Project: Investigate IMDB Movie Dataset

By Mohammed Barakah

Table of Contents

Introduction

Dataset: IMDB Movie Dataset

Dataset Description: This dataset collects information from 10866 movies, and its budget and revenue and if it's popular or not and also did the critics like or not.

Questions:

- `1` : is there any correlation between popularity and budget?
- `2` : if the movie gets a high vote average does this mean high popularity?
- `3` : What are the highest genres of movies?

.

- `id` : indicates the movie ID
- `imdb_id` : indicates movie ID, this field is unique
- `budget` : indicates the budget for the movie
- `revenue` : indicates the revenue that movie generated
- `original_title` : indicates the the name of the movie
- `cast` : indicates the cast that perform in this movie
- `homepage` : indicates the website of the movie.
- `director` : indicates the name of the director
- `tagline` : indicates a catchphrase or slogan for the movie
- `keywords` : indicates the keyword used to describe the movie
- `overview` : specific description for the movie
- `runtime` : indicates the time from start to finish, in minutes
- `genres` : indicates a style or category of the movie
- `production_companies` : indicates the company that produce the movie
- `release_date` : indicates a the release date of the movie, d/m/y
- `vote_count` : indicates how many people vote for the movie
- `vote_average` : indicates the average score out of 10, 10 means they actually like it
- `release_year` : indicates the release date, only year
- `budget_adj` : indicates the budget after adjustment to the inflation

- `revenue_adj` : indicates the revenue after adjustment to the inflation

In [105…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme()
%matplotlib inline

pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

In [106…
```
df = pd.read_csv('G:/My Drive/CS/Data analsys/udacity/DA Project#2/Data/tmdb-movies.csv'
```

Data Wrangling

General Properties

First we are going to see what the dataset looks like by seeing the last 5 rows.

In [107…
```
df.tail(5)
```

Out[107]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	directo
10861	21	tt0060371	0.08	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	NaN	Bruce Brow
10862	20379	tt0060472	0.07	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	NaN	Joh Frankenheim
10863	39768	tt0060161	0.07	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	NaN	Eld Ryazanc
10864	21449	tt0061177	0.06	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	NaN	Woody Alle
10865	22293	tt0060666	0.04	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	NaN	Harold Warre

And by looking at the data type we don't need to change anything, at least for now. And we can see that this dataset has 10866 rows, and 21 columns All of these columns are lower case and clear to read, we don't

have to change anything for now.

In [108…
```
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 [109…
```
df.isnull().sum()
```

Out[109]:
```
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
```

We can see here that there is a lot of null values, but at this early stage i don't think that these missing values are important, Such as homepage "official movie website", tagline, keywords and production company

In [110…
```
df.describe()
```

Out[110]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	bu
count	10866.00	10866.00	10866.00	10866.00	10866.00	10866.00	10866.00	10866.00	
mean	66064.18	0.65	14625701.09	39823319.79	102.07	217.39	5.97	2001.32	17
std	92130.14	1.00	30913213.83	117003486.58	31.38	575.62	0.94	12.81	34
min	5.00	0.00	0.00	0.00	0.00	10.00	1.50	1960.00	
25%	10596.25	0.21	0.00	0.00	90.00	17.00	5.40	1995.00	
50%	20669.00	0.38	0.00	0.00	99.00	38.00	6.00	2006.00	
75%	75610.00	0.71	15000000.00	24000000.00	111.00	145.75	6.60	2011.00	20
max	417859.00	32.99	425000000.00	2781505847.00	900.00	9767.00	9.20	2015.00	425

I'm not interested with some of these columns ['homepage', 'tagline', 'keywords', 'overview', 'budget_adj', 'revenue_adj'] They provide no insight, or value so we are going to delete these columns using the drop function.

In [111…

```
# drop columns
df.drop(['homepage', 'tagline', 'keywords', 'overview', 'budget_adj', 'revenue_adj', 'pr
```

And now let's check again for null values.

In [112…

```
df.isnull().sum()
```

Out[112]:

```
id                 0
imdb_id           10
popularity         0
budget             0
revenue            0
original_title     0
cast              76
director          44
runtime            0
genres            23
release_date       0
vote_count         0
vote_average       0
release_year       0
dtype: int64
```

We see here that we have duplicated movies, so we are going to delete one of them with the id 2089.

In [113…

```
df[df.duplicated(keep=False)]
```

Out[113]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	
2089	42194	tt0411951	0.60	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	Dwight H. Little	92	Crime|Dran
2090	42194	tt0411951	0.60	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	Dwight H. Little	92	Crime|Dran

In [114…

