Analysis of Movie dataset.

I chose the TMDB movie dataset.

In order investigate TMDB movie dataset, I made several questions:

1) What factors determine higher revenue?
2) I thought there could be a positive relationship between higher revenue and popularity?
   2.1 What is the relationship between higher revenue and popularity?
   2.2 I assumed that there is a positive relationship between revenue and popularity. Because the more popular the movie, the more people want to watch it.
3) There could also be a positive relationship between average vote and revenue. So, I was thinking, what is the exact relationship between average vote of movies and revenue?

4) I also thought that there could be a positive relationship between budget and revenue.
   4.1 Because more directors invest in films, they can hire better actors, and use better cameras, better places which in return make the movie very popular and with quality, so more revenue the movie will generate.
   So, what is the relationship between budget and revenue?
5) I have also thought that there could be some relationship between revenue and release year. Because if the economy was very good and it was the release_year, movies can generate more revenue. In addition, new movies due to inflation also generated more revenue than movies 50 years ago. In addition, due to programs created by Netflix and Amazon prime, movies that was released later and distribute through these platforms can generate more revenue, where movies before did not have enough streaming platforms, so they generate less revenue.
   So, what is the relationship between revenue and release year?

6) Last questions, which genres are the most popular?

First of all, I imported all the packages. Then, read the tmdb movies data.
Then checked the first 10 rows.

```
x=pd.read_csv("../input/datadomashka/tmdb-movies.csv")
x.head(10)
```

[30…

	id	imdb_id	popularity	budget	revenue	original_title	cast	homep
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.c
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.c
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurg
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-w epis
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.c
5	281957	tt1663202	9.110700	135000000	532950503	The Revenant	Leonardo DiCaprio|Tom Hardy|Will Poulter|Domhn...	http://www.foxmovies.com/movies/the-rever
6	87101	tt1340138	8.654359	155000000	440603537	Terminator Genisys	Arnold Schwarzenegger|Jason Clarke|Emilia Clar...	http://www.terminatormovie.c

1)As we can from the above example, certain columns, like ‘cast’ and ‘genres’,'release_date', 'product_companies','director' contain multiple values separated by pipe (|) characters
2) The final two columns ending with “_adj” show the budget and revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

[304]:
```
print(x.shape)
print(x.dtypes)
print(type(x['release_date'][0]))
```

```
(10866, 21)
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
<class 'str'>
```

1) As we can see, column "release_date" is string, but it should be datetime. So we need to fix it.

2) Also, there are 10866 rows and 21 columns.

[303]:
```
print(x.nunique())
print(x.describe())
```

```
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
                 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
```

We can see from the above graph that id, imdb_id is very unique.
Also, the average budget was 1.462570e+07 and average release year is 2001.

[306]:
```
print(x.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
None
```

From the above table, we can see that some columns such as imdb_id, homepage, production_companies, keywords,tagline has missing values

307]:
```
sum(x.duplicated())
x.drop_duplicates(inplace=True)
```

There is one duplicate in this data. We will use drop_duplicates to drop it.

```
sum(x.duplicated())
```

[30…] 0

As we see, no missing data now.

```
x['release_year']= pd.to_datetime(x['release_year'])
x.head()
```

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/

5 rows × 21 columns

So, we changed the type of release_year variable from string to datetime type.

So it will help us later to make observations with visuals.

```
print(x.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   id                    10865 non-null  int64         
 1   imdb_id               10855 non-null  object        
 2   popularity            10865 non-null  float64       
 3   budget                10865 non-null  int64         
 4   revenue               10865 non-null  int64         
 5   original_title        10865 non-null  object        
 6   cast                  10789 non-null  object        
 7   homepage              2936 non-null   object        
 8   director              10821 non-null  object        
 9   tagline               8041 non-null   object        
 10  keywords              9372 non-null   object        
 11  overview              10861 non-null  object        
 12  runtime               10865 non-null  int64         
 13  genres                10842 non-null  object        
 14  production_companies  9835 non-null   object        
 15  release_date          10865 non-null  object        
 16  vote_count            10865 non-null  int64         
 17  vote_average          10865 non-null  float64       
 18  release_year          10865 non-null  datetime64[ns]
 19  budget_adj            10865 non-null  float64       
 20  revenue_adj           10865 non-null  float64       
dtypes: datetime64[ns](1), float64(4), int64(5), object(11)
memory usage: 1.8+ MB
None
```

We changed release_year from string to datetime64 format.

Right now, let's try to fix certain columns, like ‘cast’ and ‘genres’, contain multiple values separated by pipe (|) characters. First we need to find number of columsn in cast and genres'

```
x.isnull().sum()
```

```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7929
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
```

+ Code + Markdown

Still we have so many missing values in columns cast, homepage director, tagline, keywords, overview, genres, production companies. Let's get rid of some of this data.

Still we have so many missing values in columns cast, homepage director, tagline, keywords, overview, genres, production companies. Let's get rid of some of this data.

```
x.drop(['imdb_id', 'homepage','tagline', 'keywords', 'overview', 'budget_adj','revenue_adj','productic
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              10865 non-null  int64
 1   popularity      10865 non-null  float64
 2   budget          10865 non-null  int64
 3   revenue         10865 non-null  int64
 4   original_title  10865 non-null  object
 5   cast            10789 non-null  object
 6   director        10821 non-null  object
 7   runtime         10865 non-null  int64
 8   genres          10842 non-null  object
 9   release_date    10865 non-null  object
 10  vote_count      10865 non-null  int64
 11  vote_average    10865 non-null  float64
 12  release_year    10865 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(5), object(5)
memory usage: 1.2+ MB
```

As we can all rows in both columns separated by pipe (|) characters

```
x.isnull().sum()
```

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

We still have some missing values, let's just delete all missing values

```
x.dropna(inplace=True)
print(x.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10731 entries, 0 to 10865
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   id              10731 non-null  int64         
 1   popularity      10731 non-null  float64       
 2   budget          10731 non-null  int64         
 3   revenue         10731 non-null  int64         
 4   original_title  10731 non-null  object        
 5   cast            10731 non-null  object        
 6   director        10731 non-null  object        
 7   runtime         10731 non-null  int64         
 8   genres          10731 non-null  object        
 9   release_date    10731 non-null  object        
 10  vote_count      10731 non-null  int64         
 11  vote_average    10731 non-null  float64       
 12  release_year    10731 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(5), object(5)
memory usage: 1.1+ MB
None
```

It looks now we don't have any missing values. Let's start our analysis:

```
revenue_zero = x['revenue'] == 0
print(x[revenue_zero].info())
print(x[revenue_zero])
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5888 entries, 48 to 10865
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              5888 non-null   int64
 1   popularity      5888 non-null   float64
 2   budget          5888 non-null   int64
 3   revenue         5888 non-null   int64
 4   original_title  5888 non-null   object
 5   cast            5888 non-null   object
 6   director        5888 non-null   object
 7   runtime         5888 non-null   int64
 8   genres          5888 non-null   object
 9   release_date    5888 non-null   object
 10  vote_count      5888 non-null   int64
 11  vote_average    5888 non-null   float64
 12  release_year    5888 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(5), object(5)
memory usage: 644.0+ KB
None
           id  popularity    budget  revenue                  original_title  \
48     265208    2.932340  30000000        0                       Wild Card
67     334074    2.331636  20000000        0                        Survivor
74     347096    2.165433         0        0          Mythica: The Darkspore
75     308369    2.141506         0        0  Me and Earl and the Dying Girl
92     370687    1.876037         0        0        Mythica: The Necromancer
...       ...         ...       ...      ...                             ...
10861      21    0.080598         0        0              The Endless Summer
10862   20379    0.065543         0        0                      Grand Prix
10863   39768    0.065141         0        0                Beregis Avtomobilya
10864   21449    0.064317         0        0           What's Up, Tiger Lily?
```

From here, we can see that there are 5888 rows have revenue zero, some of them have also budget zero 0, it is super strange, so let's just delete them

```
print(x[revenue_zero].index.values)
```

```
[   48    67    74 ... 10863 10864 10865]
```

+ Code + Markdown

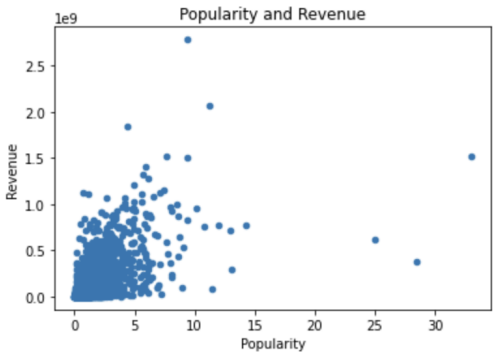
```
z=x[revenue_zero].index.values
x.drop(index=z,inplace=True)
print(x.info())
print(x[revenue_zero].index.values)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4843 entries, 0 to 10848
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              4843 non-null   int64
 1   popularity      4843 non-null   float64
 2   budget          4843 non-null   int64
 3   revenue         4843 non-null   int64
 4   original_title  4843 non-null   object
 5   cast            4843 non-null   object
 6   director        4843 non-null   object
 7   runtime         4843 non-null   int64
 8   genres          4843 non-null   object
 9   release_date    4843 non-null   object
 10  vote_count      4843 non-null   int64
 11  vote_average    4843 non-null   float64
 12  release_year    4843 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(5), object(5)
memory usage: 529.7+ KB
None
[]
```

As we can see right now there are no rows that have revenue=0. Let's start our analysis

```
import matplotlib.pyplot as plt

#Let's create scatter plot
x.plot(x='popularity',y='revenue',kind='scatter')
plt.title('Popularity and Revenue')
plt.xlabel('Popularity')
plt.ylabel('Revenue');
```



+ Code + Markdown

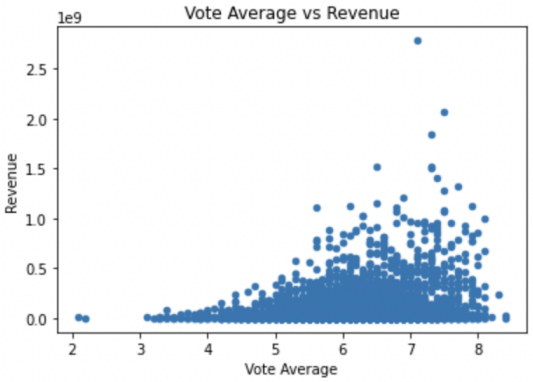
As we can't see it is not really clear with the above graph, relationship between popularity and revenue. Alternative way, could be finding correlation between popularity and revenue.

```
print(np.corrcoef(x["popularity"],x["revenue"] ))
```

```
[[1.         0.62916673]
 [0.62916673 1.        ]]
```

As we can see from here, there is a positive correlation between revenue and popularity.

```
x.plot(x='vote_average',y='revenue',kind='scatter')
plt.title('Vote Average vs Revenue')
plt.xlabel('Vote Average')
plt.ylabel('Revenue');
```



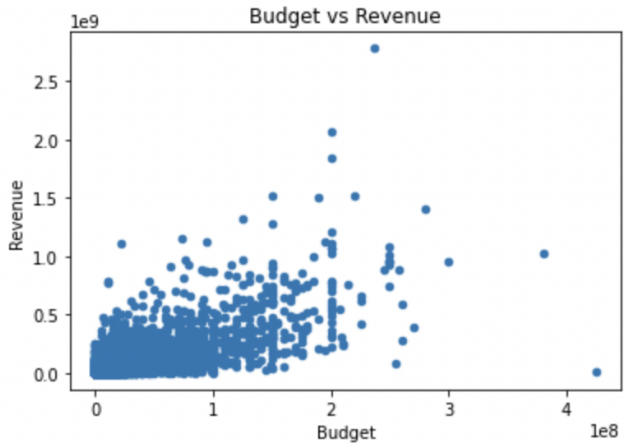
As we can from this graph, more vote average, more it generates revenue. But to get more deep understanding, let's find correlation between vote average and revenue.

```
print(np.corrcoef(x["vote_average"],x["revenue"] ))
```

```
[[1.         0.20715738]
 [0.20715738 1.        ]]
```

As we can see that there is positive correlation between vote_average and revenue. However, we can see correlation of revenue is more strong with popularity variable.

```
x.plot(x='budget',y='revenue',kind='scatter')
plt.title('Budget vs Revenue')
plt.xlabel('Budget')
plt.ylabel('Revenue');
```



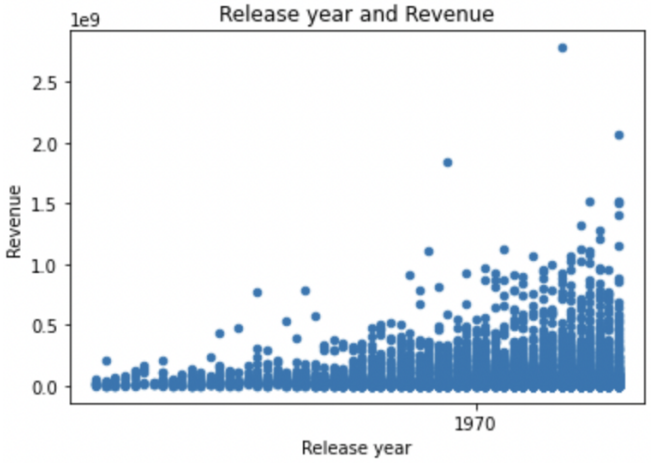
It is not clear from this graph correlation between budget and revenue. let's try to find correlation between these 2 variables.

```
print(np.corrcoef(x["budget"],x["revenue"] ))
```

```
[[1.         0.70389905]
 [0.70389905 1.        ]]
```

As we can there is higher correlation between budget and revenue. So, it means more you have budget more you can generate. It is kind of logical, because if you invest more, you can hire more popular actors and use better expensive cameras, which will increase customer's satisfaction with movie, in turn generate more revenue.

```
x.plot(x='release_year',y='revenue',kind='scatter')
plt.title('Release year and Revenue')
plt.xlabel('Release year')
plt.ylabel('Revenue');
```



As we can see, over the years movies generated more revenue, which can be also associate with growth of the economy as well as growth of inflation.

Overall, we can see that the budget had the highest correlation with revenue, which was around 0.7. It means if you increase the budget by 1000 dollars, your revenue will also increase to 700 dollars. Then the 2 highest correlation was with popularity, which was around 0.629. Which is kind of logical, a more popular movie, more customers will watch this movie, in turn it will generate more revenue. Also, there is a positive correlation between vote_average variable and revenue. Which also means, higher the average votes for movies, we expect that revenue will also increase.

Also, I made graph to check relationship between release_year variable and revenue. Overall, it seems that movies started to generated more revenue than before, which can be explained by growing economy and higher population, and different platforms to watch movies such as netflix etc.

```
b=x[x['genres'].str.contains('|')]
print(b.head())
```

```
      id  popularity     budget     revenue                original_title  \
0  135397   32.985763  150000000  1513528810                Jurassic World
1   76341   28.419936  150000000   378436354             Mad Max: Fury Road
2  262500   13.112507  110000000   295238201                      Insurgent
3  140607   11.173104  200000000  2068178225  Star Wars: The Force Awakens
4  168259    9.335014  190000000  1506249360                      Furious 7

                                                cast          director  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...    Colin Trevorrow
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...      George Miller
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...   Robert Schwentke
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...        J.J. Abrams
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...          James Wan

   runtime                                     genres release_date  \
0      124  Action|Adventure|Science Fiction|Thriller       6/9/15
1      120  Action|Adventure|Science Fiction|Thriller      5/13/15
2      119         Adventure|Science Fiction|Thriller      3/18/15
3      136   Action|Adventure|Science Fiction|Fantasy     12/15/15
4      137                     Action|Crime|Thriller       4/1/15

   vote_count  vote_average               release_year
0        5562           6.5 1970-01-01 00:00:00.000002015
1        6185           7.1 1970-01-01 00:00:00.000002015
2        2480           6.3 1970-01-01 00:00:00.000002015
3        5292           7.5 1970-01-01 00:00:00.000002015
4        2947           7.3 1970-01-01 00:00:00.000002015
```

As it can be seen from the above, genres column contains “|”.

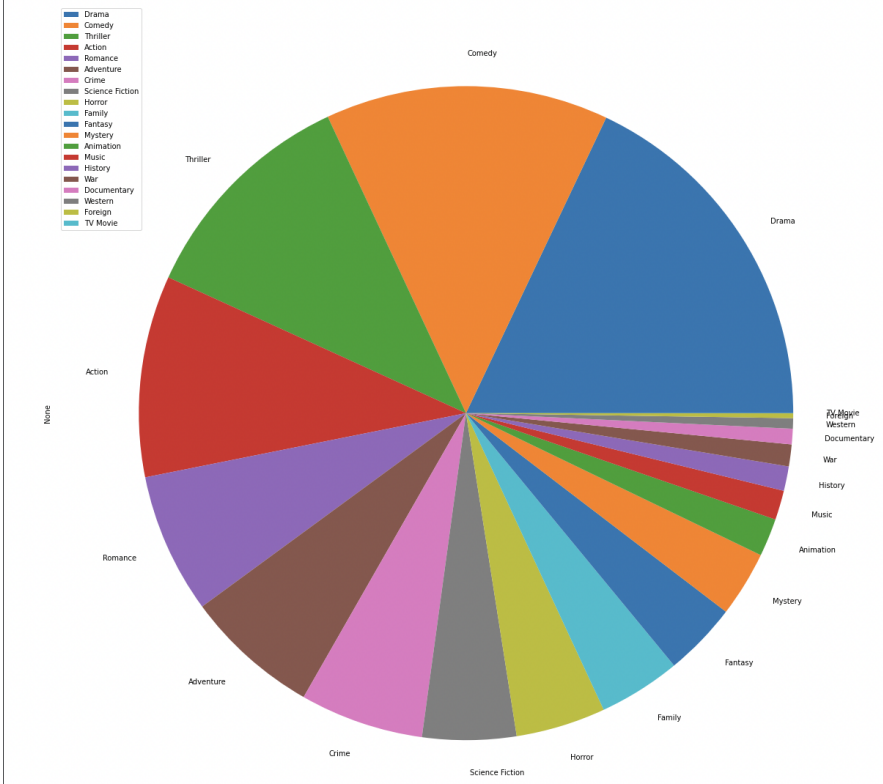
```
a = x["genres"].str.cat(sep = '|')
b = pd.Series(a.split('|'))
popular_genres=b.value_counts(ascending=False)
print(popular_genres)
```

```
Drama              2271
Comedy             1770
Thriller           1422
Action             1260
Romance             876
Adventure           843
Crime               773
Science Fiction     586
Horror              561
Family              513
Fantasy             461
Mystery             410
Animation           237
Music               184
History             153
War                 137
Documentary          97
Western              65
Foreign              30
TV Movie              1
dtype: int64
```

By doing above commands, we have found that drama and comedy are the most popular genres over the years.

```
plot = popular_genres.plot.pie(subplots=True, figsize=(20, 40))
plt.legend(loc="best")
```

```
[32…  <matplotlib.legend.Legend at 0x7fb95fa375d0>
```



As we can see from the pie chart, Drama and comedy, Thriller and action are the most popular movie genres.

References:

[1] https://stackoverflow.com/questions/19852215/how-to-add-a-legend-to-matplotlib-pie-chart
[2] https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/
[3] https://github.com/antra0497/Udacity--Project-Investigate-TMDB-Movies-Dataset/blob/master/investigate-a-dataset-template.ipynb