0trilek7d

June 20, 2023

1 Project: Investigate a Dataset - [TMDB Movies]

1.1 Table of Contents

Introduction

1.2 Objective

Inspecting the TMDB Dataset and gaining insights to understand the success factors of movies

1.2.1 Dataset Description

In this project, we are using the TMDB Movies dataset which has several attributes like 'id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title', 'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview', 'runtime', 'genres', 'production_companies', 'release_date', 'vote_count', 'vote_average', 'release_year', 'budget_adj', 'revenue_adj' and these attributes would help in analysing more about the movie data and gaining insight about the popularity and success of the movie

1.2.2 Research Questions for Analysis

1. Are there any specific Actors who have made the highest number of appearances in films and would help in determing success or failure of movies?
2. Are there any Genres which are the most famous and would help directors or producers to look for?
3. Which are the movies with highest, lowest budgets and what is the average budget of the movies?
4. which are the movies with highest profit?
5. which movies have the highest runtime and do they impact the sucess?
6. Which movie has generated the highest revenue and what were its attributes?
7. Are there any outliers in the data?
8. Is there any dependency for the profits earned and the release time of the movies?

9. Which is the best movie depending on the popularity?
10. Are any any more correlatins between different features?
11. Does the budget and revenue of movies impact their popularity?

Data Wrangling

1.2.3 Loading required Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

1.2.4 Reading Dataset

[2]:
```
# Reading dataset
movie_data = pd.read_csv("tmdb-movies.csv")
```

1.2.5 Inspecting first few rows

[3]:
```
# showing first few rows
movie_data.head()
```

[3]:
```
       id    imdb_id  popularity     budget     revenue  \
0  135397  tt0369610   32.985763  150000000  1513528810
1   76341  tt1392190   28.419936  150000000   378436354
2  262500  tt2908446   13.112507  110000000   295238201
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360

                 original_title  \
0                Jurassic World
1            Mad Max: Fury Road
2                     Insurgent
3  Star Wars: The Force Awakens
4                     Furious 7

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…
2  Shailene Woodley|Theo James|Kate Winslet|Ansel…
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D…
```

```
4  Vin Diesel|Paul Walker|Jason Statham|Michelle …

                                            homepage          director  \
0                        http://www.jurassicworld.com/   Colin Trevorrow
1                         http://www.madmaxmovie.com/     George Miller
2   http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod…       J.J. Abrams
4                            http://www.furious7.com/         James Wan

                        tagline  …  \
0             The park is open.  …
1            What a Lovely Day.  …
2      One Choice Can Destroy You  …
3  Every generation has a story.  …
4           Vengeance Hits Home  …

                                            overview runtime  \
0  Twenty-two years after the events of Jurassic …     124
1  An apocalyptic story set in the furthest reach…     120
2  Beatrice Prior must confront her inner demons …     119
3  Thirty years after defeating the Galactic Empi…     136
4  Deckard Shaw seeks revenge against Dominic Tor…     137

                                      genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                       Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda…       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ…      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago…      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights …       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

[4]:
```
movie_data.columns
```

[4]: Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')

1.2.6 Checking the shape of data

```
[5]: movie_data.shape
```

[5]: (10866, 21)

1.2.7 Checking the info about the data

```
[6]: movie_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10866 non-null  int64
 1   imdb_id               10856 non-null  object
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64
 4   revenue               10866 non-null  int64
 5   original_title        10866 non-null  object
 6   cast                  10790 non-null  object
 7   homepage              2936 non-null   object
 8   director              10822 non-null  object
 9   tagline               8042 non-null   object
 10  keywords              9373 non-null   object
 11  overview              10862 non-null  object
 12  runtime               10866 non-null  int64
 13  genres                10843 non-null  object
 14  production_companies  9836 non-null   object
 15  release_date          10866 non-null  object
 16  vote_count            10866 non-null  int64
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

Noted Observations: - The dataset contains a total of 10,866 entries and 21 columns, representing various attributes of movies. - The dataset has a mix of different data types, including integers, floats, and objects, indicating the presence of both numerical and categorical features.

1.2.8 Descriptive statistics of the data

[7]:
```
movie_data.describe()
```

[7]:
```
                  id    popularity        budget       revenue       runtime  \
count   10866.000000  10866.000000  1.086600e+04  1.086600e+04  10866.000000
mean    66064.177434      0.646441  1.462570e+07  3.982332e+07    102.070863
std     92130.136561      1.000185  3.091321e+07  1.170035e+08     31.381405
min         5.000000      0.000065  0.000000e+00  0.000000e+00      0.000000
25%     10596.250000      0.207583  0.000000e+00  0.000000e+00     90.000000
50%     20669.000000      0.383856  0.000000e+00  0.000000e+00     99.000000
75%     75610.000000      0.713817  1.500000e+07  2.400000e+07    111.000000
max    417859.000000     32.985763  4.250000e+08  2.781506e+09    900.000000

         vote_count  vote_average  release_year    budget_adj   revenue_adj
count  10866.000000  10866.000000  10866.000000  1.086600e+04  1.086600e+04
mean     217.389748      5.974922   2001.322658  1.755104e+07  5.136436e+07
std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

Noted Observations:

- The "popularity" feature has a mean of 0.646, indicating that the movies' popularity varies, with some being more popular than others.
- The "budget" feature has a mean of approximately 14.63 million, indicating that the movies in the dataset have a wide range of budgets.
- The "revenue" feature has a mean of approximately 39.82 million, indicating that the movies have varying levels of revenue generated.
- The "runtime" feature has a mean of approximately 102.07 minutes, suggesting that the movies' durations vary.
- The "vote_count" feature has a mean of approximately 217.39, indicating that the movies have received varying numbers of votes from viewers.
- The "vote_average" feature has a mean of 5.97, suggesting that the movies' average ratings range from low to high.
- The "release_year" feature indicates the year of movie release, with a range from 1960 to 2015.
- The "budget_adj" and "revenue_adj" features represent the budget and revenue adjusted for inflation, respectively.
- Some movies have missing values in the "budget," "revenue," "budget_adj," and "revenue_adj" features, as the minimum values for these features are zero.
- The maximum values in the "popularity," "budget," "revenue," "runtime," "vote_count,"

"vote_average," "budget_adj," and "revenue_adj" features suggest the presence of outliers in the dataset.

1.2.9 Checking for null values in the data

[8]:
```
movie_data.isnull().sum()
```

[8]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

Noted observations: - This data has missing values in several features. - The "imdb_id" column has 10 missing values. - The "cast" column has 76 missing values. - The "homepage" column has a significant number of missing values, with 7930 missing entries. - The "director" column has 44 missing values. - The "tagline" column has 2824 missing values. - The "keywords" column has 1493 missing values. - The "overview" column has 4 missing values. - The "genres" column has 23 missing values. - The "production_companies" column has 1030 missing values.

1.3 Data cleaning

[9]:
```
movie_data.columns
```

[9]:
```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

- It is observed that there are several features which might not be required for our analysis so we would be deleting them from our data

1.3.1 Deleting unrequired features

[10]:
```
# current unrequired features
unrequired_features = ['id','imdb_id','homepage','tagline', 'keywords',␣
↪'overview']
movie_data = movie_data.drop(unrequired_features, axis = 1)
```

1.3.2 Removing duplicate values

[11]:
```
movie_data.duplicated().sum()
```

[11]: 1

- There is only 1 duplicate feature so it can be observed and also removed from the data

[12]:
```
movie_data[movie_data.duplicated()]
```

[12]:
```
      popularity    budget  revenue original_title  \
2090     0.59643  30000000   967000         TEKKEN

                                                   cast          director  \
2090  Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian…  Dwight H. Little

      runtime                                       genres  \
2090       92  Crime|Drama|Action|Thriller|Science Fiction

        production_companies release_date  vote_count  vote_average  \
2090  Namco|Light Song Films      3/20/10         110           5.0

      release_year  budget_adj  revenue_adj
2090          2010  30000000.0     967000.0
```

[13]:
```
print("shape of data before dropping duplicates: ",movie_data.shape)
movie_data = movie_data.drop_duplicates(keep ='first')
print("shape of data after dropping duplicates: ",movie_data.shape)
```

```
shape of data before dropping duplicates:  (10866, 15)
shape of data after dropping duplicates:  (10865, 15)
```

1.3.3 Handling 0 value in budget and revenue column

[14]:
```
sorted(movie_data['budget'].unique())[:5]
```

[14]: [0, 1, 2, 3, 5]

```
[15]: sorted(movie_data['revenue'].unique())[:5]
```

```
[15]: [0, 2, 3, 5, 6]
```

- It can be observed that there are zeros in both budget and revenue column but it should be typically more so these zeros would be replaced with Nan and then handled

```
[16]: movie_data['budget'] = movie_data['budget'].replace(0, np.NAN)
      movie_data['revenue'] = movie_data['revenue'].replace(0, np.NAN)
```

```
[17]: movie_data['budget'].isnull().sum(), movie_data['revenue'].isnull().sum()
```

```
[17]: (5696, 6016)
```

- It can be observed that there are 5696 rows which have budget as 0 and 6016 rows where revenue was 0 so we would impute these values using median

```
[18]: # imputing the null values using the median
      # Calculating the median value
      median_value1 = movie_data['budget'].median()
      median_value2 = movie_data['revenue'].median()

      # Imputing null values with the median
      movie_data['budget'].fillna(median_value1, inplace=True)
      movie_data['revenue'].fillna(median_value2, inplace=True)
```

```
[19]: movie_data['budget'].isnull().sum(), movie_data['revenue'].isnull().sum()
```

```
[19]: (0, 0)
```

1.3.4 Converting budget and revenue to required data type

```
[20]: # converting budget and revenue to only numeric than float which would help in␣
       ↪plotting
      movie_data['budget'] = movie_data['budget'].astype(int)
      movie_data['revenue'] = movie_data['revenue'].astype(int)
```

1.3.5 Handling the release_date feature

- As the release_date feature was not in the correct date time format so we would convert this into the correct format using pandas

```
[21]: movie_data['release_date'] = pd.to_datetime(movie_data['release_date'])
```

1.3.6 Handling values in string features

- features like cast, genres, director and production companies are in string format separated by pipes and there are also null values so we need to handle these

[22]:
```
movie_data.isnull().sum()
```

[22]:
```
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
director                  44
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

[23]:
```
# Empty string values are read as nan in Pandas, so we will replace drop these␣
↪values as it would not be useful to impute them using mode as the cast or␣
↪directors would tend to change
movie_data = movie_data.dropna(axis =0)
```

[24]:
```
movie_data.isnull().sum()
```

[24]:
```
popularity              0
budget                  0
revenue                 0
original_title          0
cast                    0
director                0
runtime                 0
genres                  0
production_companies    0
release_date            0
vote_count              0
vote_average            0
release_year            0
budget_adj              0
revenue_adj             0
dtype: int64
```

- splitting the string values with the split function on pipe operator and converting it into array

[25]:
```
movie_data['cast'] = movie_data['cast'].str.split('|')
```

```
movie_data['production_companies'] = movie_data['production_companies'].str.
    ↪split('|')
movie_data['director'] = movie_data['director'].str.split('|')
movie_data['genres'] = movie_data['genres'].str.split('|')
```

[26]:
```
movie_data.head(1)
```

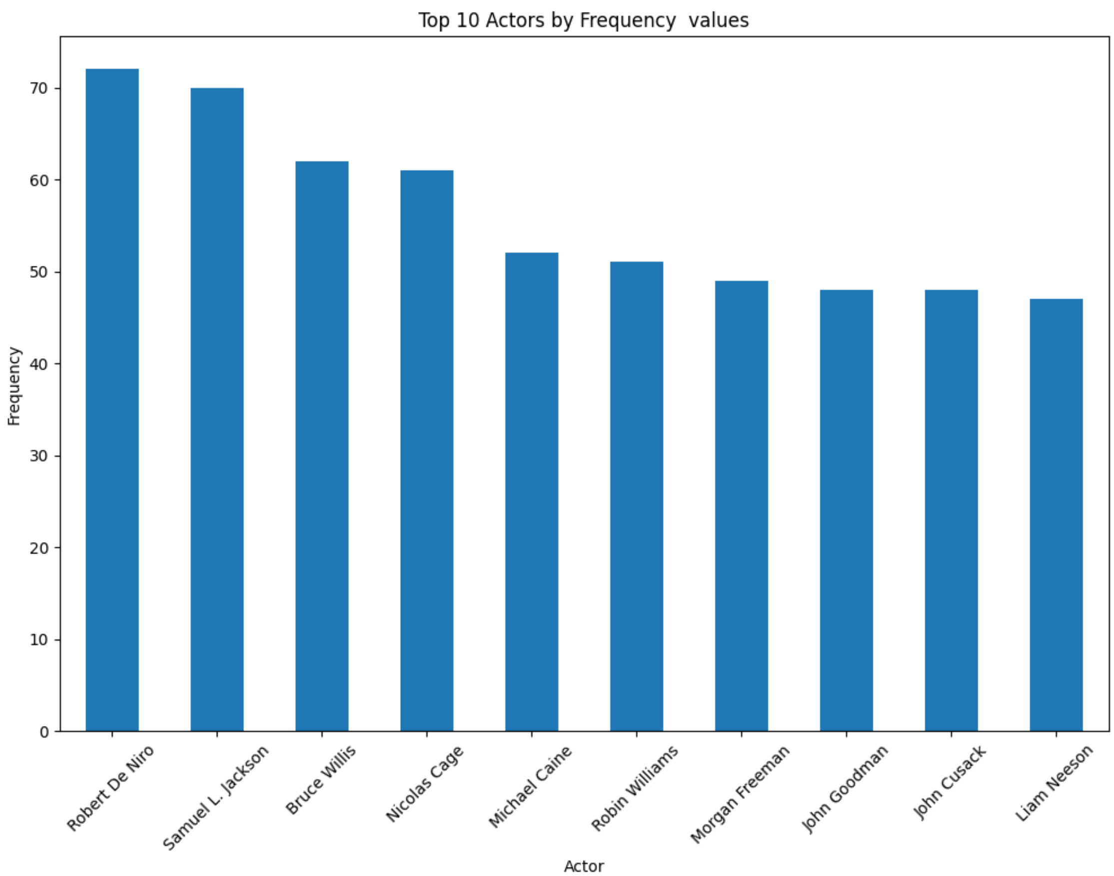
[26]:
```
   popularity     budget     revenue  original_title  \
0   32.985763  150000000  1513528810  Jurassic World

                                                cast           director  \
0  [Chris Pratt, Bryce Dallas Howard, Irrfan Khan…  [Colin Trevorrow]

   runtime                                         genres  \
0      124  [Action, Adventure, Science Fiction, Thriller]

                                production_companies release_date  vote_count  \
0  [Universal Studios, Amblin Entertainment, Lege…   2015-06-09        5562

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
```

1.4 Exploratory Data Analysis

1.4.1 1. Are there any specific Actors who have made the highest number of appearances in films and would help in determing success or failure of movies?

- From the basic view of the dataset,It is observed particularly that there is a list of actors for each movie, an intriguing question emerges regarding the frequency of actor appearances. To rephrase this inquiry, we can ask which actors have made the most appearances in the movies contained in the dataset. So this can be observed from the below code

[27]:
```
# Calculate the frequency of each actor
actors_frequency = movie_data["cast"].apply(pd.Series).stack().value_counts()

# Select the top 10 actors
top_10_actors = actors_frequency.head(10)

# Plot the figure
plt.figure(figsize=(12, 8))
top_10_actors.plot(kind="bar")
plt.title("Top 10 Actors by Frequency  values")
plt.xlabel("Actor")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.show()
```



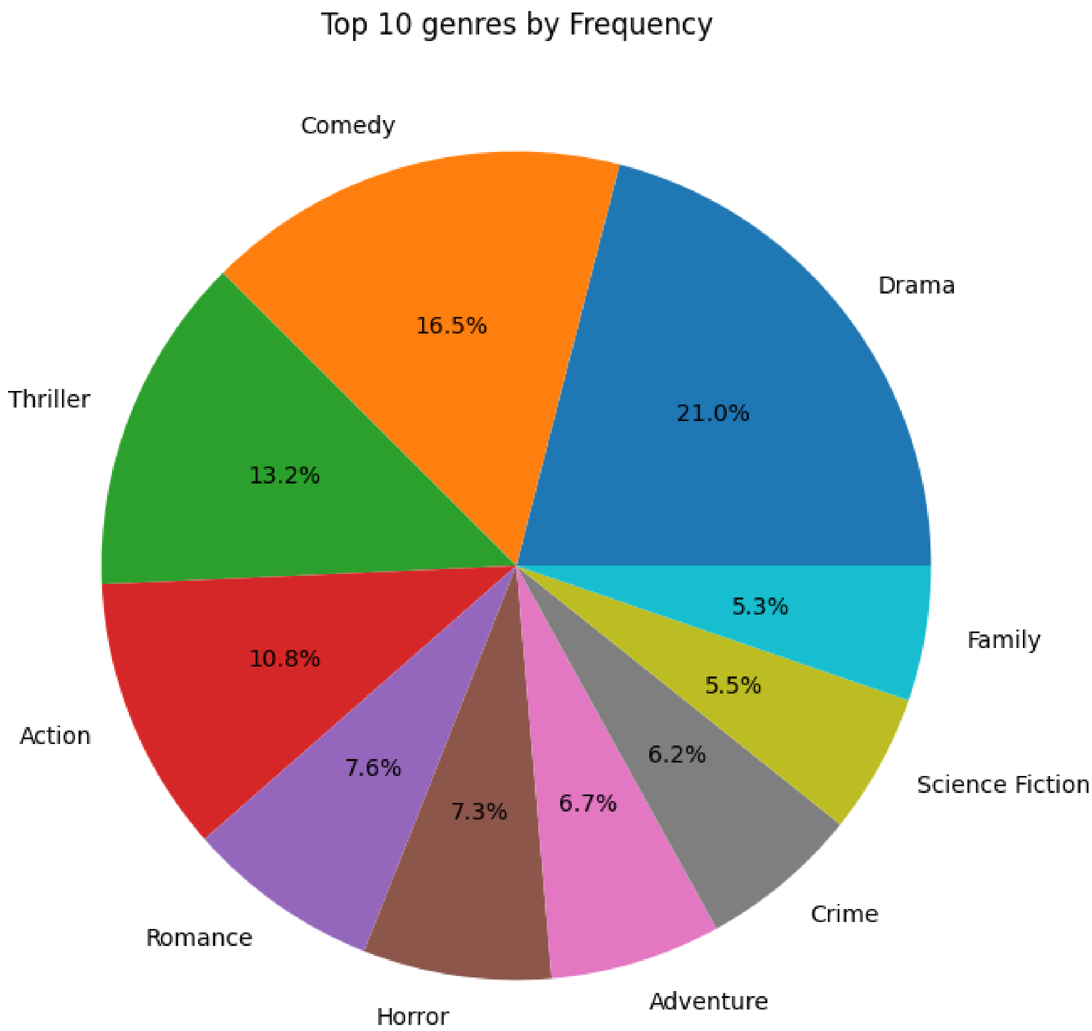
- It can be observed that the top 5 actors are Robert De Niro, Samuel L.Jackson. Bruce Wills, Nicolas Cage nd Michael Caine

1.4.2 2. Are there any Genres which are the most famous and would help directors or producers to look for?

- We will also observe the top 10 genres which are the most famous, As genres play a important role when it comes to movies being watched

[28]:
```
# Calculate the frequency of each actor
genres_frequency = movie_data["genres"].apply(pd.Series).stack().value_counts()

# Select the top 10 genres
top_10_genres = genres_frequency.head(10)

# Plot the figure
# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(top_10_genres, labels=top_10_genres.index, autopct='%1.1f%%')
plt.title("Top 10 genres by Frequency")
```

```
plt.show()
```



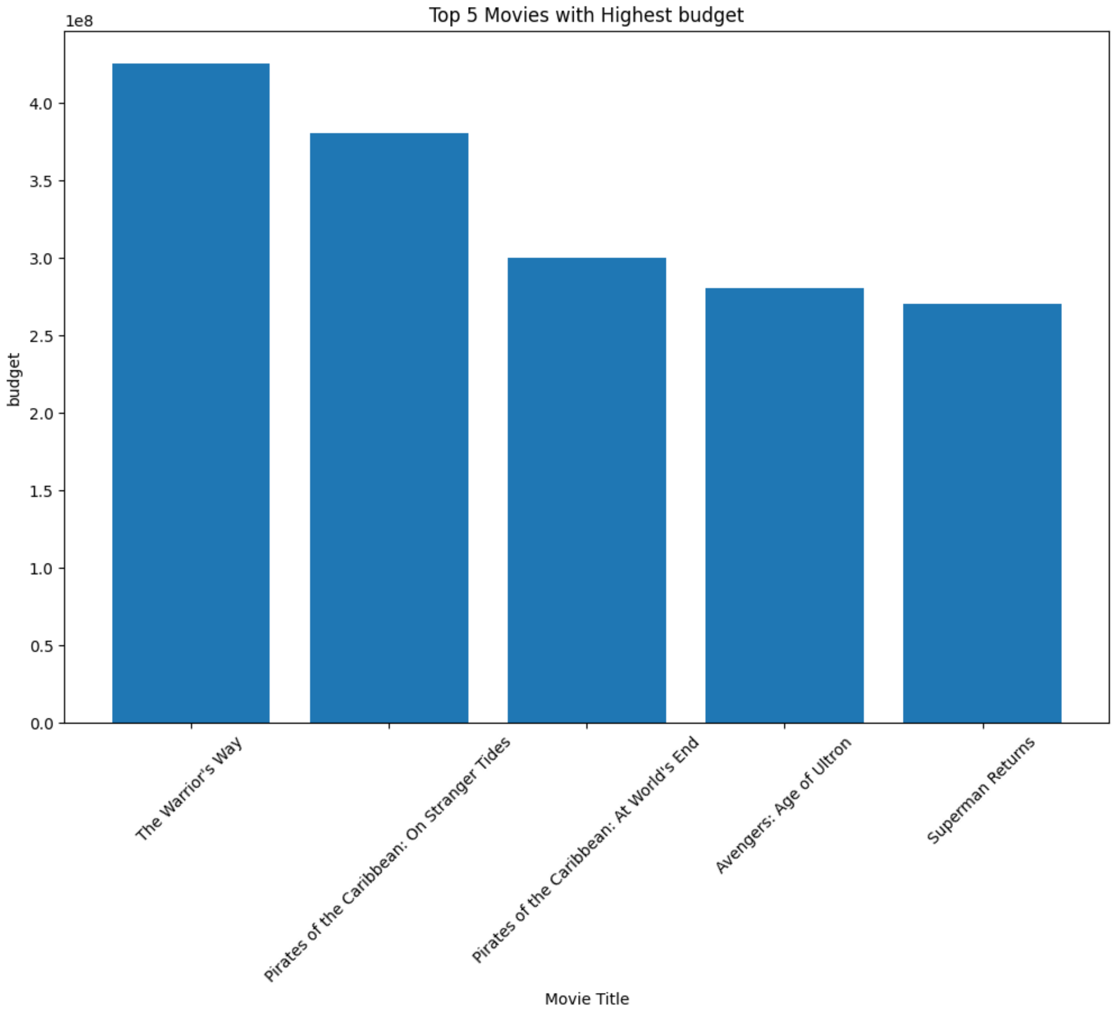
- It can be observed that drama, comedy and thriller play the most important role when it comes for genres

1.4.3 3. Which are the movies with highest, lowest budgets and what is the average budget of the movies?

3.1. Which are the movies with highest Budget?

[29]:
```
# Top 5 movies with the highest profits
top_movies = movie_data.nlargest(5, "budget")

# Plotting the top 5 movies with highest budget
```

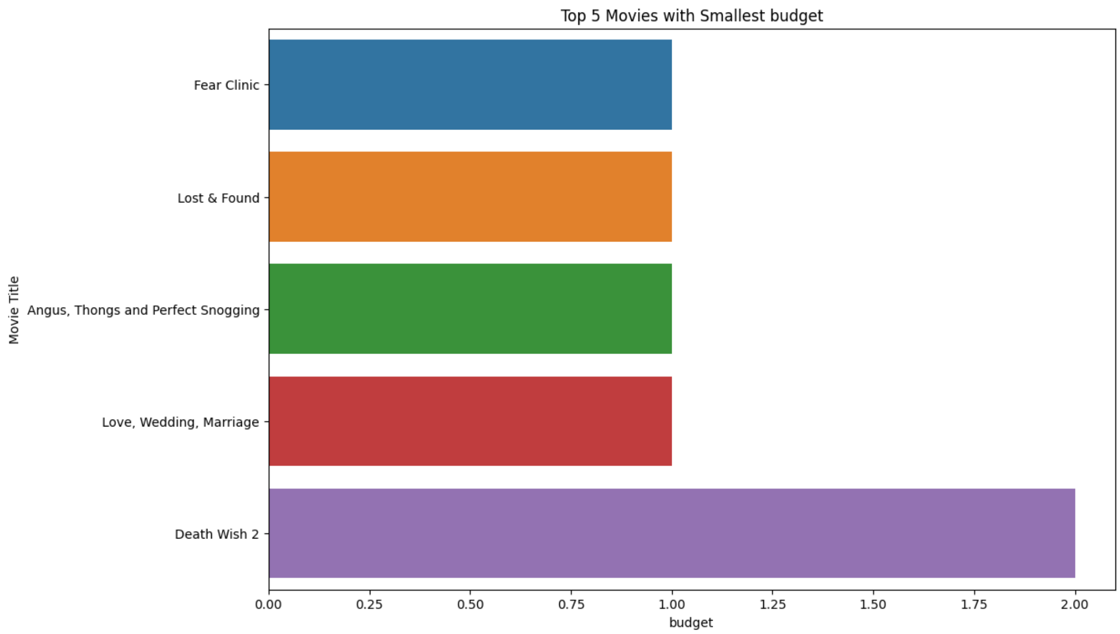
```python
plt.figure(figsize=(12, 8))
plt.bar(top_movies["original_title"], top_movies["budget"])
plt.title("Top 5 Movies with Highest budget")
plt.xlabel("Movie Title")
plt.ylabel("budget")
plt.xticks(rotation=45)
plt.show()
```



3.2. Calculating movies with lowest Budget

[30]:
```python
# Top 5 movies with the lowest budget
# Sort the movie_data by budget in ascending order
sorted_movies = movie_data.sort_values("budget", ascending=True).head(5)

# Plotting the top 5 movies with smallest profits
plt.figure(figsize=(12, 8))
```

```python
sns.barplot(data=sorted_movies, x="budget", y="original_title")
plt.title("Top 5 Movies with Smallest budget")
plt.xlabel("budget")
plt.ylabel("Movie Title")
plt.show()
```



3.3 Checking the average budget of the movies

[31]:
```python
movie_data['budget'].mean()
```

[31]: 24444813.234854687

- Average budget is about 24 millions

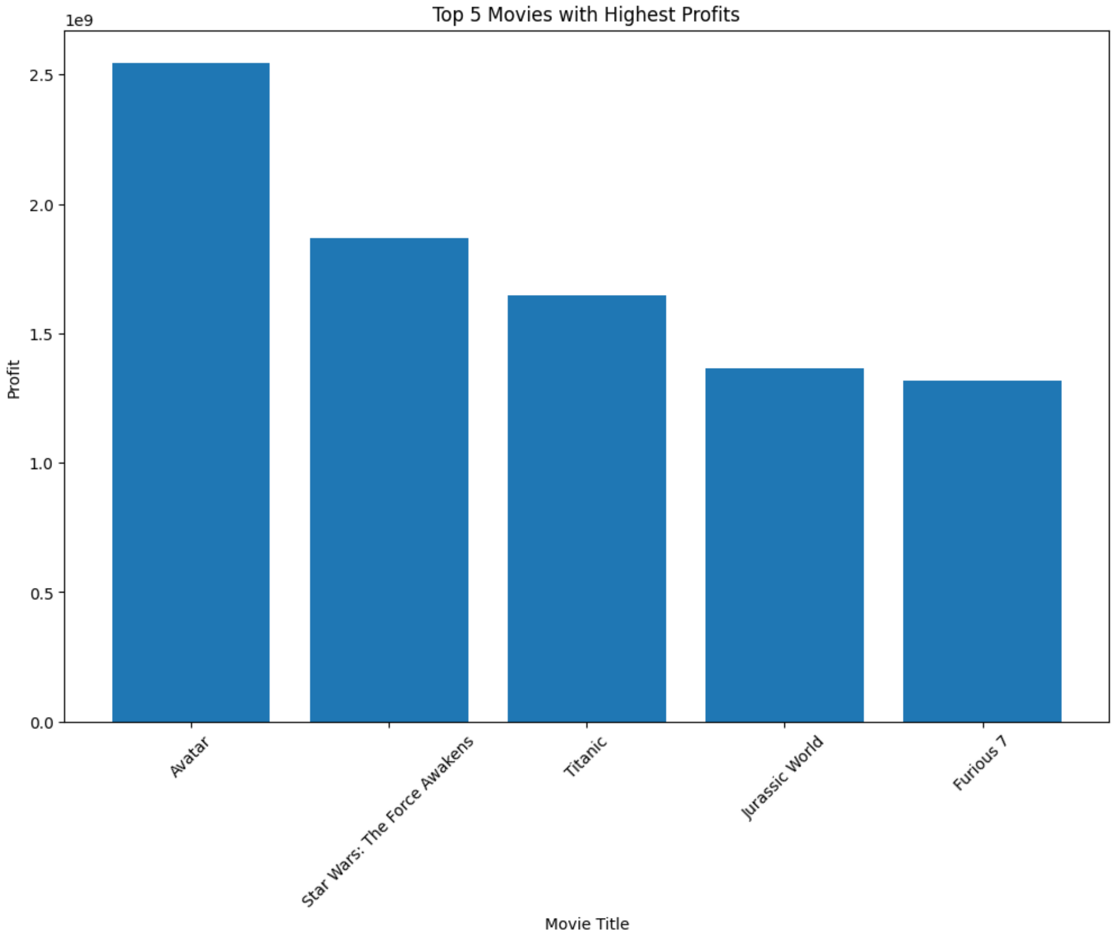
1.4.4 4. which are the movies with highest profit?

- for calculating profit, we can create a new feature profit_from_movies by using the budget and revenue features from the data

[32]:
```python
movie_data['profit_from_movies'] = movie_data['revenue'] - movie_data['budget']
```

[33]:
```python
# Top 5 movies with the highest profits
top_movies = movie_data.nlargest(5, "profit_from_movies")

# Plotting the top 5 movies with highest profits
plt.figure(figsize=(12, 8))
plt.bar(top_movies["original_title"], top_movies["profit_from_movies"])
```

```
plt.title("Top 5 Movies with Highest Profits")
plt.xlabel("Movie Title")
plt.ylabel("Profit")
plt.xticks(rotation=45)
plt.show()
```



- It can be observed that Avatar, Star wars: The force awakens. Titanic, Jurassic world and Furious 7 are the movies with max profits

1.4.5 5. which movies have the highest runtime and do they impact the sucess?

[34]:
```
# writing a function to find details of movies with highest values
def find_highest(data,column_name):
  highest_rutime= data[column_name].idxmax()
  return pd.DataFrame(data.loc[highest_rutime])
```

[35]:
```
# Code to plot the distribution
def plot_distribution(data, feature):
```

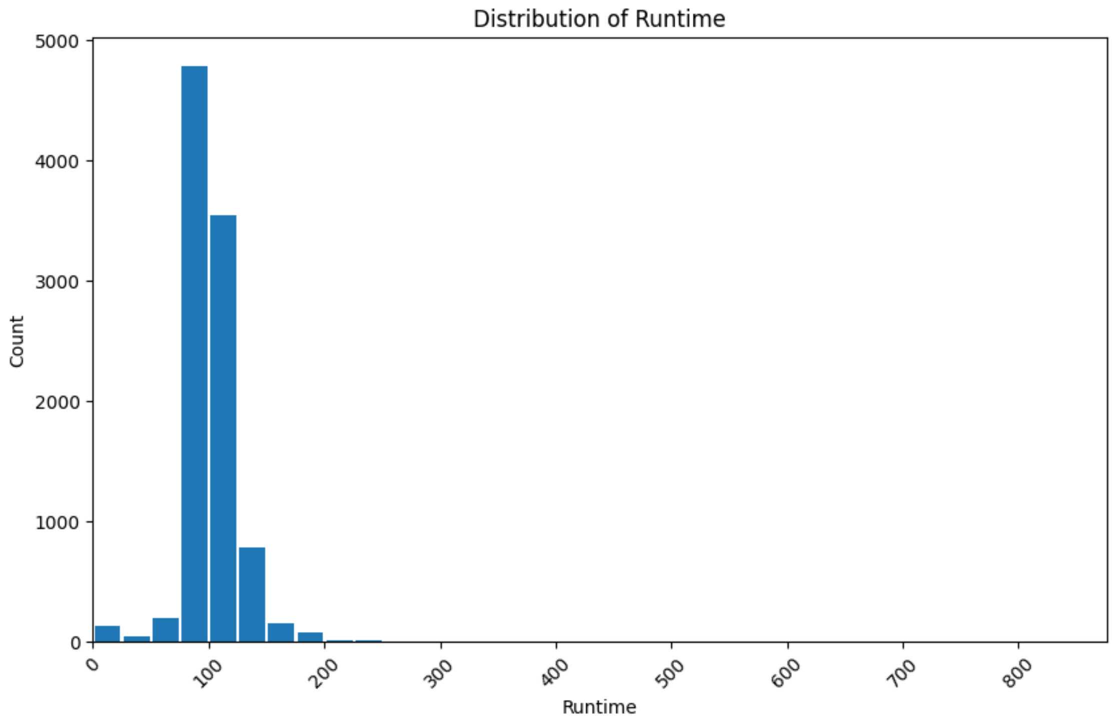
```
    plt.figure(figsize=(10, 6), dpi=100)
    plt.hist(data[feature], rwidth=0.9, bins=35)
    plt.title(f'Distribution of {feature.capitalize()}')
    plt.xlabel(feature.capitalize())
    plt.ylabel('Count')
    plt.xlim(data[feature].min(), data[feature].max())  # Adjust the x-axis␣
↪limits
    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
    plt.show()
```

[36]:
```
find_highest(movie_data,'runtime')
```

[36]:
```
                                                                  4041
popularity                                                    0.469332
budget                                                        17000000
revenue                                                       31853080
original_title                                                   Taken
cast                      [Dakota Fanning, Matt Frewer, Eric Close, Emil…
director                  [Breck Eisner, FÃ©lix EnrÃquez AlcalÃ¡, John …
runtime                                                            877
genres                                               [Science Fiction]
production_companies                                      [DreamWorks]
release_date                                       2002-12-02 00:00:00
vote_count                                                          38
vote_average                                                       6.8
release_year                                                      2002
budget_adj                                                         0.0
revenue_adj                                                        0.0
profit_from_movies                                            14853080
```

- We can observe that the movie `Taken` has the highest runtime with 877 minutes and it is a complete science fiction movie produced by DreamWorks which is released in 2002 with a profit of 14853080 dollars

[37]:
```
plot_distribution(movie_data,'runtime')
```



- The distribution shows that the runtime is right skewed and there are some outlier movies which have runtime over 800 mins

[38]:
```
movie_data['runtime'].mean()
```

[38]:
```
102.92662709783053
```

- We can also observe that the average runtime of the movies is approximately 102 minutes

1.4.6 6. Which movie has generated the highest revenue and what were its attributes?

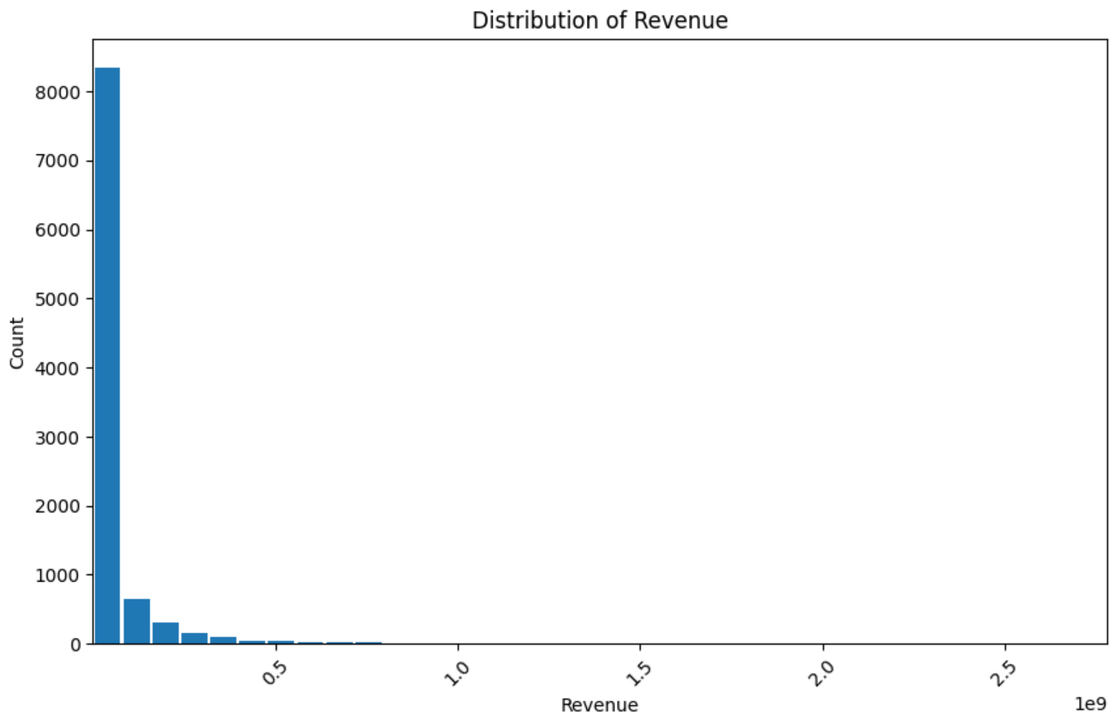
[39]:
```
find_highest(movie_data,'revenue')
```

[39]:
```
                                                                     1386
popularity                                                       9.432768
budget                                                          237000000
revenue                                                        2781505847
original_title                                                     Avatar
cast                      [Sam Worthington, Zoe Saldana, Sigourney Weave…
director                                                  [James Cameron]
runtime                                                               162
genres                        [Action, Adventure, Fantasy, Science Fiction]
production_companies      [Ingenious Film Partners, Twentieth Century Fo…
release_date                                          2009-12-10 00:00:00
```

```
vote_count                                                8458
vote_average                                               7.1
release_year                                              2009
budget_adj                                    240886902.887613
revenue_adj                                   2827123750.41189
profit_from_movies                                  2544505847
```

- It can be observed that the movie with highest revenue is Avatar which is directed by James Cameron and has a net profit of 2544505847 with a revenue of 2781505847 dollars

[40]:
```
plot_distribution(movie_data,'revenue')
```

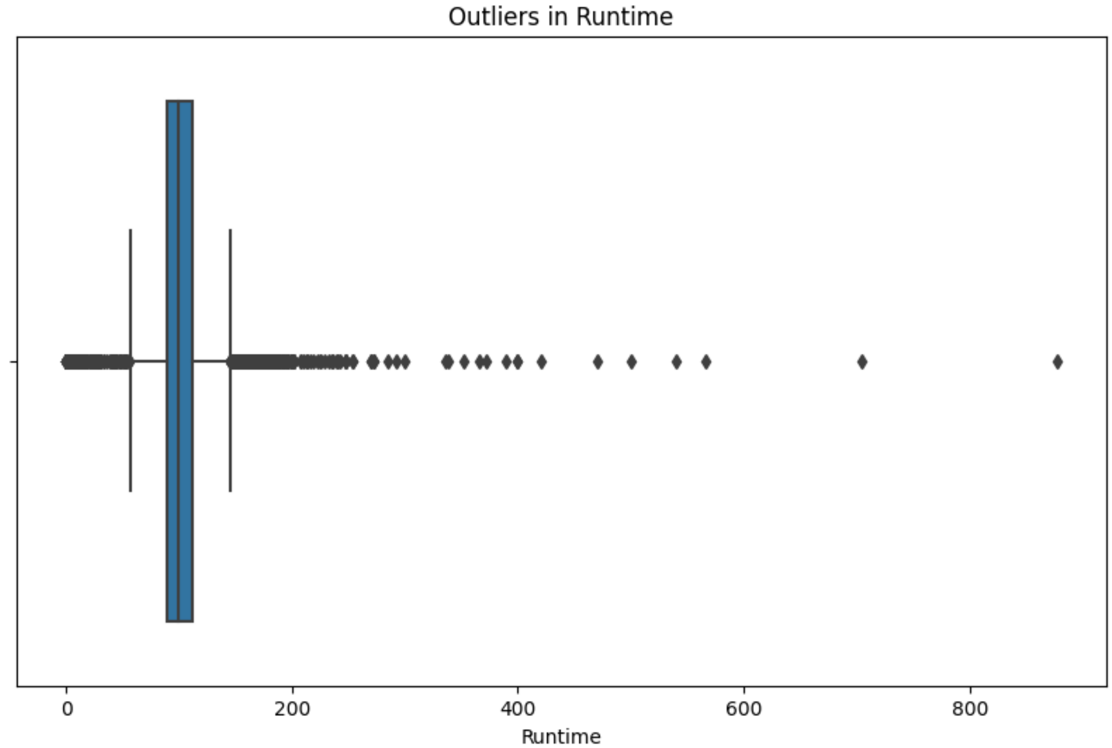


- This revenue feature even seems to be right skewed so it can also be handled using data transformation to get it into gaussian distribution if this is used as input to a ML model

1.4.7 7. Are there any outliers in the data?

[41]:
```
def plot_outliers(data, feature):
    plt.figure(figsize=(10, 6), dpi=100)
    sns.boxplot(x=data[feature])
    plt.title(f'Outliers in {feature.capitalize()}')
    plt.xlabel(feature.capitalize())
    plt.show()
```

[42]:
```
plot_outliers(movie_data,'runtime')
```



- It can be observed that there are some movies with very high runtime with >500 mins so there can be several reasons for it as it may be a movie with 1 hr or 60 min runtime but mistakenly noted as 600 or there can be a large movie in reality having runtime above 500 mins

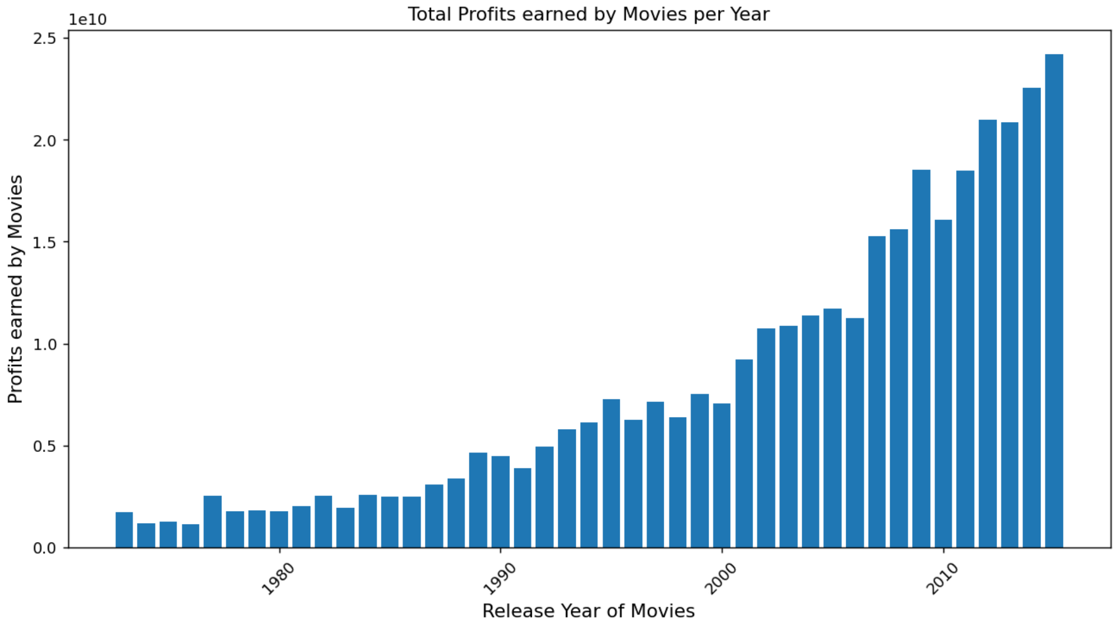
[43]:
```
# Describing runtime
movie_data['runtime'].describe()
```

[43]:
```
count    9772.000000
mean      102.926627
std        27.877432
min         0.000000
25%        90.000000
50%       100.000000
75%       112.000000
max       877.000000
Name: runtime, dtype: float64
```

Noted observations: - The dataset contains 9,772 movies with available runtime information. - The average runtime of the movies is approximately 102.93 minutes, indicating that the typical movie in the dataset is around this duration. - The standard deviation of the runtime is 27.88, which suggests a relatively wide distribution of movie lengths. - The minimum runtime is recorded as 0 minutes, which seems unusual and may indicate missing or erroneous data in some cases. - The 25th percentile indicates that 25% of the movies have a runtime of 90 minutes or less, while the 75th percentile suggests that 75% of the movies have a runtime of 112 minutes or less. - The maximum runtime is 877 minutes, indicating the longest movie in the dataset. This could be an outlier compared to the majority of movies.

1.4.8 8. Is there any dependency for the profits earned and the release time of the movies?

[44]:
```
# Extract year from release_date
movie_data['release_year'] = pd.to_datetime(movie_data['release_date']).dt.year

movie_data['release_year'] =␣
↪movie_data['release_year'][movie_data['release_year']< 2022]
# Calculate total profits earned by movies for each year
profits_year = movie_data.groupby('release_year')['profit_from_movies'].sum()

# Create a bar plot to visualize profits earned by movies over the years
plt.figure(figsize=(12, 6), dpi=130)
plt.bar(profits_year.index, profits_year.values)

# Set x-axis and y-axis labels
plt.xlabel('Release Year of Movies', fontsize=12)
plt.ylabel('Profits earned by Movies', fontsize=12)

# Set title for the plot
plt.title('Total Profits earned by Movies per Year')

# Rotate x-axis tick labels for better readability
plt.xticks(rotation=45)

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



- It can be observed that the profits are increasing year on year which is a good sign for producers to invest more into movies

1.4.9 9. Which is the best movie depending on the popularity?

[45]:
```
find_highest(movie_data,'popularity')
```

[45]:
```
                                                                     0
popularity                                                   32.985763
budget                                                       150000000
revenue                                                     1513528810
original_title                                          Jurassic World
cast                      [Chris Pratt, Bryce Dallas Howard, Irrfan Khan…
director                                             [Colin Trevorrow]
runtime                                                            124
genres                       [Action, Adventure, Science Fiction, Thriller]
production_companies      [Universal Studios, Amblin Entertainment, Lege…
release_date                                       2015-06-09 00:00:00
vote_count                                                        5562
vote_average                                                       6.5
release_year                                                    2015.0
budget_adj                                           137999939.280026
revenue_adj                                         1392445892.5238
profit_from_movies                                          1363528810
```

- It is observed that the most popular movie depending on popularity is jurassic world directed by Colin Trevorrow with a runtime of 124 mins and which was released in 2015

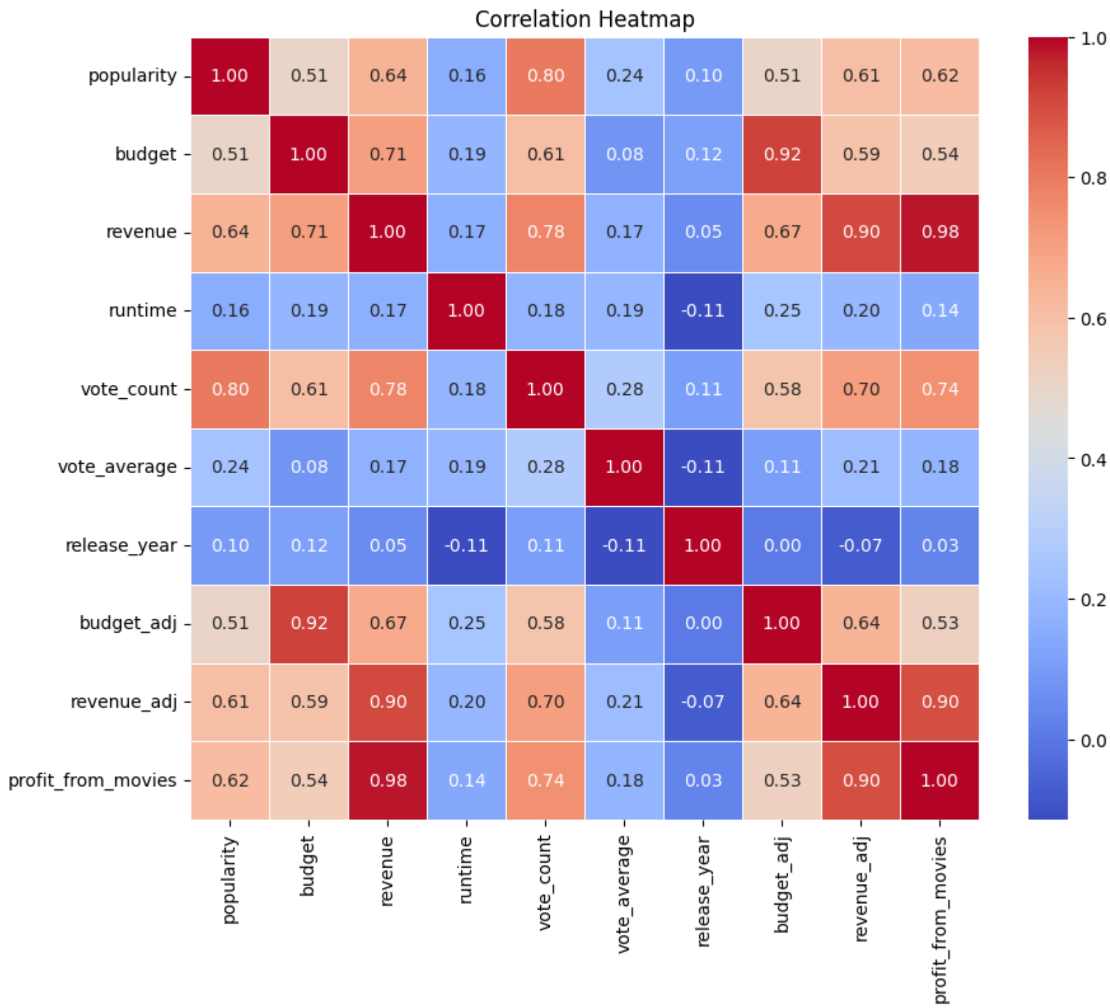
1.4.10 10. Are any any more correlatins between different features?

[46]:
```
movie_data.corr()
```

[46]:
```
                   popularity    budget   revenue   runtime  vote_count  \
popularity           1.000000  0.505283  0.643797  0.156290    0.802956
budget               0.505283  1.000000  0.706611  0.188208    0.605799
revenue              0.643797  0.706611  1.000000  0.166382    0.777009
runtime              0.156290  0.188208  0.166382  1.000000    0.184285
vote_count           0.802956  0.605799  0.777009  0.184285    1.000000
vote_average         0.239121  0.080180  0.173151  0.193973    0.279851
release_year         0.096772  0.115047  0.054683 -0.114369    0.111411
budget_adj           0.505378  0.919671  0.671505  0.249134    0.580444
revenue_adj          0.606458  0.588518  0.904420  0.198307    0.704044
profit_from_movies   0.615139  0.544920  0.978363  0.142127    0.743601

                   vote_average  release_year  budget_adj  revenue_adj  \
popularity             0.239121      0.096772    0.505378     0.606458
budget                 0.080180      0.115047    0.919671     0.588518
revenue                0.173151      0.054683    0.671505     0.904420
runtime                0.193973     -0.114369    0.249134     0.198307
vote_count             0.279851      0.111411    0.580444     0.704044
vote_average           1.000000     -0.112510    0.109789     0.214668
release_year          -0.112510      1.000000    0.004750    -0.070937
budget_adj             0.109789      0.004750    1.000000     0.640852
revenue_adj            0.214668     -0.070937    0.640852     1.000000
profit_from_movies     0.181734      0.031268    0.526809     0.899632

                   profit_from_movies
popularity                   0.615139
budget                       0.544920
revenue                      0.978363
runtime                      0.142127
vote_count                   0.743601
vote_average                 0.181734
release_year                 0.031268
budget_adj                   0.526809
revenue_adj                  0.899632
profit_from_movies           1.000000
```

[47]:
```
# Compute the correlation matrix
correlation_matrix = movie_data.corr()
plt.figure(figsize=(10, 8))
# Plot the correlation heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)
plt.title('Correlation Heatmap')
```

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



Observations noted: Observations for the correlation matrix:

1. Popularity has a strong positive correlation with budget, revenue, vote_count, and profit_from_movies. This indicates that popular movies tend to have higher budgets, generate more revenue, receive more votes, and have higher profits.
2. Budget shows a strong positive correlation with revenue, vote_count, and profit_from_movies. This suggests that movies with higher budgets tend to generate more revenue, receive more votes, and have higher profits.
3. Revenue has a strong positive correlation with budget, vote_count, and profit_from_movies. This implies that movies with higher revenues often have higher budgets, receive more votes, and have higher profits.
4. Vote_count has a strong positive correlation with popularity, budget, revenue, and profit_from_movies. This suggests that movies with more votes are likely to be popular, have higher budgets, generate more revenue, and have higher profits.

5. Runtime has a weak positive correlation with popularity, budget, revenue, and profit_from_movies. This indicates that there is a slight tendency for longer movies to be more popular, have higher budgets, generate more revenue, and have higher profits.
6. Vote_average has a weak positive correlation with popularity, budget, and revenue. This suggests that movies with higher average votes are slightly more popular, have higher budgets, and generate more revenue.
7. Release_year has a weak positive correlation with popularity and budget_adj. This indicates a slight upward trend in popularity and budget over the years.
8. Budget_adj has a strong positive correlation with budget, revenue, and profit_from_movies. This suggests that adjusted budget values follow similar patterns as the original budget in terms of revenue and profit.
9. Revenue_adj has a strong positive correlation with revenue and profit_from_movies. This implies that adjusted revenue values closely align with the original revenue and profit figures.

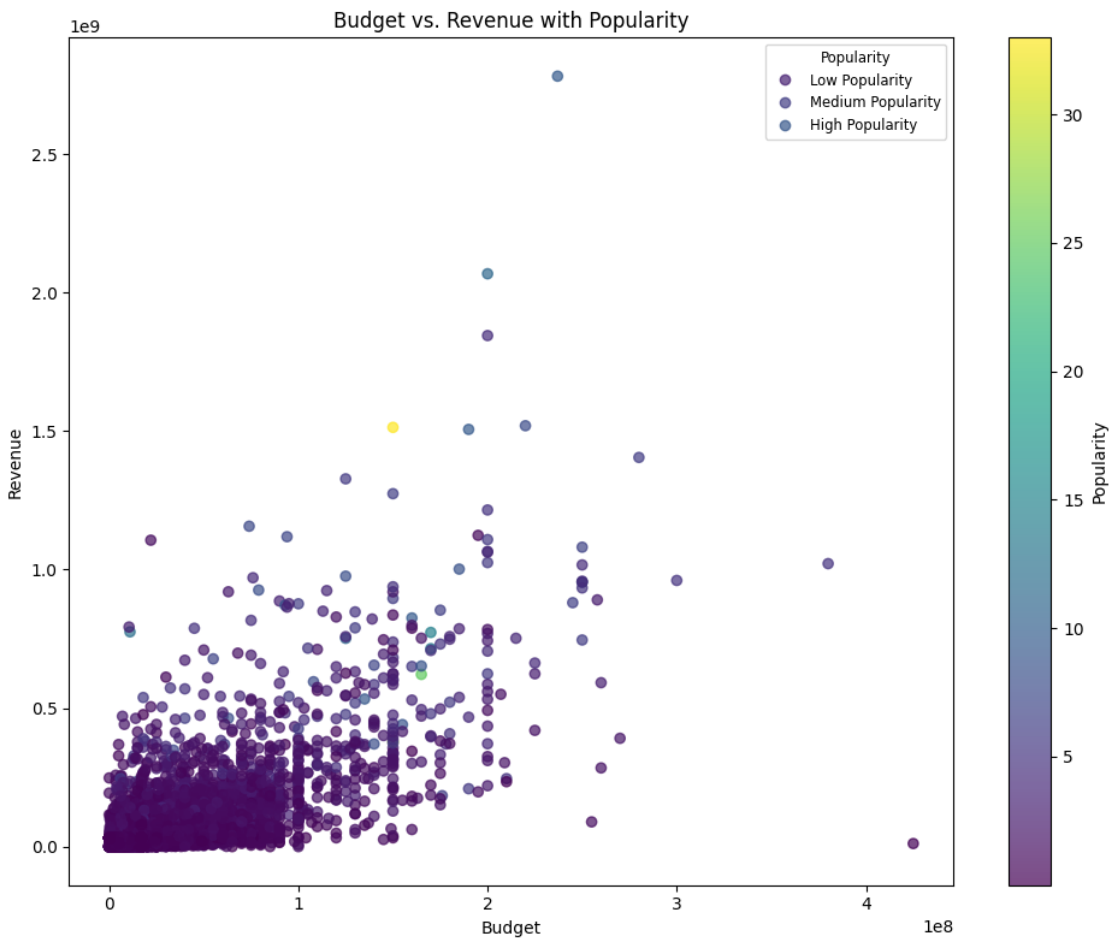
[48]:
```
movie_data.head(1)
```

[48]:
```
   popularity     budget     revenue  original_title  \
0   32.985763  150000000  1513528810  Jurassic World

                                                cast           director  \
0  [Chris Pratt, Bryce Dallas Howard, Irrfan Khan…  [Colin Trevorrow]

   runtime                                            genres  \
0      124  [Action, Adventure, Science Fiction, Thriller]

                              production_companies release_date  vote_count  \
0  [Universal Studios, Amblin Entertainment, Lege…   2015-06-09        5562

   vote_average  release_year    budget_adj   revenue_adj  profit_from_movies
0           6.5        2015.0  1.379999e+08  1.392446e+09          1363528810
```

1.4.11 11. Does the budget and revenue of movies impact their popularity?

[49]:
```
# Extracting relevant data from the dataframe
budget = movie_data['budget']
revenue = movie_data['revenue']
popularity = movie_data['popularity']

# plot size
fig, ax = plt.subplots(figsize=(10, 8))

# Creating the scatter plot
scatter = ax.scatter(budget, revenue, c=popularity, cmap='viridis', alpha=0.7)

# Setting labels and title of the plot
ax.set_xlabel('Budget')
ax.set_ylabel('Revenue')
```

```
ax.set_title('Budget vs. Revenue with Popularity')

# Add colorbar to the plot
cbar = plt.colorbar(scatter)
cbar.set_label('Popularity')

# Add bubble size legend to the plot
sizes = [50, 100, 150]
labels = ['Low Popularity', 'Medium Popularity', 'High Popularity']
legend = ax.legend(handles=scatter.legend_elements()[0], labels=labels,
                   title='Popularity', loc='upper right',
                   fontsize='small', title_fontsize='small')

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



- Yes, it is observed from the scatter plot that most popular movies require a good budget and they also generate a good revenue, there are also some empiral cases where the low budget movies have also generated good revenue thus having good popularity.

1.5 Conclusion

The project was great to be done and good observations and limitations were noted which would even help in understanding better about the movies data. - It is observed that the average runtime is around 103 minutes and average budget of movies is around 24 millions. - It is also observed that movies with higher budgets tend to generate more revenue, receive more votes, and have higher profits. - It is also observed that the profits are increasing year on year which is a good sign for producers to invest more into movies. - It is also observed that Robert De Niro has acted in most number of movies - It is also observed that high popularity has also lead in high gross movies - There are different genres but the directors and producers should try to have more movies on Drama as this seemed to be the most popular one. - It can also be further checked if any specific directors have a raise in popularity of the movie too.

1.6 Limitations

- It is also observed that there are several values in revenue and budget which have zero in them.
- There were also several missing values which were to be imputed and there are some text features where some text analytics can also be used.
- It is also observed that there are some outliers in the data, like the highest runtime of a movie was 877 minutes which is approximately 37 hrs. This is too long for a movie so these outliers can also be handled.

[50]:
```
# from subprocess import call
# call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```