```
df.drop([2089], axis=0, inplace=True)
```

In [115…
```
df.describe()
```

Out[115]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	10865.00	10865.00	10865.00	10865.00	10865.00	10865.00	10865.00	10865.00
mean	66066.37	0.65	14624286.06	39826896.08	102.07	217.40	5.98	2001.32
std	92134.09	1.00	30914284.61	117008277.46	31.38	575.64	0.94	12.81
min	5.00	0.00	0.00	0.00	0.00	10.00	1.50	1960.00
25%	10596.00	0.21	0.00	0.00	90.00	17.00	5.40	1995.00
50%	20662.00	0.38	0.00	0.00	99.00	38.00	6.00	2006.00
75%	75612.00	0.71	15000000.00	24000000.00	111.00	146.00	6.60	2011.00
max	417859.00	32.99	425000000.00	2781505847.00	900.00	9767.00	9.20	2015.00

There are 76 null values in the cast column, so we are going to see how they look by creating a new dataset with only cast null values.

In [116…
```
cast_null = df.query('cast.isnull()')
cast_null.head(5)
```

Out[116]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	genres	relea
371	345637	tt4661600	0.42	0	0	Sanjay's Super Team	NaN	Sanjay Patel	7	Animation	
441	355020	tt4908644	0.22	0	0	Winter on Fire: Ukraine's Fight for Freedom	NaN	Evgeny Afineevsky	98	Documentary	
465	321109	tt4393514	0.20	0	0	Bitter Lake	NaN	Adam Curtis	135	Documentary	
536	333350	tt3762974	0.12	0	0	A Faster Horse	NaN	David Gelb	90	Documentary	
538	224972	tt3983674	0.11	0	0	The Mask You Live In	NaN	Jennifer Siebel Newsom	88	Documentary	

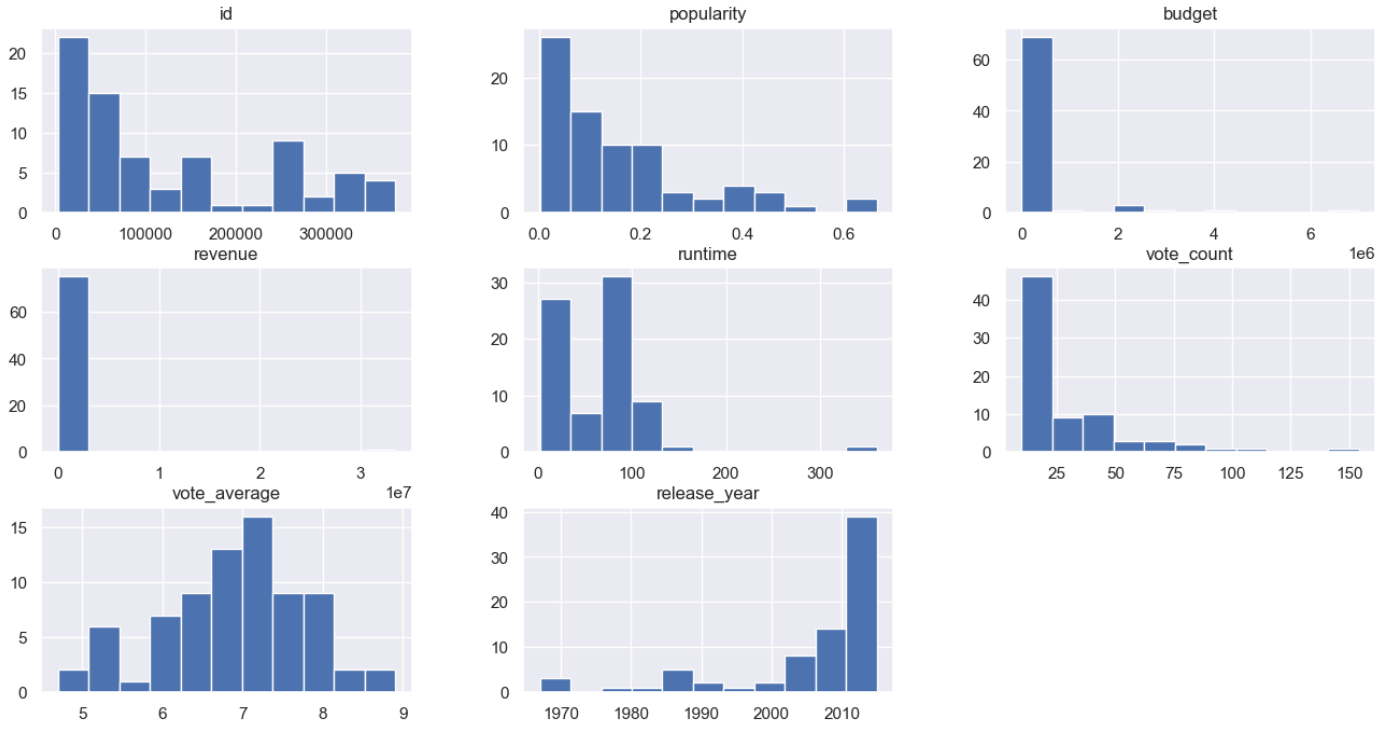
In [117…
```
cast_null.describe()
```

Out[117]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	76.00	76.00	76.00	76.00	76.00	76.00	76.00	76.00
mean	125506.53	0.15	294162.07	467265.37	63.03	29.01	6.88	2005.93
std	115445.52	0.16	1071112.54	3832448.61	53.76	25.64	0.89	11.68
min	3171.00	0.00	0.00	0.00	2.00	10.00	4.70	1967.00
25%	23685.00	0.03	0.00	0.00	7.75	12.00	6.47	2005.75
50%	77621.50	0.09	0.00	0.00	74.50	18.50	6.95	2011.00
75%	242661.25	0.21	0.00	0.00	93.25	37.25	7.53	2014.00
max	376823.00	0.66	7000000.00	33400000.00	360.00	154.00	8.90	2015.00

We can see that there are 76 rows with null values and 75% of budget and revenue are 0, so we are going to drop any cast with null values, they provide no insight.

In [118…
```
cast_null.hist(figsize=[16,8] , bins=11);
```



In [119…
```
# drop any cast with null values
df = df[df['cast'].notna()]
```

We see a lot of movies that have a budget less than 100, And these are not realistic values for movie budgets, we are going to investigate more by creating a new dataset with only a budget less than 100.

In [120…
```
odd_numrical_budget = df[df['budget'] < 100]
odd_numrical_budget['budget'].value_counts()
```

Out[120]:
```
0      5631
10        6
1         4
30        3
3         3
8         3
25        2
12        2
18        2
15        2
21        1
95        1
89        1
2         1
6         1
27        1
90        1
14        1
28        1
17        1
32        1
68        1
20        1
97        1
93        1
```

```
80        1
75        1
5         1
11        1
Name: budget, dtype: int64
```

In [121…
```
len(odd_numrical_budget)
```

Out[121]:
```
5677
```

In [122…
```
odd_numrical_budget.describe()
```

Out[122]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	5677.00	5677.00	5677.00	5677.00	5677.00	5677.00	5677.00	5677.00
mean	84032.05	0.33	0.21	3117140.43	97.99	44.09	5.91	2001.34
std	102874.11	0.31	3.50	14180405.98	36.41	70.34	0.97	13.68
min	6.00	0.00	0.00	0.00	0.00	10.00	1.50	1960.00
25%	14286.00	0.15	0.00	0.00	88.00	13.00	5.30	1994.00
50%	29756.00	0.27	0.00	0.00	95.00	21.00	6.00	2007.00
75%	121674.00	0.43	0.00	0.00	106.00	44.00	6.60	2012.00
max	409696.00	8.41	97.00	253625427.00	900.00	1329.00	9.20	2015.00

As we can see here, we have 5631 rows with budgets 0, and 4663 rows with revenues 0, and 75% popularity 0.43, 75% vote average 6.60. So we are going to delete these rows, they are not accurate.

In [123…
```
df.describe()
```

Out[123]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	10789.00	10789.00	10789.00	10789.00	10789.00	10789.00	10789.00	10789.00
mean	65647.67	0.65	14725230.49	40104153.65	102.35	218.73	5.97	2001.29
std	91819.23	1.00	30999368.92	117372462.85	31.00	577.45	0.93	12.82
min	5.00	0.00	0.00	0.00	0.00	10.00	1.50	1960.00
25%	10567.00	0.21	0.00	0.00	90.00	17.00	5.40	1995.00
50%	20497.00	0.39	0.00	0.00	99.00	38.00	6.00	2006.00
75%	74726.00	0.72	16000000.00	24637469.00	112.00	147.00	6.60	2011.00
max	417859.00	32.99	425000000.00	2781505847.00	900.00	9767.00	9.20	2015.00

In [124…
```
df = df[df.budget > 100]
df.describe()
```

Out[124]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00
mean	45231.36	1.00	31077955.90	81179129.01	107.19	412.66	6.03	2001.23
std	72455.90	1.33	38987579.08	160149504.05	22.61	791.72	0.88	11.79
min	5.00	0.00	108.00	0.00	0.00	10.00	1.50	1960.00
25%	8802.50	0.35	6000000.00	0.00	93.00	36.00	5.50	1996.00

50%	12735.50	0.63	18000000.00	21809825.00	103.00	125.00	6.10	2005.00
75%	44772.00	1.16	40000000.00	90567612.75	117.00	406.25	6.60	2010.00
max	417859.00	32.99	425000000.00	2781505847.00	540.00	9767.00	8.40	2015.00

Now we have 5132 rows, but we have another problem, some of these values are not formatted such as (5.112000e+03) and we don't want it to give us wrong insight, So we are going to change the format using the pandas function.

In [125…
```
df.head(5)
```

Out[125]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	
0	135397	tt0369610	32.99	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124	Acti
1	76341	tt1392190	28.42	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	120	Acti
2	262500	tt2908446	13.11	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	119	
3	140607	tt2488496	11.17	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	136	Acti
4	168259	tt2820852	9.34	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	137	

In [126…
```
pd.options.display.float_format = '{:.2f}'.format
df.describe()
```

Out[126]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00	5112.00
mean	45231.36	1.00	31077955.90	81179129.01	107.19	412.66	6.03	2001.23
std	72455.90	1.33	38987579.08	160149504.05	22.61	791.72	0.88	11.79
min	5.00	0.00	108.00	0.00	0.00	10.00	1.50	1960.00
25%	8802.50	0.35	6000000.00	0.00	93.00	36.00	5.50	1996.00
50%	12735.50	0.63	18000000.00	21809825.00	103.00	125.00	6.10	2005.00
75%	44772.00	1.16	40000000.00	90567612.75	117.00	406.25	6.60	2010.00
max	417859.00	32.99	425000000.00	2781505847.00	540.00	9767.00	8.40	2015.00

In [127…
```
df.info()
```

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

