Project: Investigate a Dataset - TMDb movie data

Table of Contents

Introduction

Dataset Description

> Using the Movie Database(TMDb) with data from over 10,000 movies I will analyze the data to uncover trends among different movies and genres. This analysis will explore their popularity, runtimes, genres, and budget, as well as the correlations among these factors.

Question(s) for Analysis

> **Question 1**: What are the top 10 favorite movies?

> **Question 2**: What year was the highest budget movie produce?

> **Question 3**: What is the correlation between the average runtime of movies and the passage of time?

In [2]:

```
# Import packates
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Data Wrangling

General Properties

In [3]:

```
#load data
df = pd.read_csv('tmdb-movies.csv')

#view first 3 rows
df.head(3)
```

Out[3]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.

3 rows × 21 columns

In [4]:

```
#show size of dataframe
df.shape
```

Out[4]:

```
(10866, 21)
```

In [5]:

```
#dataset information

df.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
```

In [6]:

```
#data summary
df.describe()
```

Out[6]:

	id	popularity	budget	revenue	runtime	vote_count	v
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	1
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

In [7]:

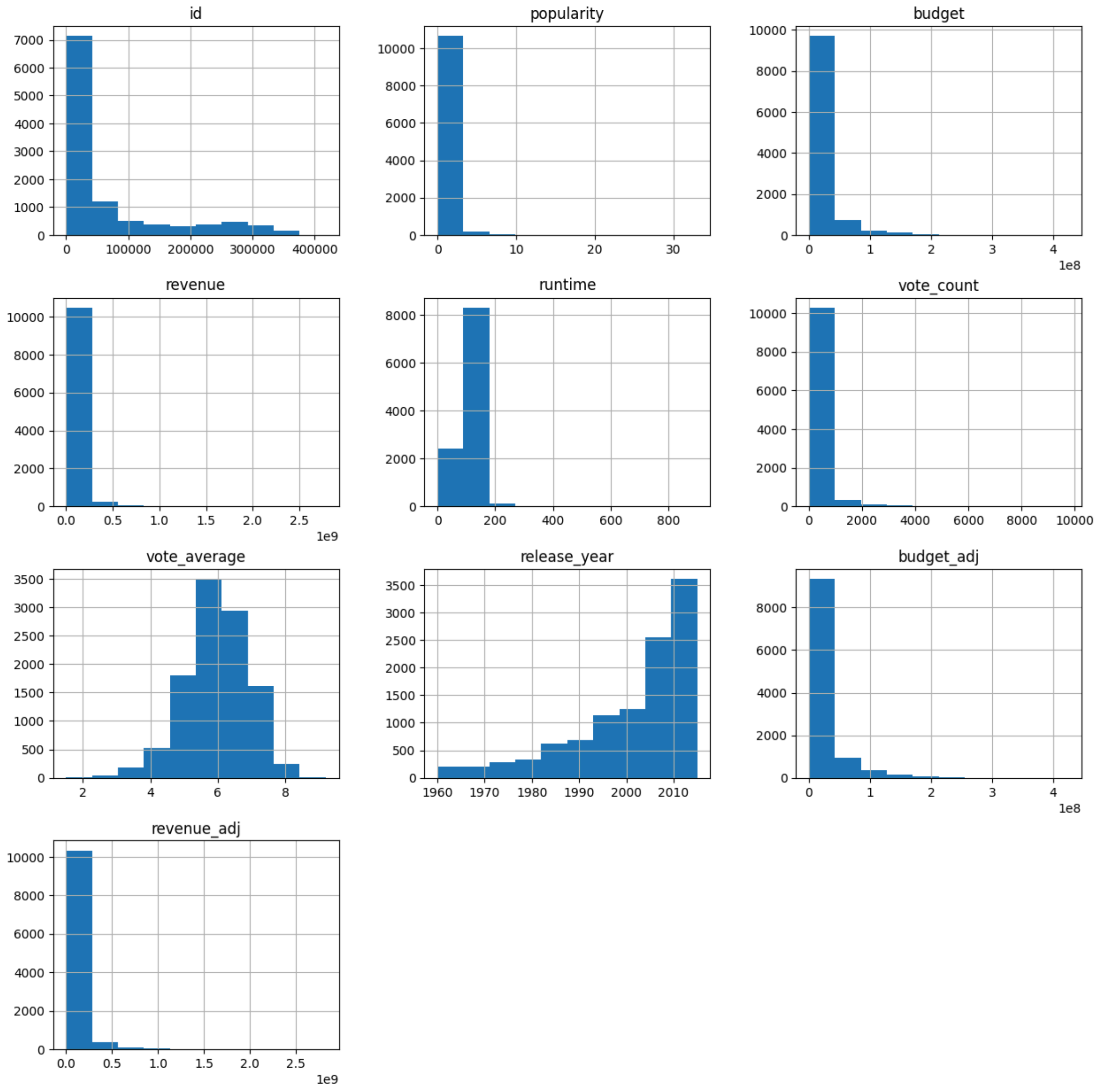
```
#show data types
df.nunique()
```

Out[7]:

```
id                      10865
imdb_id                 10855
popularity              10814
budget                    557
revenue                  4702
original_title          10571
cast                    10719
homepage                 2896
director                 5067
tagline                  7997
keywords                 8804
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj               2614
revenue_adj              4840
dtype: int64
```

```
In [8]: df.hist(figsize=(15,15))
```

```
Out[8]: array([[<AxesSubplot: title={'center': 'id'}>,
                <AxesSubplot: title={'center': 'popularity'}>,
                <AxesSubplot: title={'center': 'budget'}>],
               [<AxesSubplot: title={'center': 'revenue'}>,
                <AxesSubplot: title={'center': 'runtime'}>,
                <AxesSubplot: title={'center': 'vote_count'}>],
               [<AxesSubplot: title={'center': 'vote_average'}>,
                <AxesSubplot: title={'center': 'release_year'}>,
                <AxesSubplot: title={'center': 'budget_adj'}>],
               [<AxesSubplot: title={'center': 'revenue_adj'}>, <AxesSubplot: >,
                <AxesSubplot: >]], dtype=object)
```



Data Cleaning

In [9]:

```
#check for duplicates
df.duplicated().sum()
```

Out[9]:

```
1
```

In [10]:

```
#Drop duplicate rows found
df.drop_duplicates(inplace=True)

#check to make sure duplicates have been dropped
df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
#look for missing/NULL values
df.isna().sum()
```

Out[11]:

```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7929
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
```

In [16]:
```
#drop columns that are not necessary
df.drop(columns=['imdb_id','homepage','director','tagline','keywords','overvie
```

Out[16]:

	id	popularity	budget	revenue	original_title	cast	runtime	
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	124	Ac
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	120	Ac
2	262500	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	119	
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	136	Ac
4	168259	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	137	
...	...	...	...	...	...	...	...	
10861	21	0.080598	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	95	
10862	20379	0.065543	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	176	A
10863	39768	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	94	
10864	21449	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	80	
10865	22293	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	74	

10865 rows × 12 columns

In [12]:
```
print(df.columns)
```

```
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')
```

In [ ]:

```
#Remove rows that have '0' in Budget and revenue
```

Exploratory Data Analysis

Research Question 1:

> What are the top 10 favorite movies?

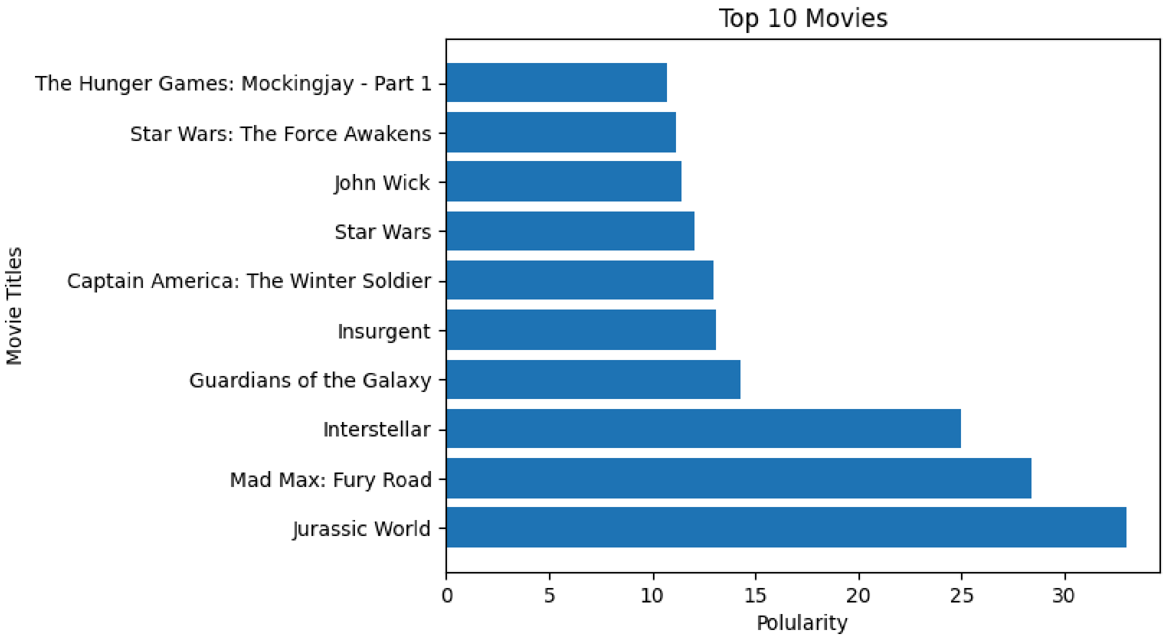
In [13]:

```
#top 10 popular movies
top_10_movies = df.sort_values(by='popularity', ascending=False).head(10)

# Display the top 10 favorite movies
print(top_10_movies[['original_title', 'popularity']])
```

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

In [15]:
```
#plot results
plt.barh(top_10_movies['original_title'], top_10_movies['popularity']);

#set title
plt.title('Top 10 Movies')

#set axis titles
plt.xlabel('Polularity')
plt.ylabel('Movie Titles')
```

Out[15]:
```
Text(0, 0.5, 'Movie Titles')
```



From the data shown you can find that Jurassic world is the leading top favorite movie

Research Question 2:

What year was the highest budget movie produced?

In [19]:

```
#top 10 popular movies
highest_budget = df.sort_values(by='budget', ascending=False).head(15)

# Display the top 10 favorite movies
print(highest_budget[['original_title','release_year','budget']])
```

```
                                    original_title  release_year     budget
2244                             The Warrior's Way          2010  425000000
3375   Pirates of the Caribbean: On Stranger Tides          2011  380000000
7387      Pirates of the Caribbean: At World's End          2007  300000000
14                         Avengers: Age of Ultron          2015  280000000
6570                              Superman Returns          2006  270000000
4411                                   John Carter          2012  260000000
1929                                       Tangled          2010  260000000
7394                                  Spider-Man 3          2007  258000000
5508                               The Lone Ranger          2013  255000000
4367             The Hobbit: An Unexpected Journey          2012  250000000
1923  Harry Potter and the Deathly Hallows: Part 1          2010  250000000
1389        Harry Potter and the Half-Blood Prince          2009  250000000
5431           The Hobbit: The Desolation of Smaug          2013  250000000
643                     X-Men: Days of Future Past          2014  250000000
4363                         The Dark Knight Rises          2012  250000000
```

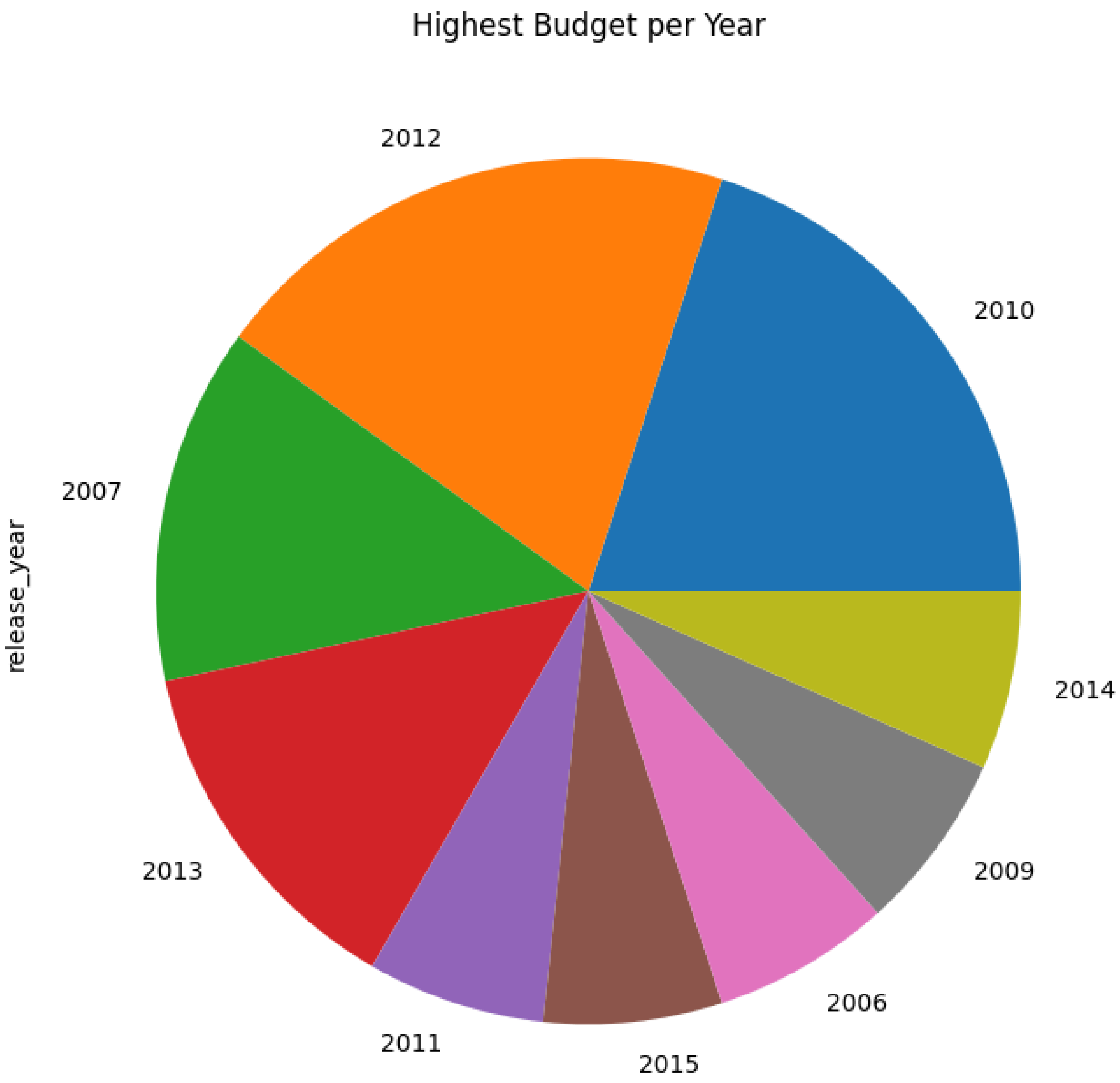
In [20]:

```
highest_budget_year = highest_budget.loc[highest_budget['budget'].idxmax(), 'r
print(f'Highest Budget Year is: {highest_budget_year}')
```

```
Highest Budget Year is: 2010
```

In [78]:
```
#plot years using pie chart to visually see what year had the highest budget

highest_budget['budget'],highest_budget['release_year'].value_counts().plot(ki
print(r)')
```

Out[78]:
```
Text(0.5, 1.0, 'Highest Budget per Year')
```

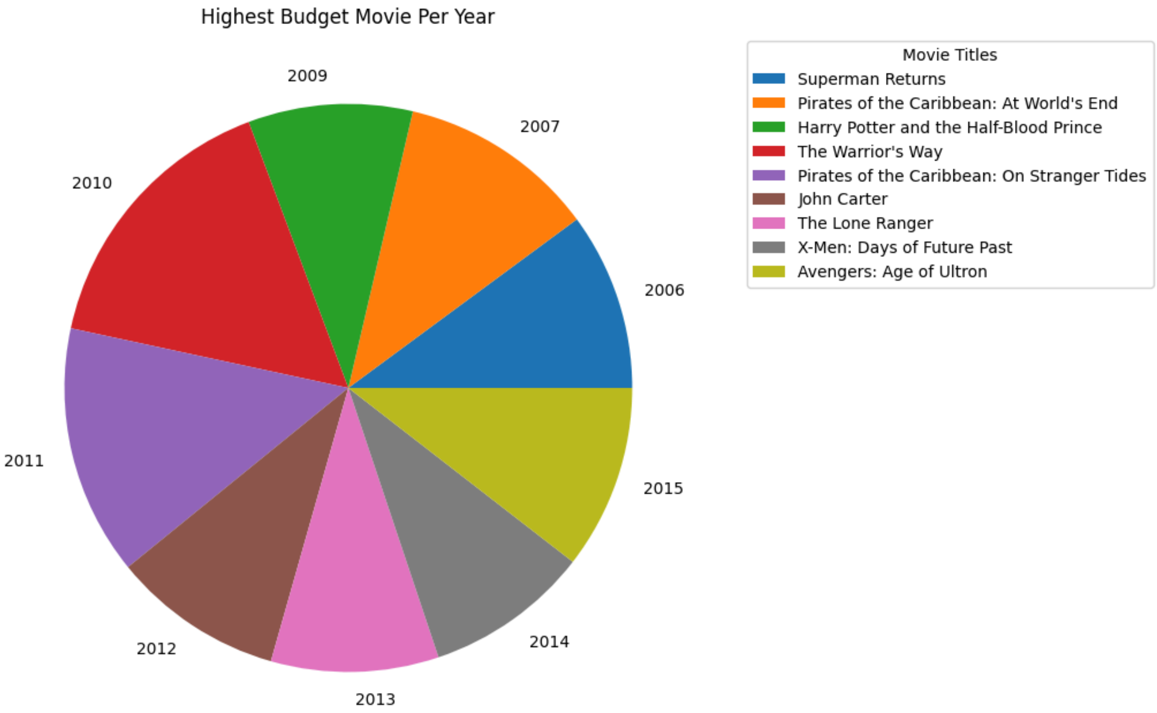


This chart shows a clear understanding of which year the highest budget was produced but after I made this Pie chart I realized without origional_title names it is difficult to tell which movie had the highest budget per year made.

I tried it another way to include a legend making the Pie chart easier to understand quickly while simultaneously revising this project to include a user-defined function as requested.

In [26]:

```
# Plot highest budget movie by release year with movie titles as legend
def plot_highest_budget_pie_chart(df):

    # Find the movie with the highest budget per year
    highest_budget_per_year = df.loc[df.groupby('release_year')['budget'].idxm

    # Create the pie chart for the highest budget by year
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.pie(highest_budget_per_year['budget'], labels=highest_budget_per_year['

    # Add a legend with the highest budget movie title for each year
    ax.legend(highest_budget_per_year['original_title'], title="Movie Titles",
              bbox_to_anchor=(1.05, 1), loc='best')

    # Display the chart
    plt.title('Highest Budget Movie Per Year')
    plt.show()

# Call the function
plot_highest_budget_pie_chart(highest_budget)
```



From analyzing the data we can find that in 2010 the movie 'The Warrior's Way' had the largest budget in the amount of 425,000,000 USD which is almost 100,000,000 USD more than the next highest budget movie made.

Research Question 3:

> What is the correlation between the average runtime of movies and the passage of time?

In [81]:

```
#find the average movie runtime
df['runtime'].mean()
```

Out[81]:

```
102.07086324314375
```

In [87]:

```
#find the minimun movie runtime
df['runtime'].min()
```

Out[87]:

```
0
```

In [88]:

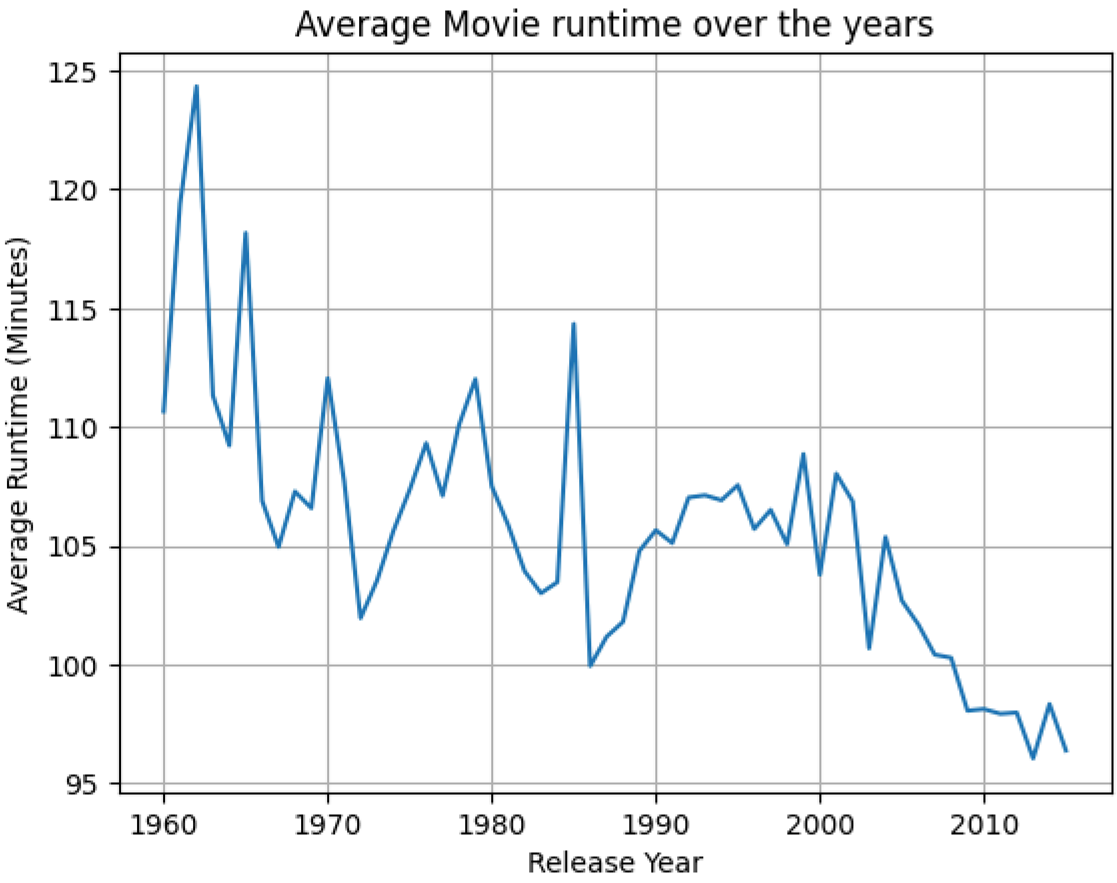
```
#find the maximum movie runtime
df['runtime'].max()
```

Out[88]:

```
900
```

In [86]:

```python
#Group by 'release_year' and calculate the average 'runtime'
average_runtime_per_year=df.groupby('release_year')['runtime'].mean()

#Plot a line chart
plt.plot(average_runtime_per_year.index, average_runtime_per_year.values)

#Add labels ad title
plt.xlabel('Release Year')
plt.ylabel('Average Runtime (Minutes)')
plt.title('Average Movie runtime over the years')

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



Conclusions

Question 1: What are the top 10 favorite movies?

> Jurassic world is the top favorite movie of all times.

Question 2: What year was the highest budget movie produce?

> The Warrior's Way was the movie that had the largest budget and was made in 2010

Question 3: What is the correlation between the average runtime of movies and the passage of time?

> The included graph of movie runtimes shows a correlation between the length of movies and time, indicating that movies are getting shorter over the years.

Side Note:

> I found the material for this project quite challenging and it took me a considerable amount of time to complete. I wish there had been a step-by-step video guide on getting started and navigating the workspaces before beginning. Loading the dataframe at the start was particularly difficult. However, after spending many hours experimenting in the workspaces, I began to enjoy the process and ultimately feel that I learned a lot.

Limitations

> After I reviewed the data a second time I realized that I should probably check the min and max runtimes when trying to find a correlation between the average runtime of movies and time. This is where I realized that some of the data may in fact be short clips, with little to no time. Or a long TV series with extra long run times which lead me to believe I may not have an accurate

In [27]:

```
# Running this cell will execute a bash command to convert this notebook to ar
!python -m nbconvert --to html Investigate_a_Dataset.ipynb
```

```
[NbConvertApp] WARNING | pattern 'Investigate_a_Dataset_revised.ipynb' mat
ched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
```

In [ ]: