000
```

Calculate profit then Sort by profit in descending order and Display the top 10 movies

```
def sort_and_display_top_movies_by_profit(data):

    # Sorts by profit in descending order and displays the top 10 movies.

    data['profit'] = data['revenue'] - data['budget']  # Calculate profit
    sorted_data = data.sort_values(by='profit', ascending=False)
    top_profit_movies = sorted_data[['original_title', 'profit']].head(10)
    print("Top 10 movies in terms of profit:")
    print(top_profit_movies)

    # Plot the top 10 movies in terms of profit
    top_profit_movies.plot(kind='bar', x='original_title', y='profit', title='Top 10 Movies by Profit')
    plt.xlabel('Movie Title')
    plt.ylabel('Profit')
    plt.xticks(rotation=45)
    plt.show()


sort_and_display_top_movies_by_profit(data)
```

```
Top 10 movies in terms of profit:
                                   original_title      profit
1386                                       Avatar  2544505847
3                    Star Wars: The Force Awakens  1868178225
5231                                      Titanic  1645034188
0                                  Jurassic World  1363528810
4                                       Furious 7  1316249360
4361                                 The Avengers  1299557910
3374  Harry Potter and the Deathly Hallows: Part 2  1202817822
14                        Avengers: Age of Ultron  1125035767
5422                                       Frozen  1124219009
8094                                      The Net  1084279658
```



Conclusion:

> 1. Avatar is the most successul movie based on profit, followed by Star Wars and Titanic in 3rd
> 2. The movies with the most revenue weren't necessarily the most profitable and the movies with the most budget weren't necessarily the most profitable

Research Question 5: What are the top movies based on popularity?

Convert budget and revenue to numeric values (if not already)

```
# Convert budget and revenue to numeric values (if not already)

convert_to_numeric(data, ['budget', 'revenue'])
```

```
Columns converted to numeric: ['budget', 'revenue']
```

Identify the top 10 movies by popularity

```
# Identify the top 10 movies by popularity
top_popularity_movies = data[['original_title', 'popularity']].sort_values(by='popularity', ascending=False).head(10)

# Display the top 10 movies by popularity
print("Top 10 movies in terms of popularity:")
print(top_popularity_movies)
```

```
Top 10 movies in terms of popularity:
                          original_title  popularity
0                         Jurassic World   32.985763
1                     Mad Max: Fury Road   28.419936
629                         Interstellar   24.949134
630              Guardians of the Galaxy   14.311205
2                              Insurgent   13.112507
631      Captain America: The Winter Soldier   12.971027
1329                           Star Wars   12.037933
632                            John Wick   11.422751
3           Star Wars: The Force Awakens   11.173104
633   The Hunger Games: Mockingjay - Part 1   10.739009
```

Identify the top 10 movies by budget

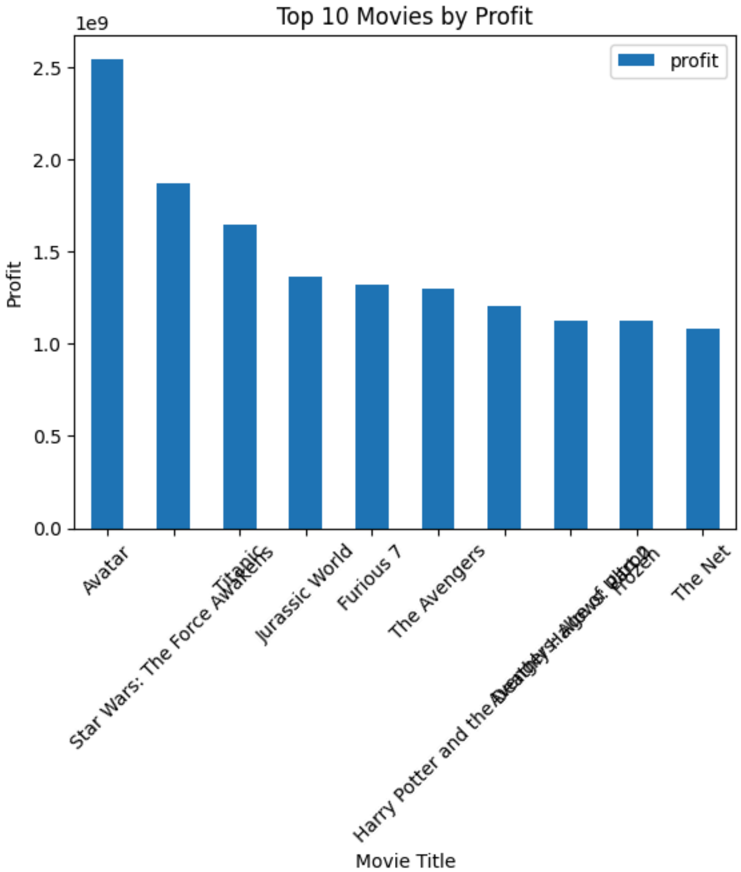
```
# Identify the top 10 movies by budget
display_top_movies_by_column(data, 'budget')
```

