Project: Investigate a Dataset - [TMDb movie data]

> Overview

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

1. Defining the Question

a) Specifying the Research Questions

- What factors determines whether a movie will be popular or not?
- Does the budget of the movie determines the revenue the movie will yield?
- Does the budget of the movie determines the popularity of the movie?

b) Defining the Metric for Success

This project will be successful when:

- The research questions are fully answered

c) Understanding the context

> What can we say about the success of a movie before it is released? Are there certain companies (Pixar?) that have found a consistent formula? Given that major films costing over $100 million to produce can still flop, this question is more important than ever to the industry. Film aficionados might have different interests. Can we predict which films will be highly rated, whether or not they are a commercial success?

> This is a great place to start digging in to those questions, with data on the plot, cast, crew, budget, and revenues of several thousand films.

d) Recording the Experimental Design

In this project, the following steps will be followed:

1. Loading and understanding data set
2. Data preparation/Wrangling
3. Exploratory Data Analysis
4. Conclusion

Tools to use:

- Seaborn
- Matplotlib
- Pandas
- Numpy

e) Data Relevance

The dataset to use for this project can be found by

following this link

Below is the dataset glossary:

- 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

2. Reading the Data

```
# Loading the libraries
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt

# Using seaborn style defaults and setting the default figure size
sb.set(rc={'figure.figsize':(30, 5)})
from warnings import filterwarnings
filterwarnings('ignore')
```

```
# Loading the Dataset
df = pd.read_csv('https://d17h27t6h515a5.cloudfront.net/topher/2017/October/59dd1c4c_tmdb-mov
```

3. Checking the Data

```
# Determining the no. of records in our dataset
print(df.shape)
```

```
(10866, 21)
```

```
# Previewing the top of our dataset
df.head()
```

	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...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
							Vin Diesel|Paul

```
# Previewing the bottom of our dataset
df.tail()
```

	id	imdb_id	popularity	budget	revenue	original_title	cast
10861	21	tt0060371	0.080598	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...
10862	20379	tt0060472	0.065543	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...

```
# Checking whether each column has an appropriate datatype
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
```

4. External Data Source Validation

> We don't have any other external data set to compare with this data set

5. Data Preperation/Tidying the Dataset

a.Validation

```
# Checking for irrelevant columns
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')
```

> So far all the columns are relevant

b. Completeness

```
# Checking for missing values
df.isnull().any().any()
```

```
True
```

```
# Identifying the Missing Data
df.isnull().sum()
```

```
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 have missing values

```
# Checking percentage of missing values per columns
missing_columns = []
for i, col in enumerate(df.columns):
  missing = (df[col].isnull().sum()/df.shape[0])*100
  if missing > 0:
    missing_columns.append(col)
  print(f'{i+1}. {col} = {(df[col].isnull().sum()/df.shape[0])*100}%')
```

```
1. id = 0.0%
2. imdb_id = 0.09203018590097553%
3. popularity = 0.0%
4. budget = 0.0%
5. revenue = 0.0%
6. original_title = 0.0%
7. cast = 0.6994294128474139%
8. homepage = 72.97993741947359%
9. director = 0.40493281796429226%
10. tagline = 25.989324498435483%
11. keywords = 13.740106755015645%
12. overview = 0.03681207436039021%
13. runtime = 0.0%
14. genres = 0.21166942757224366%
15. production_companies = 9.479109147800479%
16. release_date = 0.0%
17. vote_count = 0.0%
18. vote_average = 0.0%
19. release_year = 0.0%
20. budget_adj = 0.0%
21. revenue_adj = 0.0%
```

> homepage has a very high percentage of missing values. It has to be dropped.

```
# The columns with missing values
missing_columns
```

```
['imdb_id',
 'cast',
 'homepage',
 'director',
```

```
       'tagline',
       'keywords',
       'overview',
       'genres',
       'production_companies']
```

```
# Dropping homepage
df.drop('homepage', axis = 1, inplace = True)
```

```
# Dropping all rows with null values
df.dropna(inplace = True)
```

```
# Checking for null values again
df.isnull().any().any()
```

```
False
```

> All null values have been dealt with

c. Consistency

```
# Checking for duplicates
df.duplicated().any().any()
```

```
True
```

```
# Dropping duplicates
df.drop_duplicates(inplace = True)
# Checking changes
df.duplicated().any().any()
```

```
False
```

> The duplicate data in our dataset have been dropped. The result dataset has no duplicates.

```
# Determining the no. of records in our dataset
print(df.shape)
```

```
(7030, 20)
```

d. Uniformity

```
# Checking column names
df.columns
```

```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'director', 'tagline', 'keywords', 'overview', 'runtime',
       'genres', 'production_companies', 'release_date', 'vote_count',
       'vote_average', 'release_year', 'budget_adj', 'revenue_adj'],
      dtype='object')
```

> The columns naming is uniform. But to ensure everything is accurate, will run the below code

```
# Changing all column names to lower case
df.columns = df.columns.str.lower()
# Checking changes
df.columns
```

```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'director', 'tagline', 'keywords', 'overview', 'runtime',
       'genres', 'production_companies', 'release_date', 'vote_count',
       'vote_average', 'release_year', 'budget_adj', 'revenue_adj'],
      dtype='object')
```

e. Outliers

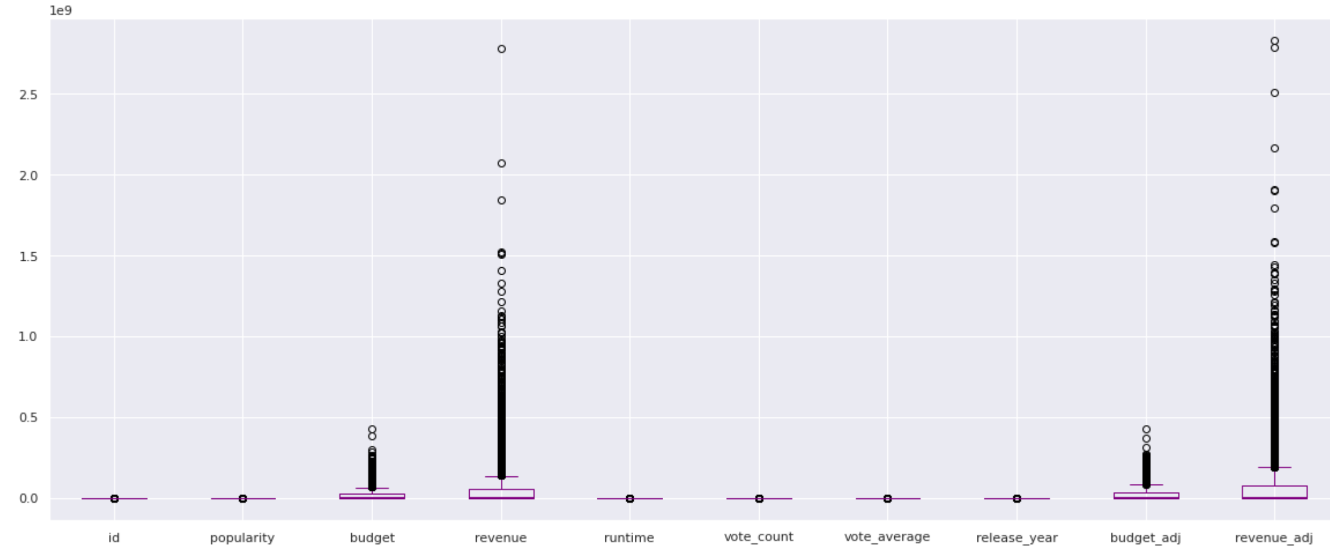
```
# Creating an outliers function
def outliers(data):
  # IQR
  Q1, Q3, IQR = 0, 0, 0
  outliers = pd.DataFrame()
  # Numerical columns
  numerical = data.select_dtypes(include = ['int64', 'float64'])
  Q1 = numerical.quantile(0.25)
  Q3 = numerical.quantile(0.75)
  IQR = Q3 - Q1
  # Outliers
  outliers = numerical[((numerical < (Q1 - 1.5 * IQR)) |(numerical > (Q3 + 1.5 * IQR))).any(a
  print(f'Number of outliers = {outliers.shape[0]}')
  print(f'Percentage = {(outliers.shape[0]/data.shape[0])*100}%')
```

```
# Checking for Outliers
outliers(df)
```

```
Number of outliers = 2687
Percentage = 38.221906116642955%
```

```
# Viewing the outliers
df.boxplot(figsize=(20,8),color='purple')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8f2e47950>
```



Though we have a great number of outliers in budget, revenue, budget_adj and revenue_adj, they form over 38% of our data set hence dropping them will hugely affect our analysis. Thus, will keep them and standadize the data later.

f. Anomalies

```
# Checking for Anomalies
df.describe()
```

pularity	budget	revenue	runtime	vote_count	vote_average	release_year
30.000000	7.030000e+03	7.030000e+03	7030.000000	7030.000000	7030.000000	7030.000000
0.829578	2.084592e+07	5.933303e+07	104.849075	312.752205	6.013329	1999.380939
0.278587	0.000000e+00	0.000000e+00	92.000000	24.000000	5.500000	1992.000000

> There seems to be no anomalies in our data set

6. Exploratory Descriptive Analysis

