Investigate_a_Dataset

July 13, 2022

1 Project: Investigate a Dataset - [The Movie Database (TMDb)]

1.1 Table of Contents

1.1.1 Dataset Description

If you are interested in movies production , this dataset will be a good refrence for you as it contains information about 10 thousands movies collected from The Movie Database (TMDb)that includes information about revenue ,budget, production companies and many other information that will give you a good overview about movies making.

columns:

id

imdb_id

Popularity: no. of people who like the movie

Budget :the cost production of the movie

Revenue :the income for each movie

Original_title: name of the movie

Cast: the actors and actresses who act in the movie

Homepage: the website of the movie

Director: name of the director

Tagline: slogan represent the idea of the movie

Overview:The idea of the movie

key words:words help in searching the movie online.

Run time: movie's time duration

Geners: The movie's type

Production Companies:the company that produced the movie

Release date:the date of the the first show at cinemas

Vote_count: number of people who vote for the movie

Vote_average:percentage

Release_year: year that movie showed in cinemas

Budget_adj:budget of the associated movie in terms of 2010 dollars, accounting for inflation over time.

Revenue-adj:revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

1.1.2 Question(s) for Analysis

What is the distribution of revenues?

What is the relation between the budget and the revenue?

What is the relation between the run time and averge vote?

What is the higher revenue in year 2010?

How many movies that its average vote is 7 or more?

How many movies that produced starting from year 2000 ?

```
In [1]: # Use this cell to set up import statements for all of the packages that you
        #   plan to use.

        # Remember to include a 'magic word' so that your visualizations are plotted
        #   inline with the notebook. See this page for more:
        #   http://ipython.readthedocs.io/en/stable/interactive/magics.html
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        % matplotlib inline
```

```
In [2]: # Upgrade pandas to use dataframe.explode() function.
        !pip install --upgrade pandas
```

```
Requirement already up-to-date: pandas in /opt/conda/lib/python3.6/site-packages (1.1.5)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.7.3 in /opt/conda/lib/python
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /opt/conda/lib/python3.6/site-p
Requirement already satisfied, skipping upgrade: numpy>=1.15.4 in /opt/conda/lib/python3.6/site-
Requirement already satisfied, skipping upgrade: six>=1.5 in /opt/conda/lib/python3.6/site-packa
```

Data Wrangling

**Here we will load the data ,check for cleanliness, and then trim and clean the dataset for analysis

1.1.3 General Properties

The checklist we will follow to clean the data:

check the data type of each column.

check the missing data.

check the duplication.

Load the data

```
In [3]: # Load  data, Perform operations to inspect data
        #   types and look for instances of missing or possibly errant data.
        df =pd.read_csv('tmdb-movies.csv')
        df.head()
```

Out[3]:

	id	imdb_id	popularity	budget	revenue	original_title	cast
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...

```
                                            homepage          director  \
0                    http://www.jurassicworld.com/   Colin Trevorrow
1                      http://www.madmaxmovie.com/     George Miller
2    http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod...       J.J. Abrams
4                         http://www.furious7.com/         James Wan

                        tagline  ...  \
0             The park is open.  ...
1            What a Lovely Day.  ...
2    One Choice Can Destroy You  ...
3  Every generation has a story.  ...
4           Vengeance Hits Home  ...

                                          overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137

                                     genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                     Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

Get the number of columns and rows of the dataset

```
In [4]: df.shape
```

Out[4]: (10866, 21)

Get the statistics of each column like count, mean,standard deviation ,minimun and maximum values.

In [5]: df.describe()

Out[5]:

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

	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086600e+04
mean	217.389748	5.974922	2001.322658	1.755104e+07	5.136436e+07
std	575.619058	0.935142	12.812941	3.430616e+07	1.446325e+08
min	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+00
25%	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+00
50%	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+00
75%	145.750000	6.600000	2011.000000	2.085325e+07	3.369710e+07
max	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

Get some information about the dataset like type of each column and count of non null values in each column

In [6]: 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
```

Get number of unique elements without null values in each column.

```
In [7]: df.nunique()
```

```
Out[7]: id                    10865
        imdb_id               10855
        popularity            10814
        budget                  557
        revenue                4702
        original_title        10571
        cast                  10719
        homepage               2896
        director               5067
        tagline                7997
        keywords               8804
        overview              10847
        runtime                 247
        genres                 2039
        production_companies   7445
        release_date           5909
        vote_count             1289
        vote_average             72
        release_year             56
        budget_adj             2614
        revenue_adj            4840
        dtype: int64
```

Get count of null values in each column in the dataset.

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

```
Out[8]: 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
```

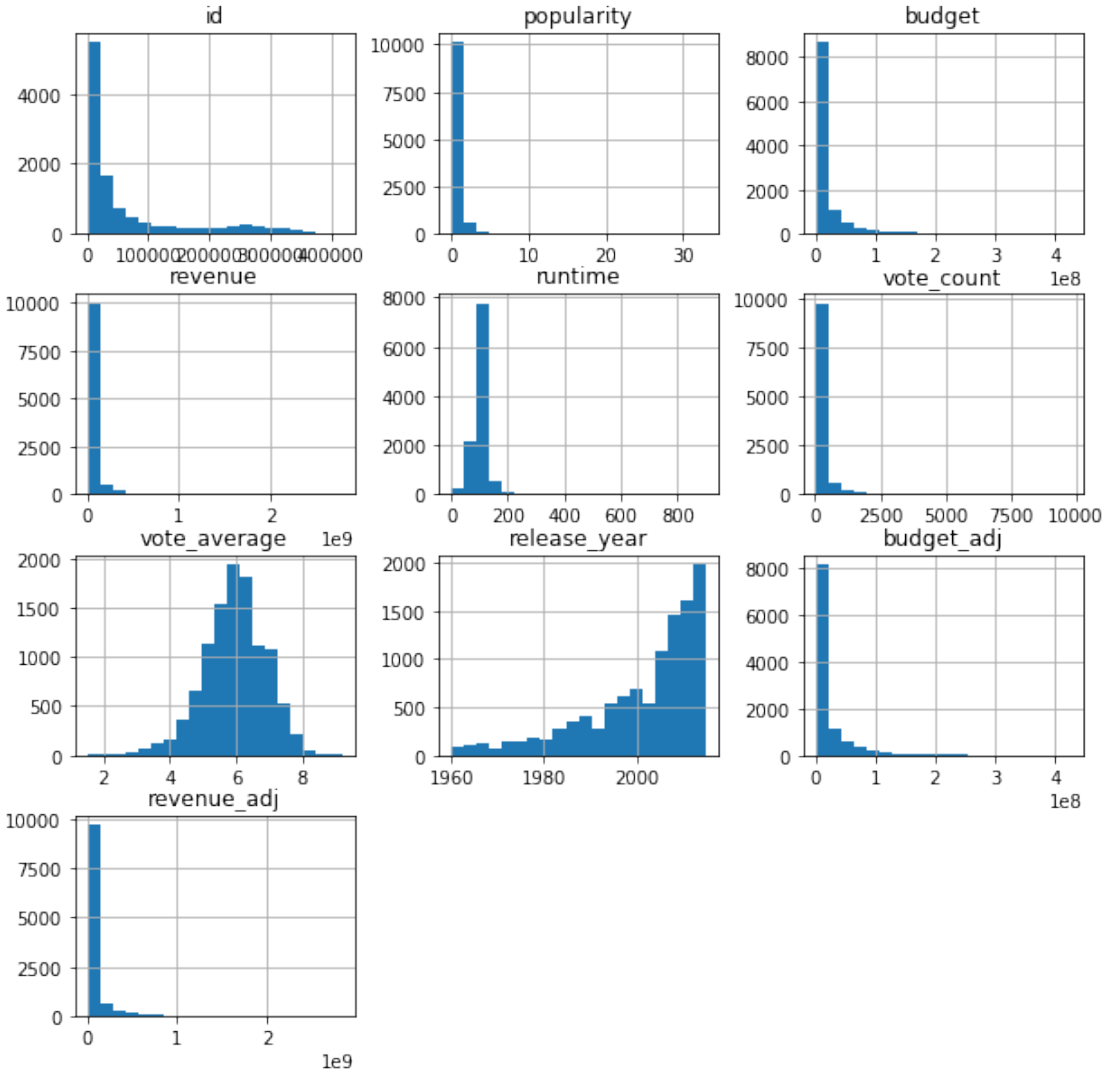
Get the sum of duplicated rows in the data set

```
In [9]: df.duplicated().sum()
```

```
Out[9]: 1
```

Get distribution of numeric data type in the dataset.

```
In [10]: df.hist(figsize=(10,10),bins=20);
```



Drop the columns that won't be used in exploratory data analysis

```
In [14]: to_drop=['id','imdb_id','popularity','original_title','cast','homepage','director',
                  'tagline','overview','keywords','genres','production_companies','release_date'
                  'vote_count','budget_adj','revenue_adj']
         df.drop(to_drop, inplace=True, axis=1)

In [15]: df.head()

Out[15]:
```

	budget	revenue	runtime	vote_average	release_year
0	150000000	1513528810	124	6.5	2015
1	150000000	378436354	120	7.1	2015
2	110000000	295238201	119	6.3	2015
3	200000000	2068178225	136	7.5	2015
4	190000000	1506249360	137	7.3	2015

1.1.4 Data Cleaning

We will perform the following to clean the dataset

Convert the datatype of release_year from integer to string.

Drop duplicated rows.

Drop columns that we won't use in exploratory data analysis as we did in the previous step.

```
In [16]: # After discussing the structure of the data and any problems that need to be
         #   cleaned, perform those cleaning steps in the second part of this section.
         #Convert the datatype of release_year from integer to string
         df['release_year']=df['release_year'].astype(str)
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   budget        10866 non-null  int64
 1   revenue       10866 non-null  int64
 2   runtime       10866 non-null  int64
 3   vote_average  10866 non-null  float64
 4   release_year  10866 non-null  object
dtypes: float64(1), int64(3), object(1)
memory usage: 424.6+ KB
```

```
In [17]: # drop duplicated rows
         df.drop_duplicates(inplace=True)
```

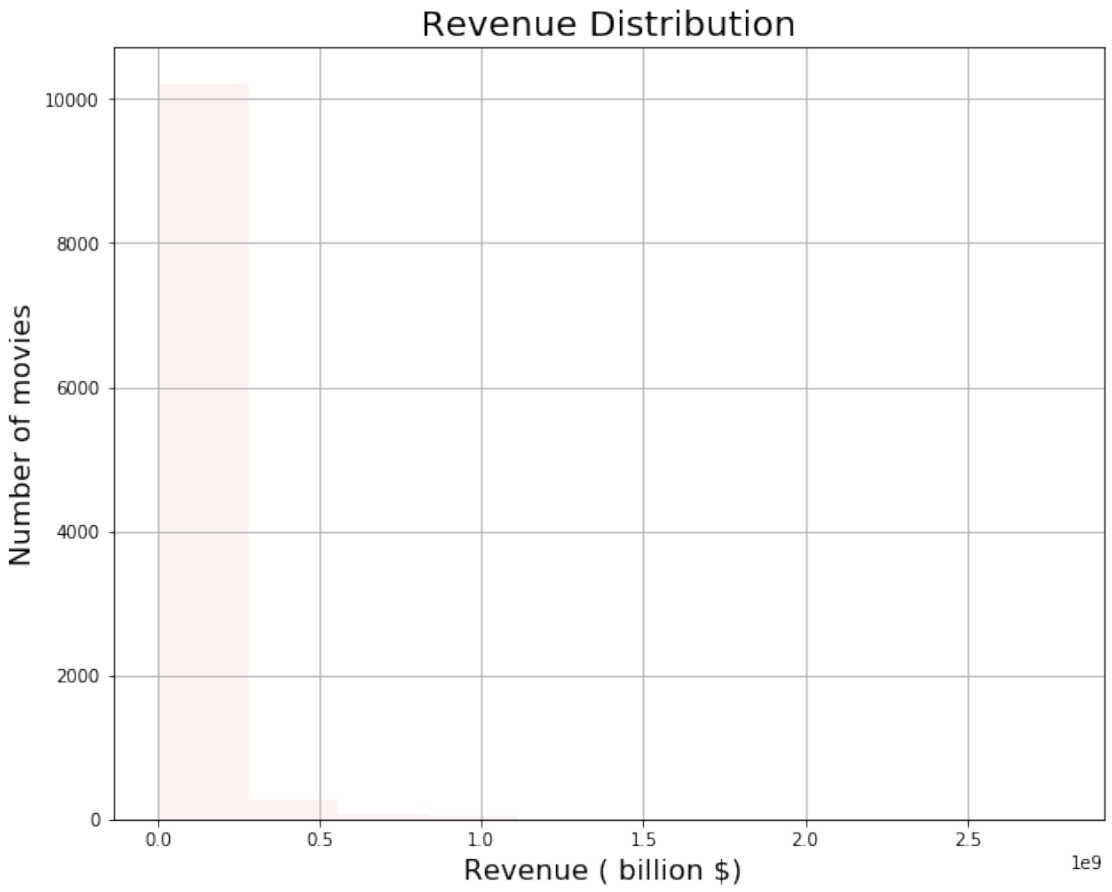
```
In [18]: df.duplicated().sum()
```

```
Out[18]: 0
```

Exploratory Data Analysis

1.1.5 What is the distribution of revenues ?

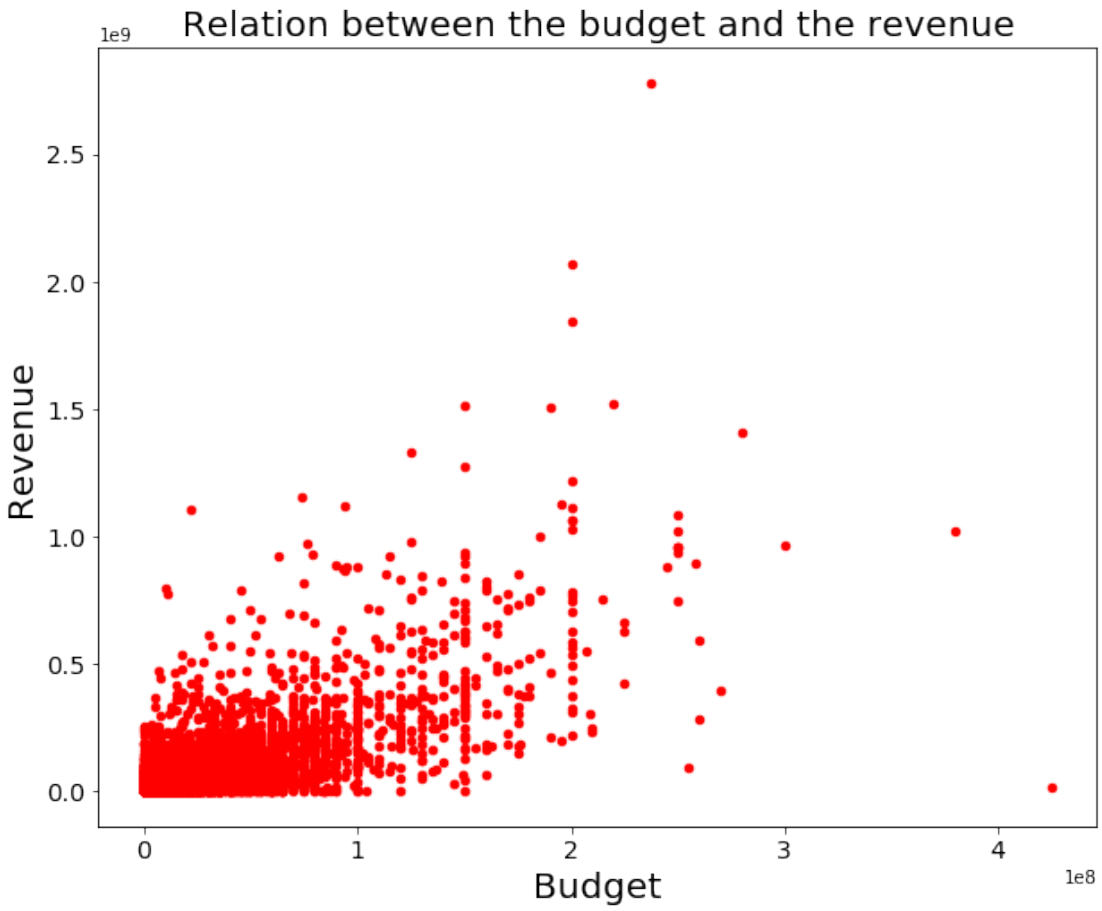
```
In [46]: # plot histogram to get the distribution

         df.hist(["revenue"],bins=10 , figsize=(10,8) , alpha =0.05 ,color ='red');
         plt.title("Revenue Distribution",fontsize=20);
         plt.xlabel('Revenue ( billion $) ',fontsize=16);
         plt.ylabel('Number of movies',fontsize=16);
```



**from the previous distribution ,we find that there is huge values of revenue recorded as zero in the dataset. Maybe it's a mistake or those films failed to get any revenue

1.1.6 What is the relation between the budget and the revenue?

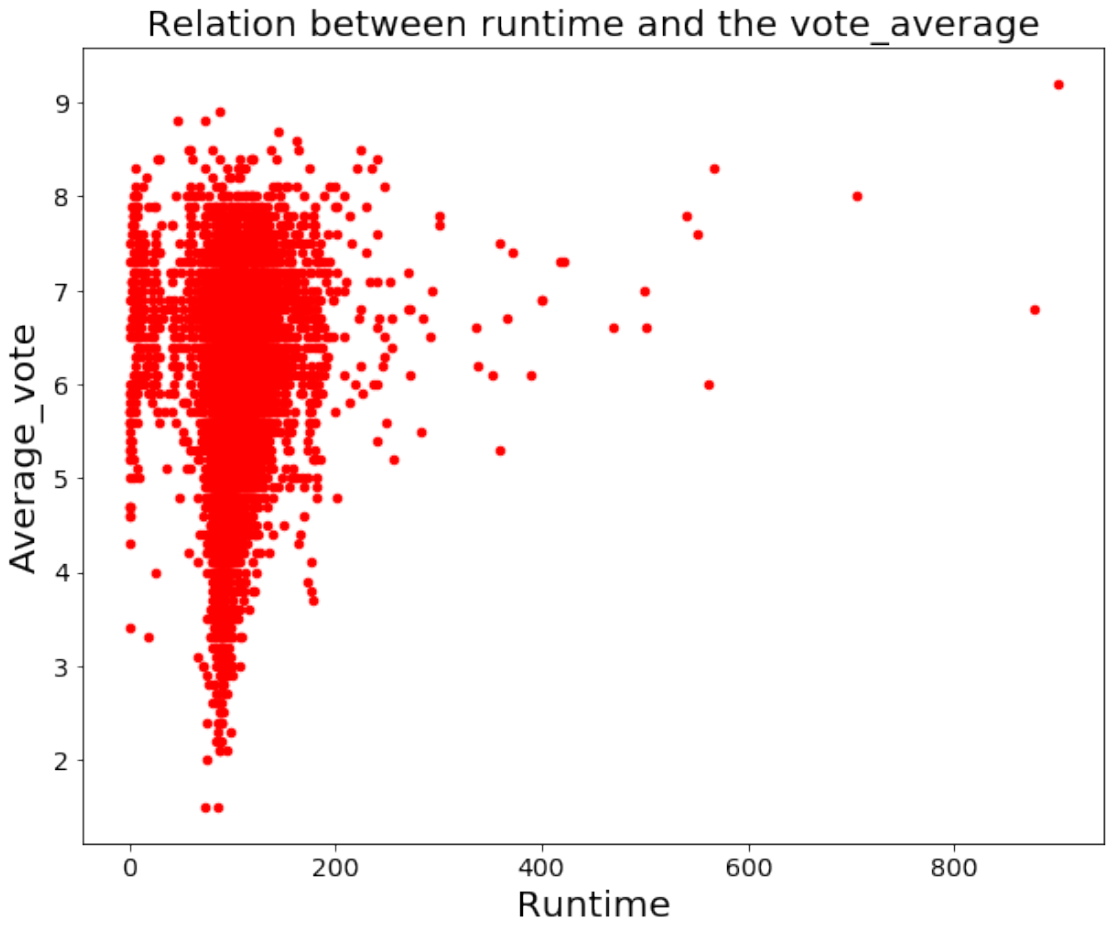
```
In [45]: #plot scatter plot to get the relation
         x=['budget']
         y=['revenue']
         df.plot.scatter(x, y, figsize=(10,8), color='red',fontsize=14)
         plt.title("Relation between the budget and the revenue",fontsize=20)
         plt.xlabel('Budget',fontsize=20)
         plt.ylabel('Revenue',fontsize=20)
         plt.show()
```



**From the previous plot, we can find that the higher budget leads to higher revenue .This can be an indicator for the company production to invest more in producing process

1.1.7 What is the relation between the run time and average vote?

```
In [71]: x=['runtime']
         y=['vote_average']
         df.plot.scatter(x, y ,figsize=(10,8), color='red',fontsize=14)
         plt.title("Relation between runtime and the vote_average",fontsize=20)
         plt.xlabel('Runtime',fontsize=20)
         plt.ylabel('Average_vote',fontsize=20)
         plt.show()
```



**There is no relation between the time duration of the movie and the vote it will take

1.1.8 What is the higher revenue in year 2010?

```
In [57]: df.query('release_year =="2010"')['revenue'].max()
```

```
Out[57]: 1063171911
```

**so the higher revenue in year 2010 was almost one billion and 63 millions dollar.

1.1.9 How many movies that its average vote is 7 or more?

```
In [65]: df.query('vote_average >= 7').count()
```

```
Out[65]: budget          1550
         revenue         1550
         runtime         1550
         vote_average    1550
         release_year    1550
         dtype: int64
```

**1550 movies have got vote average 7 or more.

1.1.10 How many movies that produced starting from year 2000?

```
In [73]: df.query('release_year >= "2000"').count()
```

```
Out[73]: budget          6926
         revenue         6926
         runtime         6926
         vote_average    6926
         release_year    6926
         dtype: int64
```

**About 6920 movies produced strating from year 2000 , and based on this dataset that collected data about movies starting from year 1960, it seemed that the movies production grows.

Conclusions

From the anaysis done previously,we can conclude the following The higher budget leads to higher revenue.

There is no relation between runtime and the vote that the movie gets

From more than 10 thousands movies only 1550 movies got vote higher than 7, if I'm on your foot,I won't waste my free time to watch a movie got vote less than 7

Number of movies produced in the last 20 years is more than those produced from 1960 till 2000, so if you are an investor,think about investing in movies' production.

Limitations

1)Large number of budget and revenue of the movies was zero and that appeared in the distribution and of cousre affects the mean of both of them

2)In cast, production companies and generes columns were more than one value ,this doesn't affect my research questions but it would be better to be separated

1.2 Submitting your Project

```
In [1]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[1]: 0
```

```
In [ ]:
```