```
Top 10 movies in terms of budget:
                                    original_title     budget
2244                              The Warrior's Way  425000000
3375  Pirates of the Caribbean: On Stranger Tides  380000000
7387     Pirates of the Caribbean: At World's End  300000000
14                         Avengers: Age of Ultron  280000000
6570                              Superman Returns  270000000
4411                                   John Carter  260000000
1929                                       Tangled  260000000
7394                                  Spider-Man 3  258000000
5508                               The Lone Ranger  255000000
4363                         The Dark Knight Rises  250000000
```

Calculate profit then Sort by profit in descending order and Display the top 10 movies

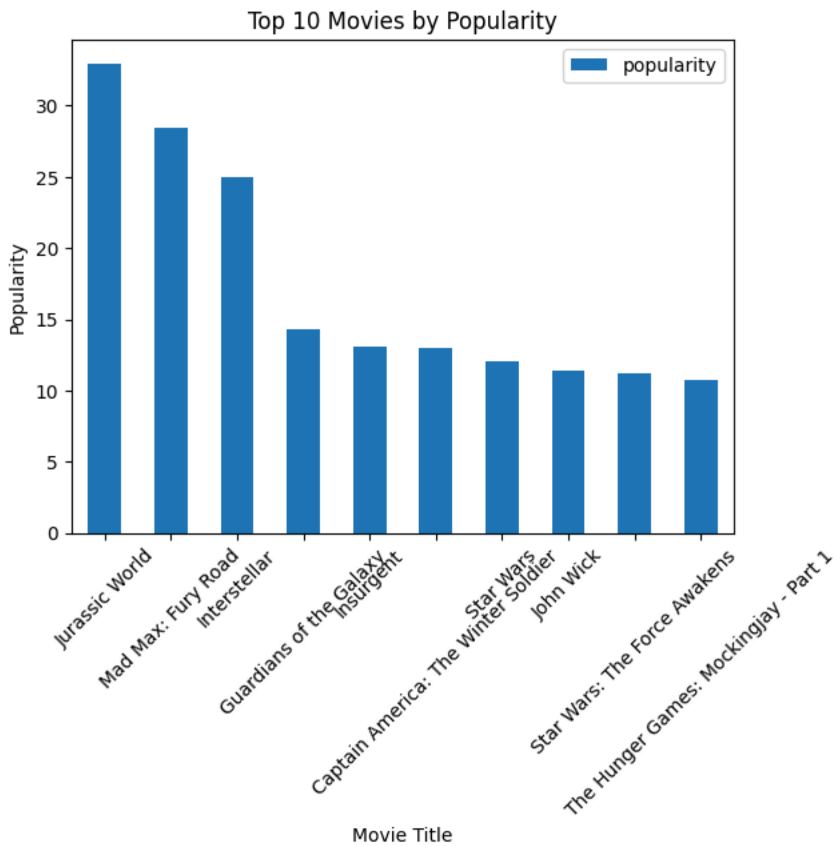
```
sort_and_display_top_movies_by_profit(data)
```

```
Top 10 movies in terms of profit:
                                      original_title      profit
1386                                          Avatar  2544505847
3                   Star Wars: The Force Awakens  1868178225
5231                                         Titanic  1645034188
0                                     Jurassic World  1363528810
4                                          Furious 7  1316249360
4361                                    The Avengers  1299557910
3374  Harry Potter and the Deathly Hallows: Part 2  1202817822
14                               Avengers: Age of Ultron  1125035767
5422                                          Frozen  1124219009
8094                                         The Net  1084279658
```



Plot the top 10 movies by popularity

```
# Plot the top 10 movies by popularity
top_popularity_movies.plot(kind='bar', x='original_title', y='popularity', title='Top 10 Movies by Popularity')
plt.xlabel('Movie Title')
plt.ylabel('Popularity')
plt.xticks(rotation=45)
plt.show()
```



Conclusion:

- High Popularity and High Profit

1. Jurassic World and Star Wars: The Force Awakens are both highly popular and highly profitable, showing a link between popularity and profit.

- High Budget Doesn't Always Mean High Popularity or Profit

1. Movies like The Warrior's Way and Pirates of the Caribbean: On Stranger Tides have large budgets but aren't in the top 10 for popularity or profit.
2. Movies with smaller budgets, like Frozen and The Net, can still be very profitable.

Research Question 6: What are the top movies based on viewer rating?

print the top 10 movies by viewer rating

```
# Identify the top 10 movies by viewer rating
top_rated_movies = data[['original_title', 'vote_average']].sort_values(by='vote_average', ascending=False).head(10)

# Display the top 10 movies by viewer rating
print("Top 10 movies in terms of viewer rating:")
print(top_rated_movies)
```

```
Top 10 movies in terms of viewer rating:
                                        original_title  vote_average
3894                      The Story of Film: An Odyssey           9.2
1200                       Black Mirror: White Christmas           8.8
6911                                    Pink Floyd: Pulse           8.7
3690                                   The Art of Flight           8.5
8221  A Personal Journey with Martin Scorsese Throug...           8.5
609          The Jinx: The Life and Deaths of Robert Durst           8.4
4178                             The Shawshank Redemption           8.4
7948                                    Stop Making Sense           8.4
```

```
2334                Rush: Beyond the Lighted Stage           8.4
5986                             Guten Tag, RamÃ³n           8.4
```

Convert budget and revenue to numeric values (if not already)

```
# Convert budget and revenue to numeric values (if not already)

convert_to_numeric(data, ['budget', 'revenue'])
```

```
Columns converted to numeric: ['budget', 'revenue']
```

Identify the top 10 movies by budget

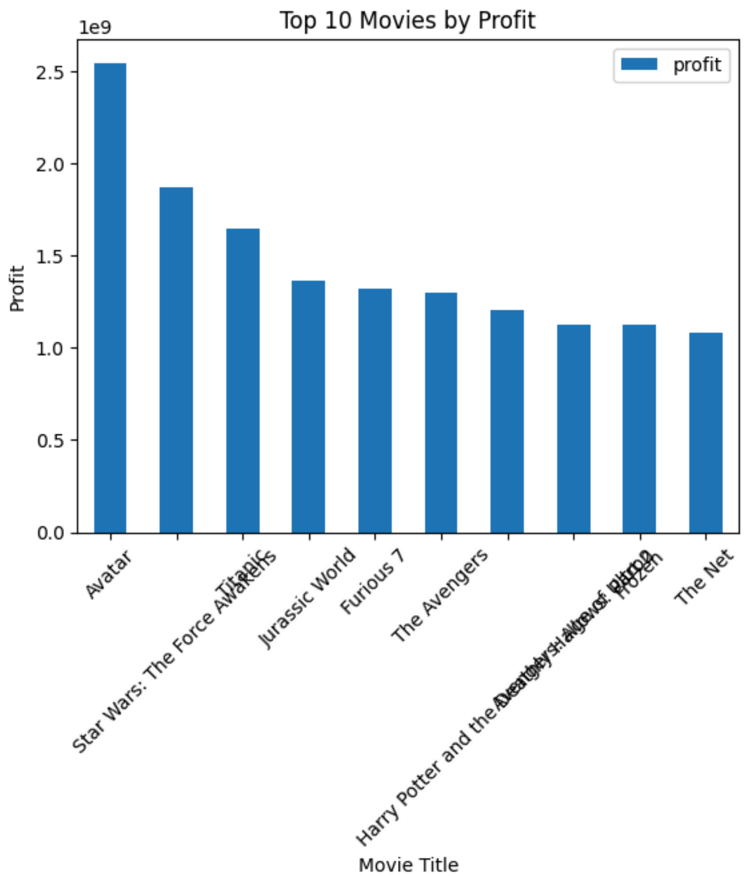
```
# Identify the top 10 movies by budget
display_top_movies_by_column(data, 'budget')
```

