# Investigate_a_Dataset

April 23, 2020

## 1 Project: Investigate a Dataset (TMDB Movie Data)

### 1.1 Table of Contents


## Introduction

This dataset contains information about 10,000 movies collected from The Movie Database (TMDb).How can we utilize this dataset to identify and predict criteria that makes a good movies and eventually increase profit. below are the questions i plan to answer from analysing this dataset

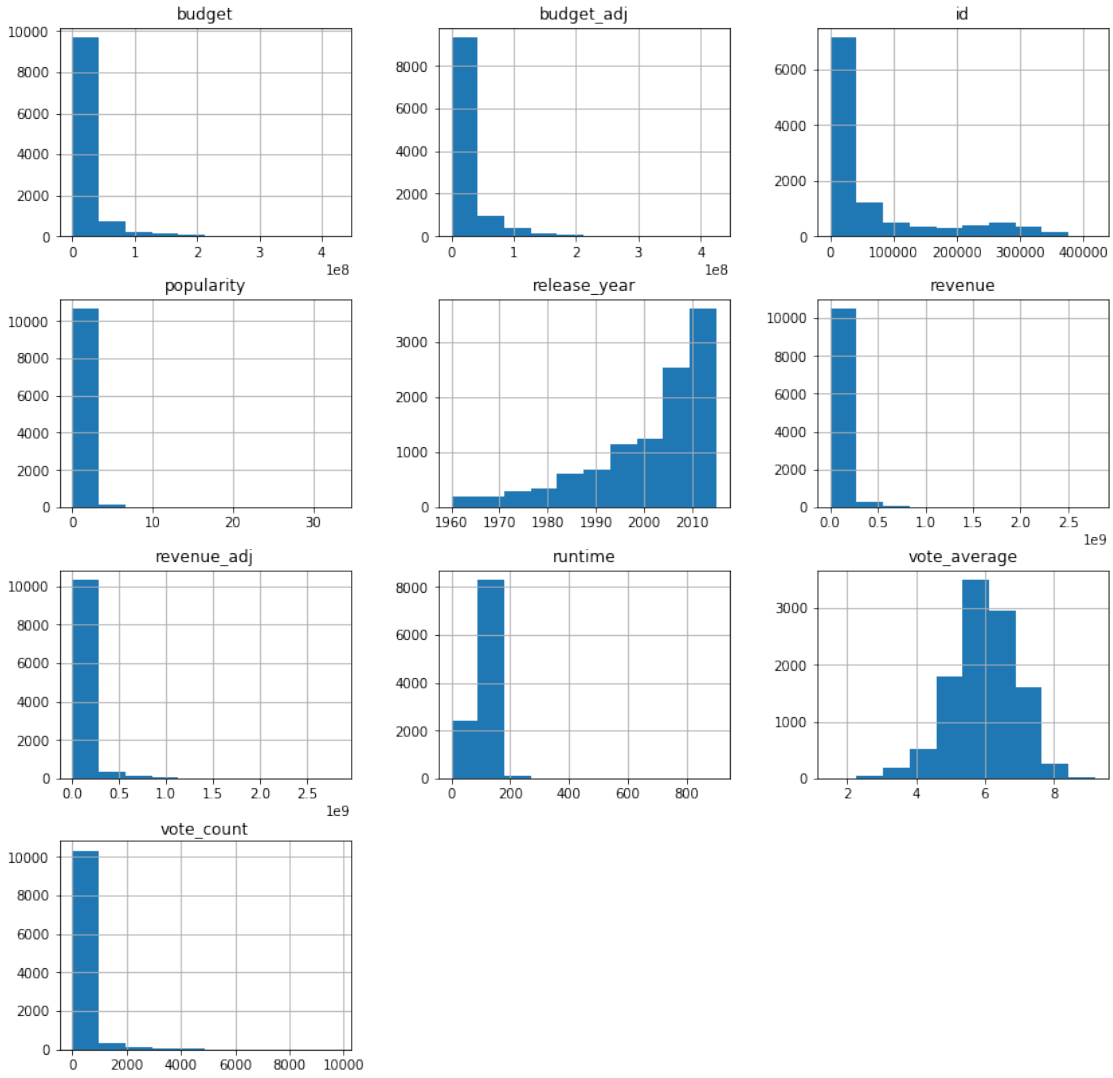
1. Q1 - Genres populareties top to bottom based on entire dataset?
2. Q2: Movie genres over time - Most populare Genres
3. Q3: Number of movies release by year
4. Q4: Top 10 Directors with most number of movies
5. Q5: Max and Min movies profits
6. Q6: Average profit per year
7. Q7: total profit over the years
8. Q8: top 10 movies with highest profit
9. Q9: Average movie runtime from year to year
10. Q10: Actors participated in highest number of movies
11. Q11. top production companies with highest number of movies
12. Q12: relation between vote count and vote average

In [153]:
```
# import libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

## Data Wrangling

In this section of the report, I will load in the data and check sample size, datatype, duplication, missing values and perform data cleanliness so the dataset is ready for analysis.

```
In [154]: # Load data and print out a few lines. Perform operations to inspect data
          #   types and look for instances of missing or possibly errant data.
          df=pd.read_csv('tmdb-movies.csv')
          df.hist(figsize=(14,14));
```




**Findings:** 1. home_page feature is not populated most of the time, only 2936 out of 10866 total records has data (27%) 2. tagline columns has 30% of missing values 3. Both features can be considered for removal

**Data sample Datatypes**

```
In [155]: df.dtypes
```

```
Out[155]: 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
```

**Findings:** 1- release_data shoud be datetime instead of object datatype
**Features with missing values**

```
In [156]: df.isnull().sum()
```

```
Out[156]: 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
```

Number of rows with at least null values in any of the columns

```
In [157]: df.isnull().any(axis=1).sum()

Out[157]: 8874
```

**Show duplidate rows**

```
In [158]: df.duplicated().sum()

Out[158]: 1
```

Findings: only one row is found duplicated in the dataset.

```
In [159]: df[df.duplicated(keep=False)]

Out[159]:          id    imdb_id  popularity    budget  revenue original_title  \
          2089  42194  tt0411951        0.60  30000000   967000         TEKKEN
          2090  42194  tt0411951        0.60  30000000   967000         TEKKEN

                                                      cast homepage  \
          2089  Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...      NaN
          2090  Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...      NaN

                director              tagline     ...       \
          2089  Dwight H. Little  Survival is no game     ...
          2090  Dwight H. Little  Survival is no game     ...

                                                overview runtime  \
          2089  In the year of 2039, after World Wars destroy ...      92
          2090  In the year of 2039, after World Wars destroy ...      92

                                               genres    production_companies  \
          2089  Crime|Drama|Action|Thriller|Science Fiction  Namco|Light Song Films
          2090  Crime|Drama|Action|Thriller|Science Fiction  Namco|Light Song Films

               release_date vote_count  vote_average  release_year    budget_adj  \
          2089      3/20/10        110          5.00          2010 30,000,000.00
          2090      3/20/10        110          5.00          2010 30,000,000.00

                revenue_adj
          2089   967,000.00
          2090   967,000.00

          [2 rows x 21 columns]

In [160]: df.describe()

Out[160]:              id  popularity         budget          revenue   runtime  \
          count  10,866.00   10,866.00      10,866.00        10,866.00 10,866.00
          mean   66,064.18        0.65  14,625,701.09    39,823,319.79    102.07
```

| | | | | | |
|---|---|---|---|---|---|
| std | 92,130.14 | 1.00 | 30,913,213.83 | 117,003,486.58 | 31.38 |
| min | 5.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 25% | 10,596.25 | 0.21 | 0.00 | 0.00 | 90.00 |
| 50% | 20,669.00 | 0.38 | 0.00 | 0.00 | 99.00 |
| 75% | 75,610.00 | 0.71 | 15,000,000.00 | 24,000,000.00 | 111.00 |
| max | 417,859.00 | 32.99 | 425,000,000.00 | 2,781,505,847.00 | 900.00 |

| | vote_count | vote_average | release_year | budget_adj | revenue_adj |
|---|---|---|---|---|---|
| count | 10,866.00 | 10,866.00 | 10,866.00 | 10,866.00 | 10,866.00 |
| mean | 217.39 | 5.97 | 2,001.32 | 17,551,039.82 | 51,364,363.25 |
| std | 575.62 | 0.94 | 12.81 | 34,306,155.72 | 144,632,485.04 |
| min | 10.00 | 1.50 | 1,960.00 | 0.00 | 0.00 |
| 25% | 17.00 | 5.40 | 1,995.00 | 0.00 | 0.00 |
| 50% | 38.00 | 6.00 | 2,006.00 | 0.00 | 0.00 |
| 75% | 145.75 | 6.60 | 2,011.00 | 20,853,251.08 | 33,697,095.72 |
| max | 9,767.00 | 9.20 | 2,015.00 | 425,000,000.00 | 2,827,123,750.41 |

**Finding** 1. About 50% of budget, revenue, budget_adj * revenue_adj have a value of 0. this will have big impact on the meanvalues. 2. Obvious outliars exist for runtime, and vote_count, and possibly populariy column.

**Identifying number of features with value 0 and combinations**