```
Int64Index: 5112 entries, 0 to 10865
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              5112 non-null   int64  
 1   imdb_id         5111 non-null   object 
 2   popularity      5112 non-null   float64
 3   budget          5112 non-null   int64  
 4   revenue         5112 non-null   int64  
 5   original_title  5112 non-null   object 
 6   cast            5112 non-null   object 
 7   director        5108 non-null   object 
 8   runtime         5112 non-null   int64  
 9   genres          5111 non-null   object 
 10  release_date    5112 non-null   object 
 11  vote_count      5112 non-null   int64  
 12  vote_average    5112 non-null   float64
 13  release_year    5112 non-null   int64  
dtypes: float64(2), int64(6), object(6)
memory usage: 599.1+ KB
```

In [128…
```
df.isnull().sum()
```

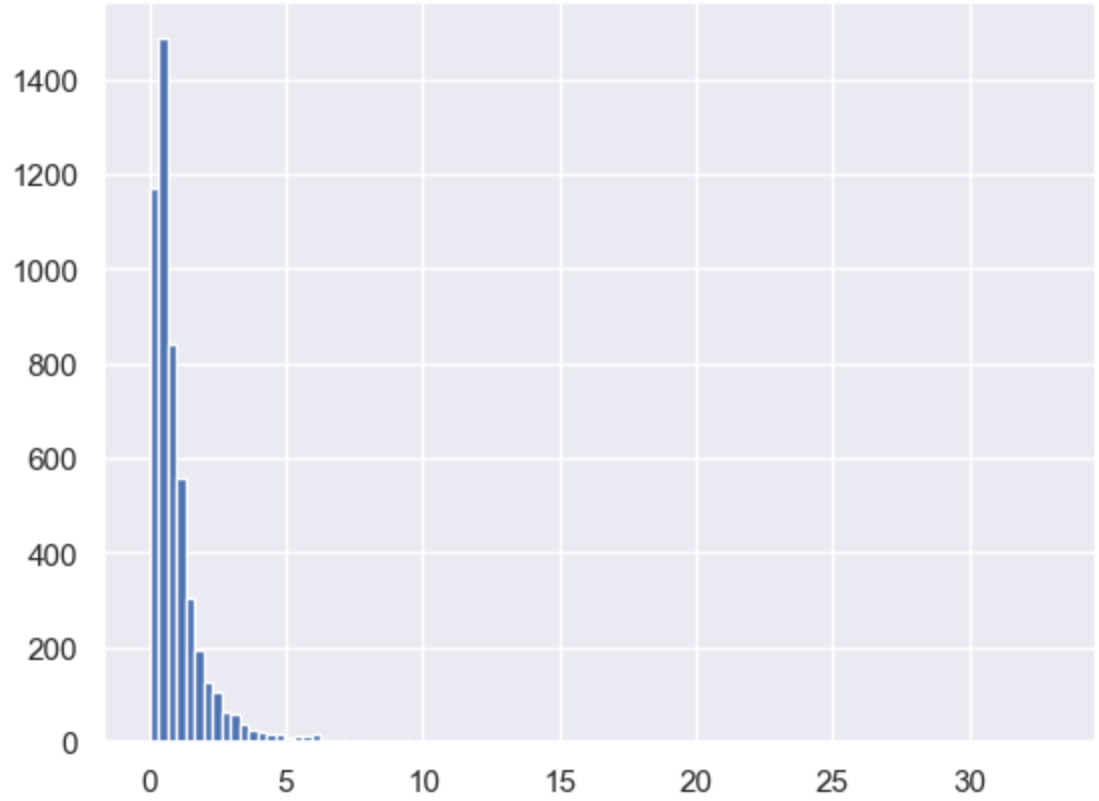
Out[128]:
```
id                0
imdb_id           1
popularity        0
budget            0
revenue           0
original_title    0
cast              0
director          4
runtime           0
genres            1
release_date      0
vote_count        0
vote_average      0
release_year      0
dtype: int64
```

Now we fix most of the problem and are almost done with cleaning.

Exploratory Data Analysis

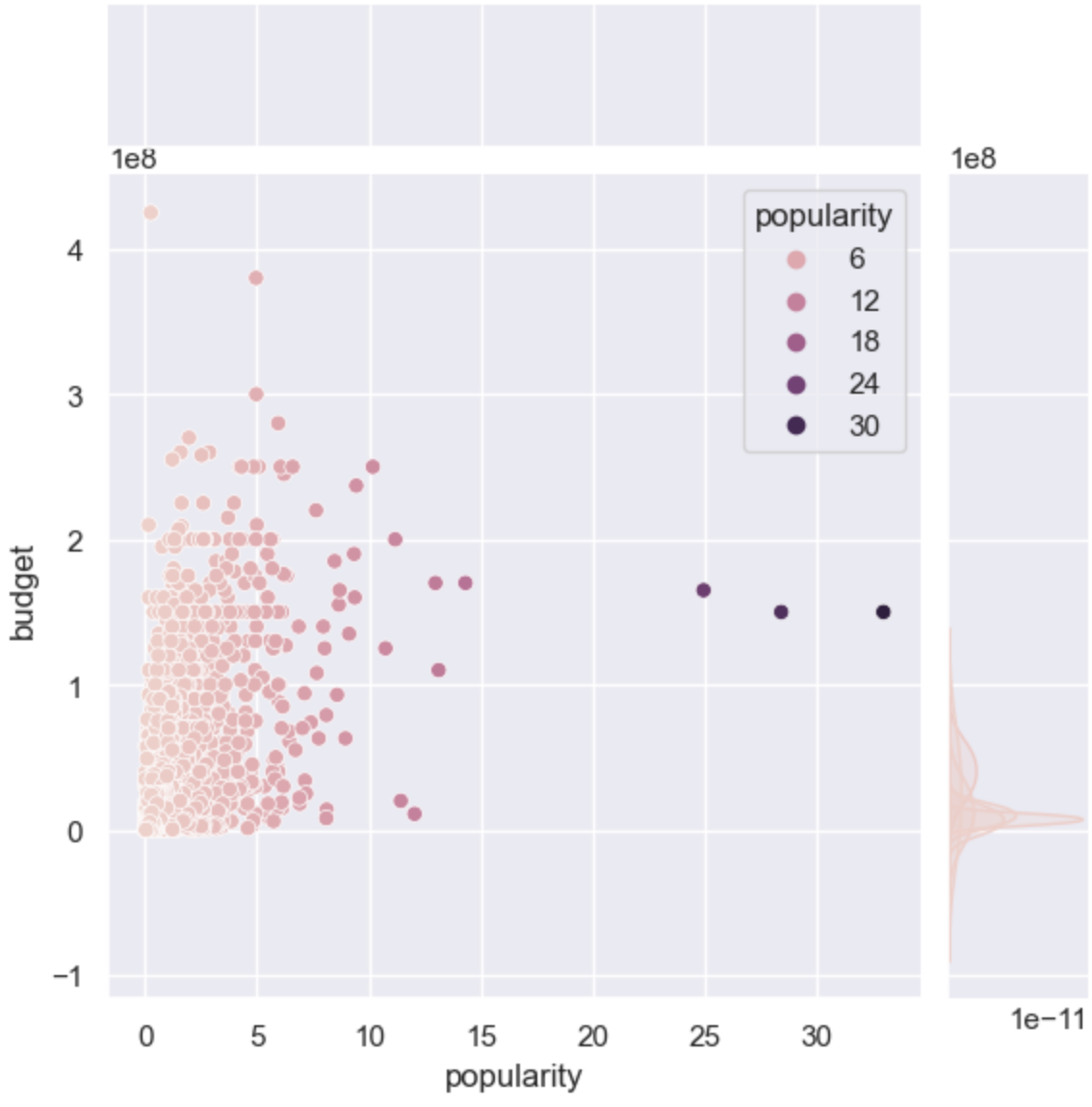
Question 1: is there any correlation between popularity and budget?

In [129…
```
df.popularity.hist(bins=100);
```



In [130…

