Project: Investigate a Dataset (TMDB Movie Data)

Table of Contents

Introduction

In this project, we will analyze TMDB movie in particular, whether runtime affects average vote, what is the biggest of budget and the revenue between 2010 and 2015 , and how many films have been released over the years

In [84]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [85]:

```
ls
```

```
 Volume in drive C is TI31420300A
 Volume Serial Number is 0456-6234

 Directory of C:\Users\badr alshaibani\Desktop\AUdacity\DAND

05/19/2020  06:06 AM    <DIR>          .
05/19/2020  06:06 AM    <DIR>          ..
05/17/2020  01:48 AM    <DIR>          .ipynb_checkpoints
04/06/2018  09:25 AM    <DIR>          __MACOSX
05/19/2020  06:06 AM           176,343 investigate-a-dataset-template.ipyn
b
05/17/2020  12:27 AM             3,420 investigate-a-dataset-template.ipyn
b (1).zip
05/17/2020  01:48 AM        10,739,535 noshowappointments-kagglev2-may-201
6.csv
05/17/2020  08:02 AM            61,194 titanic_data.csv
05/17/2020  08:04 AM         6,883,750 tmdb-movies (1).csv
               5 File(s)     17,864,242 bytes
               4 Dir(s)  428,542,631,936 bytes free
```

Data Wrangling

General Properties :

In [86]:

```
df = pd.read_csv('tmdb-movies (1).csv')
df.head()
```

Out[86]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http:/
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 ...	

5 rows × 21 columns

In [87]:

```
## Here we showed the number of rows and columns
```

In [88]:

```
df.shape
```

Out[88]:

```
(10866, 21)
```

In [89]:

```
### Here we take a general description of all the data
```

In [90]:

```
df.describe()
```

Out[90]:

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

Data Cleaning

In [91]:

```
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 [92]:

```
### Here we take information for the data and all types of storage
```

In [93]:

```
df.drop(['cast','homepage' ,'director','overview','tagline','keywords','original_title'
,'production_companies','budget_adj','revenue_adj'],axis=1 , inplace=True)
```

In [94]:

```
### Here we drop the columns we don't need in our questions
```

In [95]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 11 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   runtime       10866 non-null  int64  
 6   genres        10843 non-null  object 
 7   release_date  10866 non-null  object 
 8   vote_count    10866 non-null  int64  
 9   vote_average  10866 non-null  float64
 10  release_year  10866 non-null  int64  
dtypes: float64(2), int64(6), object(3)
memory usage: 933.9+ KB
```

In [96]:

```
### Here we take information for data and all kinds of storage after the projection pro
cess
```

In [97]:

```
df.shape
```

Out[97]:

```
(10866, 11)
```

In [98]:

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

Out[98]:

```
id               0
imdb_id         10
popularity       0
budget           0
revenue          0
runtime          0
genres          23
release_date     0
vote_count       0
vote_average     0
release_year     0
dtype: int64
```

In [99]:

```
df.dropna(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10835 entries, 0 to 10865
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            10835 non-null  int64  
 1   imdb_id       10835 non-null  object 
 2   popularity    10835 non-null  float64
 3   budget        10835 non-null  int64  
 4   revenue       10835 non-null  int64  
 5   runtime       10835 non-null  int64  
 6   genres        10835 non-null  object 
 7   release_date  10835 non-null  object 
 8   vote_count    10835 non-null  int64  
 9   vote_average  10835 non-null  float64
 10  release_year  10835 non-null  int64  
dtypes: float64(2), int64(6), object(3)
memory usage: 1015.8+ KB
```

In [100]:

```
df.shape
```

Out[100]:

```
(10835, 11)
```

In [101]:

```
df.duplicated().sum()
```

Out[101]:

```
1
```

In [102]:

```
### Here we see the total number of duplicate rows
```

In [103]:

```
df.drop_duplicates(inplace=True)
```

In [104]:

```
df.duplicated().sum()
```

Out[104]:

```
0
```

In [105]:

```
### Here we dropped the duplicate rows¶
```

In [106]:

```
df.isin([0]).any()
```

Out[106]:

```
id              False
imdb_id         False
popularity      False
budget           True
revenue          True
runtime          True
genres          False
release_date    False
vote_count      False
vote_average    False
release_year    False
dtype: bool
```

In [107]:

```
## looking for columns have zero value
```

In [108]:

```
df.replace(0, np.nan,inplace=True)
df.dropna(how='any', axis=0,inplace=True)
```

In [109]:

```
df.isin([0]).any()
```

Out[109]:

```
id              False
imdb_id         False
popularity      False
budget          False
revenue         False
runtime         False
genres          False
release_date    False
vote_count      False
vote_average    False
release_year    False
dtype: bool
```

In [110]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3854 entries, 0 to 10848
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            3854 non-null   int64  
 1   imdb_id       3854 non-null   object 
 2   popularity    3854 non-null   float64
 3   budget        3854 non-null   float64
 4   revenue       3854 non-null   float64
 5   runtime       3854 non-null   float64
 6   genres        3854 non-null   object 
 7   release_date  3854 non-null   object 
 8   vote_count    3854 non-null   int64  
 9   vote_average  3854 non-null   float64
 10  release_year  3854 non-null   int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 361.3+ KB
```

Exploratory Data Analysis

(What are the most budget and revenues between 2010 to 2015 ?)

In [111]:

```
year2010 = df[df.release_year == 2010]
year2011 = df[df.release_year == 2011]
year2012 = df[df.release_year == 2012]
year2013 = df[df.release_year == 2013]
year2014 = df[df.release_year == 2014]
year2015 = df[df.release_year == 2015]
```

In [112]:

```
y10 = year2010.revenue.mean()
y10
```

Out[112]:

```
122496407.16853933
```

In [113]:

```
y11 = year2011.revenue.mean()
y11
```

Out[113]:

```
117629373.1005025
```

In [114]:

```
y12 = year2012.revenue.mean()
y12
```

Out[114]:

153066177.4177215

In [115]:

```
y13 = year2013.revenue.mean()
y13
```

Out[115]:

135281478.35

In [116]:

```
y14 = year2014.revenue.mean()
y14
```

Out[116]:

145878602.16363636

In [117]:

```
y15 = year2015.revenue.mean()
y15
```

Out[117]:

163768267.50625

In [118]:

```
### Budget
```

In [119]:

```
yy10 = year2010.budget.mean()
yy10
```

Out[119]:

47545721.56741573

In [120]:

```
yy11 = year2011.budget.mean()
yy11
```

Out[120]:

42419851.97487437

In [121]:

```
yy12 = year2012.budget.mean()
yy12
```

Out[121]:

```
48022851.42405064
```

In [122]:

```
yy13 = year2013.budget.mean()
yy13
```

Out[122]:

```
47599570.98888889
```

In [123]:

```
yy14 = year2014.budget.mean()
yy14
```

Out[123]:

```
44810715.15151515
```

In [124]:

```
yy15 = year2015.budget.mean()
yy15
```

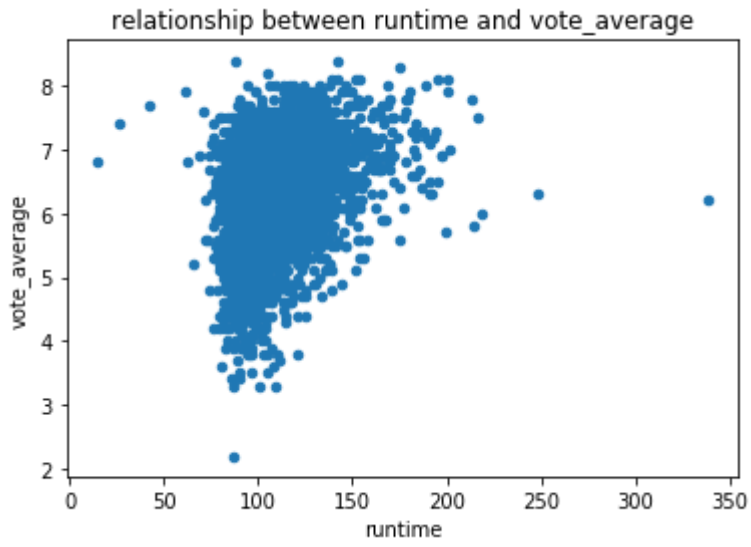
Out[124]:

```
44817359.55
```

Question 2 (Does the runtime affect the average vote?)

In [125]:

```
df.plot(x='runtime',y='vote_average',kind='scatter');
plt.title('relationship between runtime and vote_average');
```



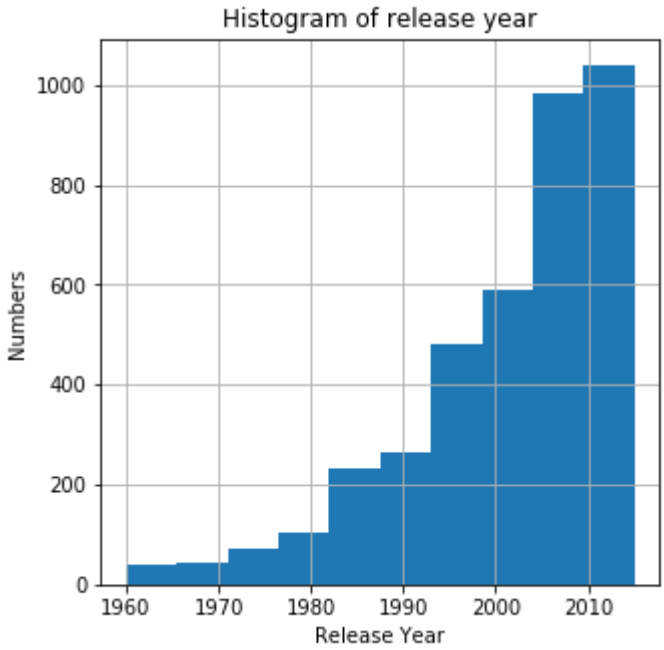
In [126]:

```
## Note here that the playback time affects the voting rate the majority of people do n
ot like to spend a long time watching a movie
```

How many films have been released over the years ?

In [127]:

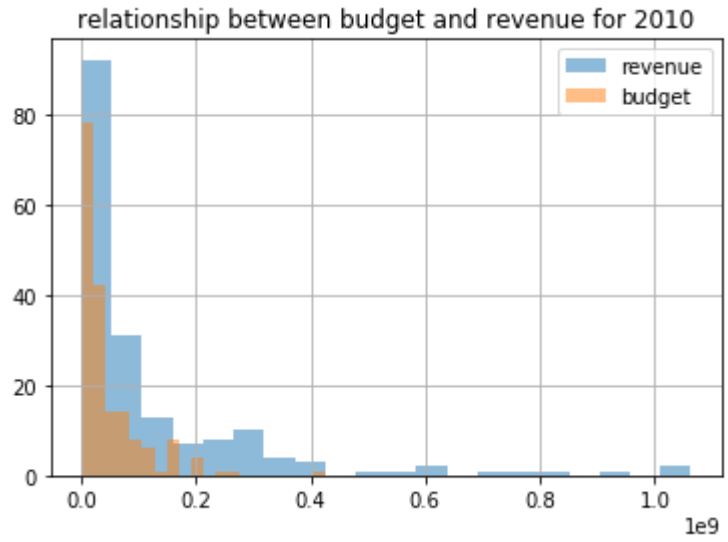
```
df['release_year'].hist(figsize=(5, 5));
# title and labels
plt.ylabel('Numbers');
plt.xlabel('Release Year');
plt.title('Histogram of release year');
```



What are the most budget and revenues between 2010 to 2015 ?

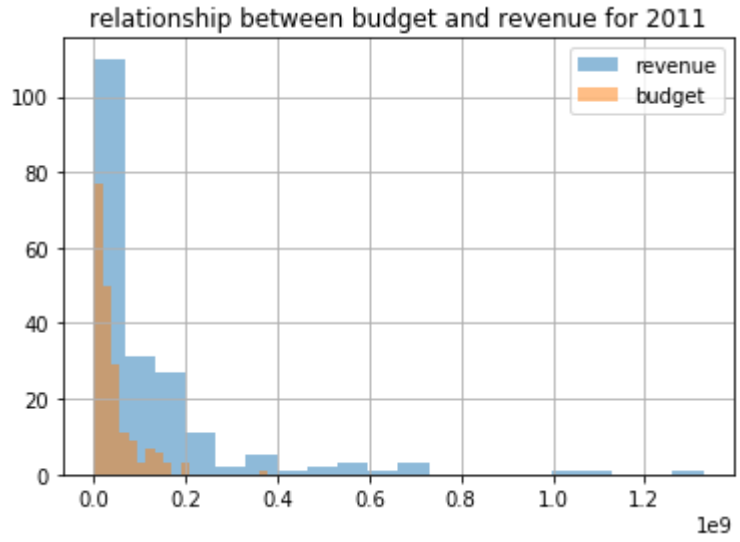
In [128]:

```
year2010.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2010.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2010')
plt.legend();
```



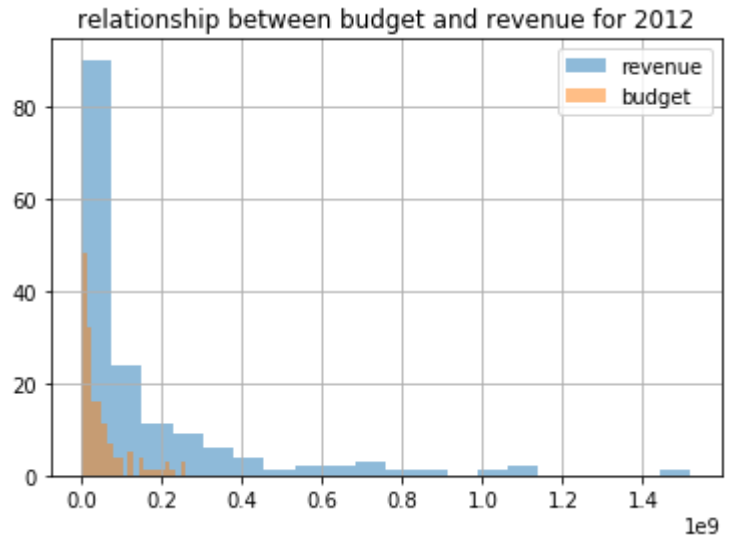
In [129]:

```
year2011.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2011.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2011')
plt.legend();
```



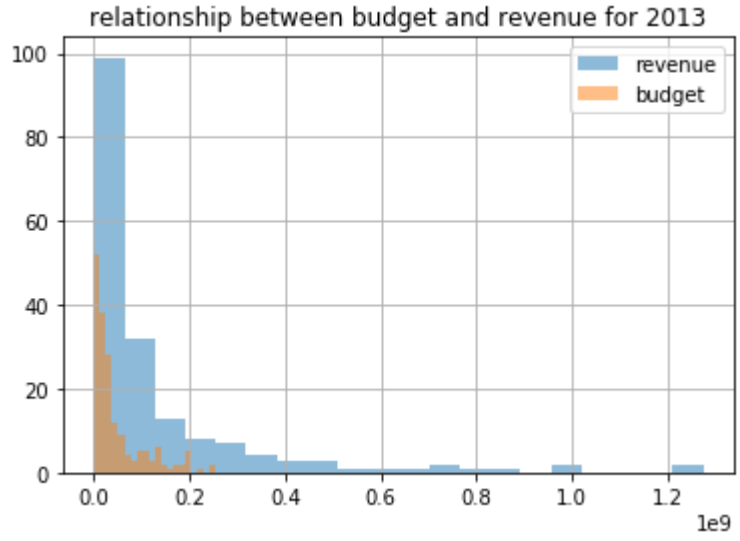
In [130]:

```
year2012.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2012.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2012')
plt.legend();
```



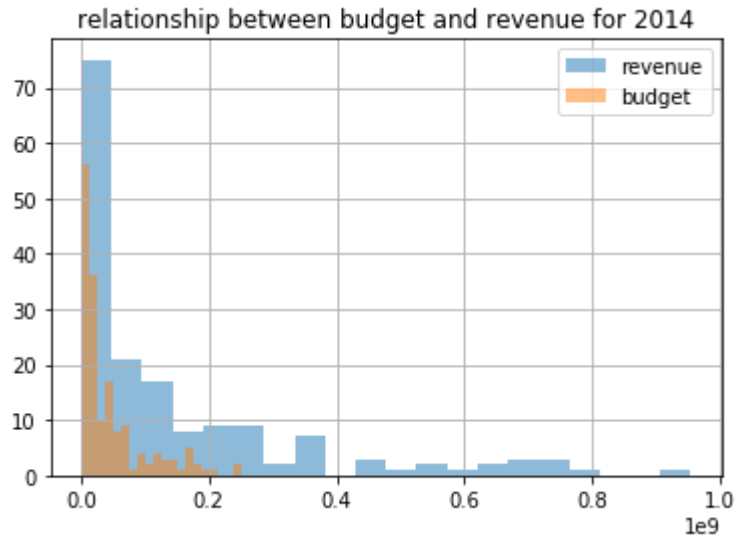
In [131]:

```
year2013.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2013.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2013')
plt.legend();
```



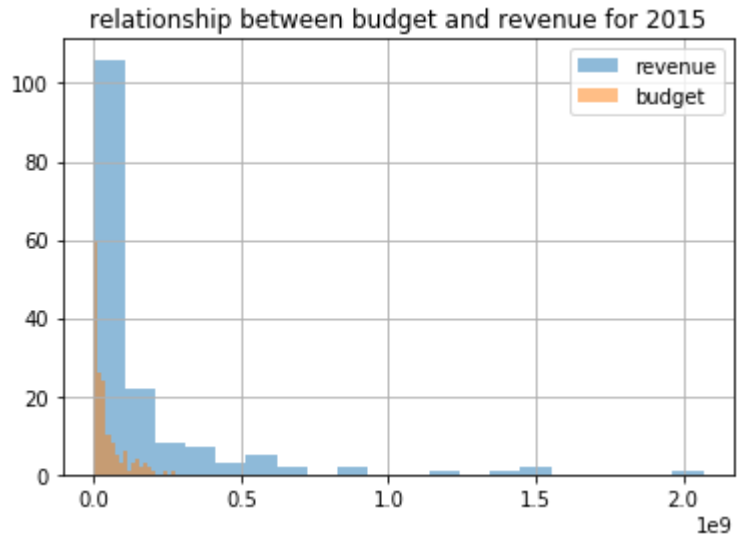
In [132]:

```
year2014.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2014.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2014')
plt.legend();
```



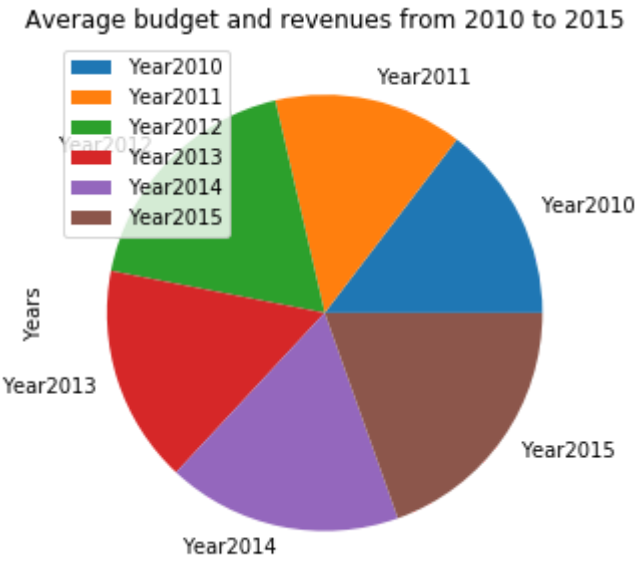
In [133]:

```
year2015.revenue.hist(alpha=0.5,bins=20,label='revenue')
year2015.budget.hist(alpha=0.5,bins=20,label='budget')
plt.title('relationship between budget and revenue for 2015')
plt.legend();
```



In [134]:

```
df_0= pd.DataFrame({'Years': [y10,y11,y12,y13,y14,y15]},index=['Year2010', 'Year2011',
'Year2012','Year2013','Year2014','Year2015'])
df_0.plot.pie(y='Years', figsize=(7,5));
plt.title('Average budget and revenues from 2010 to 2015');
```



Conclusions

-The number of movies released in general increases with the increase in years.

-Movies with less than 200 minutes are more popular.

-2010 is the most budgeted year for the film

-The year 2015 was the most revenue year for the film

-I showed the problem of zero values in the revenue and budget columns And I solved it, but I got another problem that the number of rows is very small

This project for udacity is and the results may be correct or wrong depending on the methods used in the analysis