```
In [161]: (df['budget_adj'] == 0).sum()

Out[161]: 5696

In [162]: (df['revenue_adj'] == 0).sum()

Out[162]: 6016

In [163]: (df['runtime'] == 0).sum()

Out[163]: 31

In [164]: (df['revenue']==0).sum()

Out[164]: 6016

In [165]: (df['budget']==0).sum()

Out[165]: 5696

In [166]: ((df['budget_adj']==0) & (df['revenue_adj']==0)).sum()

Out[166]: 4701

In [167]: ((df['runtime']==0) & (df['revenue_adj']==0)).sum()

Out[167]: 31

In [168]: ((df['budget_adj']==0) & (df['runtime']==0)).sum()
```

```
Out[168]: 28
```

**Findings:** 1. budget_adj has the same number of missing values as budget column. smiliar case for revenue and revenue_adj 2. since budget_adj & revenue_adj cater for inflation, they should be better used for analysis.

**Check for unique values**

```
In [169]: df.nunique()
```

```
Out[169]: 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
```

**Findings:** 1. the original_title has 295 rows with duplicate titles, but remaining of the data is diffirent. so it's not a duplicate record (only title duplication) 2. the last 35 rows has strange characters probably used the foriegn language which got corrupted during laoding or conversion. this should not matter on the analysis but overall we lost the movies titles for those movies.

```
In [170]: titles=df.groupby('original_title').count()
          titles.tail(20)
```

```
Out[170]:                                      id  imdb_id  popularity  budget  \
          original_title
          çŋäÿåäžžååą                    1        1           1       1
          çżçÿçÿçä£ą                      1        1           1       1
          èğčæåťåç                       1        1           1       1
          èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ     1        1           1       1
          èèž                                1        1           1       1
          éżåďłé¡731                         1        1           1       1
          éżçď¿äij2ïijäżěåäÿžèťţ             1        1           1       1
```

| | | | | |
|---|---|---|---|---|
| éåŋęåĺé¿ | 1 | 1 | 1 | 1 |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 1 | 1 | 1 | 1 |
| ël¡íìčij | 1 | 1 | 1 | 1 |
| ìğíĺ | 1 | 1 | 1 | 1 |
| ìğìijëą êřë êÿÿ | 1 | 1 | 1 | 1 |
| ìžìźëŕÿ | 1 | 1 | 1 | 1 |
| ììì ìăì | 1 | 1 | 1 | 1 |
| ìíìă | 1 | 1 | 1 | 1 |
| ìêÿřì ë | 1 | 1 | 1 | 1 |
| ìăì í ì | 1 | 1 | 1 | 1 |
| íňí ììijëą | 1 | 1 | 1 | 1 |
| íìň Duelist | 1 | 1 | 1 | 1 |
| íìÿëğ | 1 | 1 | 1 | 1 |

| original_title | revenue | cast | homepage | director | \ |
|---|---|---|---|---|---|
| çŋäÿåäžžååą | 1 | 1 | 1 | 1 | |
| çżçÿçÿçä£ą | 1 | 1 | 0 | 1 | |
| èğčæåťåç | 1 | 1 | 0 | 1 | |
| èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ | 1 | 1 | 0 | 1 | |
| èèž | 1 | 1 | 0 | 1 | |
| éżåďłé¡731 | 1 | 1 | 0 | 1 | |
| éżçď¿äij2ïijäžěåäÿžèťţ | 1 | 1 | 0 | 1 | |
| éåŋęåĺé¿ | 1 | 1 | 0 | 1 | |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 1 | 1 | 0 | 1 | |
| ël¡íìčij | 1 | 1 | 0 | 1 | |
| ìğíĺ | 1 | 1 | 0 | 1 | |
| ìğìijëą êřë êÿÿ | 1 | 1 | 0 | 1 | |
| ìžìźëŕÿ | 1 | 1 | 0 | 1 | |
| ììì ìăì | 1 | 1 | 0 | 1 | |
| ìíìă | 1 | 1 | 1 | 1 | |
| ìêÿřì ë | 1 | 1 | 0 | 1 | |
| ìăì í ì | 1 | 1 | 0 | 1 | |
| íňí ììijëą | 1 | 1 | 0 | 1 | |
| íìň Duelist | 1 | 1 | 0 | 1 | |
| íìÿëğ | 1 | 1 | 1 | 1 | |

| original_title | tagline | keywords | overview | runtime | \ |
|---|---|---|---|---|---|
| çŋäÿåäžžååą | 0 | 1 | 1 | 1 | |
| çżçÿçÿçä£ą | 0 | 0 | 1 | 1 | |
| èğčæåťåç | 0 | 0 | 1 | 1 | |
| èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ | 1 | 0 | 1 | 1 | |
| èèž | 0 | 0 | 1 | 1 | |
| éżåďłé¡731 | 0 | 1 | 1 | 1 | |
| éżçď¿äij2ïijäžěåäÿžèťţ | 1 | 1 | 1 | 1 | |
| éåŋęåĺé¿ | 0 | 0 | 1 | 1 | |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 0 | 0 | 1 | 1 | |

| | | | | |
|---|---|---|---|---|
| ël¡íìčij | 0 | 0 | 1 | 1 |
| ìğíÍ | 0 | 1 | 1 | 1 |
| ìğìijëą êřë êÿÿ | 0 | 1 | 1 | 1 |
| ìžìźëŕÿ | 0 | 0 | 1 | 1 |
| ìíì ìăì | 0 | 1 | 1 | 1 |
| ìíìă | 1 | 1 | 1 | 1 |
| ìêÿřì ë | 0 | 0 | 1 | 1 |
| ìăì í ì | 0 | 1 | 1 | 1 |
| íňí ììijëą | 0 | 1 | 1 | 1 |
| íìň Duelist | 0 | 1 | 1 | 1 |
| íìÿëğ | 0 | 1 | 1 | 1 |

| original_title | genres | production_companies |
|---|---|---|
| çŋäÿåäžžååą | 1 | 1 |
| çżçÿçÿçä£ą | 1 | 1 |
| èğčæåťåç | 1 | 0 |
| èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ | 1 | 1 |
| èèž | 1 | 0 |
| éżåďłé¡731 | 1 | 1 |
| éżçď¿äij2ïijäžěåäÿžèťţ | 1 | 1 |
| éåŋęåĺé¿ | 1 | 0 |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 1 | 0 |
| ël¡íìčij | 1 | 1 |
| ìğíÍ | 1 | 1 |
| ìğìijëą êřë êÿÿ | 1 | 0 |
| ìžìźëŕÿ | 1 | 1 |
| ìíì ìăì | 1 | 0 |
| ìíìă | 1 | 1 |
| ìêÿřì ë | 0 | 0 |
| ìăì í ì | 1 | 0 |
| íňí ììijëą | 1 | 1 |
| íìň Duelist | 1 | 1 |
| íìÿëğ | 1 | 1 |

| original_title | release_date | vote_count | vote_average |
|---|---|---|---|
| çŋäÿåäžžååą | 1 | 1 | 1 |
| çżçÿçÿçä£ą | 1 | 1 | 1 |
| èğčæåťåç | 1 | 1 | 1 |
| èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ | 1 | 1 | 1 |
| èèž | 1 | 1 | 1 |
| éżåďłé¡731 | 1 | 1 | 1 |
| éżçď¿äij2ïijäžěåäÿžèťţ | 1 | 1 | 1 |
| éåŋęåĺé¿ | 1 | 1 | 1 |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 1 | 1 | 1 |
| ël¡íìčij | 1 | 1 | 1 |
| ìğíÍ | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| ìğìijëą êřë êÿÿ | 1 | 1 | 1 |
| ìžìźëŕÿ | 1 | 1 | 1 |
| ìíì ìăì | 1 | 1 | 1 |
| ìíìă | 1 | 1 | 1 |
| ìêÿřì ë | 1 | 1 | 1 |
| ìăì í ì | 1 | 1 | 1 |
| íňí ììijëą | 1 | 1 | 1 |
| íìň Duelist | 1 | 1 | 1 |
| íìÿëğ | 1 | 1 | 1 |

| original_title | release_year | budget_adj | revenue_adj |
|---|---|---|---|
| çŋäÿåäžžååą | 1 | 1 | 1 |
| çżçÿçÿçä£ą | 1 | 1 | 1 |
| èğčæåťåç | 1 | 1 | 1 |
| èş¡å¿ůåÂůåůťè (äÿ)  åďłé¡æ | 1 | 1 | 1 |
| èèž | 1 | 1 | 1 |
| éżåďłé¡731 | 1 | 1 | 1 |
| éżçď¿äij2ïijäżěåäÿžèťţ | 1 | 1 | 1 |
| éåŋęåĺé¿ | 1 | 1 | 1 |
| êşăìň ë ëšìğÿ ìťìijêÿř: êţììďìţ | 1 | 1 | 1 |
| ëł¡íìčij | 1 | 1 | 1 |
| ìğíĺ | 1 | 1 | 1 |
| ìğìijëą êřë êÿÿ | 1 | 1 | 1 |
| ìžìźëŕÿ | 1 | 1 | 1 |
| ìíì ìăì | 1 | 1 | 1 |
| ìíìă | 1 | 1 | 1 |
| ìêÿřì ë | 1 | 1 | 1 |
| ìăì í ì | 1 | 1 | 1 |
| íňí ììijëą | 1 | 1 | 1 |
| íìň Duelist | 1 | 1 | 1 |
| íìÿëğ | 1 | 1 | 1 |

## 1.2 Observations

The followin columns are not important for the data analysis and at the same time not fully populated, better of be removed: 1. homepage 2. tagline 3. overview 4. budget - the budget_adj is better use for analysis 5. revenue - revenue_adj is better use for analysis

Columns holding more than one value separated by "|" needs to be separated then joined together for comprehensive data view and analysis. 1. cast 2. director 3. genres 4. production_companies 5. keywords

release_data column shoud be datetime instead of object datatype

One duplicated row needs to be removed.

Many of values in the following columns set to 0, for proper analysis these values need to set to mean value of the column. i will also consider dropping these values

and compare results. 1. budget_adj 2. revenue_adj 3. budget 4. revenue 5. runtime The original_title column has some invalid characters, this will have no impact on the report since data will be assessed on other features.

Add new column to capture profit difference betweeen revenue_adj and budget_adj

Outliars values in 1. popularity value of 32 2. runtime of 900 minutes 3. vote_count of 9,767.00 4. revenue_adj of 2.8 Billions 5. budget_adj of 425 Millions

### 1.2.1 Data Cleaning (Step one Add profit column to datafram)

**Data Clearning - Step 1: Change floating number format to show decimals)**