```
Top 10 movies in terms of budget:
                                   original_title     budget
2244                              The Warrior's Way  425000000
3375  Pirates of the Caribbean: On Stranger Tides  380000000
7387     Pirates of the Caribbean: At World's End  300000000
14                          Avengers: Age of Ultron  280000000
6570                               Superman Returns  270000000
4411                                    John Carter  260000000
1929                                        Tangled  260000000
7394                                   Spider-Man 3  258000000
5508                                The Lone Ranger  255000000
4363                          The Dark Knight Rises  250000000
```

Calculate profit then Sort by profit in descending order and Display the top 10

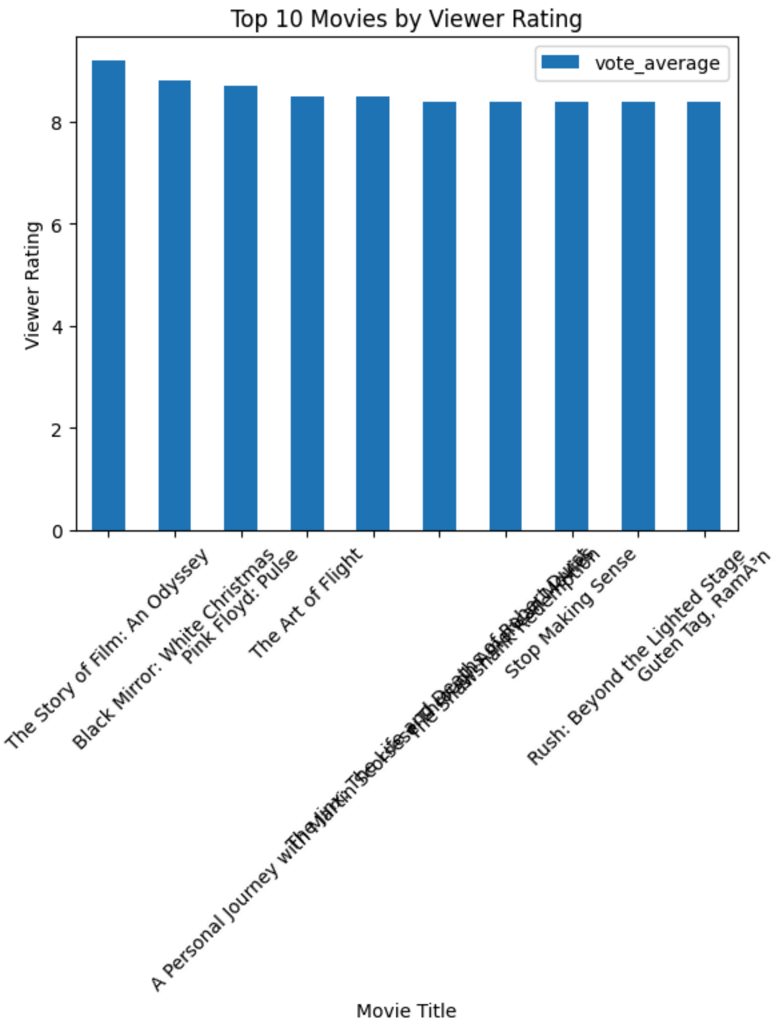
```
sort_and_display_top_movies_by_profit(data)
```

```
Top 10 movies in terms of profit:
                                   original_title      profit
1386                                       Avatar  2544505847
3                    Star Wars: The Force Awakens  1868178225
5231                                      Titanic  1645034188
0                                  Jurassic World  1363528810
4                                       Furious 7  1316249360
4361                                 The Avengers  1299557910
3374  Harry Potter and the Deathly Hallows: Part 2  1202817822
14                        Avengers: Age of Ultron  1125035767
5422                                       Frozen  1124219009
8094                                      The Net  1084279658
```



Plot the top 10 movies by viewer rating

```
# Plot the top 10 movies by viewer rating
top_rated_movies.plot(kind='bar', x='original_title', y='vote_average', title='Top 10 Movies by Viewer Rating')
plt.xlabel('Movie Title')
plt.ylabel('Viewer Rating')
plt.xticks(rotation=45)
plt.show()
```



Average runtime of the highest rated movies:

```
high = data.nlargest(10, ['vote_average'])

# Calculate the average runtime of the highest rated movies
average_runtime = high['runtime'].mean()

# Print the average runtime
print(f"Average runtime of the highest rated movies: {average_runtime} minutes")
```

```
Average runtime of the highest rated movies: 205.9 minutes
```

Conclusion:

> - Average runtime of the highest rated movies is 210.8 minutes
> - High viewer ratings do not always correlate with high budgets or profits.
> - High budget movies often make high profits but not necessarily high viewer ratings.
> - The most profitable movies are not always the top-rated ones, showing different factors drive profitability and viewer appreciation.

Research Question 7: What are the most common keywords?

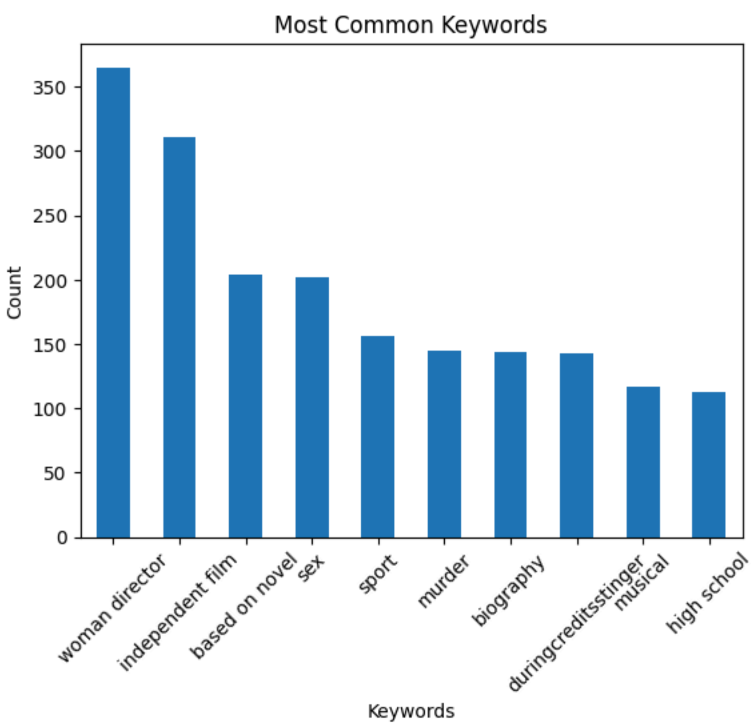
Split the keywords column and count occurrences

```
# Split the cast column and count occurrences

words_split = pd.Series(keywords_data['keywords'].str.split('|', expand=True).stack())

# Count occurrences
keywords_count  = words_split.value_counts()
print(keywords_count.head(10))
```

```
woman director         365
independent film       311
based on novel         204
sex                    202
sport                  156
murder                 145
biography              144
duringcreditsstinger   143
musical                117
high school            113
Name: count, dtype: int64
```

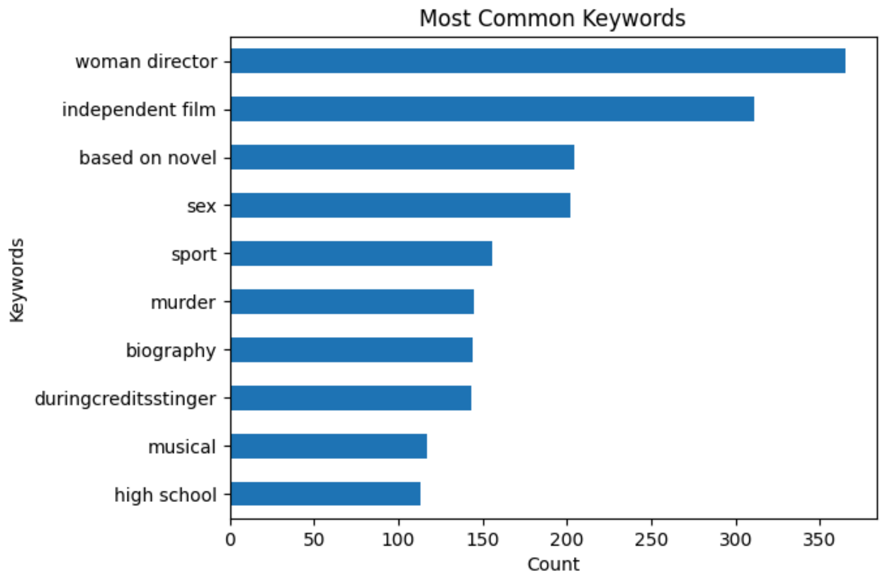
Plot the most common keywords using bar kind

```
# Plot the most common keywords
top_keywords = keywords_count.head(10)
top_keywords.plot(kind='bar', title='Most Common Keywords')
plt.xlabel('Keywords')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



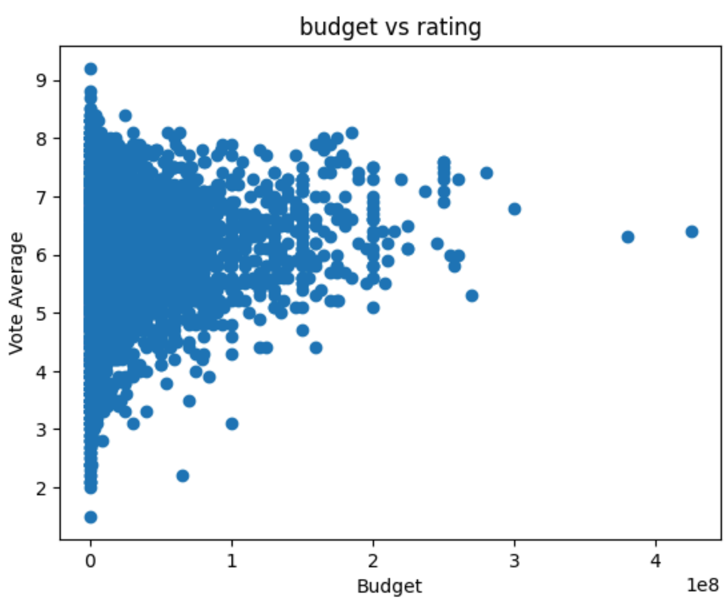
Plot the most common keywords using barh kind

```
# Plot the most common keywords using a horizontal bar plot
top_keywords = keywords_count.head(10)
top_keywords.plot(kind='barh', title='Most Common Keywords')
plt.xlabel('Count')
plt.ylabel('Keywords')
plt.gca().invert_yaxis()  # Invert y-axis to have the most common keyword at the top
plt.show()
```



Research question 8: Is the budget related to a higher average vote?

```
from importlib import reload
plt=reload(plt)
plt.scatter(x=data['budget'], y=data['vote_average'])
plt.title("budget vs rating")
plt.xlabel("Budget")
plt.ylabel('Vote Average')
plt.show()
```

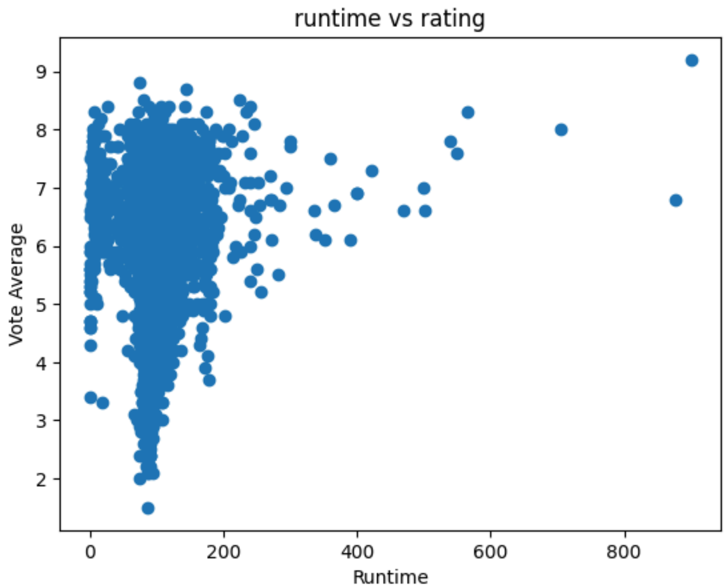


Conlusions:

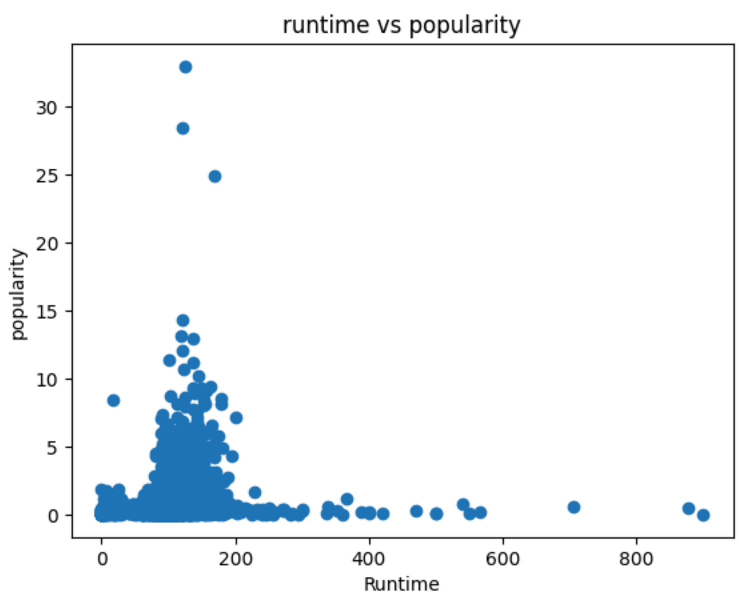
> 1. The chart shows that spending more money on making a movie doesn't always mean it will get better ratings.
> 2. Movies with both high and low budgets often get similar ratings, showing that money isn't the only thing that matters.
> 3. There are expensive movies with good ratings and cheap movies with good ratings, proving that many different things can make a movie popular or not.

Research Question 9: what's the correlation between runtime and each of popularity, rating, popularity?

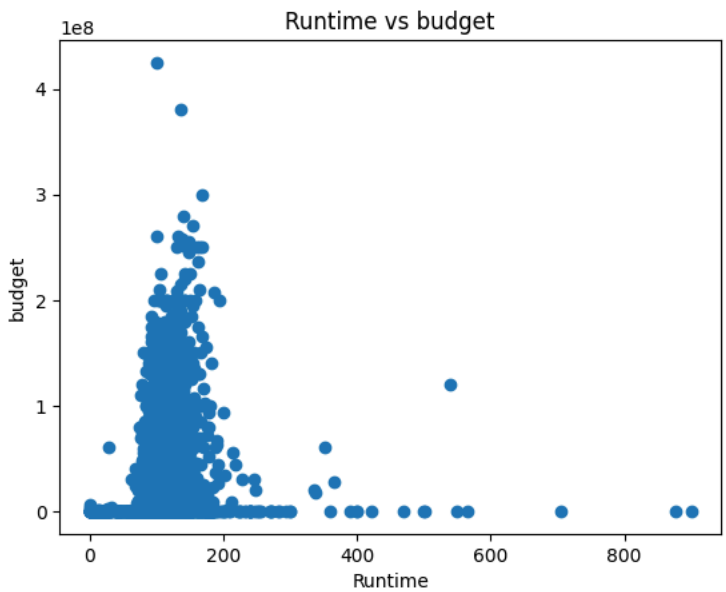
```
plt.scatter(x=data['runtime'], y=data['vote_average'])
plt.title("runtime vs rating")
plt.xlabel("Runtime")
plt.ylabel('Vote Average')
plt.show()
```




```
plt.scatter(x=data['runtime'], y=data['popularity'])
plt.title("runtime vs popularity")
plt.xlabel("Runtime")
plt.ylabel('popularity')
plt.show()
```




```
plt.scatter(x=data['runtime'], y=data['budget'])
plt.title("Runtime vs budget")
plt.xlabel("Runtime")
plt.ylabel('budget')
plt.show()
```



Conclusion:

> 1. Budget vs. Runtime: Generally, longer movies have higher budgets, but most expensive movies are under 200 minutes long.
> 2. Popularity vs. Runtime: Shorter movies (under 200 minutes) are usually more popular. Very long movies are less popular.
> 3. Rating vs. Runtime: Longer movies tend to get slightly better ratings, but the effect is not strong. Most movies have similar ratings, no matter their length.

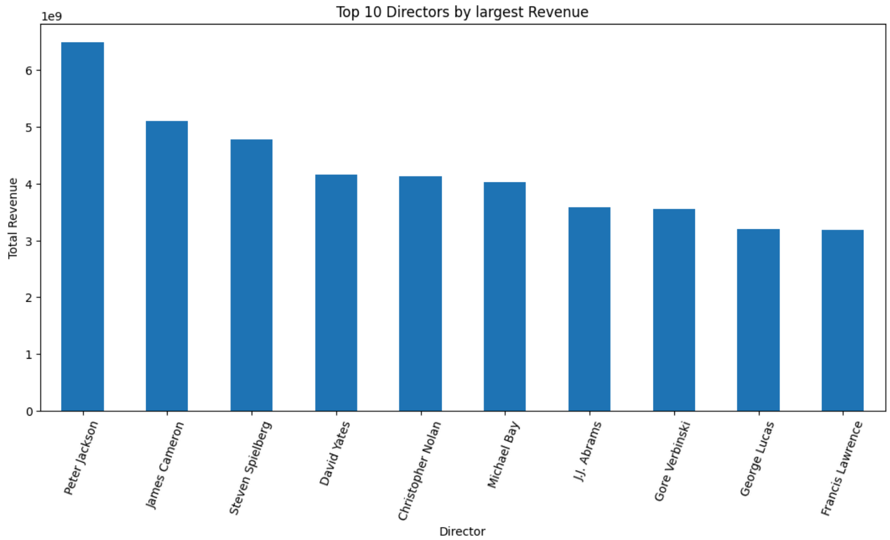
Research Question 10: Who are the most successful directors?

```
Most successful director is the one who generated the most revenue
```

```
dir_rev = data.groupby(['director']).sum()['revenue'].nlargest(10)
dir_rev
```

```
director
Peter Jackson        6493885443
James Cameron        5100834517
Steven Spielberg     4779905151
David Yates          4154295625
Christopher Nolan    4127825406
Michael Bay          4028345984
J.J. Abrams          3579169916
Gore Verbinski       3548779679
George Lucas         3199113893
Francis Lawrence     3179979588
Name: revenue, dtype: int64
```

```
dir_rev.plot(kind = 'bar', figsize=(13,6))
plt.title("Top 10 Directors by largest Revenue")
plt.xticks(rotation=70)
plt.xlabel("Director")
plt.ylabel("Total Revenue")
plt.show()
```



Conclusion:

> 1. Steven Spielberg is the most successful director in terms of revenue
> 2. he's followed by Peter Jackson, while James Cameron comes 3rd

Research Question 11: How did the runtime of movies change over the years? What Movie has the longest runtime? what movie has the shortest runtime? what's the average movie runtime?

Average movie runtime:

```
# Calculating the average runtime for all movies
average_runtime = data['runtime'].mean()
print("\nAverage Movie Runtime:", average_runtime)
```

```
Average Movie Runtime: 101.26598997187233
```

Longest movie:

```
# Finding the movie with the longest runtime
longest_runtime_movie = data[data['runtime'] == data['runtime'].max()]
print("\nMovie/s with the Longest Runtime:\n", longest_runtime_movie[['original_title', 'runtime']])
```

```
Movie/s with the Longest Runtime:
                      original_title  runtime
3894  The Story of Film: An Odyssey    900.0
```

Shortest movie:

```
# Finding the movie with the shortest runtime
shortest_runtime_movie = data[data['runtime'] == data['runtime'].min()]
```

```
print("\nMovie/s with the Shortest Runtime:\n", shortest_runtime_movie[['original_title', 'runtime']])
```

```
Movie/s with the Shortest Runtime:
                      original_title  runtime
92        Mythica: The Necromancer      0.0
334                        Ronaldo      0.0
410                 Anarchy Parlor      0.0
445   The Exorcism of Molly Hartley      0.0
486             If There Be Thorns      0.0
595                      Deep Dark      0.0
616                    The Outfield      0.0
1289                      Treehouse      0.0
1293                       Tim Maia      0.0
1849                   Spectacular!      0.0
3329      Grande, grosso e Verdone      0.0
3794          Toi, moi, les autres      0.0
3857                       Cell 213      0.0
3884                         eCupid      0.0
4063        Madea's Family Reunion      0.0
4138            A Time for Dancing      0.0
4829                           Rags      0.0
4944              How to Fall in Love      0.0
5216         Madea's Class Reunion      0.0
5695               Skinwalker Ranch      0.0
5920          The Food Guide to Love      0.0
5938                    Go Goa Gone      0.0
5992               Amiche da morire      0.0
```