```
sns.jointplot(data=df, x="popularity", y="budget", hue="popularity");
```

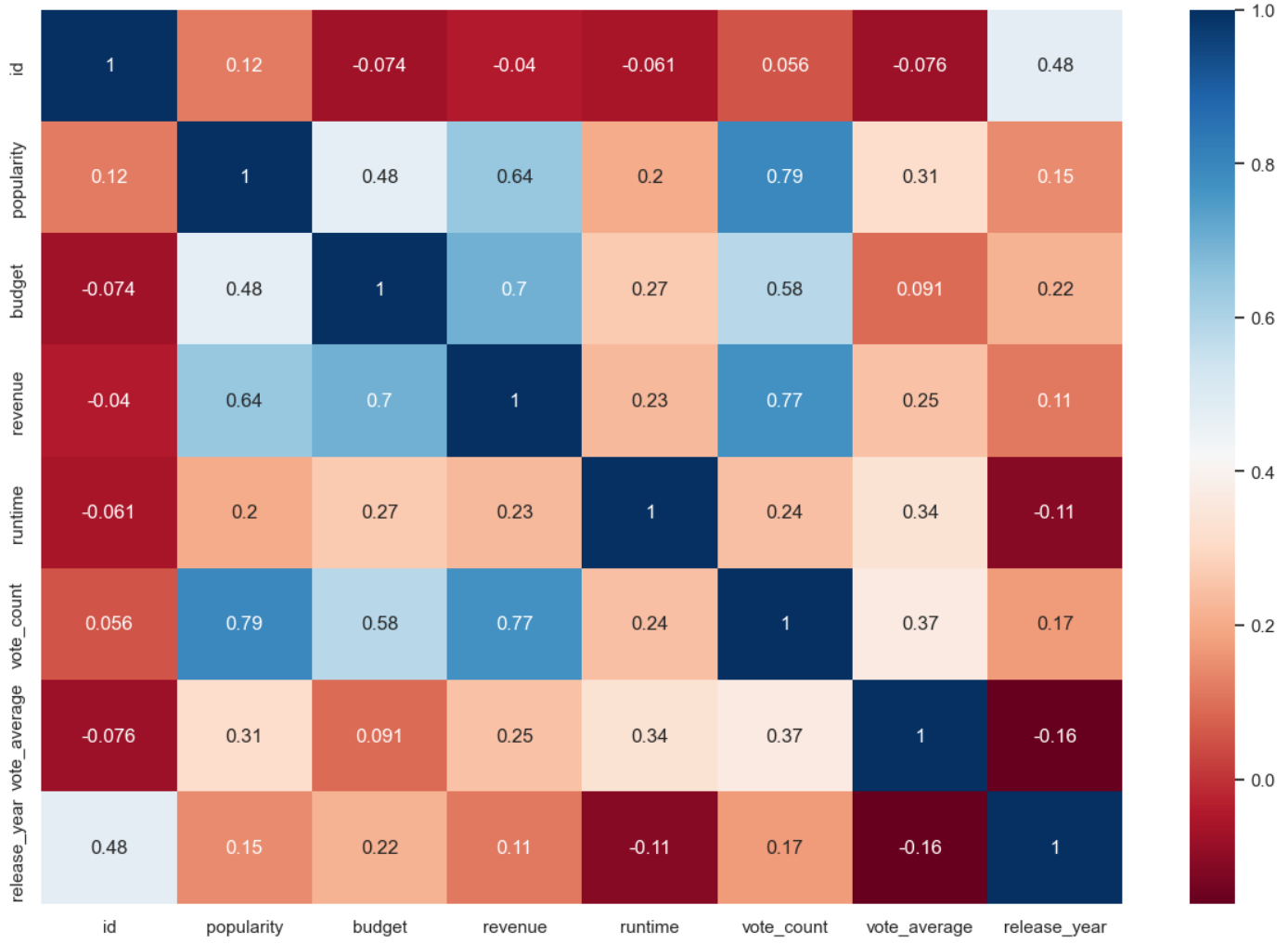


We can see from the above visualization that very few movies get popularity higher than 15, most of the movies in the range of 0 to 10. Also there is a correlation between budget and high popularity.

In [131…

```
# popularity    budget  revenue runtime vote_count      vote_average    release_year

fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(df.corr(), ax=ax, annot=True, cmap="RdBu");
```

