TMDB Movie Dataset Analysis

Ayush Gupta
Data Analyst Nanodegree
9/25/19

Introduction

In this project, I investigate a TMDB movie database file which has collection of key details of about 10k+ movies, including their details of budget, revenue, release dates, etc.

To get some information on the dataset, I wrote a function which prints out useful information about our data like number of entries, data types of columns, number of columns, etc. So, let us look at the data.

```
import pandas as pd


def data_info(dataframe):
    """
    Provides general information about the dataframe like shape, data types of
the columns, mean, standard deviation,
    inter quartile ranges, minimum, maximum. etc.

    Args:
        dataframe: data passed for assessment

    """
    # general info about the dataframe
    info = dataframe.info()
    # description of the data
    description = dataframe.describe()
    # data type of columns
    data_type = dataframe.dtypes
    # view first five rows of the dataframe
    data_view = dataframe.head()
    # shape of the dataframe before cleaning
    rows, col = dataframe.shape

    print(info)
    print(description)
    print(data_type)
    print(data_view)
    # since 'rows' includes count of a header, we need to remove its count.
    print('We have {} total entries of movies and {} columns/features of
it.\n'.format(rows - 1, col))


movie_data = pd.read_csv('tmdb-movies.csv')
```

We call this function from another function (more on this later) and pass in the argument holding our data. Using various pandas functions, it prints the output to console and running this function gets us the following output:

```
Here is the information on the data before we clean it:


<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10866 entries, 0 to 10865

Data columns (total 21 columns):

id                      10866 non-null int64

imdb_id                 10856 non-null object
```

```
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
None
                  id    popularity  ...    budget_adj   revenue_adj
count   10866.000000  10866.000000  ...  1.086600e+04  1.086600e+04
mean    66064.177434      0.646441  ...  1.755104e+07  5.136436e+07
std     92130.136561      1.000185  ...  3.430616e+07  1.446325e+08
min         5.000000      0.000065  ...  0.000000e+00  0.000000e+00
25%     10596.250000      0.207583  ...  0.000000e+00  0.000000e+00
50%     20669.000000      0.383856  ...  0.000000e+00  0.000000e+00
75%     75610.000000      0.713817  ...  2.085325e+07  3.369710e+07
max    417859.000000     32.985763  ...  4.250000e+08  2.827124e+09

[8 rows x 10 columns]
id                       int64
imdb_id                 object
popularity             float64
budget                   int64
revenue                  int64
original_title          object
cast                    object
```

```
homepage                object
director                object
tagline                 object
keywords                object
overview                object
runtime                  int64
genres                  object
production_companies    object
release_date            object
vote_count               int64
vote_average           float64
release_year             int64
budget_adj             float64
revenue_adj            float64
dtype: object
       id    imdb_id  popularity  ...  release_year    budget_adj   revenue_adj
0  135397  tt0369610   32.985763  ...          2015  1.379999e+08  1.392446e+09
1   76341  tt1392190   28.419936  ...          2015  1.379999e+08  3.481613e+08
2  262500  tt2908446   13.112507  ...          2015  1.012000e+08  2.716190e+08
3  140607  tt2488496   11.173104  ...          2015  1.839999e+08  1.902723e+09
4  168259  tt2820852    9.335014  ...          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
We have 10865 total entries of movies and 21 columns/features of it.
```

Due to space limitation horizontally, the IDE console clips out some part of the output from the middle. We can get a better visual output by running the function in a jupyter notebook. I decided to use this output as the entire output is just too big to fit into this document.

Anyway, by looking at the output we see that:

- The columns *budget, revenue, budget_adj, revenue_adj* has not given us the currency but for this dataset we will assume that it is in dollars.
- The vote count for each movie is not similar, for example, the movie 'Mad Max: Fury Road' has 6k+ votes while Sinister 2 has only 331 votes. Since the votes of the movies vary so much the *vote_average* column also is affected by it. So, we cannot calculate or assume that movie with highest votes or rating was more successful since the voters of each film vary.

What needs Brainstorming

Looking at this database...

- The first question that came in my mind was which movie gained the most profit or we can also say that which movie has been the people's favorite?

- Since this is just the glimpse of the database, this just shows the movies in the year 2015, but there are also other movies released in different years so the Second question comes in my mind is in which year the movies made the most profit?
- I also wanted to know what are the similar characteristics of movies which have gained highest profits?

What needs Wrangling and Cleaning

Based on the questions brainstormed above, I wanted to know if we have all the valid values of the variables that I wanted to calculate and how can this data be trimmed so we can only have the columns we need. This will also make our dataset clean and easy for us to calculate what we want.

- As you can see in this database, there are lot of movies where the budget or revenue have a value of '0' which means that the values of those variables of those movies has not been recorded. Calculating the profits of these movies would lead to inappropriate results. So, we need to delete these rows.
- Also, this dataset has some duplicate rows. We must clean that too for appropriate results.
- We will also calculate the average runtime of the movies so in case if we have a runtime of a movie '0' then we need to replace it with *Nan*.
- The *release_date* column must be converted into date format.
- Checking if all columns are in the desired data type, if not then we must change it.
- Mentioning the country currency in the desired columns.
- Finally, we will also remove unnecessary columns such as *id, imdb_id, popularity, budget_adj, revenue_adj, homepage, keywords, overview, production_companies, vote_count* and *vote_average*.

Analysis Questions

1. We first need to answer some general questions about the dataset such as:
    a. Which movie earns the most and least profit?
    b. Which movie had the greatest and least runtime?
    c. Which movie had the greatest and least budget?
    d. Which movie had the greatest and least revenue?
    e. What is the average runtime of all movies?
    f. In which year we had the most movies making profits? (profits of movies in each year)
2. We then move on to answer specific questions like similar characteristics of some most profitable movies such as:
    a. Average duration of movies.
    b. Average budget.
    c. Average revenue.
    d. Average profits.
    e. Which director directed most films?

    f. Which cast has appeared the most?
    g. Which genre were more successful?
    h. Which month released highest number of movies in all the years?
    i. And which month made the most profit?
3. We also analyse some trends and relations among some traits like:
    a. How have movie production trends varied over the years?
    b. What are the top 20 highest grossing movies?
    c. What are the top 20 most expensive movies?
    d. How do budgets correlate with revenues? Do higher budget movies have higher revenue?
    e. What run times are associated with each genre?

Data Cleaning

Before answering the above questions, we need to the clean dataset. I wrote a function that do just that. Each line is the code has a comment that explains what that line of code does. At the end of this function, we call the function by passing the dataset. In the code, the function *data_info(dataframe)* is also referenced twice, once before we clean the dataset and again after we clean the dataset to see how the data looks after we remove duplicate entries, null values, 0 values, fix datatypes and unnecessary columns. The output of this function is explained earlier in the document. Each cleaning step is explained after the function. Now look at the function:

```
def data_cleaning(dataframe):
    """
    This function cleans the data for further analysis. Cleaning includes
    removing duplicates, removing null values and deleting unnecessary columns.

    Args:
        dataframe: data passed for cleaning

    Returns:
        cleaned dataframe

    """
    # getting information about the data before cleaning
    print('Here is the information on the data before we clean it: \n')
    data_info(dataframe)

    # list of columns that needs to be deleted
    del_col = ['id', 'imdb_id', 'popularity', 'budget_adj', 'revenue_adj',
'homepage', 'keywords', 'overview',
               'production_companies', 'vote_count', 'vote_average']
    # deleting the columns from the database
    dataframe = dataframe.drop(del_col, axis=1)

    # dropping duplicate rows but will keep the first one
    dataframe = dataframe.drop_duplicates()

    # list of column names that needs to be checked for 0
    check_row = ['budget', 'revenue']
    # this will replace the value of '0' to NaN of columns given in the list
    dataframe[check_row] = dataframe[check_row].replace(0, np.NaN)
    # now we will drop any row which has NaN values in any of the column of the
list (check_row)
    dataframe = dataframe.dropna(subset=check_row)

    # replacing 0 with NaN of runtime column of the dataframe
    dataframe['runtime'] = dataframe['runtime'].replace(0, np.NaN)

    # changing data type of `release_date` column from string to datetime
    dataframe['release_date'] = pd.to_datetime(dataframe['release_date'])

    # renaming `budget` and `revenue` columns to include currency (assuming US
dollars)
    dataframe.rename(columns={'budget': 'budget(US-Dollars)', 'revenue':
'revenue(US-Dollars)'}, inplace=True)

    # getting information about the data after cleaning
    print('Here is the information on the data after we clean it: \n')
    data_info(dataframe)

    return dataframe



if __name__ == '__main__':
    movie_data = pd.read_csv('tmdb-movies.csv')
    movie_data_cleaned = data_cleaning(movie_data)
```

First, we clean up the columns. We only keep the columns we need and remove the rest of them. We remove these columns from the dataset: *id, imdb_id, popularity, budget_adj,*

revenue_adj, homepage, keywords, overview, production_companies, vote_count, and *vote_average*. This makes the dataset a lot cleaner and is soothing to the eyes.

After this we clean up the duplicate rows using the *drop_duplicates()* function. We use the argument *inplace=True* to make change persist as we move on in the program.

We then move on to find movies which have budget or revenue value set '0' and delete those rows from the dataset. These rows are basically useless to us as we cannot use the budget or revenue values in our analysis.

The next step includes finding movies with runtime and replacing the values from '0' to *Nan*. This is done so that the rows with '0' in their runtime do not interfere and create skews in our analysis.

Now for proper data analysis, we need to have make sure that all the columns are of proper datatypes. In our dataset, the columns *release_date* is initially set to string. We change the datatype from string to datetime using pandas function *to_datetime()*.

Finally, we change the column names to make them more descriptive. Since, the dataset does not have any currency for the budget and revenue of the movies, we will assume the currency as US-Dollars and change the column names to include the currency as well in the columns name using *rename()* function. The column *budget* is changed to *budget(US-Dollars)* and the column *revenue* is changed to *revenue(US-Dollars)*.

At this point, we have a much cleaner dataset to work with and here how it looks after the clean-up. We again use the function *data_info(dataframe)* to get the output in the console.

```
Here is the information on the data after we clean it:


<class 'pandas.core.frame.DataFrame'>
Int64Index: 3854 entries, 0 to 10848
Data columns (total 10 columns):
budget(US-Dollars)     3854 non-null float64
revenue(US-Dollars)    3854 non-null float64
original_title         3854 non-null object
cast                   3850 non-null object
director               3853 non-null object
tagline                3574 non-null object
runtime                3854 non-null int64
genres                 3854 non-null object
release_date           3854 non-null datetime64[ns]
release_year           3854 non-null int64
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 331.2+ KB
None
       budget(US-Dollars)  revenue(US-Dollars)      runtime  release_year
```

```
count        3.854000e+03         3.854000e+03  3854.000000   3854.000000
mean         3.720370e+07         1.076866e+08   109.220291   2001.261028
std          4.220822e+07         1.765393e+08    19.922820     11.282575
min          1.000000e+00         2.000000e+00    15.000000   1960.000000
25%          1.000000e+07         1.360003e+07    95.000000   1995.000000
50%          2.400000e+07         4.480000e+07   106.000000   2004.000000
75%          5.000000e+07         1.242125e+08   119.000000   2010.000000
max          4.250000e+08         2.781506e+09   338.000000   2015.000000
budget(US-Dollars)            float64
revenue(US-Dollars)           float64
original_title                 object
cast                           object
director                       object
tagline                        object
runtime                         int64
genres                         object
release_date           datetime64[ns]
release_year                    int64
dtype: object
   budget(US-Dollars)  revenue(US-Dollars)  ... release_date release_year
0         150000000.0         1.513529e+09  ...   2015-06-09         2015
1         150000000.0         3.784364e+08  ...   2015-05-13         2015
2         110000000.0         2.952382e+08  ...   2015-03-18         2015
3         200000000.0         2.068178e+09  ...   2015-12-15         2015
4         190000000.0         1.506249e+09  ...   2015-04-01         2015

[5 rows x 10 columns]
We have 3853 total entries of movies and 10 columns/features of it.
```

Here we can clearly see that the number of rows reduced from 10865 to 3853! More than half of data in our dataset had either null values, duplicates or of incorrect type. This really shows how much cleaning can reduce the faulty data volume and make the analysis easier. Now that we have our clean data, lets analyse it.

Exploratory Data Analysis

Before answering any of the questions, we will first calculate profit made by each movie. Let us assign a new column which will hold the profit values of each movie. To calculate profit of each movie, we need to subtract the budget from the revenue of each movie.

We use *insert()* function to insert the new column a position specified in the first argument, the name of the column as the second argument and the calculation it takes to output to the column as the third argument. We also set the datatype of the column for consistency.

```
# assigning a new column which will hold the profit values of each movie
dataframe.insert(2, 'profit(US-Dollars)', dataframe['revenue(US-Dollars)'] -
dataframe['budget(US-Dollars)'])
# changing the data type of the column to float for consistency
dataframe['profit(US-Dollars)'] = dataframe['profit(US-
Dollars)'].apply(np.float64)
```

Now, let us move on to answer the questions:

1. General Statistics:
    a. Which movie earns the most and least profit?
    Before we answer this question, we will define a function which calculates highest and lowest values of columns.

```
def get_highest_lowest(dataframe, column_name):
    """
    This function calculates highest and lowest values of columns specified in
the argument

    Args:
        dataframe: cleaned dataset
        column_name: column name for which highest and values are to be
calculated

    Returns:
        concatenated data frame containing highest and lowest calculated values

    """
    # taking the index value of the highest number in profit column
    highest_id = dataframe[column_name].idxmax()
    # calling by index number, storing that row info in a variable
    highest_value = pd.DataFrame(dataframe.loc[highest_id])

    # taking the index value of the lowest number in profit column
    lowest_id = dataframe[column_name].idxmin()
    # calling by index number, storing that row info in a variable
    lowest_value = pd.DataFrame(dataframe.loc[lowest_id])

    # concatenating two values in a single data frame
    combined_values = pd.concat([highest_value, lowest_value], axis=1)

    return combined_values


# movie with most and least earned profit
print('Movies that earned most and least profit: ')
print(get_highest_lowest(dataframe, 'profit(US-Dollars)'))
```

Passing the newly created column to our newly created function gets us the following result.

```
Movies that earned most and least profit:
                                                      1386                                                  2244
budget(US-Dollars)                                2.37e+08                                              4.25e+08
revenue(US-Dollars)                               2.78151e+09                                           1.10876e+07
profit(US-Dollars)                                2.54451e+09                                           -4.13912e+08
original_title                                         Avatar                                  The Warrior's Way
cast                  Sam Worthington|Zoe Saldana|Sigourney Weaver|S ...  Kate Bosworth|Jang Dong-gun|Geoffrey Rush|Dann ...
director                                         James Cameron                                         Sngmoo Lee
tagline                                 Enter the World of Pandora.                           Assassin. Hero. Legend.
runtime                                                   162                                                   100
genres                Action|Adventure|Fantasy|Science Fiction                Adventure|Fantasy|Action|Western|Thriller
release_date                              2009-12-10 00:00:00                                   2010-12-02 00:00:00
release_year                                             2009                                                  2010
```

The column names for the dataframe above are the index number. The first column shows the highest profit made by a movie and second column shows the highest in loss movie in this dataset.

As we can see the Directed by James Cameron, Avatar film has the highest profit in all, making over $2.5B in profit in this dataset. May be the highest till now in the entire human history but we cannot say for sure as this dataset doesn't have all the films released till date.

And the movie most in loss in this dataset is The Warriors Way. Going in loss by more than $400M was directed by Singmoo Lee.

b. Which movie had the greatest and least runtime?

Again we use our function *get_highest_lowest()*.

```
# movies with longest and shortest runtime
print('\nMovies which have longest and shortest runtime: ')
print(get_highest_lowest(dataframe, 'runtime'))
```

```
Movies which have longest and shortest runtime:
                                                      2107                                                  5162
budget(US-Dollars)                                 1.8e+07                                                    10
revenue(US-Dollars)                                  871279                                                     5
profit(US-Dollars)                                -1.71287e+07                                                -5
original_title                                         Carlos                                     Kid's Story
cast                  Edgar RamÃrez|Alexander Scheer|Fadi Abi Samra ...  Clayton Watson|Keanu Reeves|Carrie-Anne Moss|K ...
director                                         Olivier Assayas                                Shinichiro Watanabe
tagline                                The man who hijacked the world                                        NaN
runtime                                                   338                                                    15
genres                               Crime|Drama|Thriller|History                          Science Fiction|Animation
release_date                              2010-05-19 00:00:00                                   2003-06-02 00:00:00
release_year                                             2010                                                  2003
```

So again, the first column shows the runtime of the highest and second the lowest with column names as the index number.

I have never heard a runtime of a movie so long! Runtime of 338 min, that is approximately 3.5 hrs! So, Carlos movie has the highest runtime.

The name of the movie with shortest runtime is Kid's Story, runtime of just 15 min!

As you see both movies have one thing in common, negative profits! The Kid's Story having a budget of 10 dollars and revenue 5, just does not seem right to me. But this is what our dataset shows.

c. Which movie had the greatest and least budget?

```
# movies with largest and smallest budget
print('\nMovies that had largest and smallest budget: ')
print(get_highest_lowest(dataframe, 'budget(US-Dollars)'))
```

```
Movies that had largest and smallest budget:
                                                    2244                                                2618
budget(US-Dollars)                                  4.25e+08                                               1
revenue(US-Dollars)                                 1.10876e+07                                            100
profit(US-Dollars)                                  -4.13912e+08                                           99
original_title                                      The Warrior's Way                                      Lost & Found
cast                Kate Bosworth|Jang Dong-gun|Geoffrey Rush|Dann ...  David Spade|Sophie Marceau|Ever Carradine|Step ...
director                                            Sngmoo Lee                                             Jeff Pollack
tagline                                             Assassin. Hero. Legend.  A comedy about a guy who would do anything to  ...
runtime                                             100                                                    95
genres                                 Adventure|Fantasy|Action|Western|Thriller                           Comedy|Romance
release_date                                        2010-12-02 00:00:00                                    1999-04-23 00:00:00
release_year                                        2010                                                   1999
```

This is interesting. Same in format as above dataframe, we can see that The Warriors Way had the highest budget of all movies in the dataset of about $425M. This same movie also had the highest loss. So, it makes sense that having the highest budget in all makes the film harder to have higher revenues and earn more profits.

And the least budget of all, the Lost & Found movie of $1 has made me think how a movie with 95 min of runtime can managed with such low budget! Also making revenue of $100 and earning a profit $99, this may be a local movie release. Because it is kind of impossible to have such low budget and earning low revenues if it has released internationally.

Now let us look at the revenue side of the movies...!

d. Which movie had the greatest and least revenue?

Again, we call our function

```
# movies with largest and smallest revenue
print('\nMovies which had generated largest and smallest revenue: ')
print(get_highest_lowest(dataframe, 'revenue(US-Dollars)'))
```

And we get our output as:

```
Movies which had generated largest and smallest revenue:
                                                      1386                                                  5067
budget(US-Dollars)                                2.37e+08                                              6e+06
revenue(US-Dollars)                           2.78151e+09                                                  2
profit(US-Dollars)                            2.54451e+09                                             -6e+06
original_title                                     Avatar                                    Shattered Glass
cast                  Sam Worthington|Zoe Saldana|Sigourney Weaver|S ...  Hayden Christensen|Peter Sarsgaard|ChloÃ« Sevi ...
director                                    James Cameron                                       Billy Ray
tagline                                Enter the World of Pandora.                                    NaN
runtime                                              162                                              94
genres                        Action|Adventure|Fantasy|Science Fiction                        Drama|History
release_date                              2009-12-10 00:00:00                           2003-11-14 00:00:00
release_year                                         2009                                            2003
```

Interesting results again! Avatar movie also earning the most profit movie has made most revenue too! Making a revenue of more than $2.7B, it makes a logical sense that the more revenue you earn the more profit you gain and as we saw in earlier result, The Warriors Way having most budget had less chance of making profits. It really looks that there's correlation between profit and budget/revenue, but we cannot say just by looking at few results.
Having lowest revenue of $2, Shattered Glass movie seems like could not sell much tickets.

e. What is the average runtime of all movies?

To calculate the average runtime of movies, I wrote a little function.

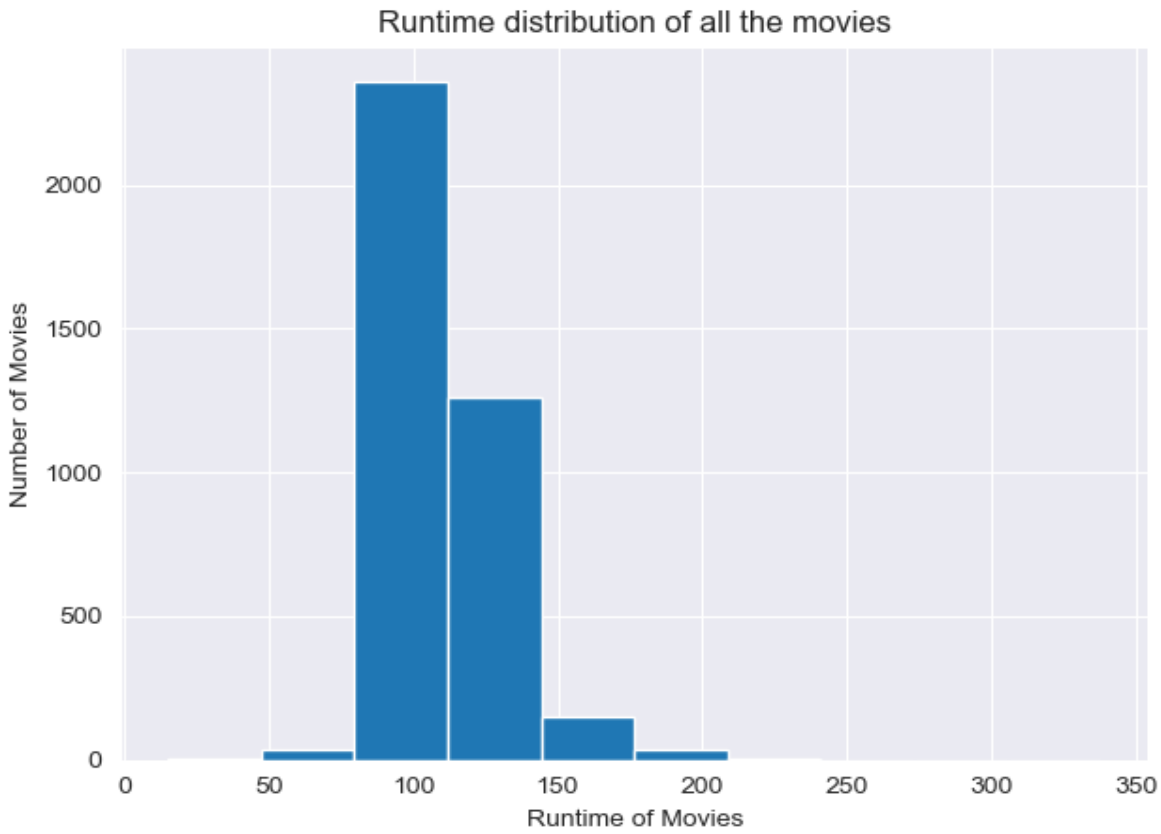
```
def get_average(dataframe, column_name):
    """
    This function calculates and returns average of the column specified

    Args:
        dataframe: cleaned dataset
        column_name: column name for which average is calculated

    Returns:
        average value of the column

    """
    return dataframe[column_name].mean()
```

Now to get the runtime, I wrote another function which calls this average function to calculate the runtime and then plot a histogram for the same.

```
def get_runtime(dataframe):
    """
    This function gets runtime average of all movies including graphs for runtime
distribution, swarmplot and box plot
    for deeper insights

    Args:
        dataframe: cleaned dataset containing movie runtime for analysis

    """
    # average runtime of all movies
    print('\nAverage runtime of movies is approximately:
{}'.format(get_average(dataframe, 'runtime')))

    # gives styles to background
    sns.set_style('darkgrid')
    plt.xlabel('Runtime of Movies')
    plt.ylabel('Number of Movies')
    plt.title('Runtime distribution of all the movies')

    # plotting runtime distribution of all movies using a histogram plot
    plt.hist(dataframe['runtime'])
    plt.show()

    # plotting interquartile range of movie runtime using box plot
    sns.boxplot(dataframe['runtime'])
    plt.show()

    # key insights on runtime
    print('\nHere are some key insights on movie runtime: ')
    print(dataframe['runtime'].describe())
```

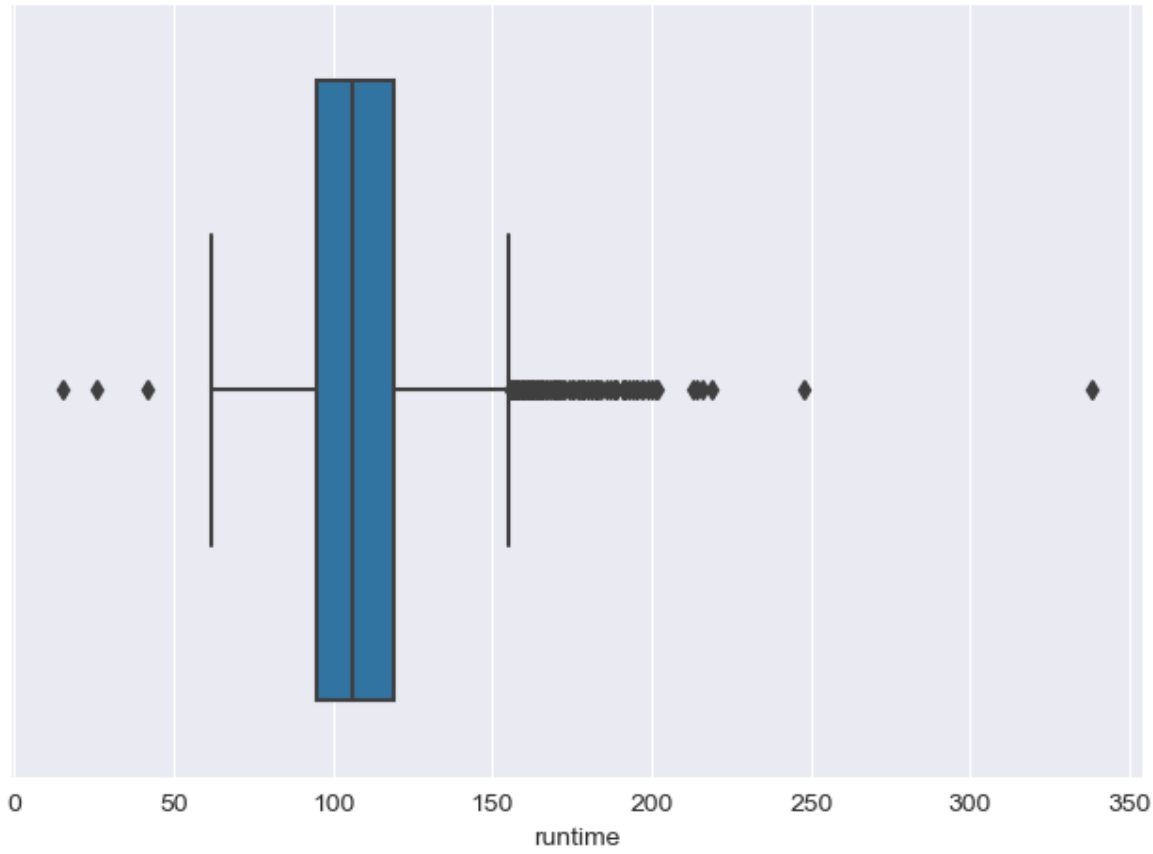
In the console, we get the average runtime and a graph for the same.

```
Average runtime of movies is approximately: 109.22029060716139
```



The above graph shows us that most movies lie between the time interval x1 to x2. For example, as you can see the tallest bar here is time interval between ~85-100 min and around 1000 movies out of 3854 movies have the runtime between these time intervals. So, through this graph we can also say that mode time of movies is ~85-110 min, has the highest concentration of data points around this time interval. The distribution of this graph is positively skewed or right skewed!

Let us dig deeper and find out the outliers in this distribution



From the above function, we also get a boxplot showing the outliers and some key insights including the interquartile range of the runtime.

```
Here are some key insights on movie runtime:
count    3854.000000
mean      109.220291
std        19.922820
min        15.000000
25%        95.000000
50%       106.000000
75%       119.000000
max       338.000000
Name: runtime, dtype: float64
```

The boxplot is that it gives us an overall idea of how spread the distribution in our case the runtime of movies is.

As we already saw in our previous calculations of least and highest runtime, this is the appropriate visualization in the comparison of other movies runtime. By looking at the boxplot we do not get the exact values, for example you can guess that the median is will around 100-110 min but by giving the describe function above we get the exact values.

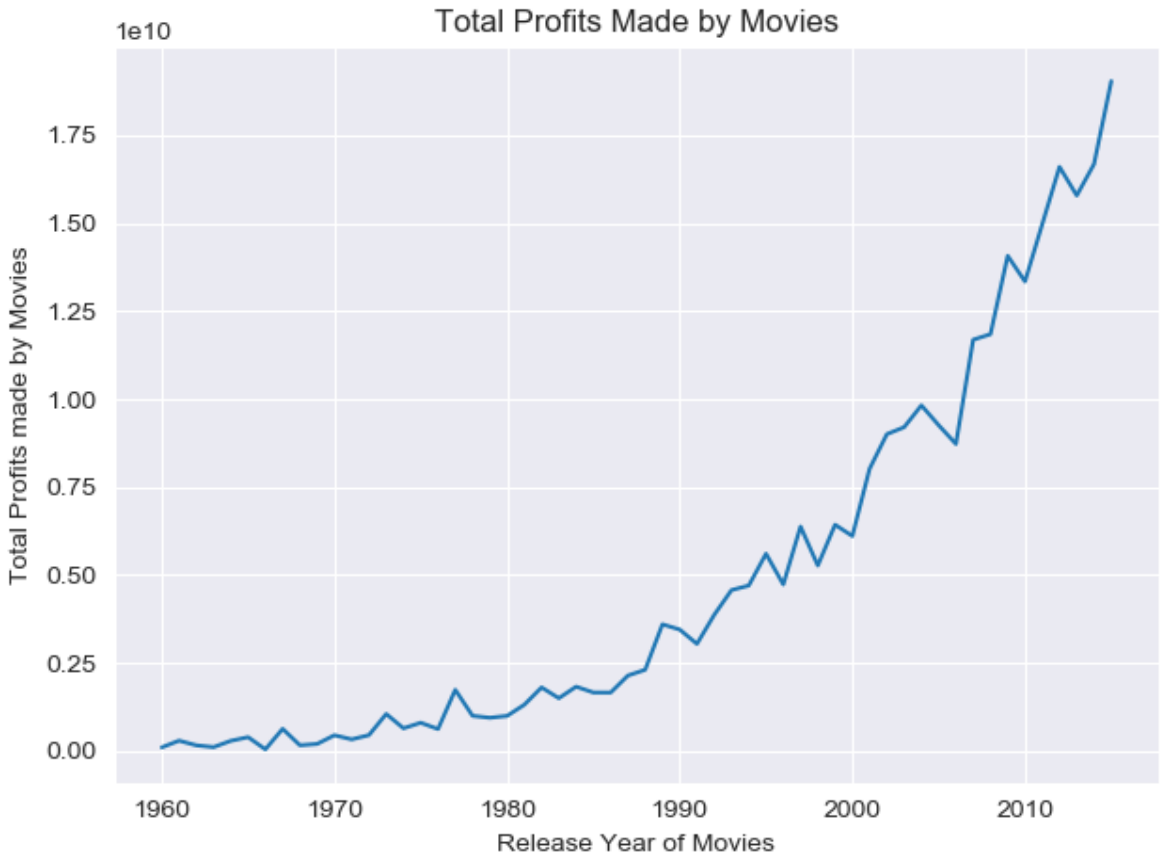
So, by looking at both, visualizations, and calculations, we can say that:

- There are 25% of movies having a runtime of less than 95 min
- There are 50% of movies having a runtime of less than 109 min. This is also the median of runtimes.
- There are 75% of movies having a runtime of less than 119 min
- 50% of movies have a runtime of between 95 min and 119 min. This is also our IQR.

As we can see there are more movies after the 3rd quartile range than the 1st. This makes the mean of the runtime pull towards the right or increases it.

f. In which year we had the most movies making profits? (profits of movies in each year)

```
def profits_each_year(dataframe):
    """
    This function return profits made by movies in each year

    Args:
        dataframe: cleaned dataset containing release year and profits made by
movies

    """
    # Since we want to know the profits of movies for every year we need to group
all the movies for those years
    profits_per_year = dataframe.groupby('release_year')['profit(US-
Dollars)'].sum()

    # giving the figure size(width, height)
    plt.xlabel('Release Year of Movies')
    plt.ylabel('Total Profits made by Movies')
    plt.title('Total Profits Made by Movies')
    # using a line plot
    plt.plot(profits_per_year)
    plt.show()

    # shows which year made the highest profit
    print('\nThe year which made most profit is:
{}'.format(profits_per_year.idxmax()))

    # using a dataFrame just to get a clean and better visual output
    profits_per_year = pd.DataFrame(profits_per_year)
    print('\nProfits made by movies in the last 5 years: ')
    print(profits_per_year.tail())
```



Each value in the y-axis is been multiplied to '1e10' (as shown above the plot). Since the profits of movies are high, having 9+ digits, cannot fit the axis. So, for example at the year 2010, the y-axis value is around 1.35, which means that the profit at that year made by al movies released in that year is 1.35x1e10 = 13500000000 which is $13.5 billion.

The year 2015, shows us the highest peak, having the highest profit than in any year, of more than $18 billion. This graph does not exactly prove us that every year pass by, the profits of movies will increase but when we see in terms of decades it does show significant up rise in profits. At the year 2000, profits were around $8 billion, but in just 15 years it increased by $10+ billion. Last 15 years had a significant rise in profits compared to any other decades as we can see in the graph.

Not every year had same number of movies released, the year 2015 had the most movie releases than in any other year. The older the movies, the less releases at that year (this is what the dataset shows us).

This dataset also does not show all the movies that has been released in each year. If it would the graph might would show some different trend.

Also, in the dataset, there were also movies that had negative profits which drags down the profits of other movies in those years. So, we are not just calculating the movies which made profits, but also which went in loss! The highest profit-making movie Avatar in 2009 alone drags the profit up by $2.5 billion out of $14 billion.

Also, we will look at the profits of each year with exact figures.

```
The year which made most profit is: 2015


Profits made by movies in the last 5 years:
              profit(US-Dollars)
release_year
2011                1.496669e+10
2012                1.659685e+10
2013                1.578274e+10
2014                1.667620e+10
2015                1.903215e+10
```

2015 was the year where movies made the highest profit of about $19+ billion which released in that year.

We are now done with exploring the dataset given. Now we want to find similar characteristics of most profitable movies.

2. Specific Statistics:
    a. Average duration of movies.

Before answering this question, we need to first clean the dataset, so we only have the data of movies that made profit not loss. Also, we need movies not only who just made profit by some dollars, but we need movies who made significant profits and then analysing similar characteristics of it. The following code does exactly that.

```
# assigning new dataframe which holds values only of movies having profit $50M or
more
profit_movie_data = dataframe[dataframe['profit(US-Dollars)'] >= 50000000]
# reindexing new dataframe
profit_movie_data.index = range(len(profit_movie_data))
# will initialize dataframe from 1 instead of 0
profit_movie_data.index += 1
```

We create a new dataframe containing movies that made a profit of >=$50000000 and we use this dataframe in our function to get the statistics. Now let us answer our question:

```
# average runtime of movies
print('\nAverage runtime of movies: {}
min(s)'.format(get_average(profit_movie_data, 'runtime')))
```

```
Average runtime of movies: 113.66741405082212 min(s)
```

    b. Average budget.

We write a similar code for this one.

```
# average budget of movies
print('Average budget of movies: ${}'.format(get_average(profit_movie_data,
'budget(US-Dollars)')))
```

```
Average budget of movies: $60444957.76083707
```

c. Average revenue.

```
# average revenue of movies
print('Average revenue of movies: ${}'.format(get_average(profit_movie_data,
'revenue(US-Dollars)')))
```

```
Average revenue of movies: $254957662.59491777
```

d. Average profits.

```
# average profit of movies
print('Average profit of movies: ${}'.format(get_average(profit_movie_data,
'profit(US-Dollars)')))
```

```
Average revenue of movies: $194512704.83408073
```

e. Which director directed most films?

For this and following questions, we introduce a new function which will get the count of values in a column.

```
def get_column_count(dataframe, column_name):
    """
    This function calculate count of specified column elements

    Args:
        dataframe: cleaned dataset
        column_name: column containing the elements for which count is calculated

    Returns:
        count of elements

    """
    # will take a column, and separate the string by '|'
    all_data = dataframe[column_name].str.cat(sep='|')

    # giving pandas series and storing the values separately
    all_data = pd.Series(all_data.split('|'))

    count = all_data.value_counts()

    return count
```

We call this function to get our directors count like this:

```
# count of movies directed by each director
director_count = get_column_count(profit_movie_data, 'director')
print('\nCount of movies directed by each directed: ')
print(director_count.head())
```

Which outputs:

```
Count of movies directed by each directed:
Steven Spielberg    23
Robert Zemeckis     13
Clint Eastwood      12
Tim Burton          11
Ridley Scott        10
dtype: int64
```

Steven Spielberg takes the crown! Directing 23 movies with over $50M+ in profit! Also, other directors following along the list such as Robert Zemeckis, Clint Eastwood, Tim Burton, and others prove to be great directors. Movies directed by these directors are more likely to make huge profits. Since we don't really know how many movies the directors directed in total in their lifetime, we can't say for sure that movies directed by above directors will always earn this much but gives us the idea that how much likely it is when it is directed by them.

f. Which cast has appeared the most?

```
# count of cast starring in a particular movie
cast_count = get_column_count(profit_movie_data, 'cast')
print('\nCount of cast starring in a particular movie: ')
print(cast_count.head())
```

```
Count of cast starring in a particular movie:

Tom Cruise            27

Brad Pitt             25

Tom Hanks             22

Sylvester Stallone    21

Cameron Diaz          20

dtype: int64
```

Tom Cruise tops the list for appearing the most in movies profiting more than $50M. Directors hiring these actors will have higher probability of making huge profits. Also, this does not mean that actors other than these acting in a film will make less profit. Famous actors such as Tom Cruise, Brad Pitt, Tom Hanks, have a huge fanbase, making the audience attract to the movie more than actors other than these hence this would affect the revenue of the movie but not always, ultimately it comes down to storyline and other influential factors. By looking at this dataset we can at least say that these actors acting in a film have a higher probability of attraction to a movie, hence increasing the advantage of earing high profits!

Since we don't know how many movies… Just Like the directors, we can say for actors well! We can't always be sure that movies acted by these actors will always earn this much but gives us the idea that how much likely it is when it is acted by them.

g. Which genre were more successful?

We create a separate function for this question, and we call this function with the dataframe as the argument.

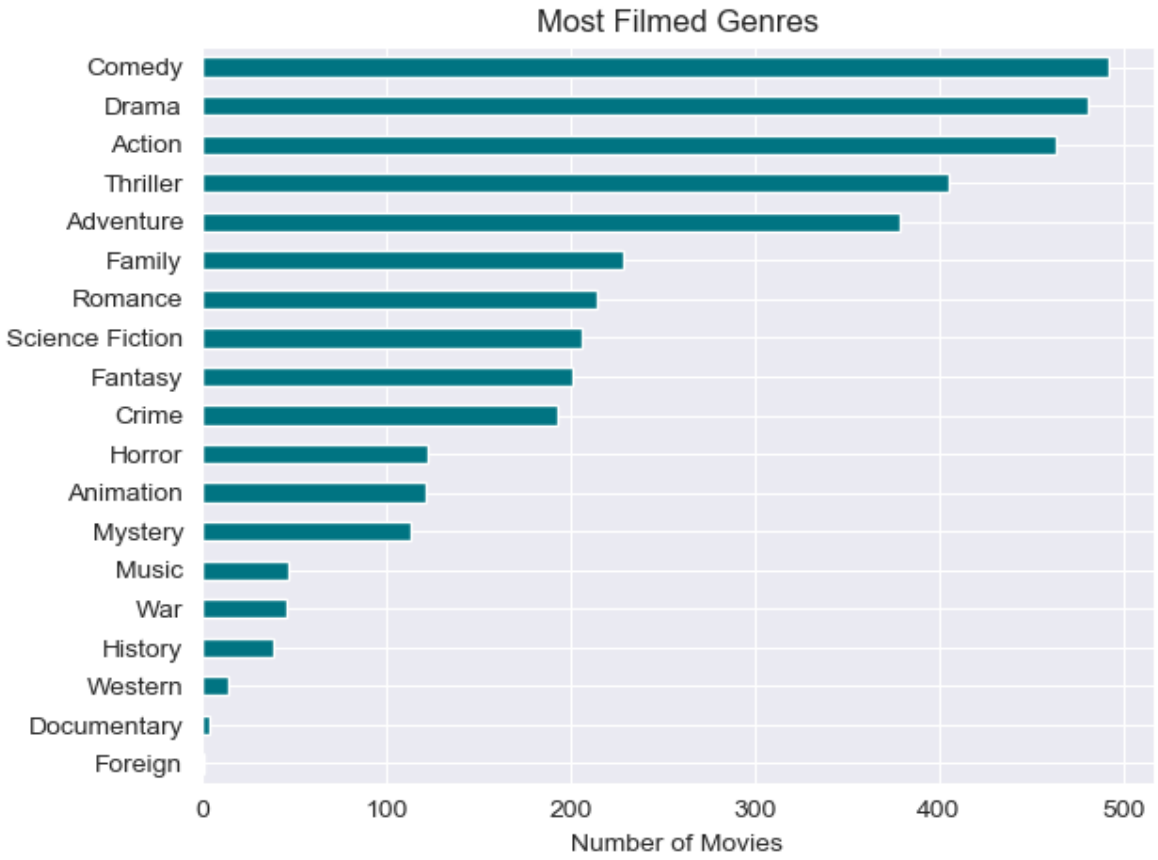
```
def successful_genre(dataframe):
    """
    This function counts number of movies in a particular genre and plots it in a
bar graph

    Args:
        dataframe: cleaned dataset

    """
    genre_count = get_column_count(dataframe, 'genres')
    print('\nNumber of movies in each genre: ')
    print(genre_count.head())

    genre_count.sort_values(ascending=True, inplace=True)

    successful_genre_graph = genre_count.plot.barh(color='#007482')
    successful_genre_graph.set(title='Most Filmed Genres')
    successful_genre_graph.set_xlabel('Number of Movies')
    plt.show()


# count of successful movies in a particular genre
successful_genre(profit_movie_data)
```

In the console we get the following result along with the graph.

```
Number of movies in each genre:

Comedy       492

Drama        481

Action       464

Thriller     405

Adventure    379

dtype: int64
```

Action, Drama and Comedy genres are the most as visualized, but Comedy takes the prize, about 492 movies have genres comedy which make $50M+ in profit. In comparison, even Adventure and Thriller really play the role. These five genres have a greater number of movies than rest of the genres as shown by visualization. Probability of earning more than $50M for these genres are higher, although other genres do count, it depends on a lot of other factors. Western, war, history, music, documentary, and the most least foreign genres have less probability to make this much in profit as in comparison to other genre.

This also does not prove that if you have a movie with an Action, comedy, and drama genre in it will have a guarantee to make more than $50M but it would have a significant interest and attraction to the population.

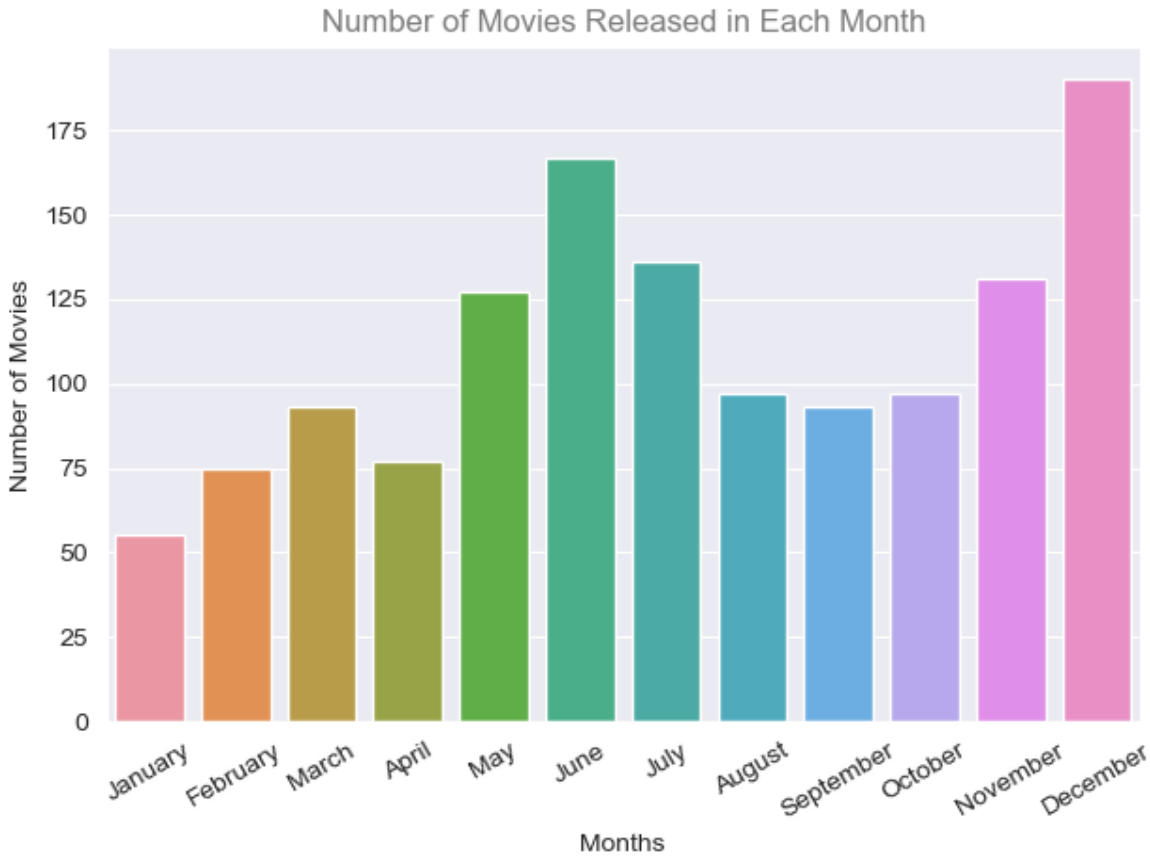


h. Which month released highest number of movies in all the years?

We again use another function to get and plot the number of movies released each month. We call the function and pass the dataframe as an argument.

```
def highest_movie_month(dataframe):
    """
    This function calculates highest number of movies in a particular month

    Args:
        dataframe: cleaned dataset

    """
    # grouping all of the months of years and then calculate the profits of those
months
    index_release_date = dataframe.set_index('release_date')
    # now we need to group all the data by month, since release date is in form
of index, we extract month from it
    group_index = index_release_date.groupby([index_release_date.index.month])

    monthly_movie_count = group_index['profit(US-Dollars)'].count()

    monthly_movie_count = pd.DataFrame(monthly_movie_count)
    print('\nNumber of movies released in each month: ')
    print(monthly_movie_count)

    month_list = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
                  'November', 'December']

    monthly_movie_count_bar = sns.barplot(x=monthly_movie_count.index,
y=monthly_movie_count['profit(US-Dollars)'],
                                          data=monthly_movie_count)
    monthly_movie_count_bar.axes.set_title('Number of Movies Released in Each
Month', alpha=0.6)
    monthly_movie_count_bar.set_xlabel("Months")
    monthly_movie_count_bar.set_ylabel("Number of Movies")
    monthly_movie_count_bar.set_xticklabels(month_list, rotation=30)
    plt.show()


# count of movies in a month
highest_movie_month(profit_movie_data)
```

```
Number of movies released in each month:

              profit(US-Dollars)

release_date

1                             55

2                             75

3                             93

4                             77

5                            127

6                            167

7                            136

8                             97

9                             93

10                            97

11                           131

12                           190
```



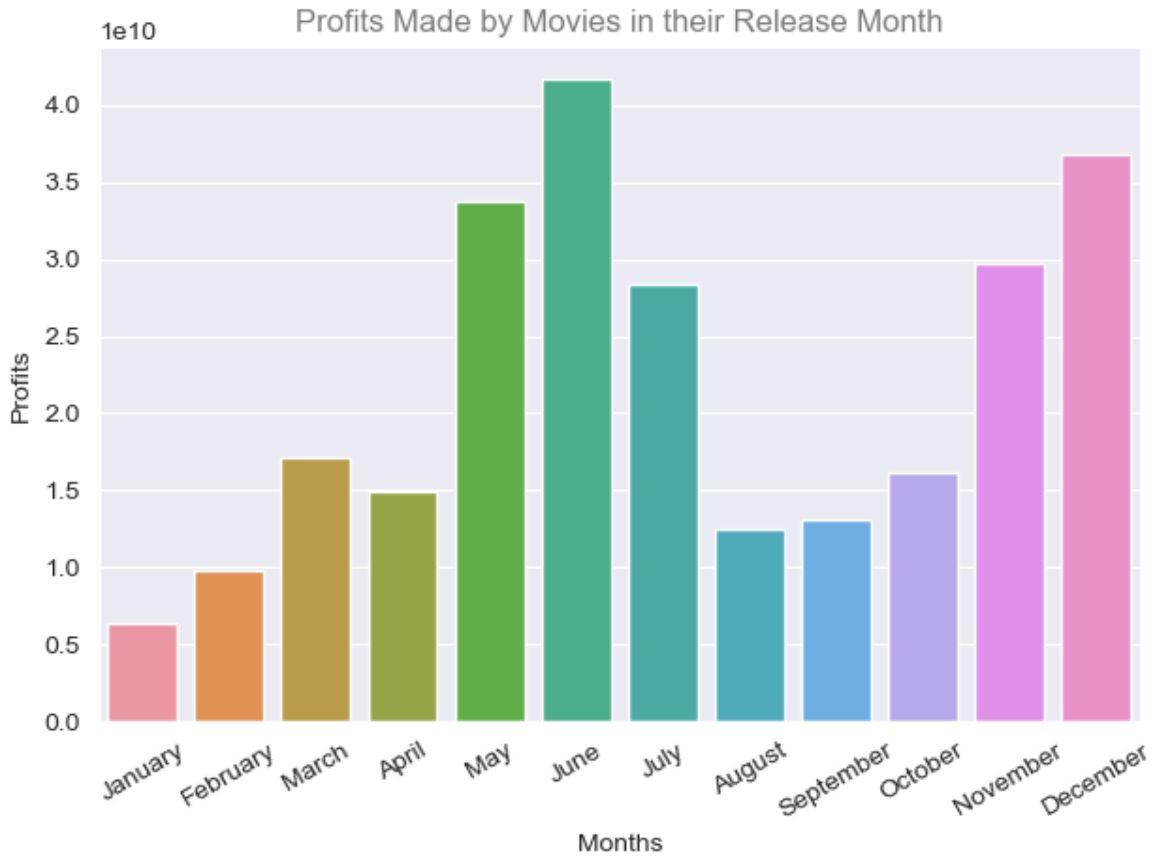
We find that December has seen the highest number of releases followed by June, July, May, and November. The summers and the holiday season seem to work great for movies as people look for more entertainment during these seasons.

i. And which month made the most profit?

Moving on to our final question for most profitable movies, we again use a similar function to get our results and the graph.

```
def most_profit_month(dataframe):
    """
    This function returns the month which made most profit

    Args:
        dataframe: cleaned dataset

    """
    index_release_date = dataframe.set_index('release_date')
    group_index = index_release_date.groupby([index_release_date.index.month])
    monthly_profit = group_index['profit(US-Dollars)'].sum()

    monthly_profit = pd.DataFrame(monthly_profit)
    print('\nProfits made by movies in their release month: ')
    print(monthly_profit)

    monthly_profit_bar = sns.barplot(x=monthly_profit.index,
y=monthly_profit['profit(US-Dollars)'],
                                     data=monthly_profit)

    month_list = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
                  'November', 'December']

    monthly_profit_bar.axes.set_title('Profits Made by Movies in their Release
Month', alpha=0.6)
    monthly_profit_bar.set_xlabel("Months")
    monthly_profit_bar.set_ylabel("Profits")
    monthly_profit_bar.set_xticklabels(month_list, rotation=30)
    plt.show()


# most profitable month
most_profit_month(profit_movie_data)
```

```
Profits made by movies in their release month:

              profit(US-Dollars)

release_date

1                    6.379417e+09

2                    9.739829e+09

3                    1.713751e+10

4                    1.495503e+10

5                    3.371841e+10

6                    4.164675e+10

7                    2.836506e+10

8                    1.253449e+10

9                    1.308814e+10

10                   1.612032e+10

11                   2.972130e+10

12                   3.685174e+10
```



We see that June see the second greatest number of releases but earns the most profit followed by December, May, November, and July. The season trend is quite similar as the previous one.

Now we are in the final leg of our exploration. Let us look at some key traits and analyse them. Here, I wrote a function that contains all the code for our analysis. To make the code simpler and more readable, each statistic is housed in its own function and this function calls each one to give out the result.

```
def general_analysis(dataframe):
    """
    This function gets general analysis and correlation between various factors

    Args:
        dataframe: cleaned data passed for analysis

    """
    # movie production trend over the years
    movie_production_trend(dataframe)

    # top 20 highest grossing movies
    highest_grossing_movies(dataframe)

    # top 20 most expensive movies
    most_expensive_movies(dataframe)

    # budget-revenue correlation
    budget_revenue_corr(dataframe)

    # runtime associated with genre
    genre_runtime(dataframe)
```

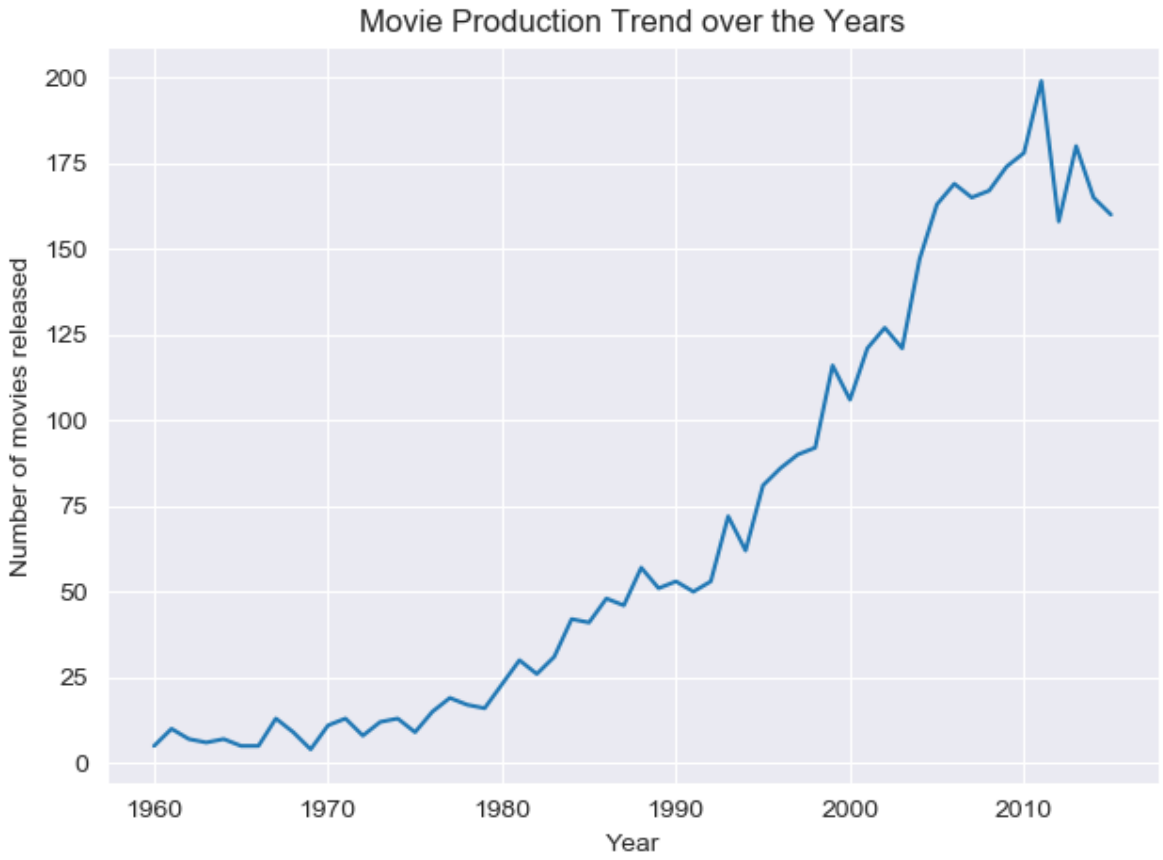
Now let us get right into analysis.

3. General analysis:
    a. How have movie production trends varied over the years?

```
def movie_production_trend(dataframe):
    """
    This function analyse the trend movie production has taken over the years

    Args:
        dataframe: cleaned dataset

    """
    # Number of movies produced each year
    movies_per_year = dataframe['release_year'].value_counts().sort_index()
    # Years with maximum and minimum movie production
    print('\nYear with lowest movie production:
{}'.format(movies_per_year.idxmin()))
    print('Year with highest movie production:
{}'.format(movies_per_year.idxmax()))
    plt.title('Movie Production Trend over the Years')
    plt.xlabel('Year')
    plt.ylabel('Number of movies released')
    plt.plot(movies_per_year)
    plt.show()
```

On the console we get our results as:

```
Year with lowest movie production: 1969

Year with highest movie production: 2011
```

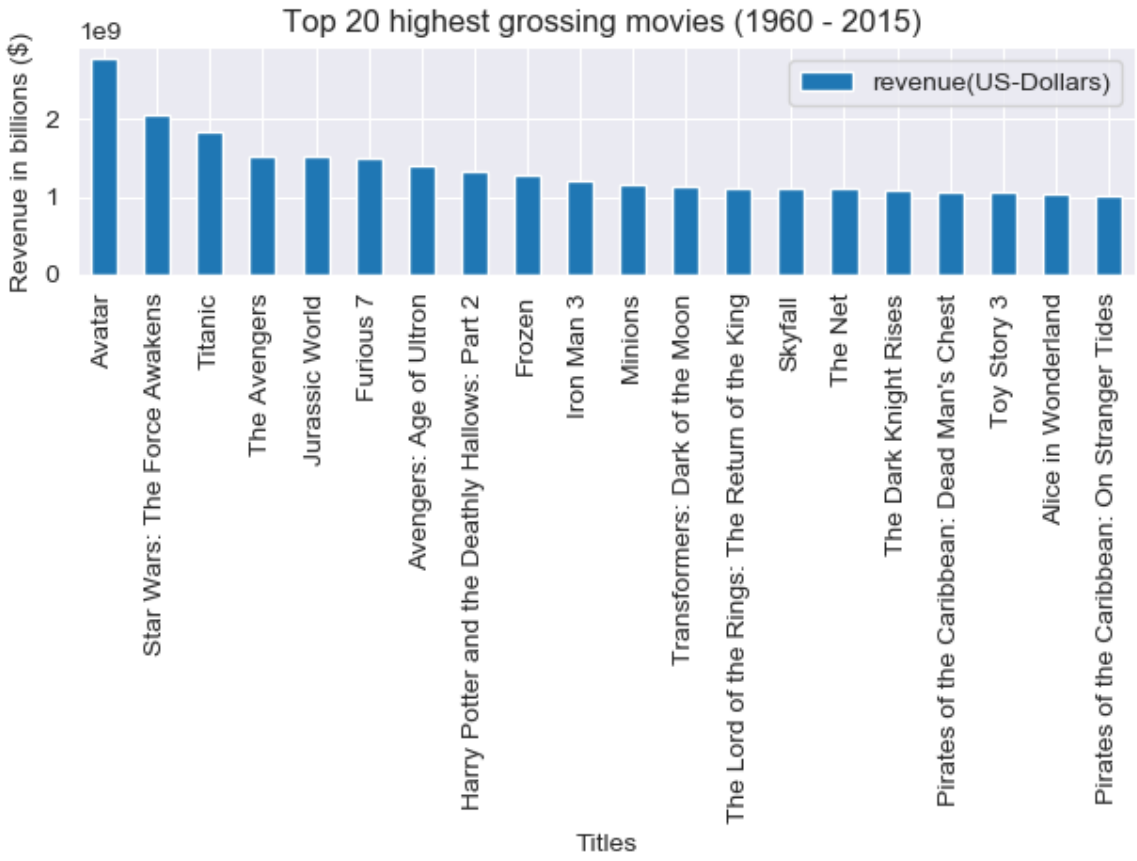


Movie production has increased over the years from 1960 to 2015. The decade of 2000 - 2010 shows a steep increase in production compared to previous decades. The year 2011 saw the maximum number of movie production, and 1969 with the least production.

b. What are the top 20 highest grossing movies?

We call up our function to get top 20 highest grossing movies.

```
def highest_grossing_movies(dataframe):
    """
    This function gets the top 20 highest grossing movies

    Args:
        dataframe: cleaned dataset

    """
    sorted_revenue = dataframe['revenue(US-
Dollars)'].sort_values(ascending=False)[:20]
    high_grossing = pd.DataFrame(sorted_revenue)

    titles = []
    revenues = []
    for i in sorted_revenue.index:
        titles.append(dataframe.loc[i, 'original_title'])
        revenues.append(sorted_revenue.loc[i])

    high_grossing['Titles'] = titles
    # high_grossing['Revenues'] = revenues

    high_grossing.set_index('Titles', inplace=True)
    high_grossing.plot(kind='bar')
    plt.title('Top 20 highest grossing movies (1960 - 2015) ')
    plt.ylabel('Revenue in billions ($)')
    plt.show()

    # List of top 20 highest grossing movies and their revenue
    print('\nTop 20 highest grossing movies: ')
    print(high_grossing)
```



In the console we get the exact numbers for these movies.

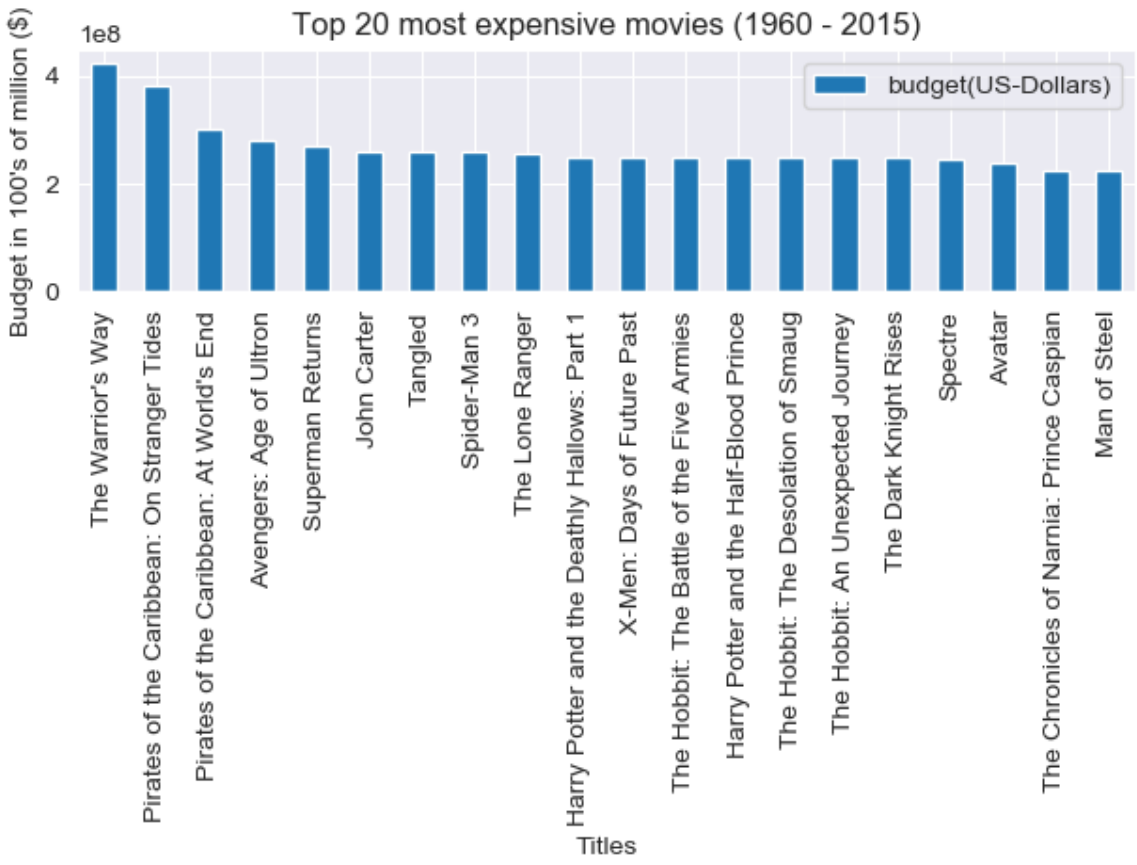
```
Top 20 highest grossing movies:
                                                revenue(US-Dollars)
Titles
Avatar                                                 2.781506e+09
Star Wars: The Force Awakens                           2.068178e+09
Titanic                                                1.845034e+09
The Avengers                                           1.519558e+09
Jurassic World                                         1.513529e+09
Furious 7                                              1.506249e+09
Avengers: Age of Ultron                                1.405036e+09
Harry Potter and the Deathly Hallows: Part 2           1.327818e+09
Frozen                                                 1.274219e+09
Iron Man 3                                             1.215440e+09
Minions                                                1.156731e+09
Transformers: Dark of the Moon                         1.123747e+09
The Lord of the Rings: The Return of the King          1.118889e+09
Skyfall                                                1.108561e+09
The Net                                                1.106280e+09
The Dark Knight Rises                                  1.081041e+09
Pirates of the Caribbean: Dead Man's Chest             1.065660e+09
Toy Story 3                                            1.063172e+09
Alice in Wonderland                                    1.025467e+09
Pirates of the Caribbean: On Stranger Tides            1.021683e+09
```

The highest grossing movie over the time period 1960 - 2015 has been Avatar with a revenue of 2.8 billion dollars.

c. What are the top 20 most expensive movies?

Once again, we call the function.

```
def most_expensive_movies(dataframe):
    """
    This function gets top 20 most expensive movies

    Args:
        dataframe: cleaned dataset

    """
    sorted_budget = dataframe['budget(US-
Dollars)'].sort_values(ascending=False)[:20]
    high_budget = pd.DataFrame(sorted_budget)
    titles_exp = []
    budgets = []
    for i in sorted_budget.index:
        titles_exp.append(dataframe.loc[i, 'original_title'])
        budgets.append(sorted_budget.loc[i])

    high_budget['Titles'] = titles_exp
    # high_budget['Budgets'] = budgets

    high_budget.set_index('Titles', inplace=True)
    high_budget.plot(kind='bar')
    plt.title('Top 20 most expensive movies (1960 - 2015)')
    plt.ylabel('Budget in 100\'s of million ($)')
    plt.show()

    # List of top 20 most expensive movies and their revenue
    print('\nTop 20 most expensive movies of all time: ')
    print(high_budget)
```



In the console we get the exact numbers for these movies.

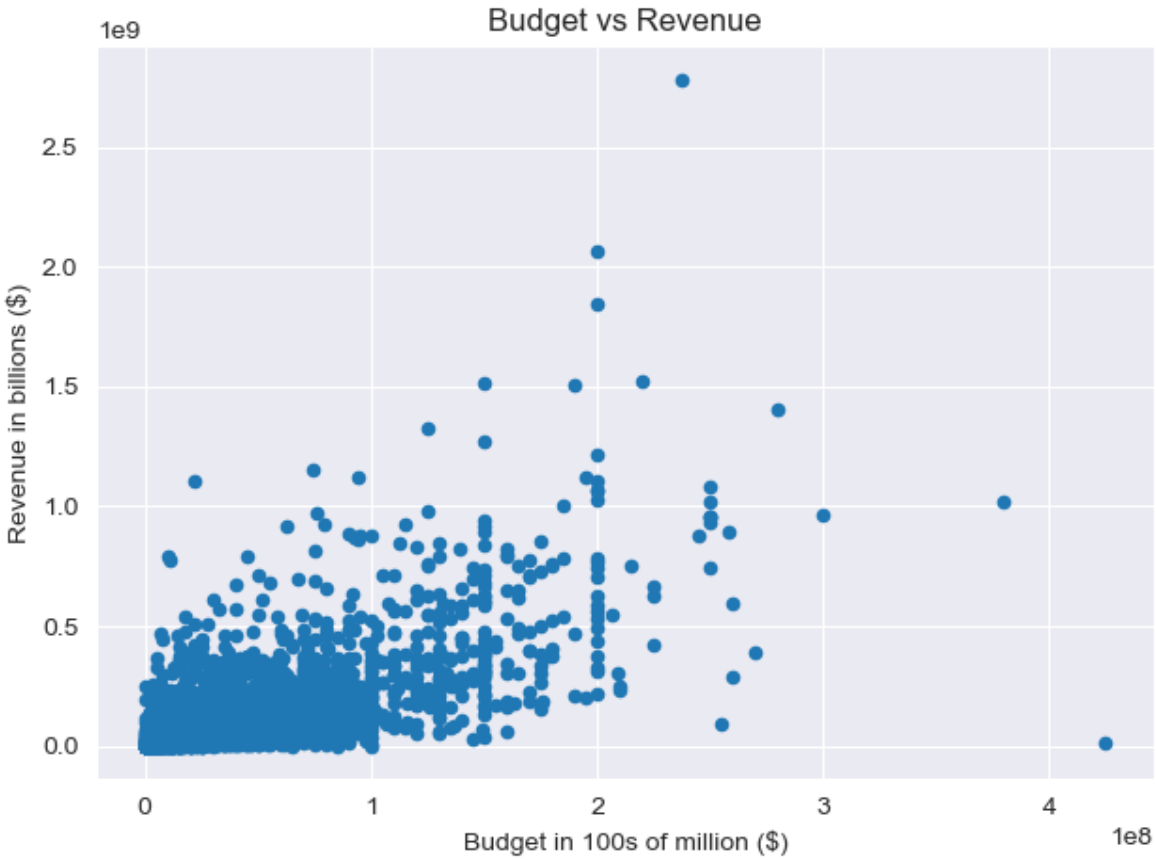
Top 20 most expensive movies of all time:

Titles	budget(US-Dollars)
The Warrior's Way	425000000.0
Pirates of the Caribbean: On Stranger Tides	380000000.0
Pirates of the Caribbean: At World's End	300000000.0
Avengers: Age of Ultron	280000000.0
Superman Returns	270000000.0
John Carter	260000000.0
Tangled	260000000.0
Spider-Man 3	258000000.0
The Lone Ranger	255000000.0
Harry Potter and the Deathly Hallows: Part 1	250000000.0
X-Men: Days of Future Past	250000000.0
The Hobbit: The Battle of the Five Armies	250000000.0
Harry Potter and the Half-Blood Prince	250000000.0
The Hobbit: The Desolation of Smaug	250000000.0
The Hobbit: An Unexpected Journey	250000000.0
The Dark Knight Rises	250000000.0
Spectre	245000000.0
Avatar	237000000.0
The Chronicles of Narnia: Prince Caspian	225000000.0
Man of Steel	225000000.0

The most expensive movie (highest budget) over the time period 1960 - 2015 has been The Warrior's way. This movie does not feature in the top 20 grossing movies.

d. How do budgets correlate with revenues?

```
def budget_revenue_corr(dataframe):
    """
    This function plots correlation between budget and revenue

    Args:
        dataframe: cleaned dataset

    """
    dataframe.plot(x='budget(US-Dollars)', y='revenue(US-Dollars)',
kind='scatter')
    plt.title('Budget vs Revenue')
    plt.xlabel('Budget in 100s of million ($)')
    plt.ylabel('Revenue in billions ($)')
    plt.show()
    person_correlation_coeff = dataframe['budget(US-
Dollars)'].corr(dataframe['revenue(US-Dollars)'], method='pearson')
    print('\nCorrelation between budget and revenue:
{}'.format(person_correlation_coeff))
```

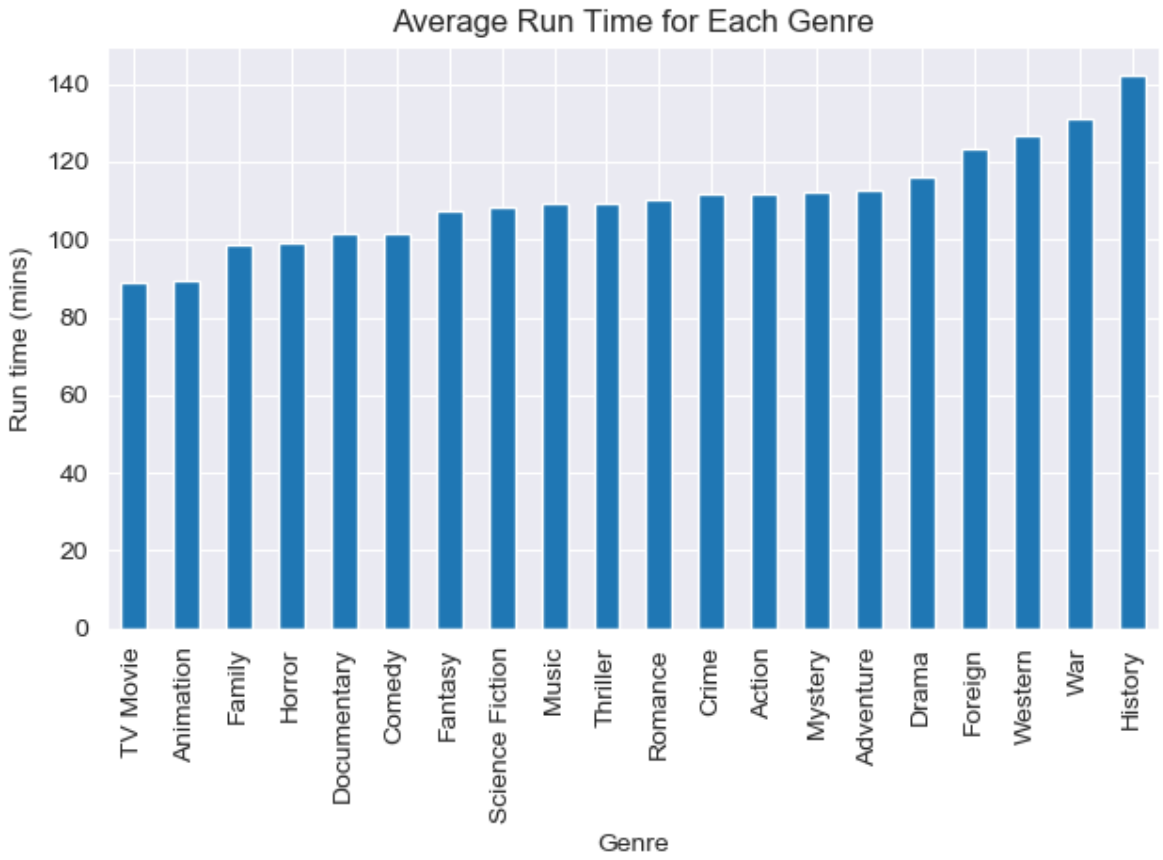



```
Correlation between budget and revenue: 0.6885561524636744
```

From the uphill scatterplot and the value of 0.69 for the Pearson's coefficient we can infer that there is a positive relationship between budget and revenue. However, not a perfectly positive correlation. It can be seen from the plot that there are some movies with high budgets but low revenues and some with low budgets and high revenues. The outliers are more with high budget movies which get low to moderate revenues.

e. What run times are associated with each genre?

```
def genre_runtime(dataframe):
    """
    This function shows what run times are associated with genres

    Args:
        dataframe: cleaned dataset

    """
    # Drop rows with null values in genre and director columns
    dataframe.dropna(subset=['genres'], inplace=True)

    # Converting the 'genre' column into a list of genres by splitting at the
pipe symbol
    dataframe['genres'] = np.where((dataframe['genres'].str.contains('|')),
dataframe['genres'].str.split('|'),
                                   dataframe['genres'])

    # Making sure every row has data as a list, even if only one genre is present
    dataframe.loc[:, 'genres'] = dataframe.genres.apply(np.atleast_1d)

    # Horizontally stacking all the lists from all rows into one big list
    all_genres = np.hstack(dataframe.genres)

    # Repeating the runtime as many times as the length of list genre and merging
it all into one list
    all_runtimes = []
    for runtime, genre in dataframe[['runtime', 'genres']].values:
        all_runtimes += [runtime] * len(genre)

    # Assigning the merged lists / arrays to a new dataframe
    genre_runtime_combined = pd.DataFrame({'genre': all_genres, 'runtime':
all_runtimes})

    # Group by genre and find the average of run times sorted in ascending order
    runtime_by_genre =
genre_runtime_combined.sort_values(['runtime']).groupby('genre')['runtime'].mean(
)
    runtime_by_genre.sort_values().plot(kind='bar')
    plt.title('Average Run Time for Each Genre')
    plt.ylabel('Run time (mins)')
    plt.xlabel('Genre')
    plt.show()
```



We find that History is the genre with the longest movies while Animation movies are the shortest.

Limitations

The dataset was assessed, and necessary cleaning steps were performed as documented above. Datatypes were made relevant to the context of the columns, duplicates were removed, and zeros and null values were dealt with under the exploration of each research question, according to the reasoning provided. Removal of rows containing nulls and/or zeros reduced the data available for analysis, which may impact the results. The accuracy of the results can be greatly increased if we include more number of movies like from around the world, different production houses and languages.

Conclusion

The dataset was assessed, and necessary cleaning steps were performed as documented above. Datatypes were made relevant to the context of the columns, duplicates were removed, and zeros and null values were dealt with as necessary. The correlations explained do not imply causations. The inferences made are tentative and have scope for further refinement.

All this analysis gave us some form of an idea of what a successful movie could usually look like, meaning what the cast, runtimes, release month, genre, directors and more would be. However, all these directors, actors, genres and released dates have a common trend of attraction. If we release a movie with these characteristics, it gives people high expectations from this movie. Thus, by attracting more people towards the movie, but it ultimately comes down to story and other important influential factors. People having higher expectations give us less probability of meeting their expectations. Even if the movie was worth, people's high expectations would lead in biased results ultimately effecting the profits. We also see this in real life specially in sequels of movies. This was just one example of an influential factor that would lead to different results, there are many that must be taken care of!

And by this we conclude with our analysis. Thank you!