```
# Number of unique values
cols = df.columns.tolist()
print(f'Number of unique values\n')
for col in cols:
  print(f'{col}: {len(df[col].unique().tolist())}')
```

```
Number of unique values

id: 7030
imdb_id: 7030
popularity: 7015
budget: 447
revenue: 4076
original_title: 6860
cast: 7003
director: 3190
tagline: 6993
keywords: 6815
overview: 7029
runtime: 188
genres: 1560
production_companies: 5387
release_date: 4621
vote_count: 1277
vote_average: 64
release_year: 56
budget_adj: 2296
revenue_adj: 4169
```

Our dataset has multiple unique values therefore it's not ideal to do categorical data analysis. Will do more analysis on numerical data.

```
# Describing the Data
df.describe()
```

	id	popularity	budget	revenue	runtime	vote_count
count	7030.000000	7030.000000	7.030000e+03	7.030000e+03	7030.000000	7030.000000
mean	51923.701422	0.829578	2.084592e+07	5.933303e+07	104.849075	312.752205
std	81410.657714	1.180330	3.602527e+07	1.404243e+08	23.794219	693.268737
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000
25%	9540.250000	0.278587	0.000000e+00	0.000000e+00	92.000000	24.000000
50%	14738.500000	0.506241	5.000000e+06	4.859580e+06	101.000000	73.000000
75%	46964.750000	0.956461	2.600000e+07	5.473358e+07	114.000000	263.000000
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	705.000000	9767.000000

```
# Function that determines the measures of central tendency.
def MeasureCentral(measure, columns, data):
  for column in columns:
    if measure == 'mean':
      print(f"{column} column mean = {data[column].mean()}")
    elif measure == 'median':
      print(f"{column} column median = {data[column].median()}")
    elif measure == 'mode':
      print(f"{column} column mode = {data[column].mode()}")


# Function used to determine the measures of distribution.
def MeasureDistribution(measure, columns, data):
  for column in columns:
    if measure == 'range':
      print(f"{column} column range = {data[column].max() - data[column].min()}")
    elif measure == 'IQR':
      Q1 = data[column].quantile(0.25)
      Q3 = data[column].quantile(0.75)
      IQR = Q3 - Q1
      print(f"{column} column IQR = {IQR}")
    elif measure == 'var':
      print(f"{column} column variance = {data[column].var()}")
    elif measure == 'std':
      print(f"{column} column std = {data[column].std()}")
    elif measure == 'skew':
      print(f"{column} column skew = {data[column].skew()}")
    elif measure == 'kurt':
      print(f"{column} column kurt = {data[column].kurt()}")


# Distribution and Boxplot functions
def NumericalPlots(column, data1, data2):
  fig, ax = plt.subplots(2,2, figsize = (12,10))
  # Outliers
```

```
    # Distribution plot
    sb.distplot(data1[column], hist=True, ax=ax[0,0], color = 'green')
    ax[0,0].set_title('Outliers: Freq dist '+ column, fontsize=10)
    ax[0,0].set_xlabel(column, fontsize=8)
    ax[0,0].set_ylabel('Count', fontsize=8)
    # Box plot
    sb.boxplot(y = data1[column], ax = ax[0,1], color = 'green')
    ax[0,1].set_title(f'Outliers: Box Plot - {column}')
    ax[0,1].set_xlabel(column)

    # No outliers
    # Distribution plot
    sb.distplot(data2[column], hist=True, ax=ax[1,0], color = 'green')
    ax[1,0].set_title('No outliers: Freq dist '+ column, fontsize=10)
    ax[1,0].set_xlabel(column, fontsize=8)
    ax[1,0].set_ylabel('Count', fontsize=8)
    # Box plot
    sb.boxplot(y = data2[column], ax = ax[1,1], color = 'green')
    ax[1,1].set_title(f'No outliers: Box Plot - {column}')
    ax[1,1].set_xlabel(column)
    plt.show()


# Numerical columns
numerical = df.select_dtypes(exclude = 'object').columns.tolist()
numerical
```

```
['id',
 'popularity',
 'budget',
 'revenue',
 'runtime',
 'vote_count',
 'vote_average',
 'release_year',
 'budget_adj',
 'revenue_adj']
```

```
# Mean
MeasureCentral('mean', numerical, df)
```

```
id column mean = 51923.701422475104
popularity column mean = 0.8295776179231873
budget column mean = 20845918.90910384
revenue column mean = 59333034.54708393
runtime column mean = 104.84907539118065
vote_count column mean = 312.7522048364154
vote_average column mean = 6.013328591749624
release_year column mean = 1999.3809388335703
budget_adj column mean = 25012344.34636352
revenue_adj column mean = 76452322.38897239
```

```
# Median
MeasureCentral('median', numerical, df)
```

```
id column median = 14738.5
popularity column median = 0.506240999999999
budget column median = 5000000.0
revenue column median = 4859580.5
runtime column median = 101.0
vote_count column median = 73.0
vote_average column median = 6.1
release_year column median = 2003.0
budget_adj column median = 6951083.69480127
revenue_adj column median = 6457480.762482705
```

```
# Mode
MeasureCentral('mode', numerical, df)
```

```
id column mode = 0            5
1            6
2           11
3           12
4           13
         ...
7025    378373
7026    395560
7027    395883
7028    414419
7029    417859
Length: 7030, dtype: int64
popularity column mode = 0     0.078482
1     0.109305
2     0.111351
3     0.126283
4     0.186995
5     0.187319
6     0.227580
7     0.317301
8     0.340804
9     0.468552
10    0.506241
11    0.623706
12    0.701814
13    0.984256
14    1.107689
dtype: float64
budget column mode = 0    0
dtype: int64
revenue column mode = 0    0
dtype: int64
runtime column mode = 0    90
dtype: int64
vote_count column mode = 0    10
dtype: int64
vote_average column mode = 0    6.1
dtype: float64
```

```
release_year column mode = 0    2014
dtype: int64
budget_adj column mode = 0    0.0
dtype: float64
revenue_adj column mode = 0    0.0
dtype: float64
```

```
# Range
MeasureDistribution('range', numerical, df)
```

```
id column range = 417854
popularity column range = 32.985575
budget column range = 425000000
revenue column range = 2781505847
runtime column range = 705
vote_count column range = 9757
vote_average column range = 6.9
release_year column range = 55
budget_adj column range = 425000000.0
revenue_adj column range = 2827123750.41189
```

```
# IQR
MeasureDistribution('IQR', numerical, df)
```

```
id column IQR = 37424.5
popularity column IQR = 0.67787375
budget column IQR = 26000000.0
revenue column IQR = 54733579.75
runtime column IQR = 22.0
vote_count column IQR = 239.0
vote_average column IQR = 1.0999999999999996
release_year column IQR = 18.0
budget_adj column IQR = 34633361.9943644
revenue_adj column IQR = 75283594.78765467
```

```
# Variance
MeasureDistribution('var', numerical, df)
```

```
id column variance = 6627695189.356135
popularity column variance = 1.3931787430094003
budget column variance = 1297819992939106.0
revenue column variance = 1.971899091548283e+16
runtime column variance = 566.1648696813245
vote_count column variance = 480621.5413772665
vote_average column variance = 0.7683313573698984
release_year column variance = 181.41525761896435
budget_adj column variance = 1563157936991113.2
revenue_adj column variance = 2.9948686414151332e+16
```

```
# Standard Deviation
MeasureDistribution('std', numerical, df)
```

```
id column std = 81410.6577135705
popularity column std = 1.1803299297270236
budget column std = 36025268.81147601
revenue column std = 140424324.51496014
runtime column std = 23.79421924924885
vote_count column std = 693.2687367661018
vote_average column std = 0.8765451256894299
release_year column std = 13.46904813336727
budget_adj column std = 39536792.19399461
revenue_adj column std = 173056887.79748505
```

```
# Skew
MeasureDistribution('skew', numerical, df)
```

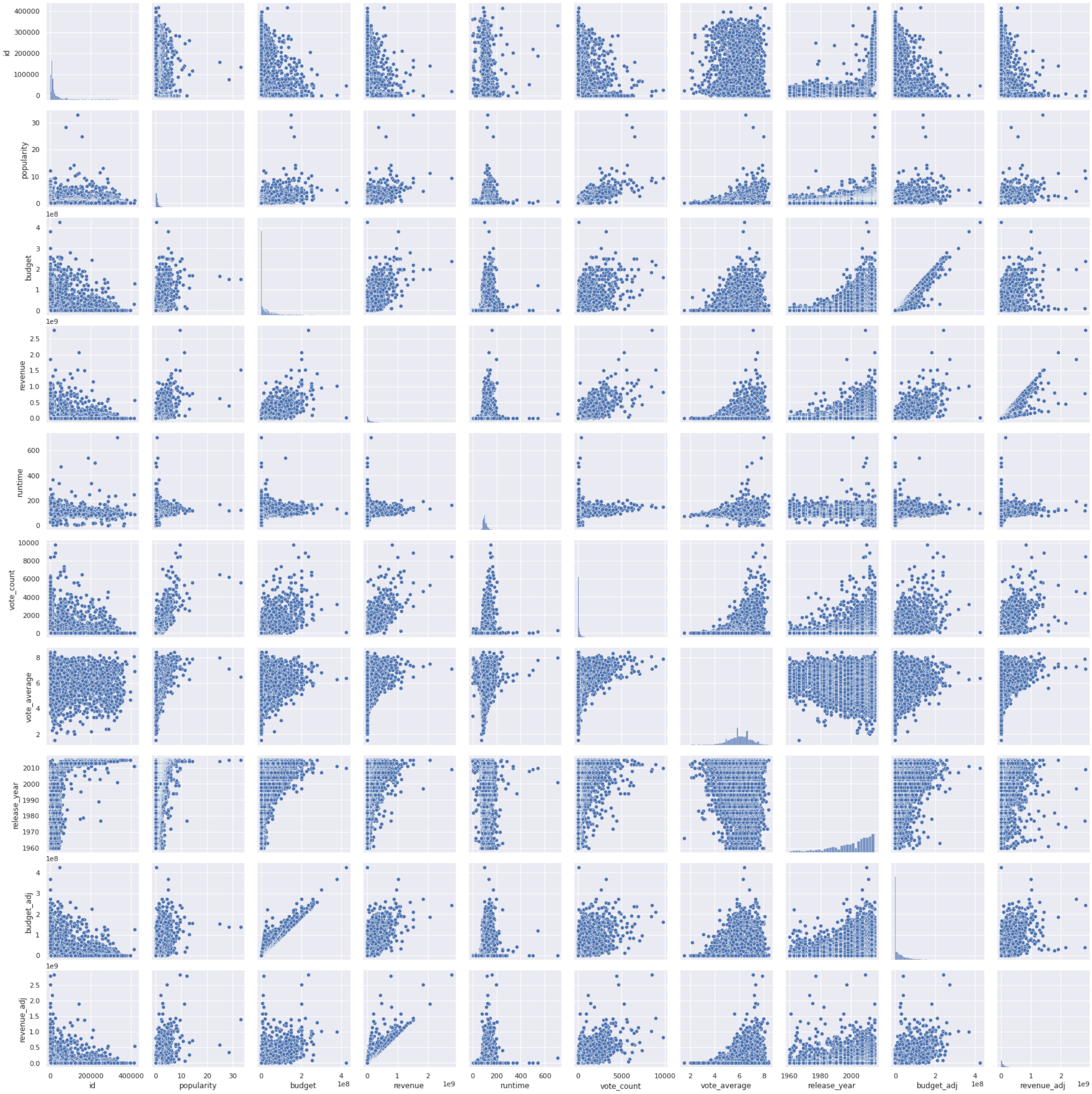
```
id column skew = 2.1418487409244342
popularity column skew = 8.777619671334422
budget column skew = 3.0405516707266615
revenue column skew = 5.499690147162847
runtime column skew = 5.552847912472253
vote_count column skew = 5.047857893970133
vote_average column skew = -0.47539388182016146
release_year column skew = -0.9876836182075454
budget_adj column skew = 2.50581958688648
revenue_adj column skew = 5.17366848452647
```

```
# Kurtosis
MeasureDistribution('kurt', numerical, df)
```

```
id column kurt = 3.5480853022042145
popularity column kurt = 161.61459319496856
budget column kurt = 12.90355172580694
revenue column kurt = 50.36318293386408
runtime column kurt = 100.23851171399662
vote_count column kurt = 35.484756936523
vote_average column kurt = 0.6822037854560352
release_year column kurt = 0.21457150599067853
budget_adj column kurt = 8.452202269157084
revenue_adj column kurt = 43.77957345118952
```

```
# Explore the types of relationships across the entire data set.
plt.figure(figsize=(20,8))
sb.pairplot(df)
```

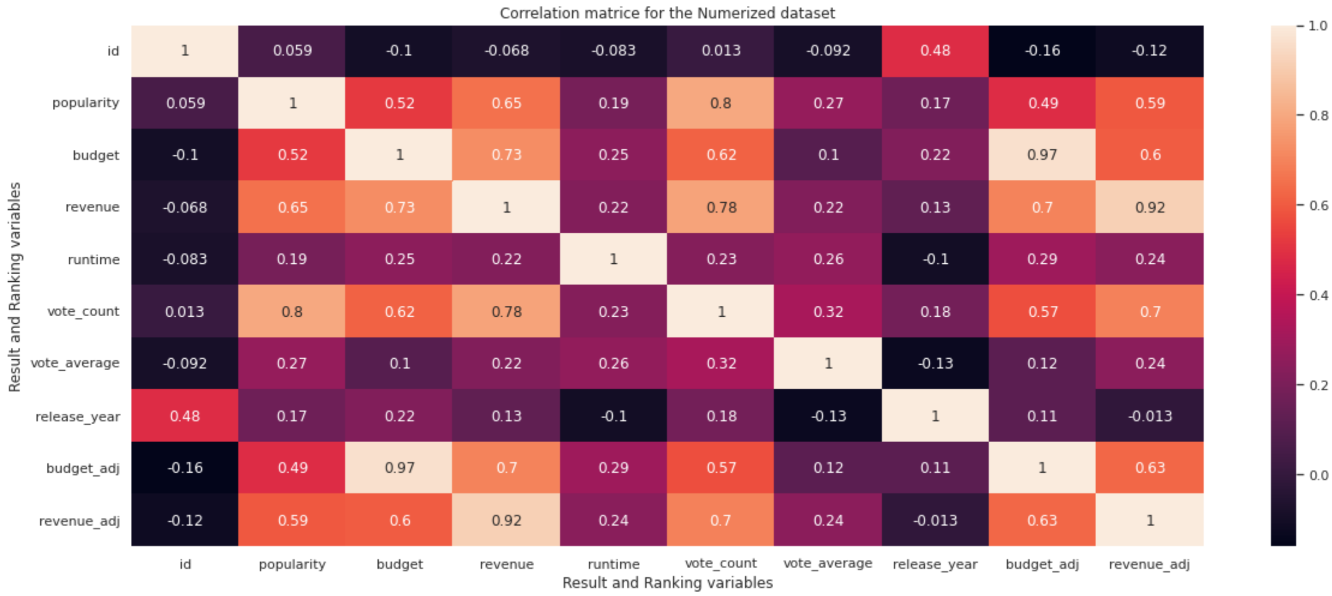
```
<seaborn.axisgrid.PairGrid at 0x7fb8f20759d0>
<Figure size 1440x576 with 0 Axes>
```



> From the pairplots, we cannot clearly see the correlations, will do further analysis to investigate this.

```
# Heatmap of correlation
plt.figure(figsize=(20,8))
corr_matrix = df.corr(method = 'pearson')

sb.heatmap(corr_matrix, annot = True)

plt.title("Correlation matrice for the Numerized dataset")
```

```
plt.xlabel("Result and Ranking variables")

plt.ylabel("Result and Ranking variables")

plt.show()
```



We see a strong correlation (>0.5) between:

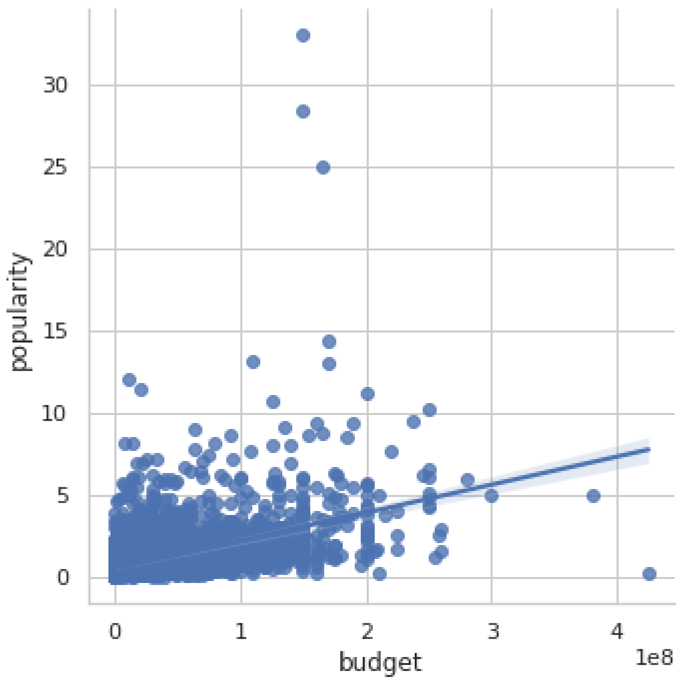
- budget and popularity
- revenue and popularity
- vote_count and popularity
- revenue_adj and popularity
- revenue and budget
- vote_count and budget
- budget_adj and budget
- vote_count and revenue
- budget_adj and revenue
- revenue_adj and revenue
- budget_adj and vote_count
- revenue_adj and vote_count
- revenue_adj and budget_adj

```
# defining a function to plot regression relation between two variables
```

```
def cor(col1,col2,d):
  ans = sb.regplot(x = col1, y = col2, data = d, scatter_kws = {"color": "red"}, line_kws = {
  return ans
```

```
# budget and popularity
sb.set_style('whitegrid')
sb.lmplot(x='budget',y='popularity',data=df)
```

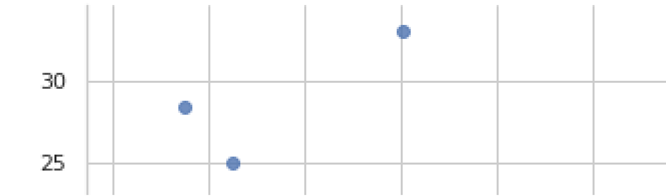
```
<seaborn.axisgrid.FacetGrid at 0x7fb8eceb1210>
```



The budget is positively correlated to the popularity of the movie.

```
# revenue and popularity
sb.set_style('whitegrid')
sb.lmplot(x='revenue',y='popularity',data=df)
```

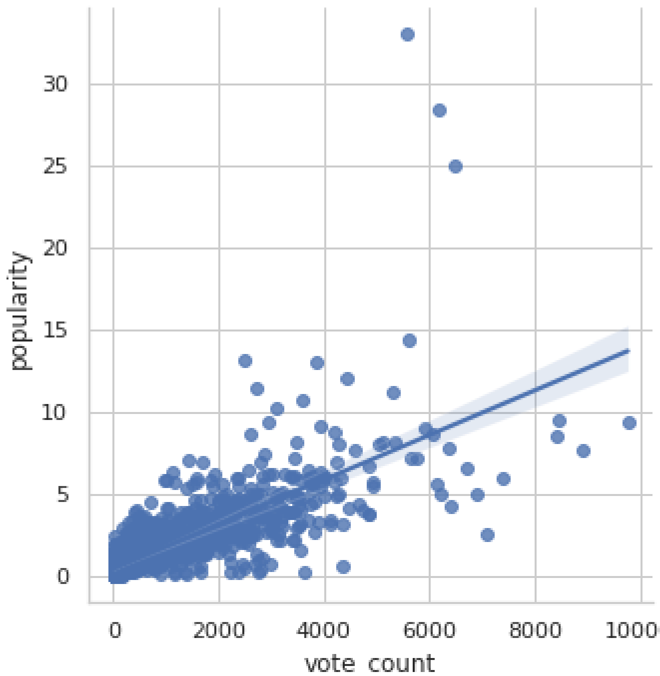
```
<seaborn.axisgrid.FacetGrid at 0x7fb8eb7fbd10>
```



The revenue is positively correlated to the popularity of the movie.

```
# vote_count and popularity
sb.set_style('whitegrid')
sb.lmplot(x='vote_count',y='popularity',data=df)
```

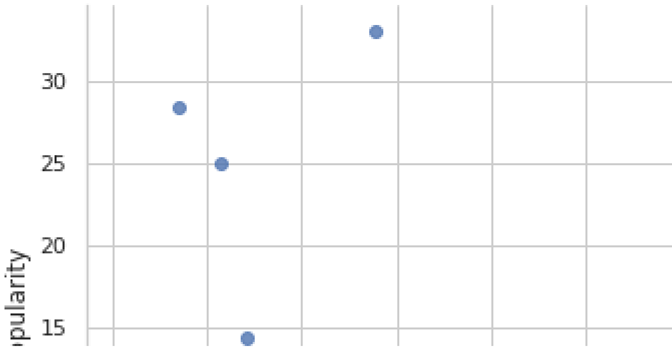
```
<seaborn.axisgrid.FacetGrid at 0x7fb8e9c8e110>
```



The vote_count is positively correlated to the popularity of the movie.

```
# revenue_adj and popularity
sb.set_style('whitegrid')
sb.lmplot(x='revenue_adj',y='popularity',data=df)
```

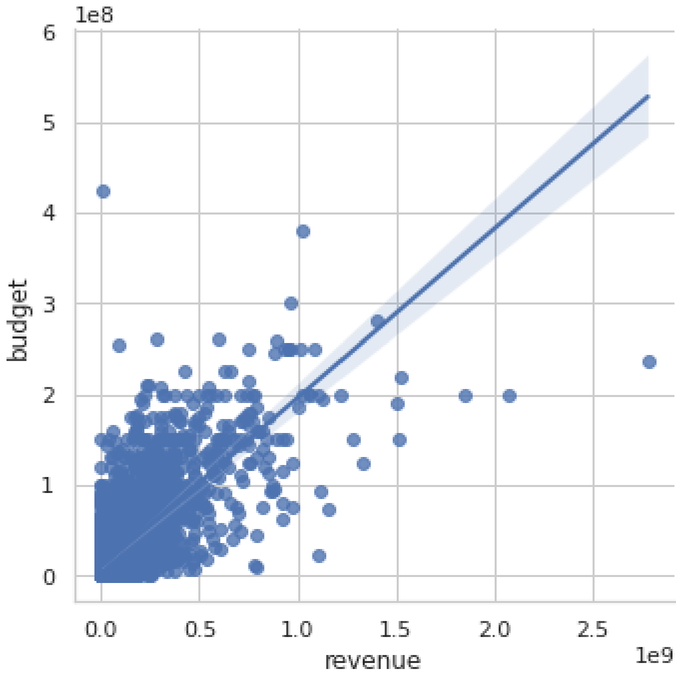
<seaborn.axisgrid.FacetGrid at 0x7fb8e9c62150>



The revenue_adj is positively correlated to the popularity of the movie.

```
# revenue and budget
sb.set_style('whitegrid')
sb.lmplot(x='revenue',y='budget',data=df)
```

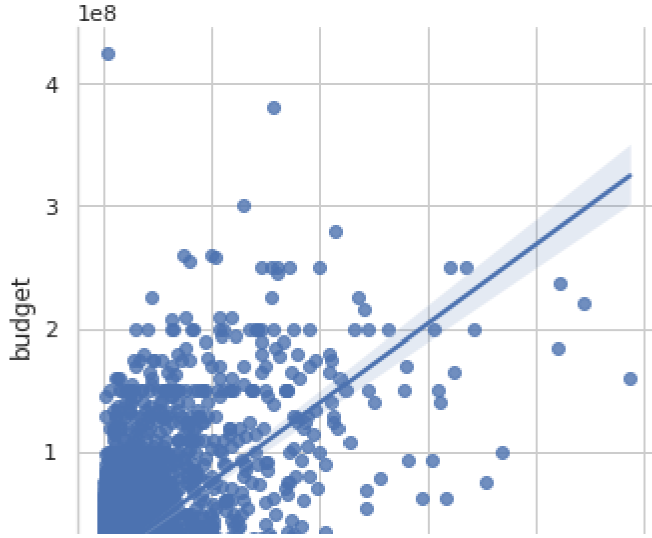
<seaborn.axisgrid.FacetGrid at 0x7fb8e9bfc550>



The revenue is positively correlated to the budget of the movie.

```
# vote_count and budget
sb.set_style('whitegrid')
sb.lmplot(x='vote_count',y='budget',data=df)
```

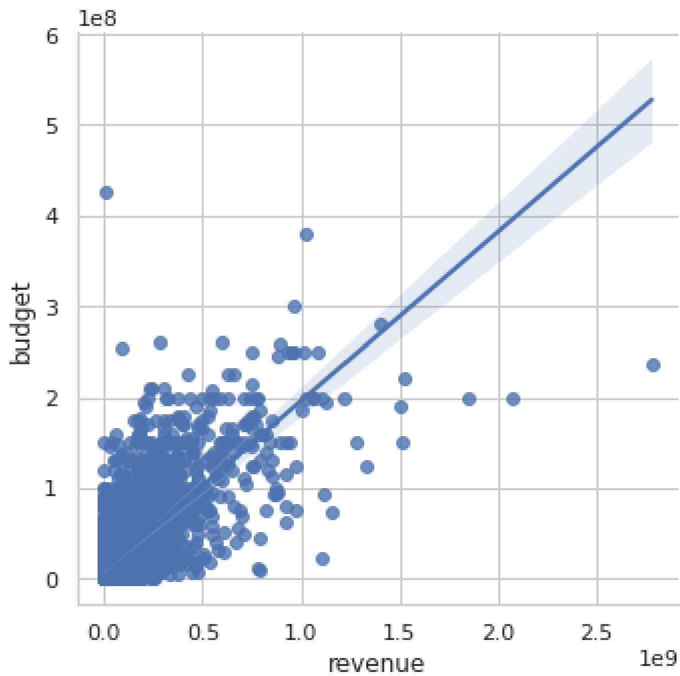
```
<seaborn.axisgrid.FacetGrid at 0x7fb8e9b628d0>
```



The budget is positively correlated to the vote_count of the movie.


```
# budget and revenue
sb.set_style('whitegrid')
sb.lmplot(x='revenue',y='budget',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb8e9bbea50>
```



The budget is positively correlated to the revenue of the movie.

Conclusion

From the Analysis, we can now answer our research questions:

1. What factors determines whether a movie will be popular or not?

> The following factors are postively correlated to the popularity of the movie i.e. with their increase, th e popularity of the movie increases and vice versa.
>
> - budget
> - revenue
> - vote_count

2. Does the budget of the movie determines the revenue the movie will yield?

> Yes.The budget of the movie is positively correlated to the revenue of the movie. This means the higher the budget of the movie, the emore likely that it will attract much revenue.

3. Does the budget of the movie determines the popularity of the movie?

> Yes. The budget of the movie is positively correlated to the popularity of the movie. This means the higher the budget of the movie, th emore likely that it will be popular.

✓ 1s completed at 8:38 AM