Exploring TMDB Dataset

Prepared By:

Ezz Eldin Ahmed

Exploring the TMDB dataset

Movies -as we know them- are simply a long-form medium for telling a story and to entertain. Same as a play or a book.

From the days of Georges Méliès and the 3-minute movies, to the age of the Marvel Cinematic Universe, movies have changed drastically over the years. They became bigger and more expensive to make. They also generated a huge revenue for some, and were a complete failure for others.

Analyzing the TMDB (The Movie Database) dataset, extending from 1960 to 2015, this project aims to answer the following questions:

1. What are the most financially successful movies?
2. Which movies “bombed” at the box office?
3. What are the most popular movies?
4. Did movies get longer or shorter over time?
5. Does spending more money on a movie mean higher revenue?
6. Which genres made the highest profit?

Answering these questions could serve as a basis to analyze the best and worst performing movies in order to understand why they succeeded or failed and learn from the mistakes to create better movies.

This data analysis project is utilizing the Python programming language and the Jupyter notebook and the following libraries:

- Pandas.
- NumPy.
- MatPlotLib.
- Seaborn.
- Plotly Express.

Part 1: Exploratory Data Analysis (EDA):

After viewing the head of the data frame, as well as the summary statistics and the null values, we concluded that safely drop the following columns: [homepage, tagline, overview, imdb_id, id] as they are irrelevant to the analysis.

The revenue, budget columns are dropped as well, because the revenue_adj and budget_adj columns are adjust to 2010 inflation levels, so they're better suited for this data which extends around 55 years.

There are missing values in some columns to various degrees, however, we already decided to drop the most of these columns, since we wouldn't need them in our analysis.

From the summary statistics:

- We noticed that the minimum values for runtime (the movie's length) and both the budget and revenue columns is zero, that doesn't make sense considering that all three must be positive (greater than 0) values.
- We can also notice that the min, 25th and 50th percentiles are all zeroes! Which implies that most of our data points in those columns are missing.

Part 2: Data Preprocessing:

Running a for loop to check the percentage of missing data points the [budget_adj, revenue_adj, runtime] columns showed us that:

- Around 55% of the budget_adj data are zeroes.
- Around 52% of the revenue_adj data are zeroes.
- Around 0.28% of the runtime data are zeroes.

As a general rule, there are ways to deal with missing data depending on the data itself:

- If the missing values are around 5%, it's safe to remove the observation altogether.
- If the missing values are more than 5%, we could:
    1. Estimate the missing values using mean, median or mode.
    2. Estimate the missing data using machine learning (ML) techniques, such as Linear regression or KNN.

We start asking ourselves about the importance of the variable we're dealing with. In our case, the budget_adj and revenue_adj columns are essential to the analysis, so we can't omit any of its data. Estimating using mean, median or mode is impractical since for mean or median: the data would be altered to much and may result in unrealistic budgets and revenues for the different time periods, as for mode: the most reoccurring value is zero so it won't help. Same applies for ML techniques.

The best possible solution was to convert all zeroes to NaNs (Not a number) so that we could keep the data intact with just the zeroes out of the way instead of removing half of our data points.

As for the other columns that weren't dropped but still had missing values [cast, genres, director, runtime], they observations could be omitted without much problem since they're insignificant (less than 5%).

```
tmdb[['revenue_adj', 'budget_adj']].describe()
```

[23]:

	revenue_adj	budget_adj
count	1.086600e+04	1.086600e+04
mean	5.136436e+07	1.755104e+07
std	1.446325e+08	3.430616e+07
min	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00
50%	0.000000e+00	0.000000e+00
75%	3.369710e+07	2.085325e+07
max	2.827124e+09	4.250000e+08

Summary statistics before converting replacing 0 with NaNs

```
tmdb_clean[['revenue_adj', 'budget_adj']].describe()
```

7]:

	revenue_adj	budget_adj
count	4.850000e+03	5.170000e+03
mean	1.150774e+08	3.688774e+07
std	1.988419e+08	4.195701e+07
min	2.370705e+00	9.210911e-01
25%	1.046262e+07	8.102293e+06
50%	4.392749e+07	2.272271e+07
75%	1.315644e+08	5.007483e+07
max	2.827124e+09	4.250000e+08

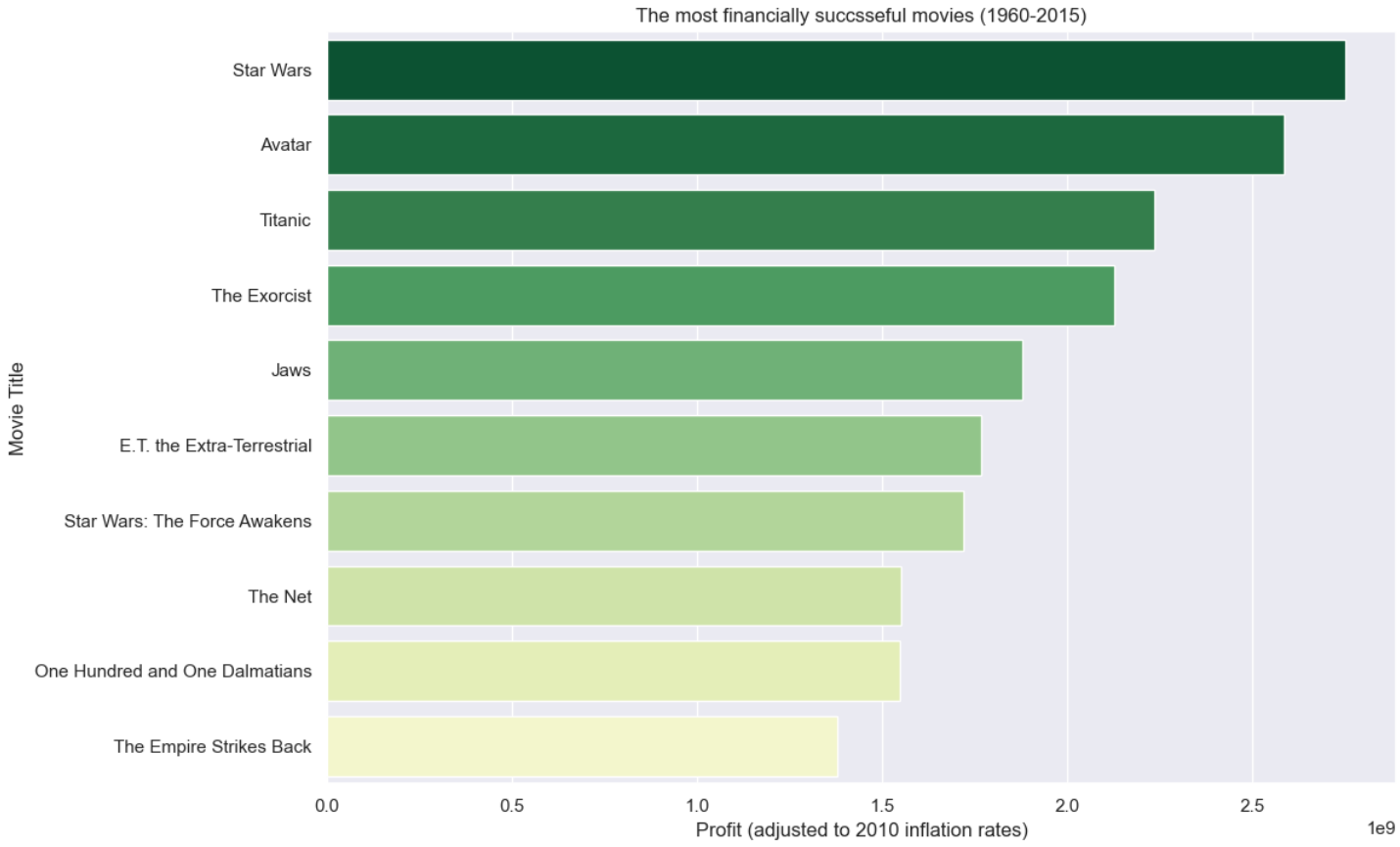
Summary statistics after converting replacing 0 with NaNs

Part 3. Data Analysis and Visualization:

Answering the project's main questions

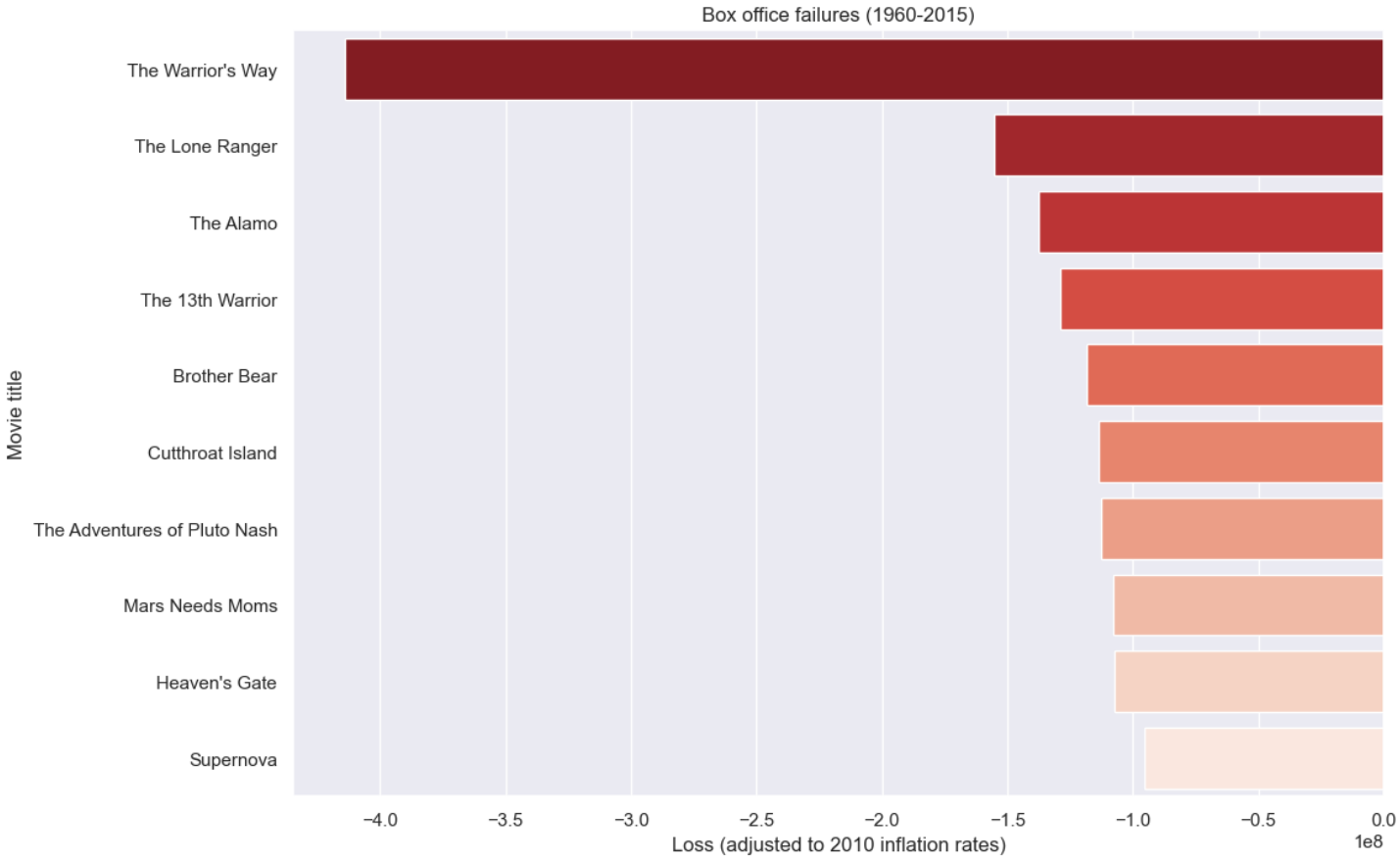
Q_1: What are the most financially successful movies?

A_1: The most financially successful movies (in billion USD):



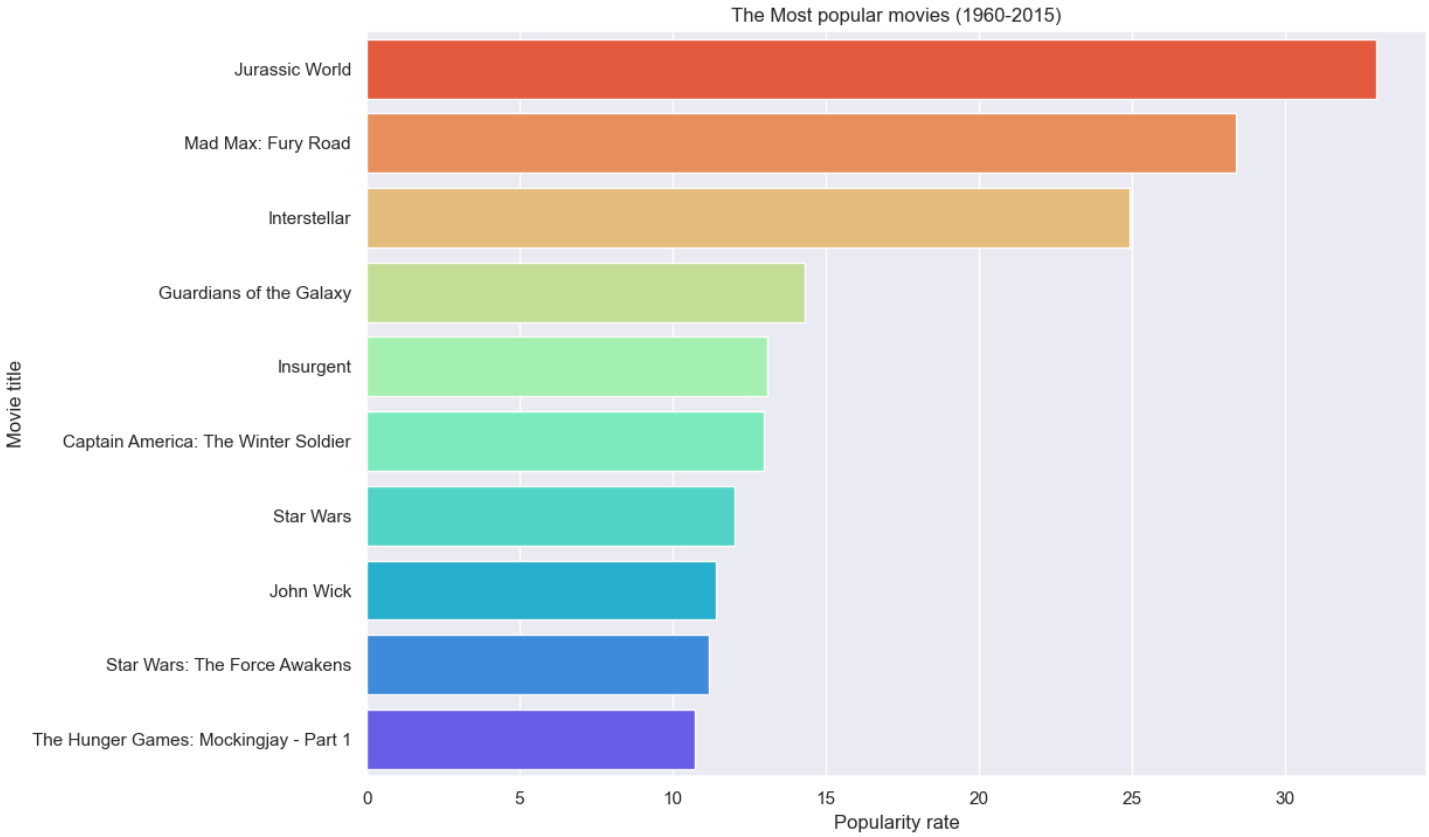
Q_2: Which movies “bombed” at the box office?

A_2: The worst performing movies (in million USD):



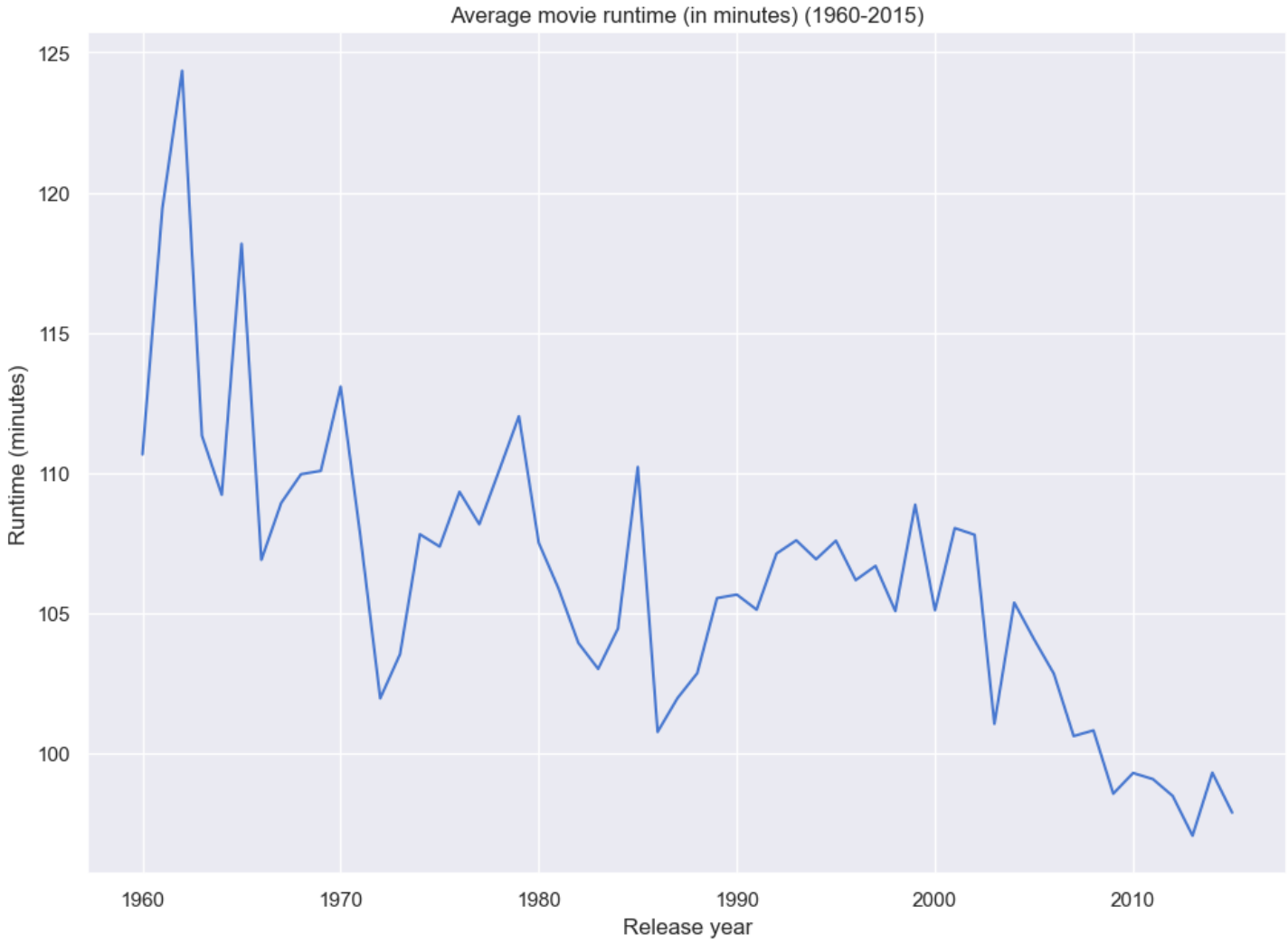
Q_3: What are the most popular movies?

A_3: The most popular movies:



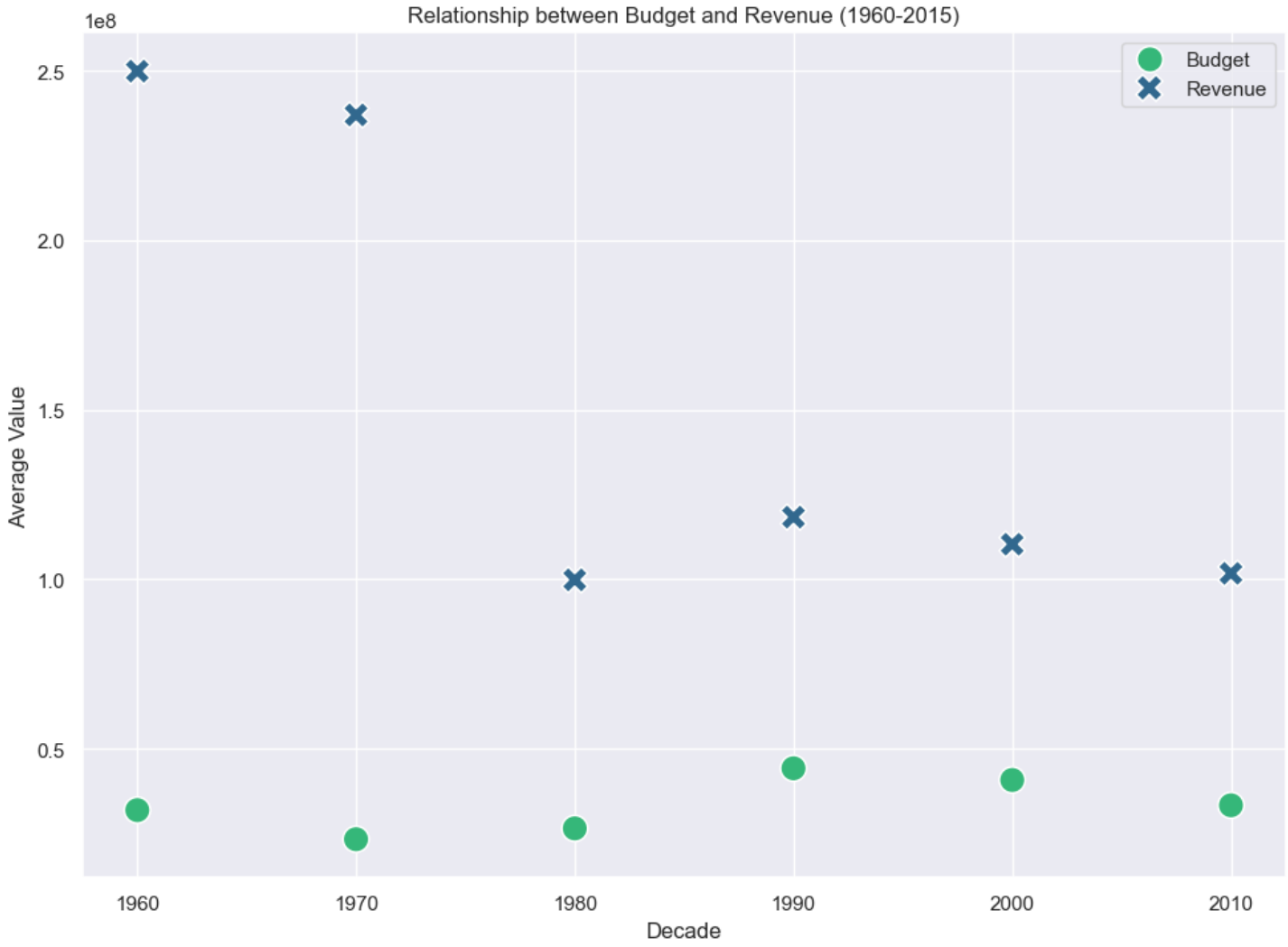
Q_4: Did movies get longer or shorter over time?

A_4: Movies lengths have gotten slightly shorter over time.



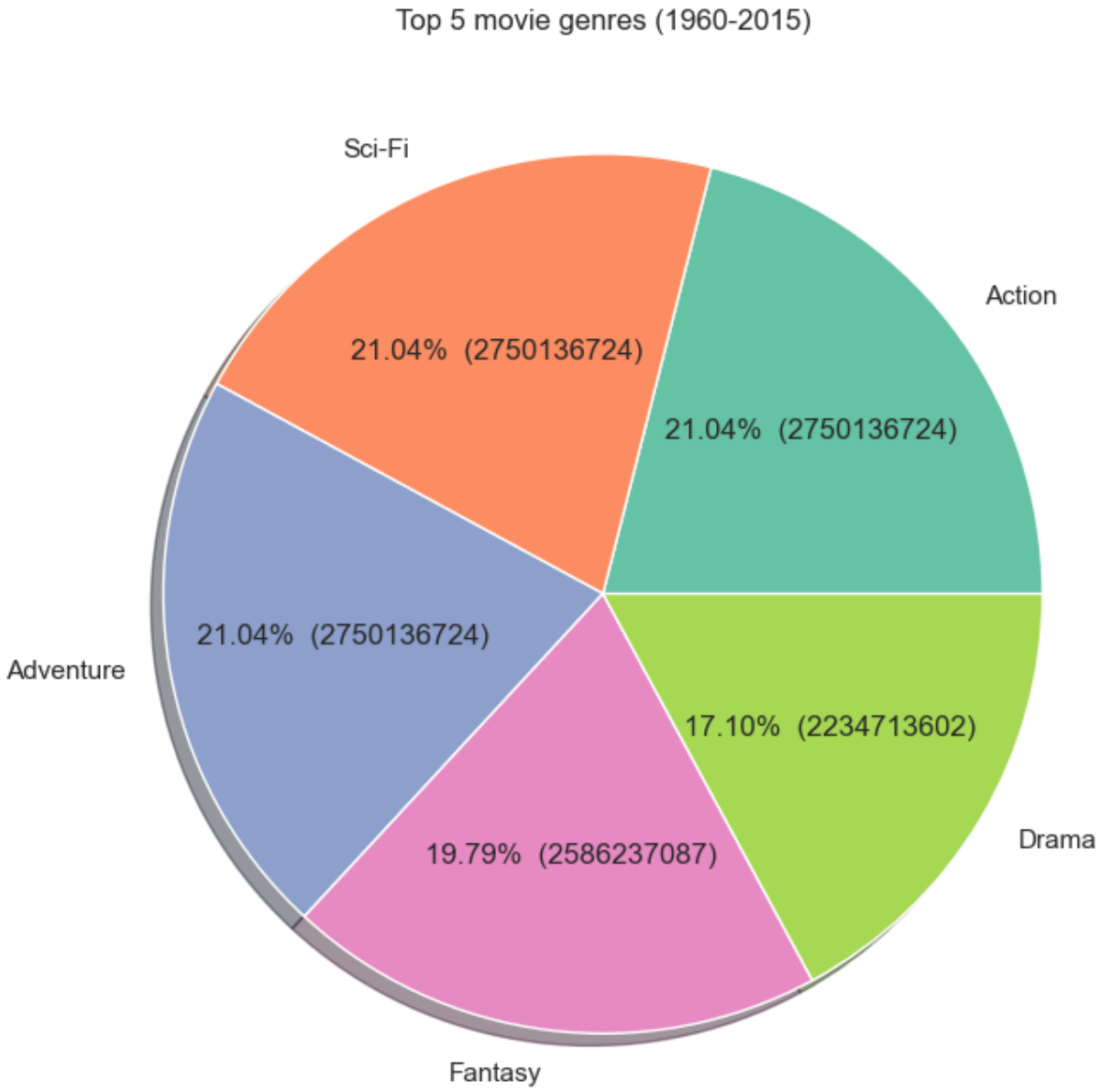
Q_5: Does spending more money on a movie mean higher revenue?

A_5:



Q_6: Which genres made the highest profit?

A_6: Science Fiction, Action, Adventure then Fantasy and drama.



Conclusions:

- The best financially-performing movies were -overall- made in the same successful genres (Action, Adventure, Science fiction, etc.). When done right, people will race to the theaters will watch them.
- Movie length tends to decrease overtime. Peaking at a little less than 125 minutes in the late 60s, and averaging around 100 minutes in mid 2010s. That might be because of the increase of amount of work that goes into the newer movies (CGI and visual effects) and other issues such as the writers' strike in 2007-2008 (that affected many movies namely the James Bond movie "Quantum of Solace" in 2008).
- The scatter plot shows that there isn't necessarily a strong relationship between spending and earning, the 1970s had -relatively- the lowest budgets yet the second highest earnings.
- From the bar plots, most financially successful movies aren't the same as the most popular movies, this is because most of the popular movies were released around 2014-2015 (the final year in the dataset). However, this is likely due to the popularity metrics being limited to TMDB only.

Limitations:

- The huge amount of missing data points in both revenue and budget columns, while they didn't affect the results, they definitely prevented us from reaching accurate ones.
- The dataset stops at 2015 -which is nearly a decade ago-. In those 9 years the movie scene has changed quite a bit (especially the last 5 years, post Avengers: Infinity war/Endgame era), so the outcomes could potentially change.
- There were some data entry errors, for example: the movie "The Net" -that shows up in the highest grossing movies- has a budget of $110M, but in the data frame it's registered as (1106279658) which is around $1.1B!

Appendix

The following code was used to obtain the results

```
#Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#Importing and exploring data
tmdb = pd.read_csv('Downloads/tmdb-movies.csv')
tmdb.head()
tmdb.describe()
tmdb.isna().sum()

#Data Preprocessing
#columns to drop (not needed in the analysis): [homepage, tagline, overview, imdb_id,
revenue, budget, id]
tmdb_clean = tmdb.drop(['homepage','tagline', 'overview', 'imdb_id', 'id','revenue',
'budget'], axis=1)

#Checking the percentage of budget and revenue that = 0
missing_tmdb_clean_percentages = []

for i in tmdb_clean[['budget_adj', 'revenue_adj', 'runtime']]:
    missing_tmdb_clean = tmdb_clean[tmdb_clean[i] == 0].shape[0]/tmdb_clean.shape[0]*100
    missing_tmdb_clean_percentages.append(missing_tmdb_clean)
    print("Percentage of missing monetary tmdb_clean is: " +
str(missing_tmdb_clean_percentages))

#More than half of our budegt and revenue data is missing! we can't omit more than 5% of
missing data, otherwise the statistics become biased.
#Replacing zeroes with NaN, since dropping them would alter the results too much.
for col in tmdb_clean[['budget_adj', 'revenue_adj']]:
    tmdb_clean[col] = tmdb_clean[col].replace(0, np.NaN)

#However, the percentage of zeroes in the runtime, cast, genres and director columns are
insignficant (<5%), so it's safe to drop the missing values.
na_drop = ['cast', 'genres', 'director']
tmdb_clean = tmdb_clean.dropna(subset = na_drop, how = 'any')
tmdb_clean = tmdb_clean[tmdb_clean['runtime'] != 0]

#Creating an adjusted profit column
tmdb_clean['profit_adj'] = tmdb_clean['revenue_adj'] - tmdb_clean['budget_adj']

#Extracting the main production company from production_companies
tmdb_clean['production_company'] =
tmdb_clean['production_companies'].str.split('|').str[0]
tmdb_clean = tmdb_clean.drop('production_companies', axis=1)
```

```
#Creating a decade column
tmdb_clean['decade'] = (tmdb_clean['release_year'] // 10) * 10

#Splitting the genres column
movies_genres = tmdb_clean['genres'].str.split('|', expand=True)
genres_combined = movies_genres.stack().droplevel(1).rename('genres')
tmdb_clean.drop(['genres'], axis=1, inplace=True)
tmdb_clean = pd.merge(tmdb_clean, genres_combined, left_index=True, right_index=True)

#Dropping duplicates
tmdb_clean=tmdb_clean.drop_duplicates()

#Data Analysis and Visualization
#What are the most profitable movies from 1960 to 2015?
successful_movies =
tmdb_clean.groupby('original_title')['profit_adj'].max().sort_values(ascending=False).head
(10)
print('The most proftiable movies are: ' + str(successful_movies))
ax = sns.barplot(successful_movies, orient='h', palette='YlGn_r')
ax.set(xlabel = "Profit (adjusted to 2010 inflation rates)", ylabel = "Movie Title")
ax.set_title('The most financially succsseful movies (1960-2015)')
plt.show()

#Box office failures
failed_movies =
tmdb_clean.groupby('original_title')['profit_adj'].max().sort_values(ascending=True).head(
10)
print('The most proftiable movies are: ' + str(failed_movies))
ax = sns.barplot(failed_movies, orient='h', palette='YlOrRd_r')
ax.set(xlabel = 'Loss (adjusted to 2010 inflation rates)', ylabel = 'Movie title')
ax.set_title('Box office failures (1960-2015)')
plt.show()

#What are the most popular movies from 1960 to 2015?
popular_movies =
tmdb_clean.groupby('original_title')['popularity'].max().sort_values(ascending=False).head
(10)
print('The most popular movies are: ' + str(popular_movies))
ax = sns.barplot(popular_movies, orient='h', palette='rainbow_r')
ax.set(xlabel = "Popularity rate", ylabel = "Movie title")
ax.set_title('The Most popular movies (1960-2015)')
plt.show()

#Did movies (on average) get longer or shorter overtime?
ax = sns.lineplot(data=tmdb_clean, x='release_year', y='runtime', errorbar=None,
color='b')
ax.set(xlabel = "Release year", ylabel = "Runtime (minutes)")
ax.set_title('Average movie runtime (in minutes) (1960-2015)')
plt.show()

#Does spending (on average) more result in gaining more?
bud_v_rev = tmdb_clean.groupby('decade').agg({'budget_adj':'mean',
                                        'revenue_adj':'mean'})
```

```
ax = sns.scatterplot(bud_v_rev, s=200, palette='viridis_r')
ax.set(xlabel = "Decade", ylabel = "Average Value")
handles, labels  =  ax.get_legend_handles_labels()
ax.legend(handles, ['Budget', 'Revenue'])
ax.set_title('Relationship between Budget and Revenue (1960-2015)')
plt.show()

#Which movie genres made the most profit?[1]
genre_profit =
tmdb_clean.groupby('genres')['profit_adj'].max().sort_values(ascending=False).head(5)
print('The genres that made the most profit are: ' + str(genre_profit))
labels = ['Action', 'Sci-Fi', 'Adventure', 'Fantasy', 'Drama']
def make_autopct(genre_profit):
    def my_autopct(pct):
        total = sum(genre_profit)
        val = int(round(pct*total/100.0))
        return '{p:.2f}%  ({v:d})'.format(p=pct,v=val)
    return my_autopct
plt.pie(genre_profit, labels=labels, autopct=make_autopct(genre_profit), shadow=True,
colors=sns.color_palette('Set2'))
plt.title('Top 5 movie genres (1960-2015)')
plt.show()
```

[1] make_autopct function from: https://stackoverflow.com/a/6170354