```
C:\Users\lpkmy\AppData\Local\Temp\ipykernel_9556\3240713052.py:4: FutureWarning: The def
ault value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  sns.heatmap(df.corr(), ax=ax, annot=True, cmap="RdBu");
```

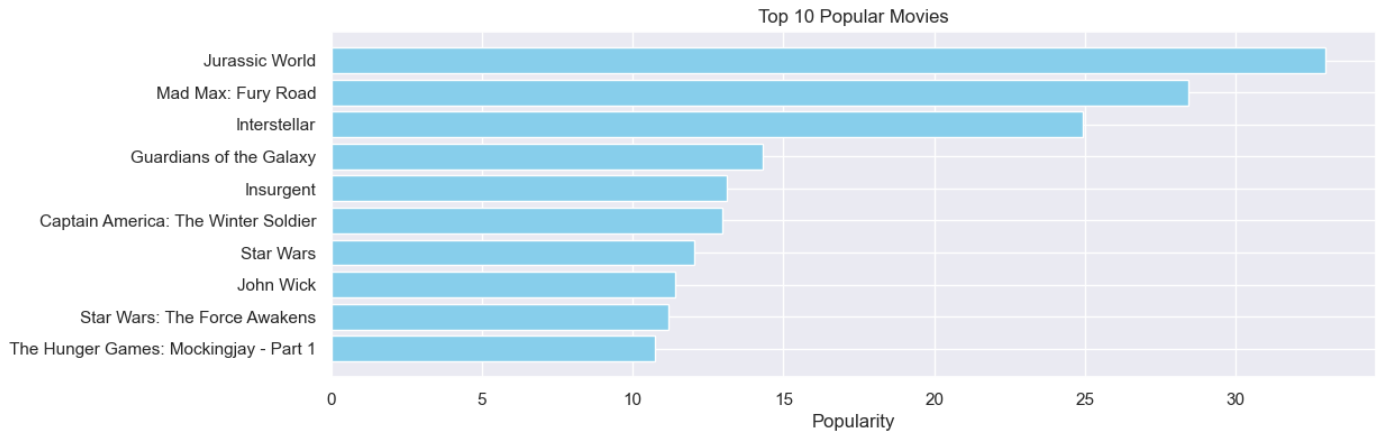


From the above heatmap we can see there is high correlation between popularity and revenue and that makes sense, also between popularity and budget the correlation value is 0.48, also the vote count “how many people vote” also makes sense, because if the movie is popular that will lead to many people knowing about the movie and vote.

In [132…

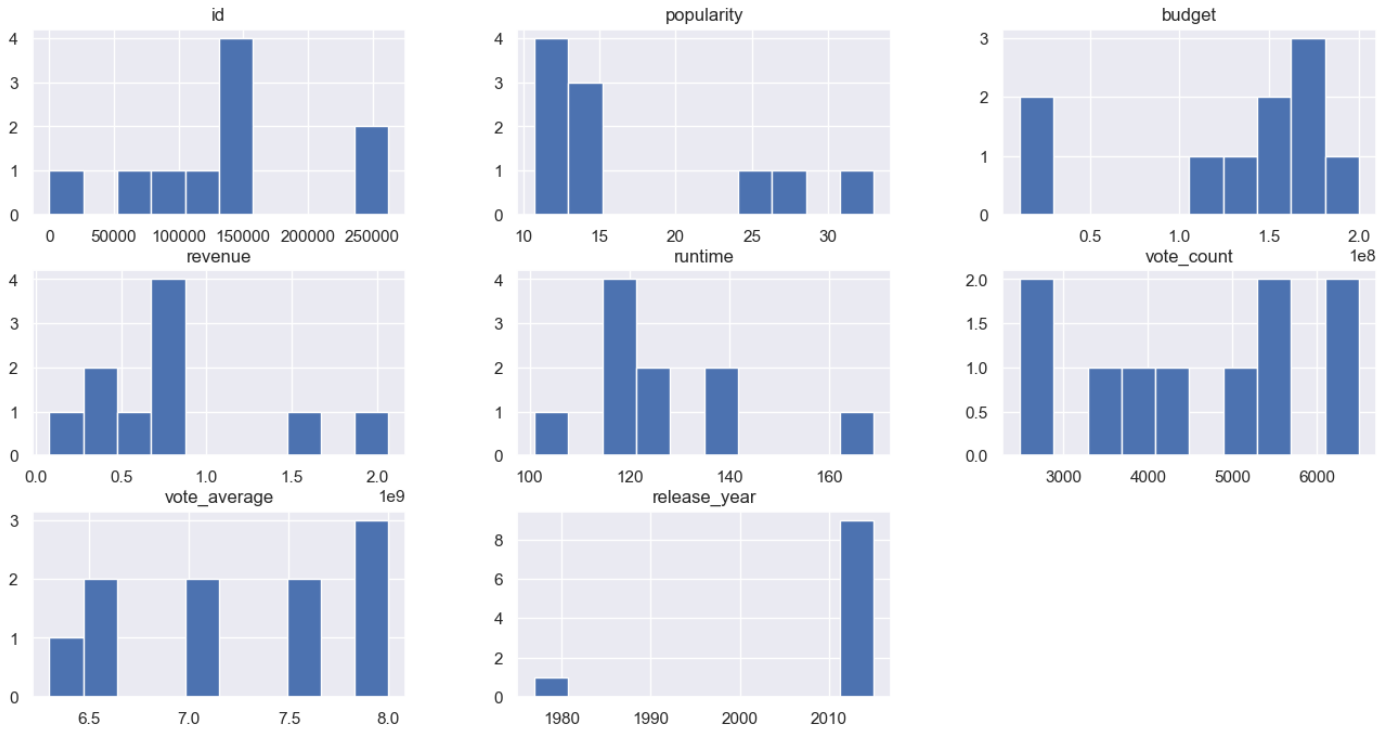
```
# top_10= df.sort_values('popularity', ascending=False)
top_popularity = df.sort_values('popularity',ascending=False)[:10]

plt.figure(figsize=(12,4))

plt.barh(top_popularity['original_title'],top_popularity['popularity'], align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Top 10 Popular Movies")
```

Out[132]: Text(0.5, 1.0, 'Top 10 Popular Movies')



And here are the top 10 popular movies.

In [133…
```
top_popularity.hist(figsize=[16,8], bins=10);
```



In [134…
```
popularty_mean = df['popularity'].mean()
print(popularty_mean)
df_p_high = df[df['popularity'] >= popularty_mean]
df_p_low = df[df['popularity'] < popularty_mean]

df_p_high.describe()
```

```
1.000051259389671
```

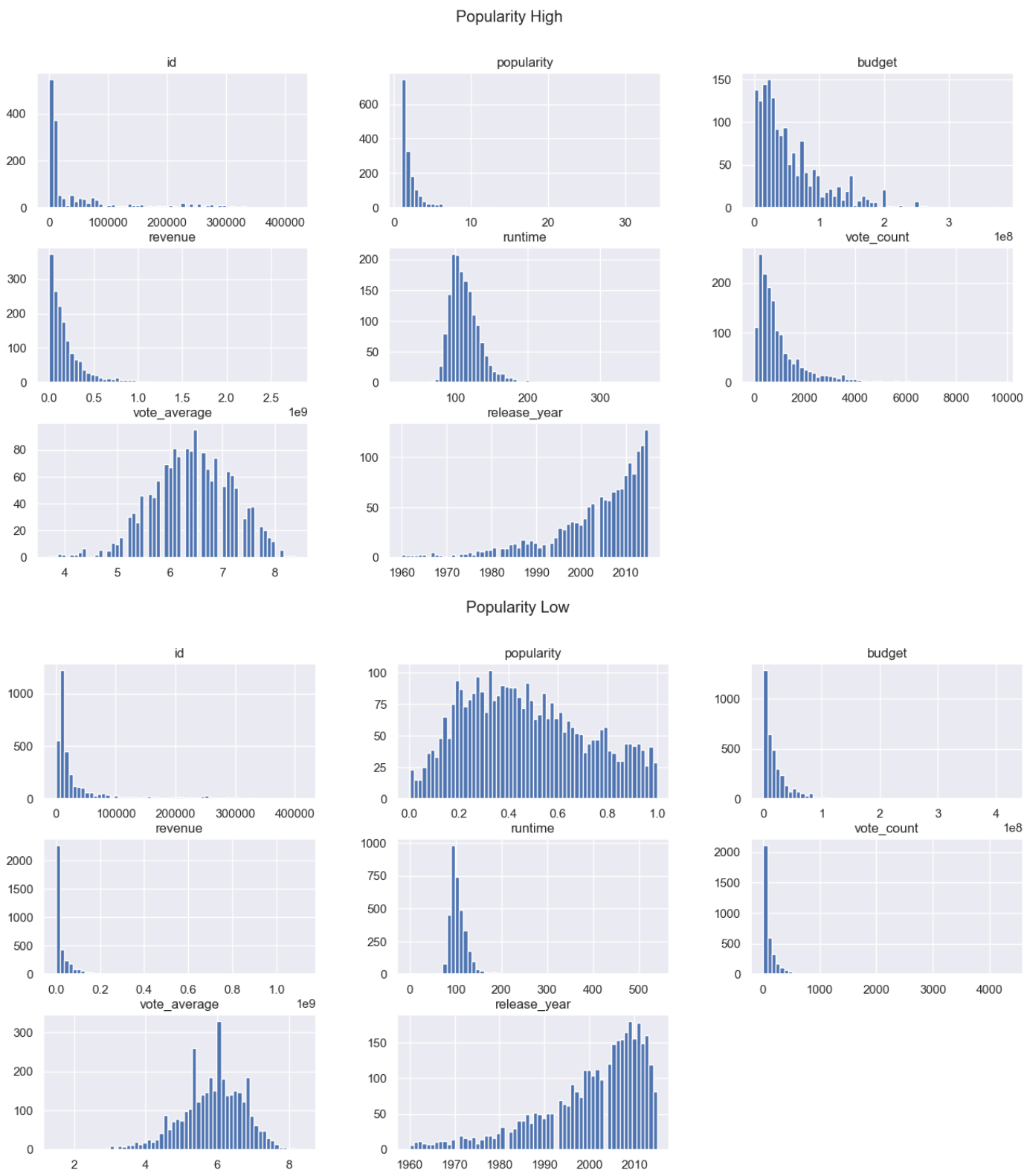
Out[134]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year
count	1590.00	1590.00	1590.00	1590.00	1590.00	1590.00	1590.00	1590.00
mean	53011.96	2.17	56364813.02	196881488.25	112.47	1057.19	6.42	2003.63
std	82231.39	1.90	52446927.65	234815020.77	21.66	1142.56	0.77	10.92
min	5.00	1.00	15000.00	0.00	26.00	10.00	3.70	1960.00
25%	1897.00	1.22	19000000.00	51086254.75	97.00	357.00	5.90	1999.00
50%	10266.50	1.61	40000000.00	123961185.00	109.00	676.00	6.40	2007.00

75%	68583.25	2.43	79000000.00	249333726.00	124.00	1275.00	7.00	2012.00
max	417859.00	32.99	380000000.00	2781505847.00	366.00	9767.00	8.40	2015.00

In [135…
```
df_p_high.hist(figsize=[16, 8] , bins=60), plt.suptitle('Popularity High') ,
df_p_low.hist(figsize=[16, 8], bins=60); plt.suptitle('Popularity Low')
```

Out[135]:
```
Text(0.5, 0.98, 'Popularity Low')
```



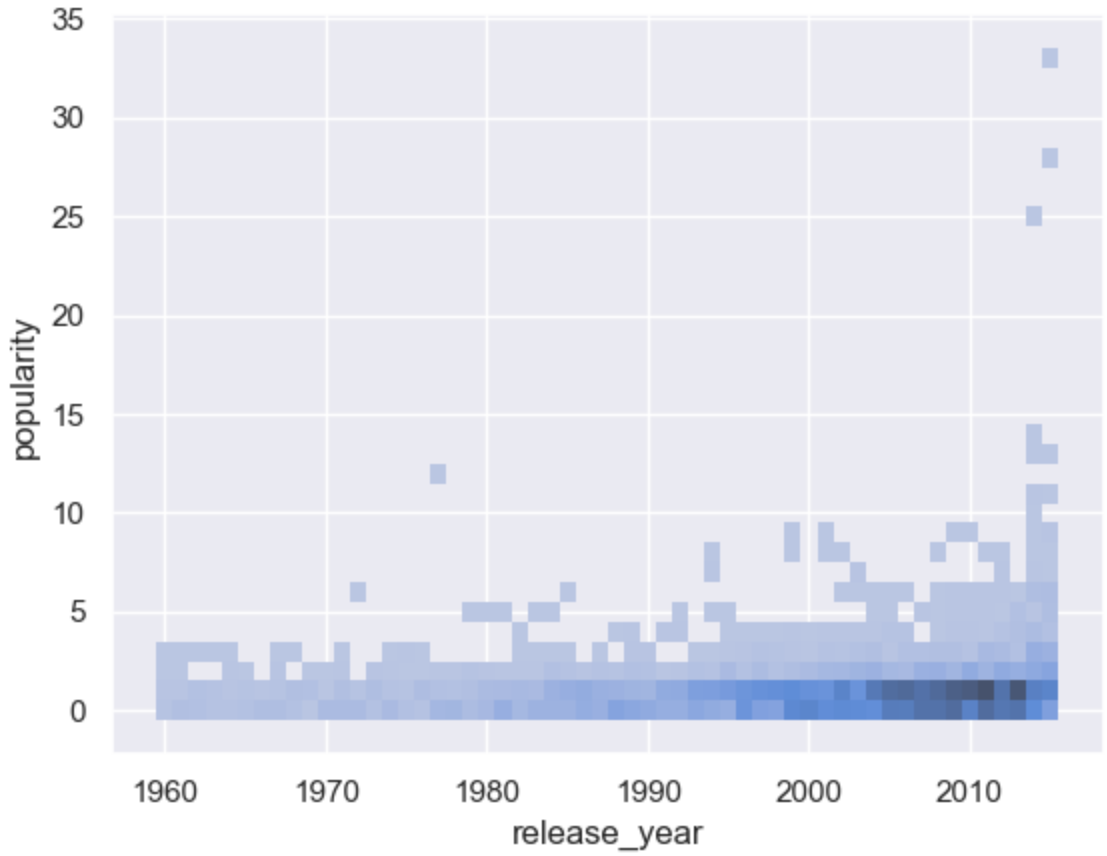
And from these visualizations we can tell that the most popular movies produced after 2008.

In [136…
```
df_uni = pd.unique(df.genres)
len(df_uni)
```

Out[136]: 1282

In [137…
```
sns.histplot(data=df, x="release_year", y="popularity",  element="poly",  discrete=True)
```

Out[137]: <Axes: xlabel='release_year', ylabel='popularity'>



We can see from this visualization that a lot of movies with low popularity were produced between 2005 and 2012.

Question 2: if the movie gets a high vote average does this mean high popularity?

In [138…
```
# best Movie by vote Vs popularity
top_vote = df.sort_values('vote_average',ascending=False)[:10]
top_vote.sort_values('vote_average', ascending=False)
```

Out[138]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime
4178	278	tt0111161	7.19	25000000	28341469	The Shawshank Redemption	Tim Robbins|Morgan Freeman|Bob Gunton|William ...	Frank Darabont	142
5986	242575	tt2876428	0.05	4000000	0	Guten Tag, RamÃ³n	Adriana Barraza|RÃ¼diger Evers|Hector Kotsifak...	Jorge RamÃrez SuÃ¡rez	119
7948	24128	tt0088178	0.28	1200000	4978922	Stop Making Sense	David Byrne|Tina Weymouth|Chris Frantz|Jerry H...	Jonathan Demme	88
7269	238	tt0068646	5.74	6000000	245066411	The Godfather	Marlon Brando|Al Pacino|James Caan|Richard S. ...	Francis Ford Coppola	175

650	244786	tt2582802	4.78	3300000	13993093	Whiplash	Miles Teller|J.K. Simmons|Melissa Benoist|Aust...	Damien Chazelle	105
10222	424	tt0108052	2.38	22000000	321265768	Schindler's List	Liam Neeson|Ben Kingsley|Ralph Fiennes|Carolin...	Steven Spielberg	195
2409	550	tt0137523	8.95	63000000	100853753	Fight Club	Edward Norton|Brad Pitt|Meat Loaf|Jared Leto|H...	David Fincher	139
9758	240	tt0071562	3.26	13000000	47542841	The Godfather: Part II	Al Pacino|Robert Duvall|Diane Keaton|Robert De...	Francis Ford Coppola	200
8043	92060	tt0088263	0.21	1100000	0	Michael Jackson's Thriller	Michael Jackson|Ola Ray|Vincent Price	John Landis	13
4177	680	tt0110912	8.09	8000000	213928762	Pulp Fiction	John Travolta|Samuel L. Jackson|Uma Thurman|Br...	Quentin Tarantino	154

In [139…

```
df.head(1)
```

Out[139]:

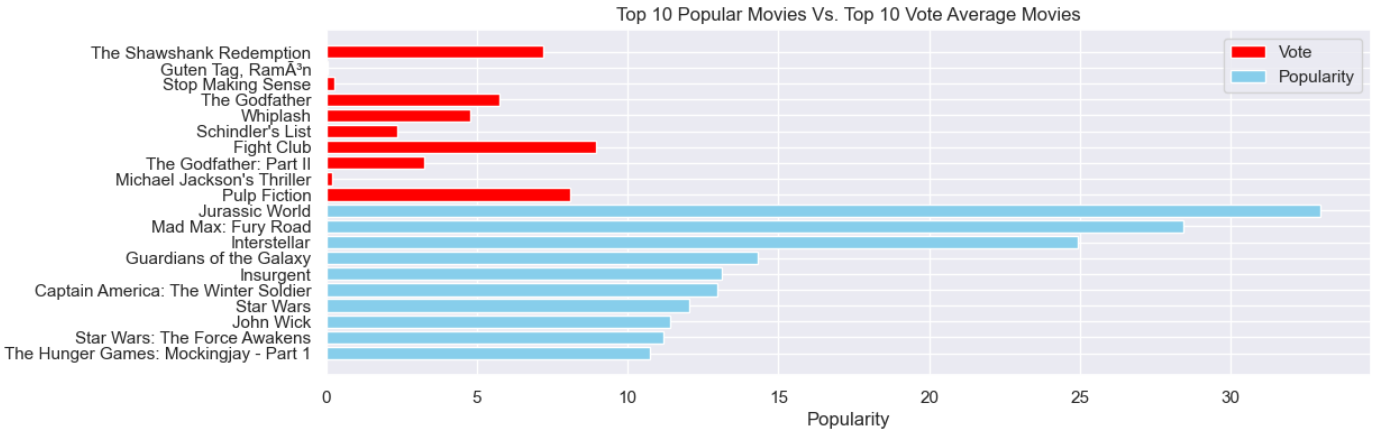
	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	
0	135397	tt0369610	32.99	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124	Action|A

In [140…

```
# top_10= df.sort_values('popularity', ascending=False)
top_popularity = df.sort_values('popularity',ascending=False)[:10]

plt.figure(figsize=(12,4))

plt.barh(top_vote['original_title'],top_vote['popularity'], align='center',
        color='red')
plt.barh(top_popularity['original_title'],top_popularity['popularity'], align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")


plt.legend(['Vote', 'Popularity'],loc='upper right', )


plt.title("Top 10 Popular Movies Vs. Top 10 Vote Average Movies")
```

Out[140]:

```
Text(0.5, 1.0, 'Top 10 Popular Movies Vs. Top 10 Vote Average Movies')
```



From the above bar chart we clearly see that high average vote does not mean high popularity.

Question 3: What are the highest genres of movies?

the first problem I encountered was the format of the column genres, they grouped together like this (Action|Adventure|Science Fiction|Thriller), so we need to separate it.

In [141…
```
def mySplitFunc(df, column_name, split_key):
    df_g_notnull = df[df[column_name].notnull()]
    genres_list = set()
    for i in df_g_notnull[column_name].str.split(split_key):
        genres_list = set().union(i,genres_list)
        genres_list = list(genres_list)
        genres_list


    return genres_list
```

In [142…
```
genres_list = mySplitFunc(df, 'genres', '|')
```

In [143…
```
# # hist geners
# df_g_notnull = df[df['genres'].notnull()]

# genres_list = set()
# for i in df_g_notnull['genres'].str.split('|'):
#     genres_list = set().union(i,genres_list)
#     genres_list = list(genres_list)
#     genres_list

# len(genres_list), genres_list
genres_list
```

Out[143]:
```
['Crime',
 'War',
 'History',
 'Animation',
 'Drama',
 'Adventure',
 'Western',
 'TV Movie',
 'Thriller',
 'Science Fiction',
 'Horror',
 'Foreign',
 'Comedy',
 'Mystery',
 'Fantasy',
```

```
 'Family',
 'Music',
 'Action',
 'Documentary',
 'Romance']
```

Now we have 20 different genres.

In [144…
```
df['release_date'] = pd.to_datetime(df['release_date']).dt.year
columns = {'release_date':'year'}
df.rename(columns=columns,inplace=True)
df['year'].apply(int).head()
```

Out[144]:
```
0    2015
1    2015
2    2015
3    2015
4    2015
Name: year, dtype: int64
```

In [145…
```
# new dataframe
df_with_genres = df.copy()
```

In [146…
```
for genre in genres_list:
    df_with_genres[genre] = df['genres'].str.contains(genre).apply(lambda x:1 if x else
genre_year = df_with_genres.loc[:,genres_list]
```

In [147…
```
df_with_genres.head(1)
```

Out[147]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	
0	135397	tt0369610	32.99	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124	Action|A

In [148…
```
df.head(1)
```

Out[148]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	director	runtime	
0	135397	tt0369610	32.99	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124	Action|A

In [149…
```
genre_year.head(1)
```

Out[149]:

	Crime	War	History	Animation	Drama	Adventure	Western	TV Movie	Thriller	Science Fiction	Horror	Foreign	Cor
0	0	0	0	0	0	1	0	0	1	1	0	0	

In [150…
```
genre_year.index = df['year']
genresdf = genre_year.groupby('year').sum()
genresdf.head(3)
```

Out[150]:

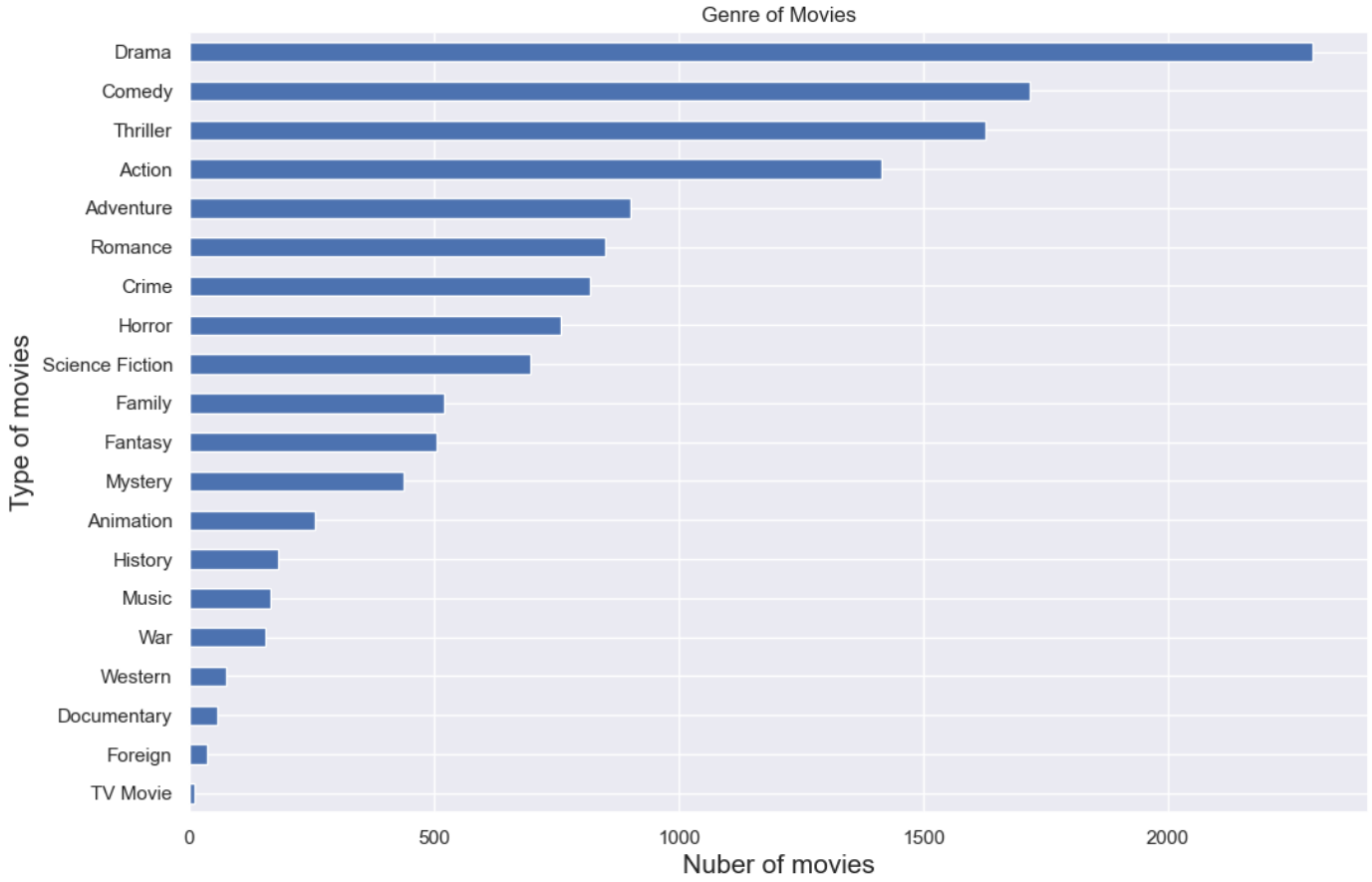
	Crime	War	History	Animation	Drama	Adventure	Western	TV Movie	Thriller	Science Fiction	Horror	Foreign

year											
1973	6	0	1	1	10	2	1	0	6	2	5	0
1974	9	0	0	0	9	3	2	0	10	3	3	0
1975	1	1	1	0	7	3	0	0	3	4	3	0

In [151…
```
genresdfSum = genresdf.sum(axis=0).sort_values(ascending=False)
genresdfSum
```

Out[151]:
```
Drama              2295
Comedy             1719
Thriller           1628
Action             1415
Adventure           902
Romance             851
Crime               818
Horror              759
Science Fiction     698
Family              520
Fantasy             504
Mystery             438
Animation           256
History             181
Music               165
War                 155
Western              75
Documentary          56
Foreign              36
TV Movie             10
dtype: int64
```

In [152…
```
fig=plt.figure(figsize=(12,8))
ax1 = plt.subplot(111)
rects = genresdfSum.sort_values(ascending=True).plot(kind='barh',label='genres')
plt.title('Genre of Movies')
plt.xlabel('Nuber of movies',fontsize=15)
plt.ylabel('Type of movies',fontsize=15)
plt.show()
```



We can see here the highest genres is Drama than comedy

Conclusions

First I started getting more familiar with data by seeing how the data looks.

I found that the dependent variable is popularity and the two independent variables are budget and revenue.

Then I checked the data type to see if there was anything wrong with it. And see if there are any null values in the dataset.

I found out that names of the columns are good, no need to change the name or merge at this stage, there are a lot of null values but luckily it's not important such as the homepage “website” of the movie and tagline which provide no insight especially in this early stage.

Then i dropped 7 columns that we don’t need, I fix some problems with the data such as duplicated rows, and investigate more about some of the rows that are null or have 0 budget, to see if they are important or not.

Then i started to do Exploratory Data Analysis, to answer 3 questions i have

Is there any correlation between popularity and budget? if the movie gets a high vote average does this mean high popularity? What are the highest genres of movies?

And I found answers to these questions with the help of visualizations and other pandas functions.

And here is a summary of what I found:

- `1` : There is correlation between popularity and budget, if the budget is high then there is a high chance to be popular.
- `2` : If the movie is popular then vote count “how many people vote” is high
- `3` : There no correlation between popularity and release year or runtime
- `4` : If the movie is popular then it will generate a lot of revenue.
- `5` : If the movie gets a high vote average that does not mean it is a popular movie.
- `6` : Most of the movies are not popular
- `7` : The highest genres of movies are drama and comedy and thriller .

The limitations that I faced is that the dataset does not contain new movies and that may introduce bias, some budgets are missing or have wrong values such as 0, if i can update the dataset that will help me a lot.

In [153…
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

Out[153]:
```
4294967295
```

References

seaborn https://seaborn.pydata.org/generated/seaborn.jointplot.html

pandas .hist https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html

pandas .drop https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

In [ ]:

In [ ]:

In [ ]: