

Investigating TMDb dataset

Introduction

TMDb dataset is a dataset that contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

The columns selected for the context of this report are: "id", "popularity", "budget", "revenue", "cast", "genres"

Data Wrangling

General Properties

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [2]: df = pd.read_csv('./tmdb-movies.csv')
df.head()
```

```
Out[2]:
```

| | 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.thed |
| 3 | 140607 | tt2488496 | 11.173104 | 200000000 | 2068178225 | Star Wars: The Force Awakens | Harrison Ford Mark Hamill Carrie Fisher Adam D... | http://ww |

| | id | imdb_id | popularity | budget | revenue | original_title | cast |
|---|--------|-----------|------------|-----------|------------|----------------|--|
| 4 | 168259 | tt2820852 | 9.335014 | 190000000 | 1506249360 | Furious 7 | Vin Diesel Paul Walker Jason Statham Michelle ... |

5 rows × 21 columns



In [3]: `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 [4]: `df.isnull().sum()`

```
Out[4]: 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
```

```
In [5]: df.dtypes
```

```
Out[5]: 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
```

```
In [6]: df.describe()
```

```
Out[6]:
```

| | id | popularity | budget | revenue | runtime | vote_count | vote_aver |
|-------|---------------|--------------|--------------|--------------|--------------|--------------|-----------|
| count | 10866.000000 | 10866.000000 | 1.086600e+04 | 1.086600e+04 | 10866.000000 | 10866.000000 | 10866.000 |
| mean | 66064.177434 | 0.646441 | 1.462570e+07 | 3.982332e+07 | 102.070863 | 217.389748 | 5.974 |
| std | 92130.136561 | 1.000185 | 3.091321e+07 | 1.170035e+08 | 31.381405 | 575.619058 | 0.935 |
| min | 5.000000 | 0.000065 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 10.000000 | 1.500 |
| 25% | 10596.250000 | 0.207583 | 0.000000e+00 | 0.000000e+00 | 90.000000 | 17.000000 | 5.400 |
| 50% | 20669.000000 | 0.383856 | 0.000000e+00 | 0.000000e+00 | 99.000000 | 38.000000 | 6.000 |
| 75% | 75610.000000 | 0.713817 | 1.500000e+07 | 2.400000e+07 | 111.000000 | 145.750000 | 6.600 |
| max | 417859.000000 | 32.985763 | 4.250000e+08 | 2.781506e+09 | 900.000000 | 9767.000000 | 9.200 |

Remove unnecessary columns

```
In [7]: df.drop(['imdb_id', 'homepage', 'director', 'overview', 'tagline', 'keywords',
                'budget_adj', 'revenue_adj', 'production_companies', 'original_title', 'release_d
                axis=1, inplace=True)
df.head()
```

```
Out[7]:
```

| | id | popularity | budget | revenue | cast | runtime | genres | vote_c |
|--|----|------------|--------|---------|------|---------|--------|--------|
|--|----|------------|--------|---------|------|---------|--------|--------|

| | id | popularity | budget | revenue | cast | runtime | genres | vote_c |
|---|--------|------------|-----------|------------|---|---------|---|--------|
| 0 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | Action Adventure Science Fiction Thriller | |
| 1 | 76341 | 28.419936 | 150000000 | 378436354 | Tom Hardy Charlize Theron Hugh Keays-Byrne Nic... | 120 | Action Adventure Science Fiction Thriller | |
| 2 | 262500 | 13.112507 | 110000000 | 295238201 | Shailene Woodley Theo James Kate Winslet Ansel... | 119 | Adventure Science Fiction Thriller | |
| 3 | 140607 | 11.173104 | 200000000 | 2068178225 | Harrison Ford Mark Hamill Carrie Fisher Adam D... | 136 | Action Adventure Science Fiction Fantasy | |
| 4 | 168259 | 9.335014 | 190000000 | 1506249360 | Vin Diesel Paul Walker Jason Statham Michelle ... | 137 | Action Crime Thriller | |

Remove rows with null values

```
In [8]: df.isna().sum()
```

```
Out[8]: id                0
popularity              0
budget                 0
revenue               0
cast                  76
runtime               0
genres                23
vote_count             0
vote_average           0
release_year          0
dtype: int64
```

```
In [9]: df.dropna(inplace=True)
df.isna().sum()
```

```
Out[9]: id                0
popularity              0
budget                 0
revenue               0
cast                  0
runtime               0
genres                0
vote_count             0
vote_average           0
release_year          0
dtype: int64
```

```
In [10]: df.shape
```

Out[10]: (10768, 10)

Remove duplicates

```
In [11]: df.drop_duplicates(inplace=True)
df.shape
```

Out[11]: (10767, 10)

Fixing the genres column

The genres column contains the genres of each movie separated with "|". To make it easier to work with, the genres cell in each row will be transformed to be a list with genres names then it will be spreaded over multiple rows with one genre per row for each movie in another dataframe

Change the genres field to an array:

```
In [12]: df['genres'] = df['genres'].apply(lambda x: x.split('|'))
```

```
In [13]: df.head(1)
```

Out[13]:

| | id | popularity | budget | revenue | cast | runtime | genres | vote_count | vote_ave |
|---|--------|------------|-----------|------------|---|---------|--|------------|----------|
| 0 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | |

```
In [14]: genres_df = pd.DataFrame(df['genres'].tolist(), index=df['id']).stack()
genres_df = genres_df.reset_index([0, 'id'])
genres_df.columns=['id', 'genres']
genres_df.head()
```

Out[14]:

| | id | genres |
|---|--------|-----------------|
| 0 | 135397 | Action |
| 1 | 135397 | Adventure |
| 2 | 135397 | Science Fiction |
| 3 | 135397 | Thriller |
| 4 | 76341 | Action |

```
In [15]: df_copy = df.copy()
df_copy.drop(['genres'], axis=1, inplace=True)
genres_df = pd.merge(df_copy, genres_df, how='inner', left_on='id', right_on='id')
genres_df.head()
```

Out[15]:

| | id | popularity | budget | revenue | cast | runtime | vote_count | vote_average | release_date |
|--|----|------------|--------|---------|------|---------|------------|--------------|--------------|
|--|----|------------|--------|---------|------|---------|------------|--------------|--------------|

| | id | popularity | budget | revenue | cast | runtime | vote_count | vote_average | release_date |
|---|--------|------------|-----------|------------|---|---------|------------|--------------|--------------|
| 0 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | 5562 | 6.5 | |
| 1 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | 5562 | 6.5 | |
| 2 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | 5562 | 6.5 | |
| 3 | 135397 | 32.985763 | 150000000 | 1513528810 | Chris Pratt Bryce Dallas Howard Irrfan Khan Vi... | 124 | 5562 | 6.5 | |
| 4 | 76341 | 28.419936 | 150000000 | 378436354 | Tom Hardy Charlize Theron Hugh Keays-Byrne Nic... | 120 | 6185 | 7.1 | |

Fixing the cast column

The same process on the genres will be applied on the cast

```
In [16]: df['cast'] = df['cast'].apply(lambda x: x.split('|'))
df.head(1)
```

| | id | popularity | budget | revenue | cast | runtime | genres | vote_count | vote_average |
|---|--------|------------|-----------|------------|---|---------|--|------------|--------------|
| 0 | 135397 | 32.985763 | 150000000 | 1513528810 | [Chris Pratt, Bryce Dallas Howard, Irrfan Khan... | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 |

```
In [17]: cast_df = pd.DataFrame(df['cast'].tolist(), index=df['id'].stack())
cast_df = cast_df.reset_index([0, 'id'])
cast_df.columns=['id', 'cast']
cast_df.head()
```

| | id | cast |
|--|----|------|
|--|----|------|

| | id | cast |
|---|--------|---------------------|
| 0 | 135397 | Chris Pratt |
| 1 | 135397 | Bryce Dallas Howard |
| 2 | 135397 | Irrfan Khan |
| 3 | 135397 | Vincent D'Onofrio |
| 4 | 135397 | Nick Robinson |

```
In [18]: df_copy = df.copy()
df_copy.drop(['cast'], axis=1, inplace=True)
cast_df = pd.merge(df_copy, cast_df, how='inner', left_on='id', right_on='id')
cast_df.head()
```

```
Out[18]:
```

| | id | popularity | budget | revenue | runtime | genres | vote_count | vote_average | release_y |
|---|--------|------------|-----------|------------|---------|--|------------|--------------|-----------|
| 0 | 135397 | 32.985763 | 150000000 | 1513528810 | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 | 2 |
| 1 | 135397 | 32.985763 | 150000000 | 1513528810 | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 | 2 |
| 2 | 135397 | 32.985763 | 150000000 | 1513528810 | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 | 2 |
| 3 | 135397 | 32.985763 | 150000000 | 1513528810 | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 | 2 |
| 4 | 135397 | 32.985763 | 150000000 | 1513528810 | 124 | [Action, Adventure, Science Fiction, Thriller] | 5562 | 6.5 | 2 |

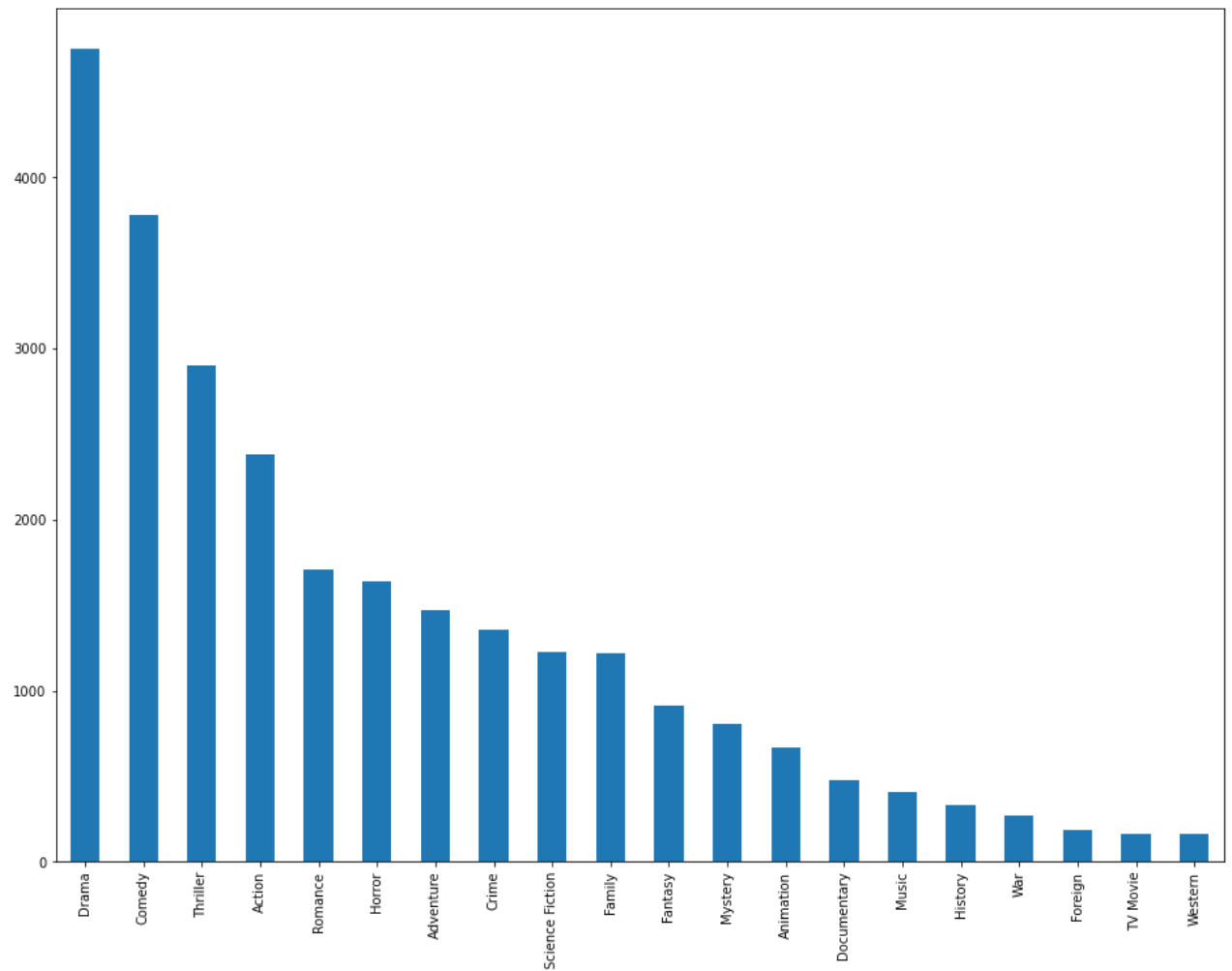
Each of the cast and genres dataframes are saved separately away from the main dataframe to not influence the result by the duplications in the other columns

EDA

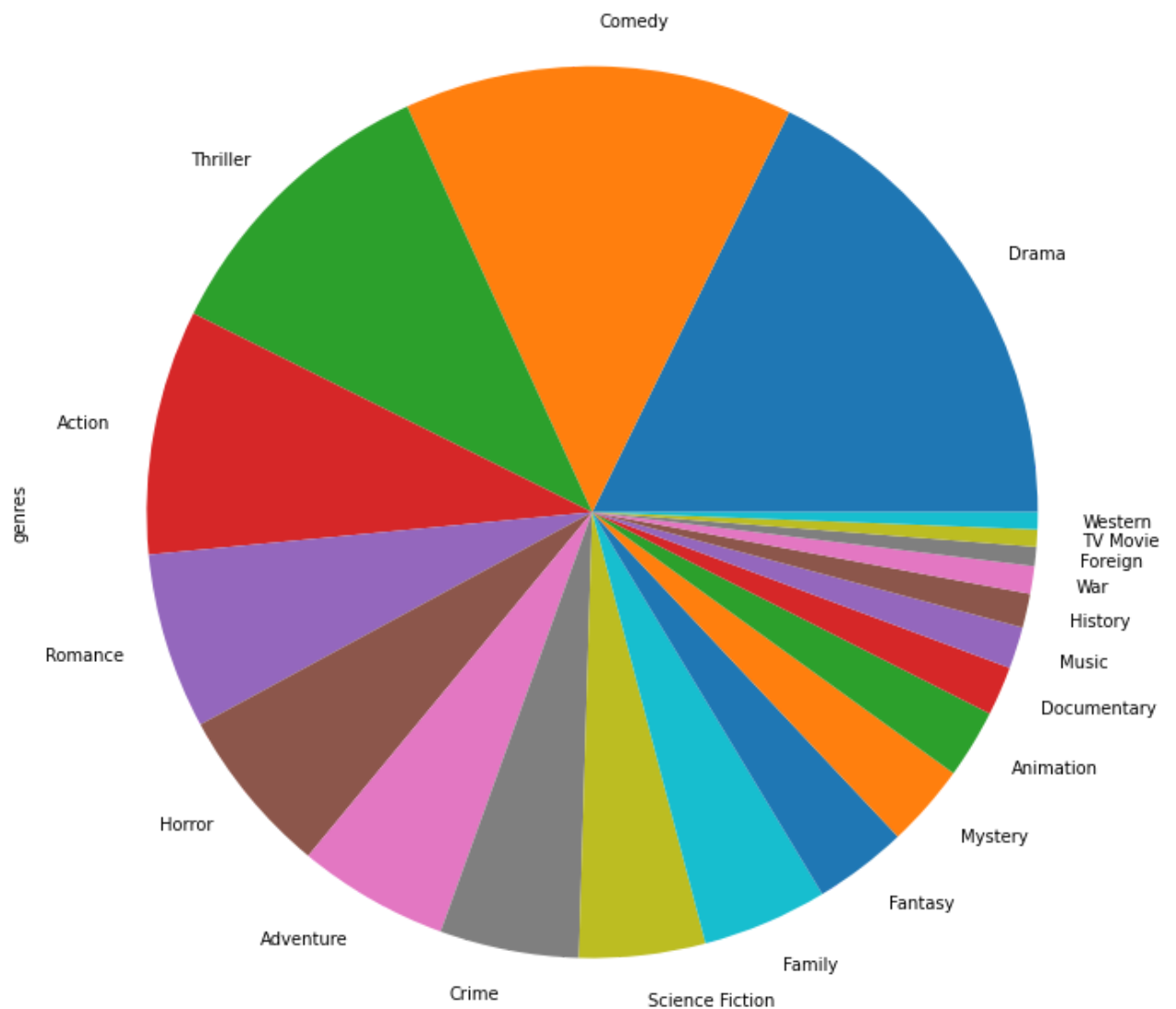
Time to ask questions, find conclusions, and have some fun!

1. What are the most used genres in movies?

```
In [19]: genres_df['genres'].value_counts().plot(kind='bar', figsize=(16,12))  
plt.show()
```



```
In [20]: genres_df['genres'].value_counts().plot(kind='pie', figsize=(16,12))  
plt.show()
```

The most used genres are Drama, Comedy, Thriller, and Action. While TV movie, War, Western, and Foreign, and other genres are lightly produced.

2. What is the average movie budget for each genre?

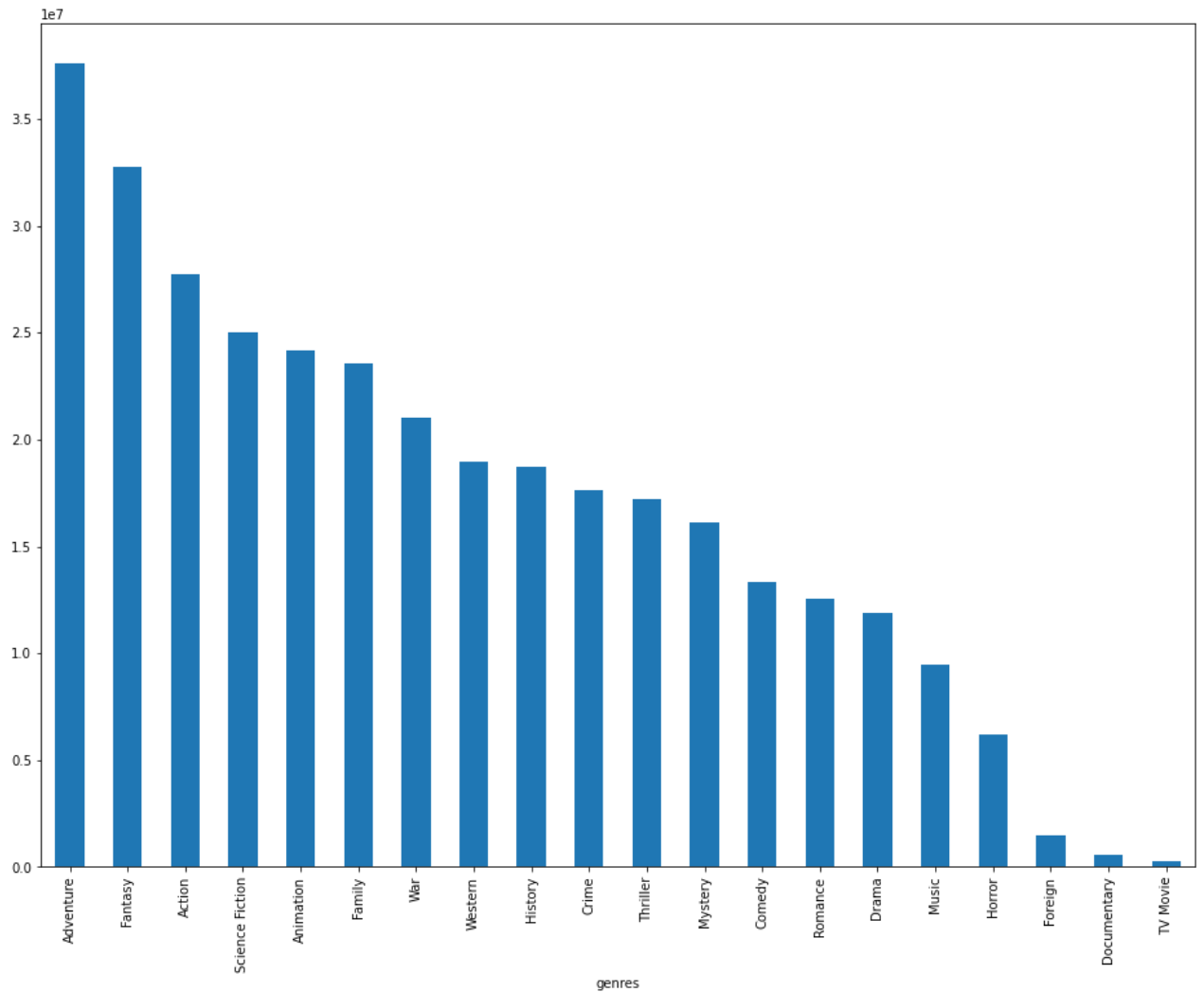
```
In [21]: avg_budget = genres_df.groupby('genres').mean()['budget']

# round the numbers
avg_budget = avg_budget.apply(lambda x: round(x))

# sort the values
avg_budget = avg_budget.sort_values(ascending=False)
avg_budget.head()
```

```
Out[21]: genres
Adventure      37594809
Fantasy        32791579
Action         27762757
Science Fiction 25013386
Animation      24194674
Name: budget, dtype: int64
```

```
In [22]: avg_budget.plot(kind='bar', figsize=(16,12))
plt.show()
```



The most expensive genres are Adventure, Fantasy, Family, History, and Science Fiction. While TV Movie, Foreign, Documentary movies are the least expensive.

3. What is the average movie revenue for each genre?

```
In [23]: avg_revenue = genres_df.groupby('genres').mean()['revenue']

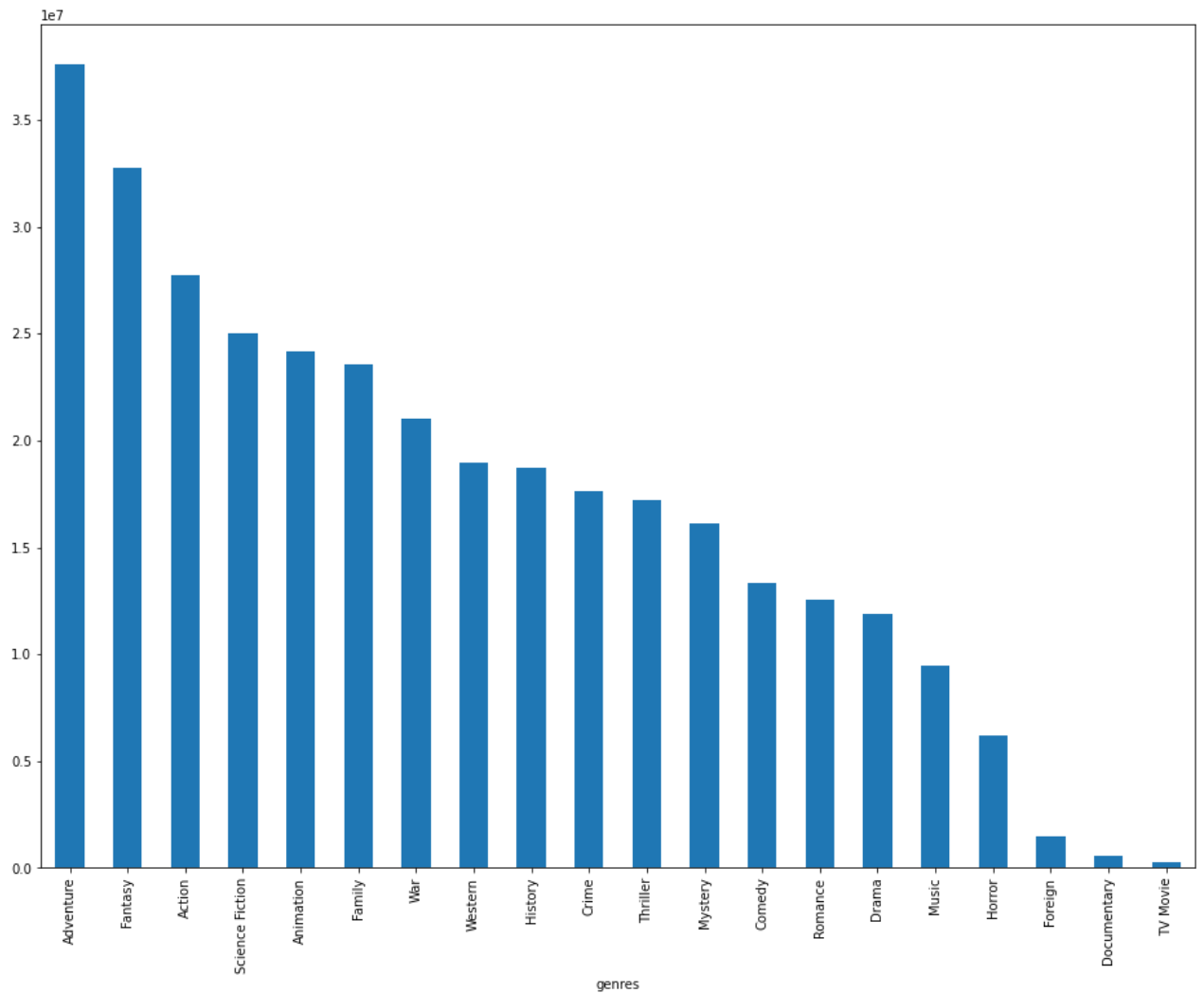
# round the numbers
avg_revenue = avg_revenue.apply(lambda x: round(x))

# sort the values
avg_revenue = avg_revenue.sort_values(ascending=False)
avg_revenue.head()
```

```
Out[23]: genres
Adventure    113291895
Fantasy      96842272
Animation    78630774
Family       73146218
Action       72886452
Name: revenue, dtype: int64
```

```
In [24]: avg_budget.plot(kind='bar', figsize=(16,12))
```

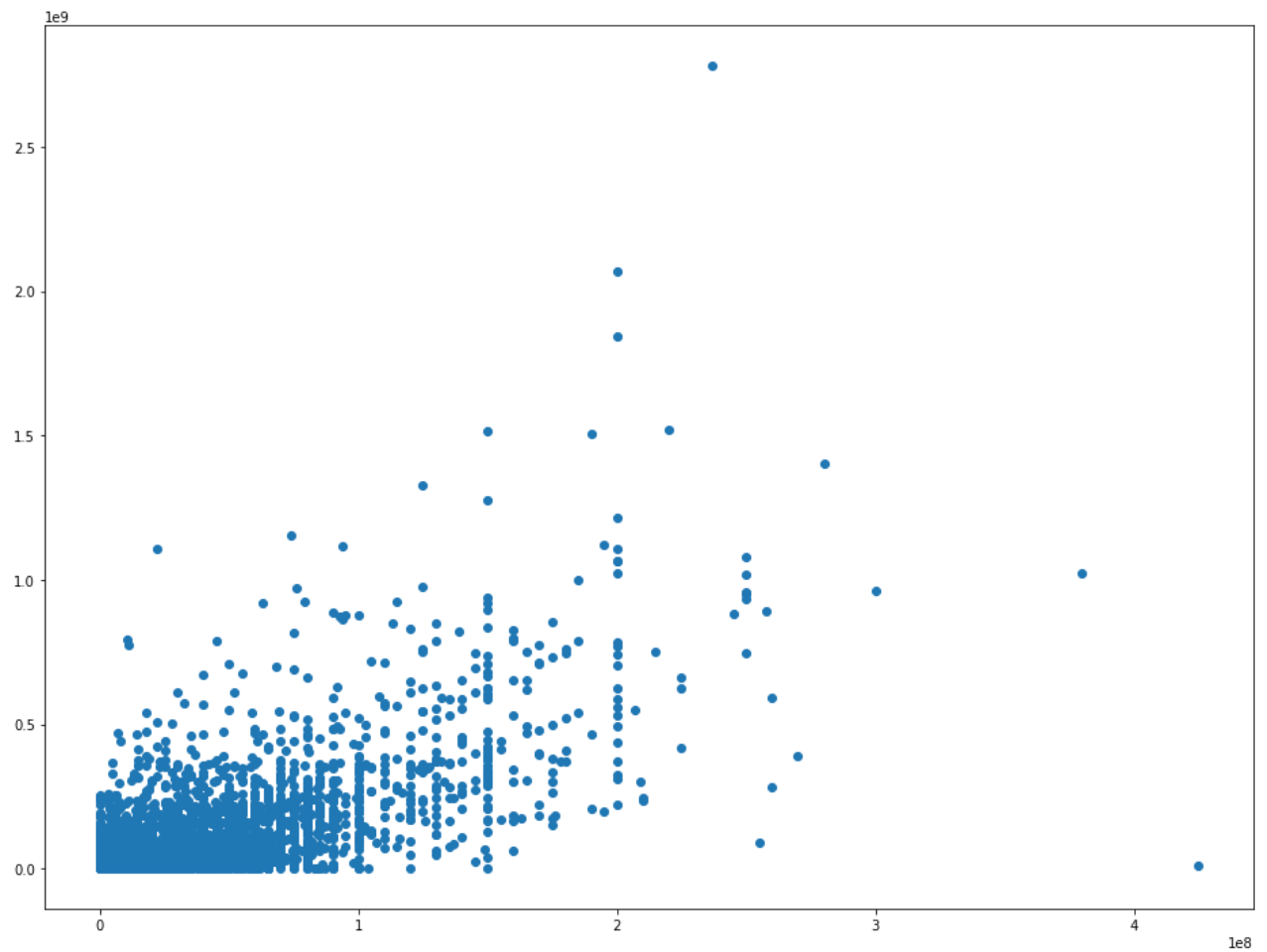
```
plt.show()
```



The genres that generates highest revenues are Adventure, Action, Fantasy, and Family. While Documentary movies don't generate much of a revenue.

4. Does the budget influence the revenue?

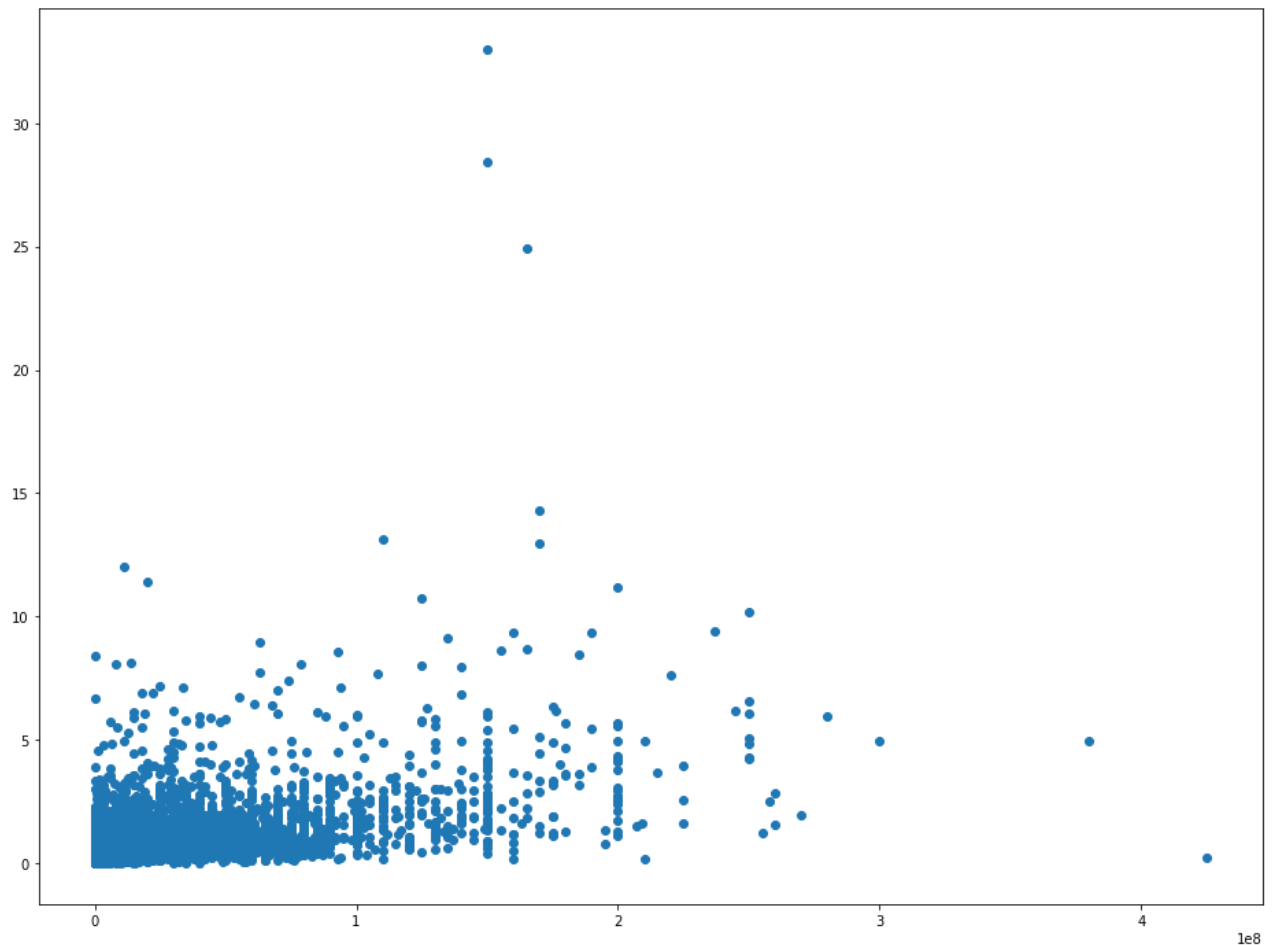
```
In [25]: plt.figure(figsize=(16,12))
plt.scatter(df['budget'], df['revenue'])
plt.show()
```



It appears that there is a very low positive correlation between the budget spent and the revenue gained. In other words, budget doesn't have that large influence on the revenue.

5. Does the budget spent influence the popularity?

```
In [26]: plt.figure(figsize=(16,12))  
plt.scatter(df['budget'], df['popularity'])  
plt.show()
```



Again, although the budget spent have small influence on the popularity, it still not a large influence to consider.

6. How the cast influence the revenue

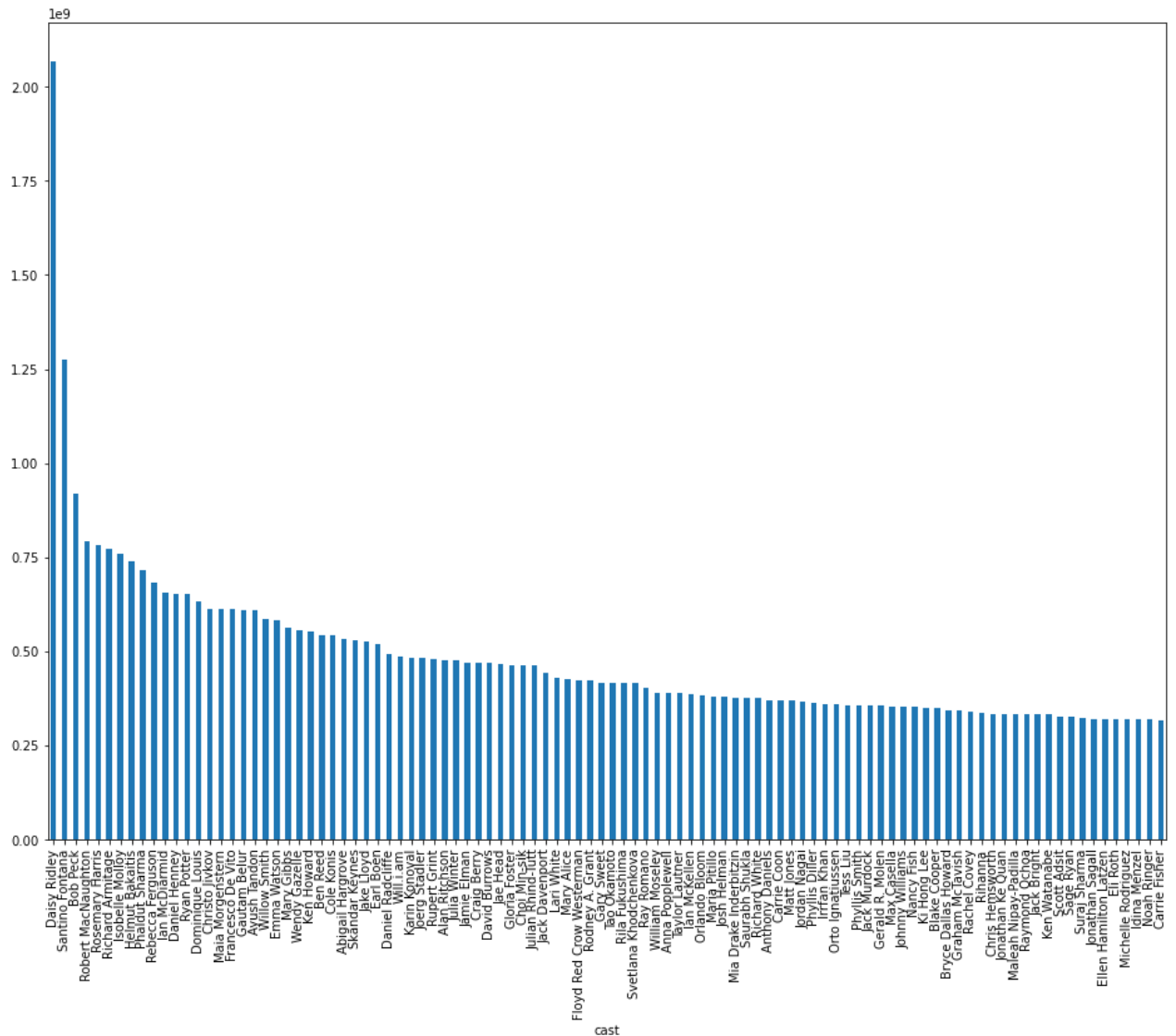
```
In [27]: avg_cast_revenue = cast_df.groupby('cast').mean()['revenue']

# round the numbers
avg_cast_revenue = avg_cast_revenue.apply(lambda x: round(x))

# sort the values
avg_cast_revenue = avg_cast_revenue.sort_values(ascending=False)
avg_cast_revenue.head()
```

```
Out[27]: cast
Daisy Ridley          2068178225
Santino Fontana       1274219009
Bob Peck              920100000
Robert MacNaughton    792910554
Rosemary Harris       783766341
Name: revenue, dtype: int64
```

```
In [28]: avg_cast_revenue[:100].plot(kind='bar', figsize=(16,12))
plt.show()
```



Cast members rarely have effect on the revenue of the movies they participate in. Daisy Ridley had the highest effect on the revenue of the movies she participated in while other cast members had similar effect on the movies they participate in.

Conclusions

- The most used genres are Drama, Comedy, Thriller, and Action with the Drama on top. People tend to be "Drama" queens
- The highest budget spent on Adventure, Fantasy, Family, and History.
- The highest revenue making genres are the ones most produced. That didn't need a data analyst to find out.
- Budget spent has low influence on both revenues on popularity. Money doesn't buy everything after all. At least not the popularity of the movies.
- Cast members has no influence on the revenues. It turned out to be just a myth.