```
In [171]: # set pandas options to see full number
          pd.options.display.float_format = '{:,.2f}'.format
          # value results
          df.head(1)
```

```
Out[171]:        id    imdb_id  popularity     budget     revenue  original_title  \
          0  135397  tt0369610       32.99  150000000  1513528810  Jurassic World

                                                    cast  \
          0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...

                              homepage         director            tagline  \
          0  http://www.jurassicworld.com/  Colin Trevorrow  The park is open.

                 ...                                                overview runtime  \
          0      ...         Twenty-two years after the events of Jurassic ...     124

                                              genres  \
          0  Action|Adventure|Science Fiction|Thriller

                                        production_companies release_date vote_count  \
          0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562

             vote_average  release_year     budget_adj      revenue_adj
          0          6.50          2015 137,999,939.28 1,392,445,892.52

          [1 rows x 21 columns]
```

**Data Clearning - Step 2: change dtype for release_date to datetime)**

```
In [172]: df['release_date']=pd.to_datetime(df['release_date'])
          df['release_date'].head(1)
```

```
Out[172]: 0   2015-06-09
          Name: release_date, dtype: datetime64[ns]
```

### Data Clearning - Step 3: Drop unwanted columns

```
In [173]: df.drop(['budget','revenue','homepage','tagline','overview'],axis=1, inplace=True)
```

### Data Clearning - Step1: setting option 1 dataframe df

```
In [174]: #First replace 0 with NaN
          df['revenue_adj'].replace(0, np.NAN, inplace=True)
          df['budget_adj'].replace(0, np.NAN, inplace=True)
          df['runtime'].replace(0, np.NAN, inplace=True)

          #Second update NaN with mean values
          df['budget_adj'].fillna(df['budget_adj'].mean(), inplace=True)
          df['revenue_adj'].fillna(df['revenue_adj'].mean(), inplace=True)
          df['runtime'].fillna(df['runtime'].mean(), inplace=True)

          # add new profit column
          df['profit']=df['revenue_adj']-df['budget_adj']

          #remove duplicates
          df.drop_duplicates(inplace=True)
          df.duplicated().sum()
          df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 17 columns):
id                      10865 non-null int64
imdb_id                 10855 non-null object
popularity              10865 non-null float64
original_title          10865 non-null object
cast                    10789 non-null object
director                10821 non-null object
keywords                9372 non-null object
runtime                 10865 non-null float64
genres                  10842 non-null object
production_companies    9835 non-null object
release_date            10865 non-null datetime64[ns]
vote_count              10865 non-null int64
vote_average            10865 non-null float64
release_year            10865 non-null int64
budget_adj              10865 non-null float64
revenue_adj             10865 non-null float64
profit                  10865 non-null float64
dtypes: datetime64[ns](1), float64(6), int64(3), object(7)
memory usage: 1.5+ MB
```

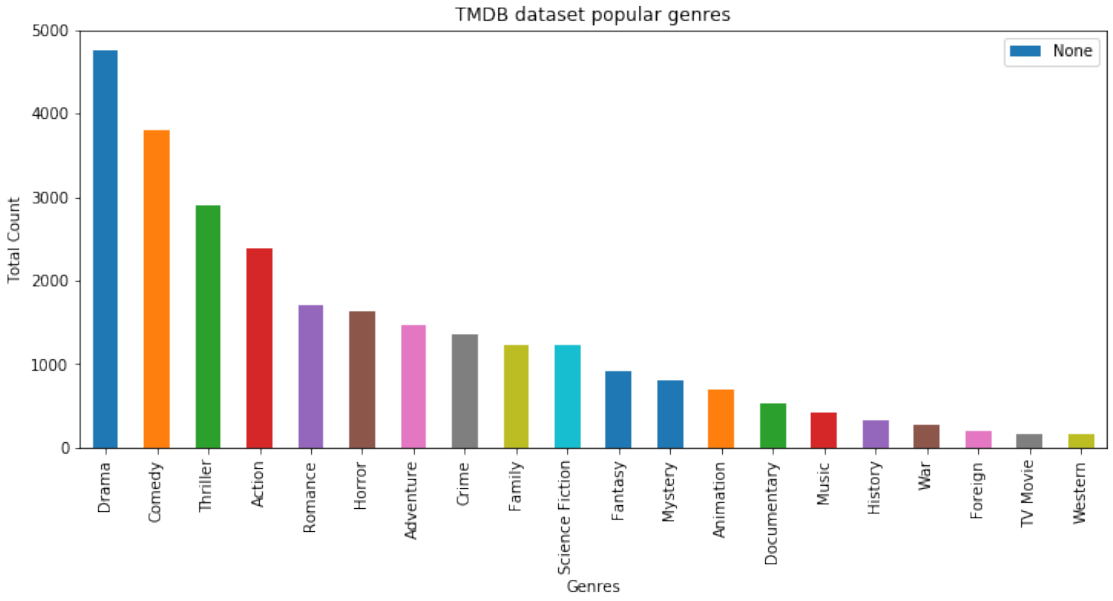
## Exploratory Data Analysis

**Tip**: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

### 1.2.2 Q1 - Genres populareties top to bottom based on entire dataset?

```
In [175]: # Extract all entries in genres column as one long string
          data = df['genres'].str.cat(sep = '|')
          # Split genres into a Pandas Series
          data = pd.Series(data.split('|'))
          # get Value count each genere
          count = data.value_counts(ascending = False)
          ax=count.plot.bar(x=count.index, y=count,figsize=(12,5), legend=True, title='TMDB data
          ax.set_xlabel("Genres")
          ax.set_ylabel("Total Count")
          plt.figure(figsize=(20,10))
```

```
Out[175]: <matplotlib.figure.Figure at 0x7fa9aefbb588>
```




```
<matplotlib.figure.Figure at 0x7fa9aefbb588>
```

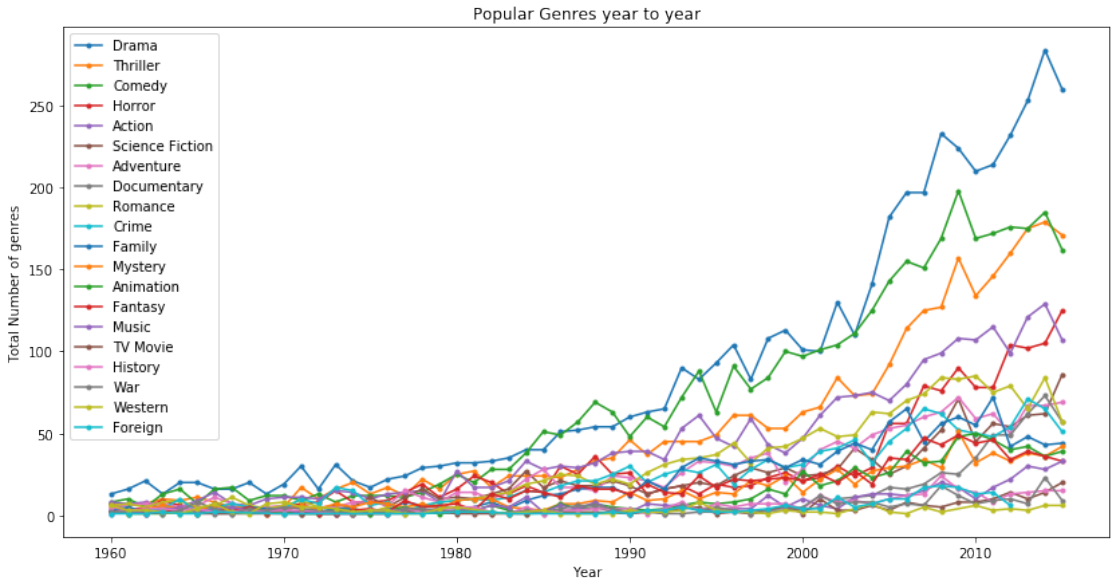
### 1.2.3 Q2: Movie genres over time - Most populare Genres

```
In [176]: year_list=[]
          genres_list=[]
```

```
count_list=[]
common_genres=[]
all_data_list_index=[]
all_data_list_values=[]
all_data_list_years=[]
all_data_dict={}

def popular_genres_year(year, column_name):
    data = df[df['release_year']==year][column_name].str.cat(sep = '|')
    # Create pandas series and store the values separately
    data = pd.Series(data.split('|'))
    # Display value count in descending order
    count = data.value_counts(ascending = False)
    #all_data_list_year=[]
    all_data_list_year=np.repeat(year,len(count.index.tolist())).tolist()
    all_data_list_index.extend(count.index.tolist())
    all_data_list_values.extend(count.tolist())
    all_data_list_years.extend(all_data_list_year)
    return

years = df['release_year'].unique()
for year in years:
    popular_genres_year(year,'genres')

my_dict={'year':all_data_list_years, 'genres':all_data_list_index, 'count':all_data_li

# genres_list
df_dict=pd.DataFrame(my_dict)

gens = df_dict['genres'].unique()

plt.figure(figsize=(14, 7))
plt.xlabel('Year')
plt.ylabel('Total Number of genres')
plt.title('Popular Genres year to year')
for n in gens:
    df_temp=[]
    df_temp = df_dict[df_dict['genres']==n]
    df_temp.sort_values(by=['year'], ascending=True, inplace=True)
    x_vals =df_temp['year'].tolist()
    y_vals=df_temp['count'].tolist()
    plt.plot(x_vals, y_vals, '.-', label=n)
plt.legend()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:41: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

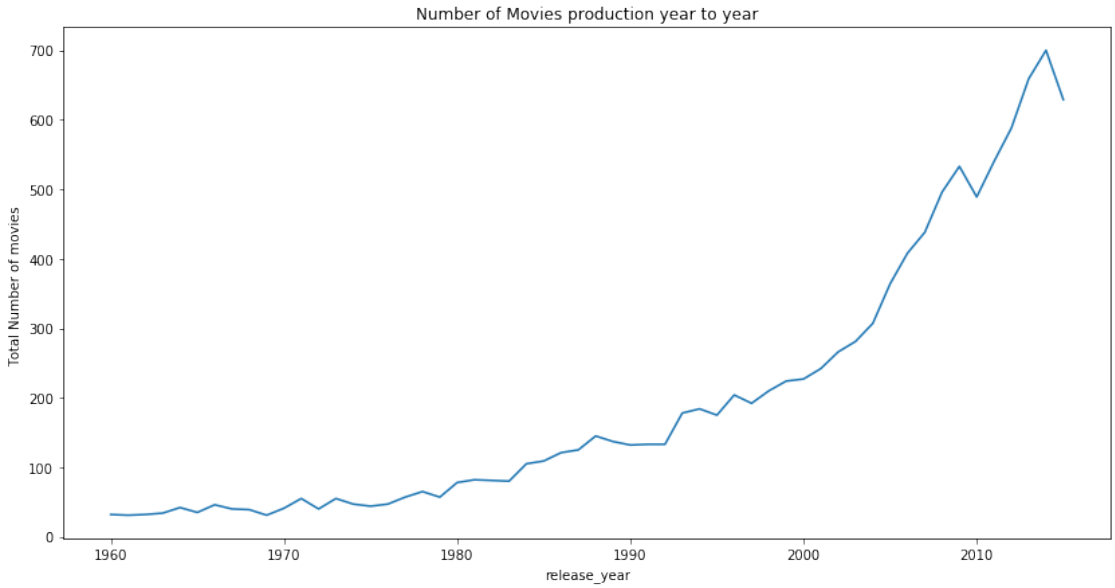
Out[176]: <matplotlib.legend.Legend at 0x7fa9acdac0b8>




### 1.2.4 Q3: Number of movies release by year

```
In [177]: plt.figure(figsize=(14, 7))
          plt.xlabel(' Year ')
          plt.ylabel('Total Number of movies ')
          plt.title('Number of Movies production year to year')
          df.groupby('release_year')['id'].count().plot()
```

Out[177]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa9acde61d0>




```
In [178]: df.groupby('release_year')['id'].count()
```

```
Out[178]: release_year
          1960     32
          1961     31
          1962     32
          1963     34
          1964     42
          1965     35
          1966     46
          1967     40
          1968     39
          1969     31
          1970     41
          1971     55
          1972     40
          1973     55
          1974     47
          1975     44
          1976     47
          1977     57
          1978     65
          1979     57
          1980     78
          1981     82
          1982     81
          1983     80
          1984    105
```

```
1985    109
1986    121
1987    125
1988    145
1989    137
1990    132
1991    133
1992    133
1993    178
1994    184
1995    175
1996    204
1997    192
1998    210
1999    224
2000    227
2001    242
2002    266
2003    281
2004    307
2005    364
2006    408
2007    438
2008    496
2009    533
2010    489
2011    540
2012    588
2013    659
2014    700
2015    629
Name: id, dtype: int64
```

### 1.3 Q4: Top 10 Directors with most number of movies

```
In [179]: directors_list = df['director'].value_counts()
          directors_list[directors_list > 20].head(10)
```

```
Out[179]: Woody Allen          45
          Clint Eastwood       34
          Martin Scorsese      29
          Steven Spielberg     29
          Ridley Scott         23
          Steven Soderbergh    22
          Ron Howard           22
          Joel Schumacher      21
          Name: director, dtype: int64
```

### 1.4 check which of the directors also is top in profit

```
In [180]: df.groupby('director')['profit'].sum().sort_values(ascending=False)[:10]
```

```
Out[180]: director
          Steven Spielberg   13,280,831,549.61
          James Cameron       6,489,495,710.14
          Peter Jackson       5,874,277,821.26
          George Lucas        5,844,159,207.50
          Robert Zemeckis     4,520,236,846.64
          Chris Columbus      4,435,946,487.26
          Michael Bay         3,958,675,886.72
          David Yates         3,472,619,726.15
          Tim Burton          3,471,916,282.25
          Ron Howard          3,327,018,018.88
          Name: profit, dtype: float64
```

### 1.5 Q5: Max and Min movies profits

```
In [181]: max_profit=df.iloc[df['profit'].idxmax()]
          min_profit=df.iloc[df['profit'].idxmin()]
          max_min = {'max':max_profit, 'min':min_profit}
          df_max_min_profit=pd.DataFrame(max_min)
          df_max_min_profit
```

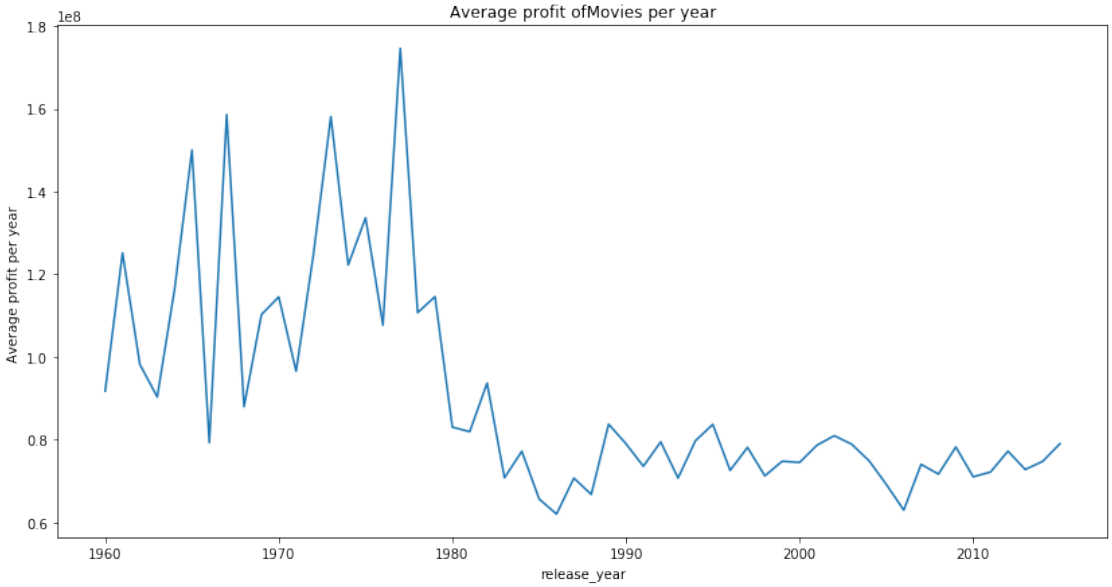
```
Out[181]:                                                                  max  \
          id                                                                11
          imdb_id                                                     tt0076759
          popularity                                                      12.04
          original_title                                              Star Wars
          cast                  Mark Hamill|Harrison Ford|Carrie Fisher|Peter ...
          director                                                   George Lucas
          keywords                    android|galaxy|hermit|death star|lightsaber
          runtime                                                        121.00
          genres                                  Adventure|Action|Science Fiction
          production_companies   Lucasfilm|Twentieth Century Fox Film Corporation
          release_date                                         1977-03-20 00:00:00
          vote_count                                                        4428
          vote_average                                                      7.90
          release_year                                                      1977
          budget_adj                                              39,575,591.36
          revenue_adj                                          2,789,712,242.28
          profit                                               2,750,136,650.92

                                                                        min
          id                                                          44992
          imdb_id                                                 tt1433813
          popularity                                                   0.38
          original_title                                          Hubble 3D
```

```
cast                                          Leonardo DiCaprio
director                                             Toni Myers
keywords                space|imax|space shuttle|woman director|3d
runtime                                                  44.00
genres                                             Documentary
production_companies                  Warner Bros.|IMAX Space Ltd.
release_date                                  2010-03-19 00:00:00
vote_count                                                    31
vote_average                                                6.50
release_year                                                2010
budget_adj                                          36,887,736.70
revenue_adj                                        115,077,354.87
profit                                              78,189,618.17
```

### 1.5.1 Q6: Average profit per year

```
In [182]: plt.figure(figsize=(14, 7))
          plt.xlabel(' Year ')
          plt.ylabel('Average profit per year ')
          plt.title('Average profit ofMovies per year')
          df.groupby('release_year')['profit'].mean().plot()
```

```
Out[182]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa9af628748>
```



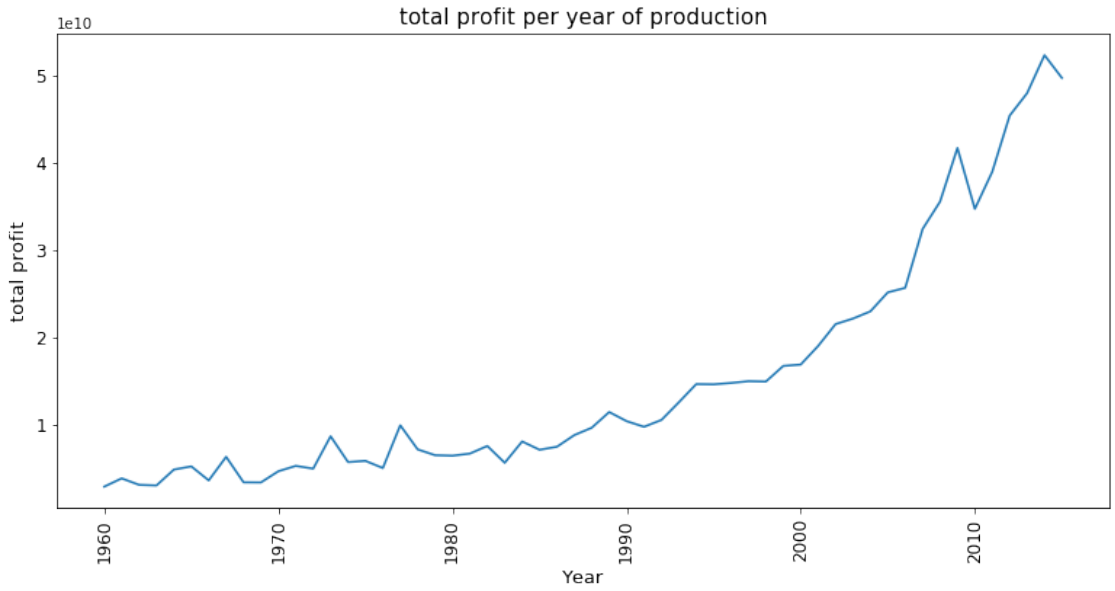

Highes profit between 1960 and 1990

## 1.6 Q7: total profit over the years

```
In [183]: profit_by_year = df.groupby('release_year')['profit'].sum()
```

```
profit_by_year.plot(kind='line', figsize=(13,6),fontsize=12)
plt.title("total profit per year of production",fontsize=15)
plt.xticks(rotation = 90)
plt.xlabel('Year',fontsize=13)
plt.ylabel("total profit",fontsize= 13)

#figure size(width, height)
plt.figure(figsize=(12,6), dpi = 130)
```

Out[183]: <matplotlib.figure.Figure at 0x7fa9acd84c88>




<matplotlib.figure.Figure at 0x7fa9acd84c88>

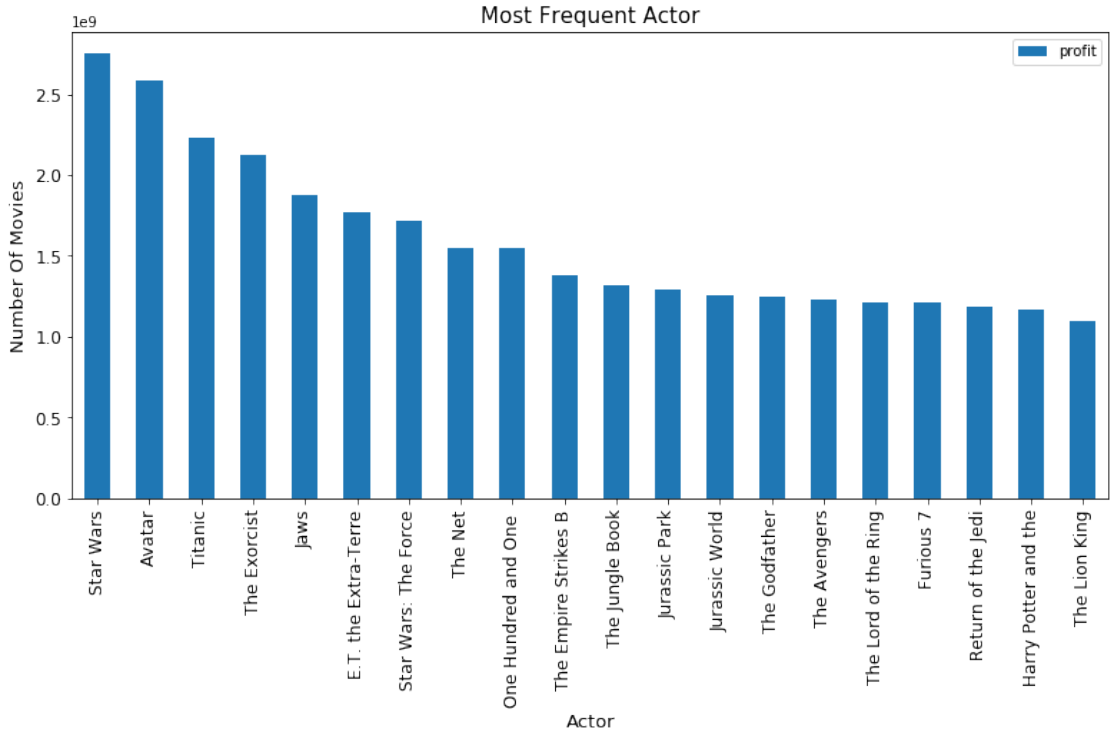
Finding: This is due to the number of movies increase

### 1.6.1 Q8: top 10 movies with highest profit

In [184]:
```
#make a plot which contain top 10 movies which earn highest profit.
#sort the 'Profit' column in decending order and store it in the new dataframe,
profit_df = pd.DataFrame(df['profit'].sort_values(ascending = False))
profit_df['original_title'] = df['original_title']
# limit title lengh
profit_df['original_title']=profit_df['original_title'].apply(lambda x:x[0:20])
profit_df.set_index('original_title', inplace=True)
profit_df.iloc[:20].plot.bar(figsize=(13,6),fontsize=12)
plt.title("Most Frequent Actor",fontsize=15)
plt.xticks(rotation = 90)
```

```
plt.xlabel('Actor',fontsize=13)
plt.ylabel("Number Of Movies",fontsize= 13)
```

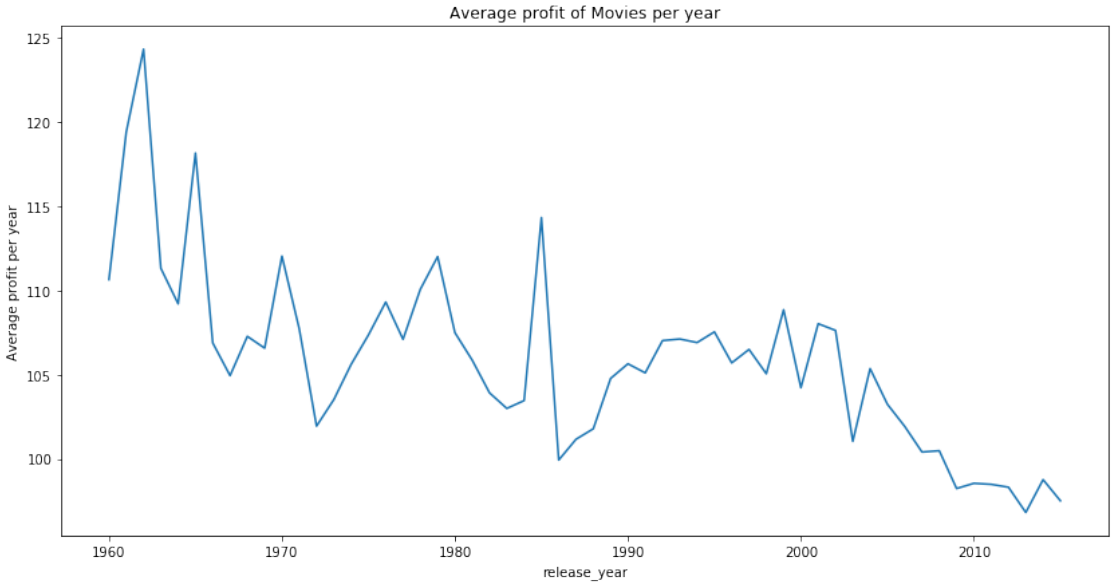
Out[184]: Text(0,0.5,'Number Of Movies')




## 1.7 Q9: Average movie runtime from year to year

In [185]:
```
plt.figure(figsize=(14, 7))
plt.xlabel(' Year ')
plt.ylabel('Average profit per year')
plt.title('Average profit of Movies per year')
df.groupby('release_year')['runtime'].mean().plot()
```

Out[185]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa9b4971da0>




### 1.8 Q10: Actors participated in highest number of movies

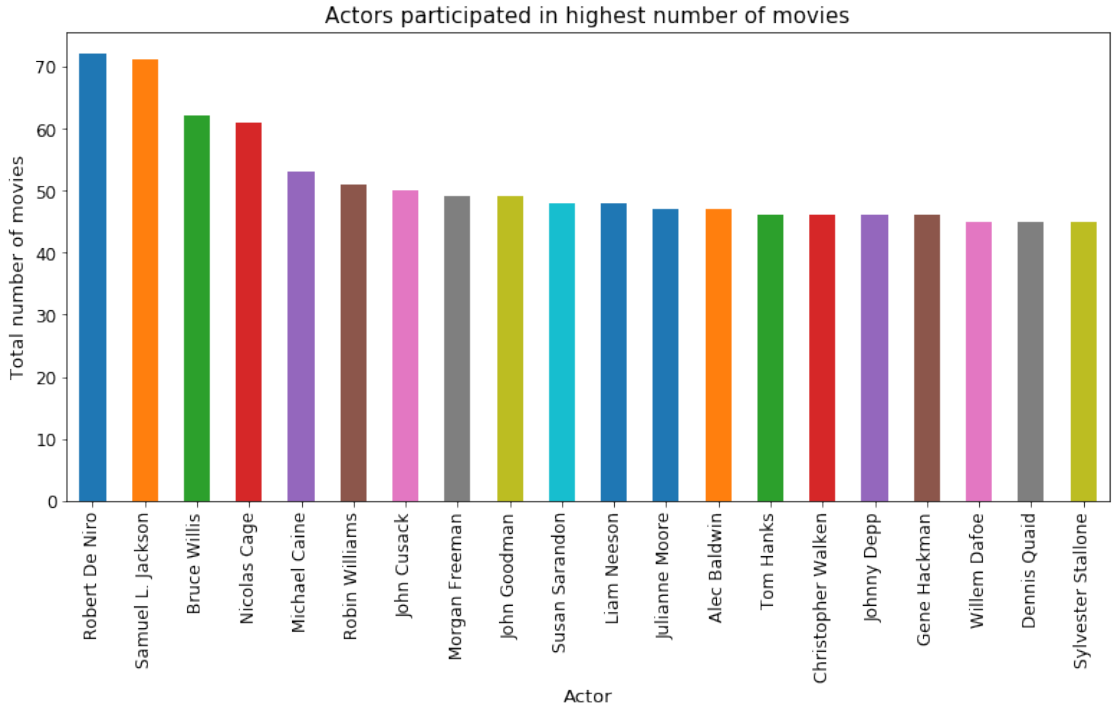
```
In [186]: # Extract all entries in actors column as one long string
          data = df['cast'].str.cat(sep = '|')
          # Split acrtor into a Pandas Series
          data = pd.Series(data.split('|'))
          # get Value count each actor
          count = data.value_counts(ascending = False)
          count[:10]
```

```
Out[186]: Robert De Niro       72
          Samuel L. Jackson    71
          Bruce Willis         62
          Nicolas Cage         61
          Michael Caine        53
          Robin Williams       51
          John Cusack          50
          Morgan Freeman       49
          John Goodman         49
          Susan Sarandon       48
          dtype: int64
```

```
In [187]: # Extract all entries in actors column as one long string
          data = df['cast'].str.cat(sep = '|')
          # Split acrtor into a Pandas Series
          data = pd.Series(data.split('|'))
          # get Value count each actor
          count = data.value_counts(ascending = False)
```

```
count.iloc[:20].plot.bar(figsize=(13,6),fontsize=12)
plt.title("Actors participated in highest number of movies",fontsize=15)
plt.xticks(rotation = 90)
plt.xlabel('Actor',fontsize=13)
plt.ylabel("Total number of movies",fontsize= 13)
```

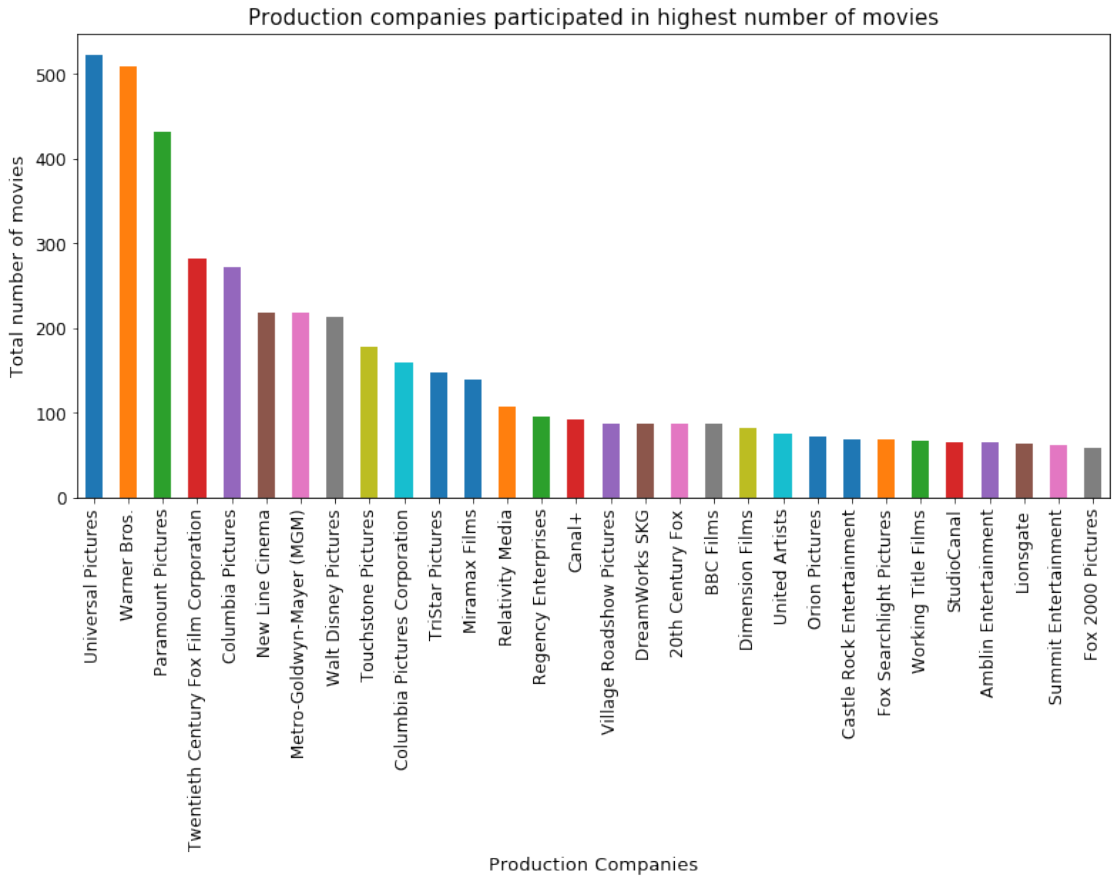
Out[187]: Text(0,0.5,'Total number of movies')




## 1.9 Q11. top production companies with highest number of movies

```
# Extract all entries in production company column as one long string
data = df['production_companies'].str.cat(sep = '|')
# Split production company name into a Pandas Series
data = pd.Series(data.split('|'))
# get Value count each production company
count = data.value_counts(ascending = False)

count.iloc[:30].plot.bar(figsize=(13,6),fontsize=12)
plt.title("Production companies participated in highest number of movies",fontsize=15)
plt.xticks(rotation = 90)
plt.xlabel('Production Companies',fontsize=13)
plt.ylabel("Total number of movies",fontsize= 13)
```

Out[188]: Text(0,0.5,'Total number of movies')



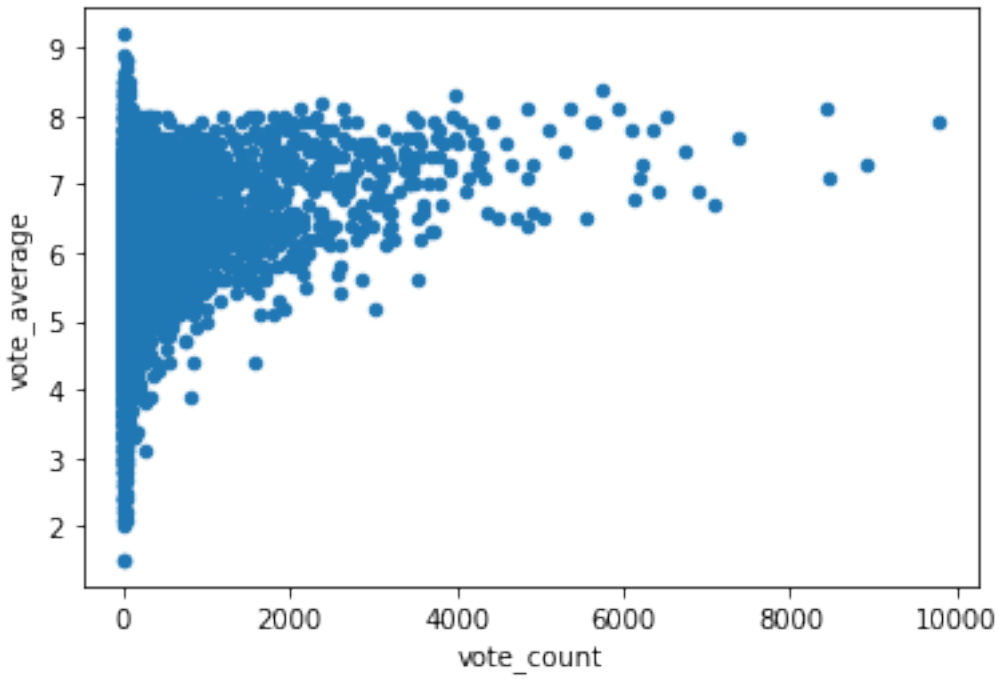

## 1.10 Q12: relation between various features in the dataset

```
In [189]: df_votes=df[['vote_count','vote_average']]#.describe['vote_count','vote_average'].desc
          df_votes.describe()
```

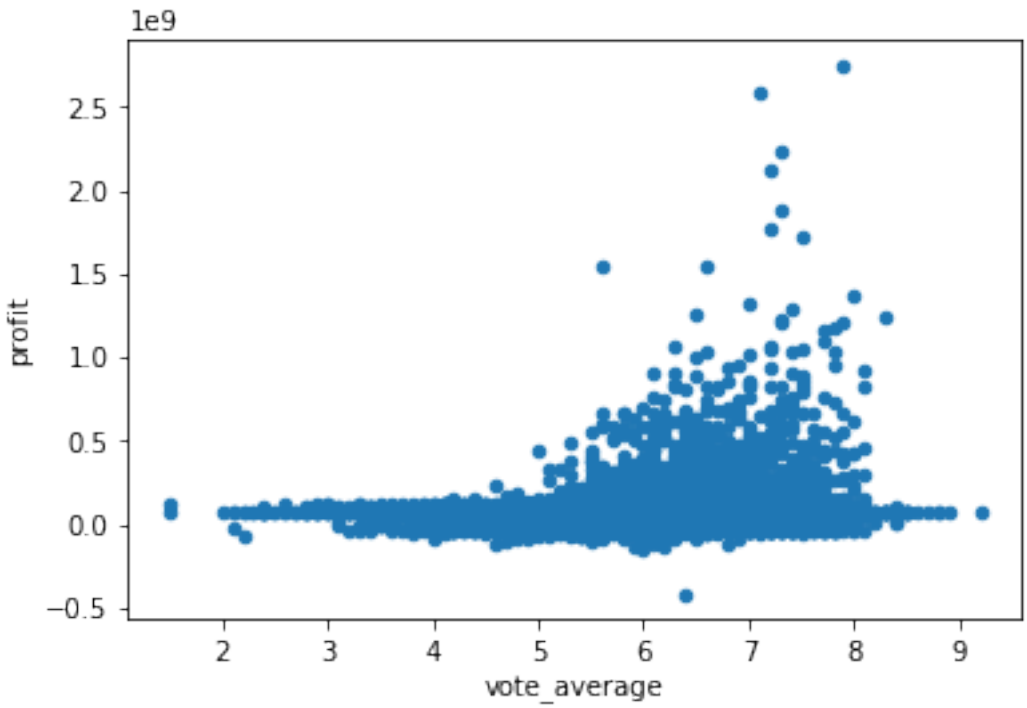
Out[189]:

| | vote_count | vote_average |
|---|---|---|
| count | 10,865.00 | 10,865.00 |
| mean | 217.40 | 5.98 |
| std | 575.64 | 0.94 |
| min | 10.00 | 1.50 |
| 25% | 17.00 | 5.40 |
| 50% | 38.00 | 6.00 |
| 75% | 146.00 | 6.60 |
| max | 9,767.00 | 9.20 |

```
In [190]: df_votes.plot(x='vote_count', y='vote_average', kind='scatter');
```



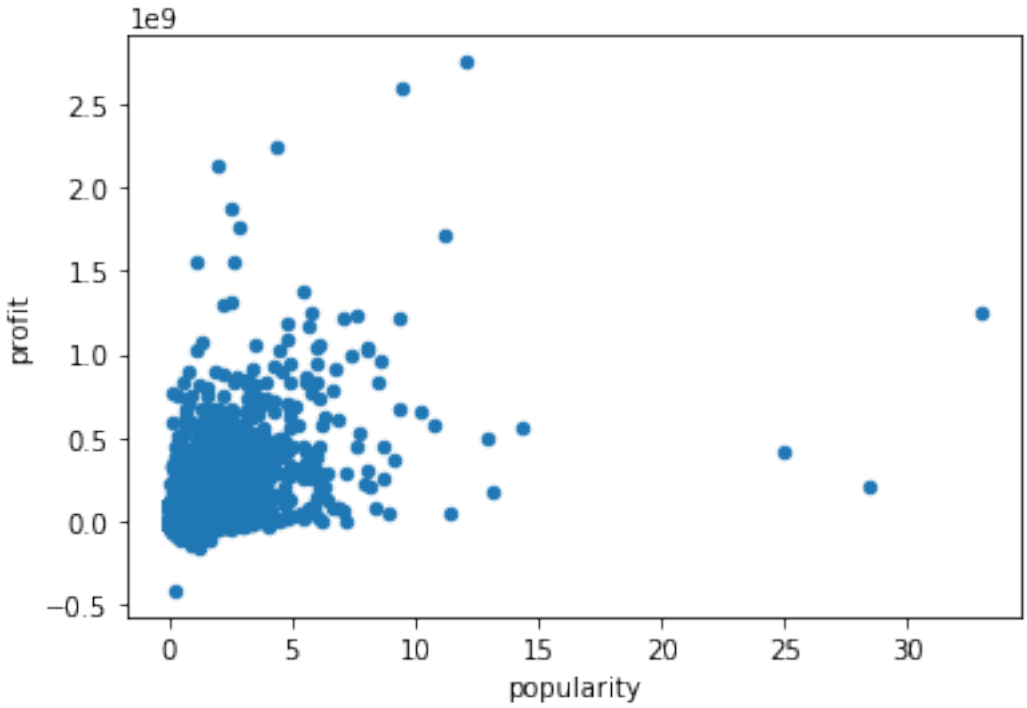

```
In [191]: df_votes['vote_count'].corr(df_votes['vote_average'])

Out[191]: 0.25381786600245509

In [192]: df_avg_votes_profit=df[['vote_average','profit']]
          df_avg_votes_profit.describe()
          df_avg_votes_profit.plot(x='vote_average', y='profit', kind='scatter');
```




```
In [193]: df_avg_votes_profit['vote_average'].corr(df_avg_votes_profit['profit'])
```

```
Out[193]: 0.1351776672548729
```

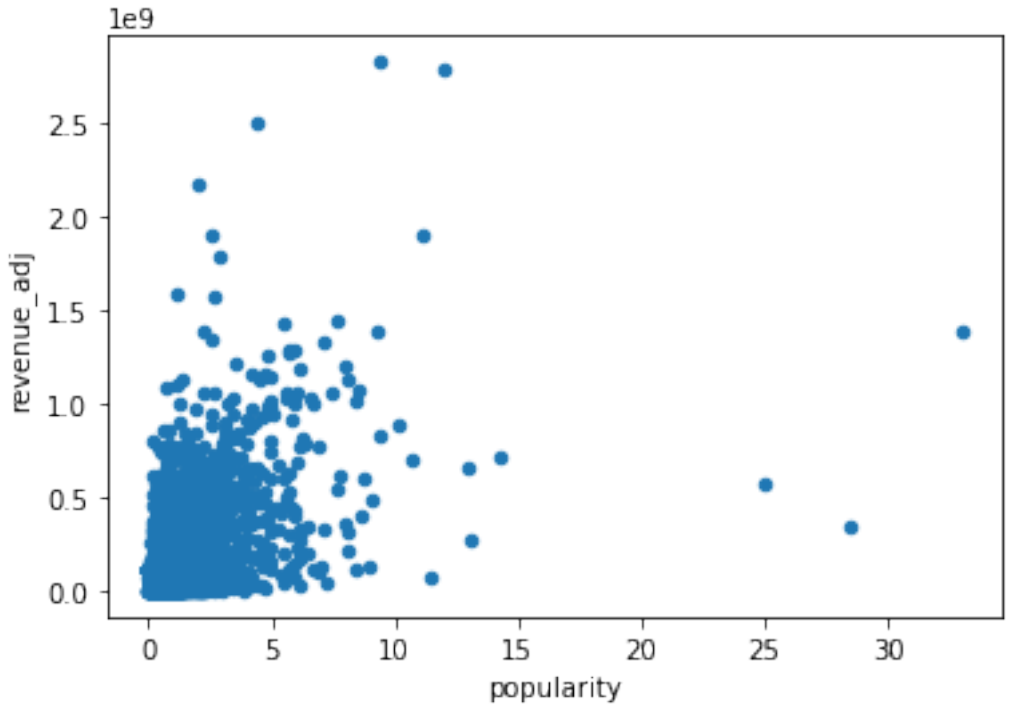
```
In [194]: df_popularity_profit=df[['popularity','profit']]
          df_popularity_profit.describe()
          df_popularity_profit.plot(x='popularity', y='profit', kind='scatter');
```




```
In [195]: df_popularity_profit['popularity'].corr(df_popularity_profit['profit'])
```

```
Out[195]: 0.46644745169735652
```

```
In [196]: df_popularity_revenue=df[['popularity','revenue_adj']]
          df_popularity_revenue.describe()
          df_popularity_revenue.plot(x='popularity', y='revenue_adj', kind='scatter');
```




```
In [197]: df_popularity_revenue['popularity'].corr(df_popularity_revenue['revenue_adj'])
```

```
Out[197]: 0.509041785282606 13
```

there dones't seem to be a big relationship between votes and profit, better correlation between popularity and profit but not very big and can't be used as an indicator

## Conclusions

1. Drama, comedy and thriller are the top populare genres
2. The top genres (Drama, Comedy, and Thriller) also continue to increase in popularity over time as seen in Q2 plot
3. Number of movies is growing exponentially over time until 2014, then a big drop happened in 2015. not clear why
4. Out of the top 10 directors, only two who made it to the top 10 most profitable
5. Steven Spielberg as number one in profitability, Ron Howard comes at 10th most profitable director
6. Star Wars is the all time most profitable movie ever. and Hubble 3D comes the least profitable
7. Average profit per year spike between 1960, and 1990, then it flaten. Total profit on the other hand keeps increasing over time. this is due to increase number of movies everyday except for 2015 which had a drop in the number of movies and profitablity as well.
8. Top 3 most profitable movies, start wars, Avatar, and Titanic
9. Average runtime of the movies are decreasing year by year.
10. Warner Bros, Universal Pictures, and Paramount Pictures have the lion share of movie production. Robert De Niro, Samuel L. Jackson, Bruce Willis, and Nicolas Cage are the most populare actors with over 60 movies.

### 1.11 Limitations:

1. there are huge number of movies with missing data specially in the budget and revenue columns.
2. there are not clear correlation or indication to what features are the secret to produce a high profitable movie

In [ ]: