Introduction:

The data set I chose is Movie dataset

Here are some notes and comments about this datasets : This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

Certain columns, like 'cast' and 'genres', contain multiple values separated by pipe (|) characters.

There are some odd characters in the 'cast' column. Don't worry about cleaning them. You can leave them as is.

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.

The questions about this dataset:

1. Does higher budget mean higher popularity ? Is there a coefficent relationship ?
2. Will the runtime affect the vote count and popularity?
3. Higher popularity means higher profits ?
4. What Features are Associate with Top 10 Revenue Movies ?
5. Which genres are most popular from year to year?

Data Wrangling:

Get familiar with the data types, data structure. I did delete the duplicates and unuseful columns like imdb_id,homepage etc.

When handling the missing data. I use two ways: for all the missing data with data type object, i fill the null with string "missing". For budget, datatype integer,I fill 0 with np.NAN.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# Import all the libraries which will be used later
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
```

```
# load in data and print out the head
df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/tmdb-movies.csv')
df.head()
```

	id	imdb_id	popularity	budget	revenue	original_title	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pr Howa K
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Hardy Ther By
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Wood Jar Winsle
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	F Han Fisher|A
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Di Walk Statham

5 rows × 21 columns

```
# check the rows and columns of this dataset
df.shape
```

```
(10866, 21)
```

```
# check datatypes to see if there are some wrongly categorized types
df.dtypes
```

```
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
```

```
# check each columns number of unique values
df.nunique()
```

```
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
```

```
# statistic values for this data
df.describe()
```

	id	popularity	budget	revenue	runtime	vote_
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.

```
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
```

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

```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
```

```
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

```
# drop unuseful columns
df.drop(['id','imdb_id', 'homepage','overview'],axis=1,inplace=True)  # do not forg

# Ways to handle missing data
# For all missing data with object as datatype , I fill in with string "missing"
df['cast'].fillna('missing',inplace=True )
df['director'].fillna('missing',inplace=True)
df['tagline'].fillna('missing',inplace=True)
df['keywords'].fillna('missing',inplace=True)
df['genres'].fillna('missing',inplace=True)
df['production_companies'].fillna('missing',inplace=True)
df['budget'] = df['budget'].replace(0, np.NAN)
# although there is no null in budget, but we would find there is a problem when we
# Will deal with all the 0 value in budget later.
```

```
# confirm the data
df.isnull().sum()
```

```
popularity                 0
budget                  5696
revenue                    0
original_title             0
cast                       0
director                   0
tagline                    0
keywords                   0
runtime                    0
genres                     0
production_companies       0
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

```
# check if there are some duplicates
df.duplicated().sum()
```

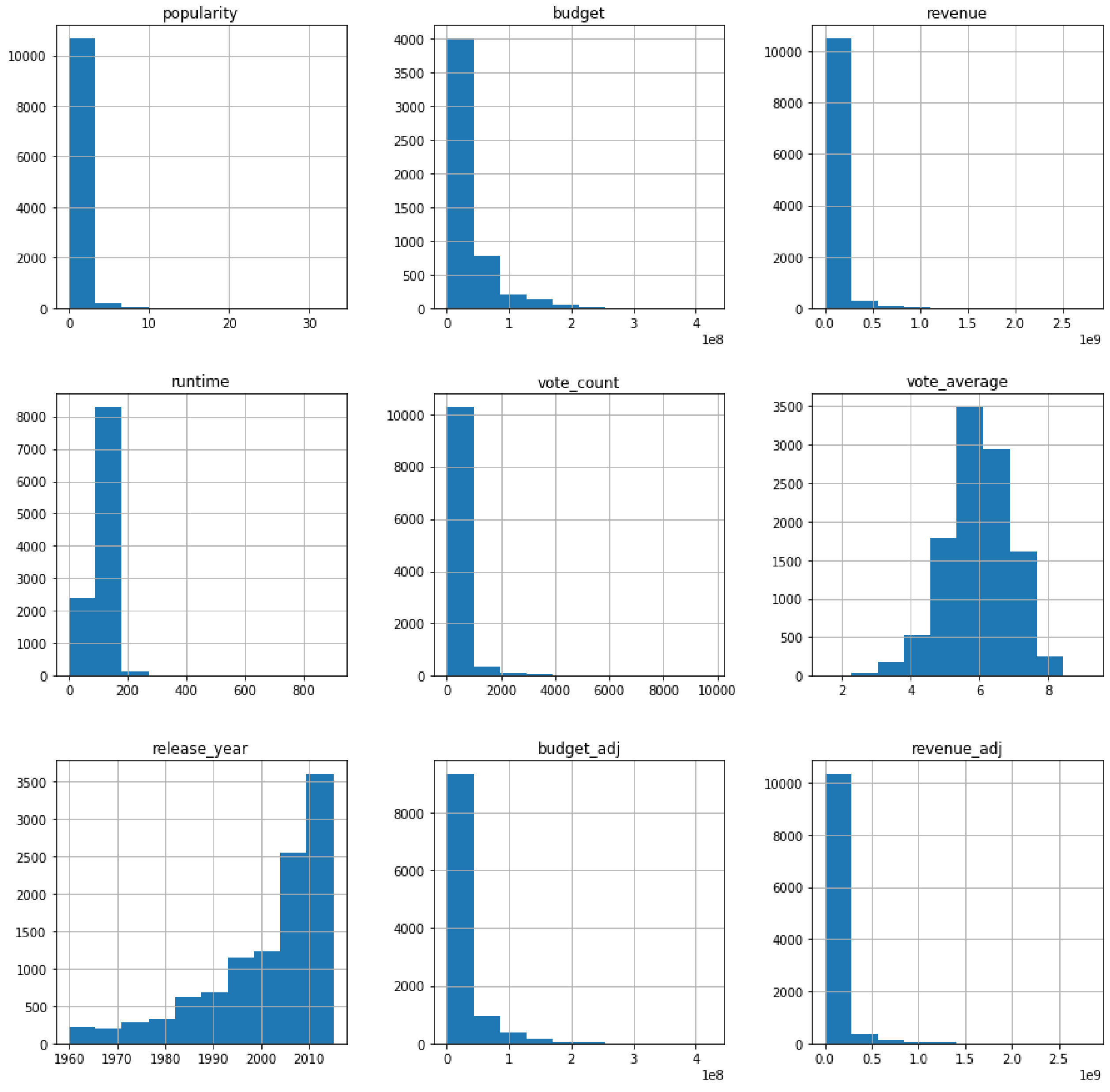
```
1
```

```
# drop the duplicates
df.drop_duplicates(inplace=True)     # do not forget inplace = True
```

```
# confirm again
df.duplicated().sum()
```

```
0
```

```
# visulize each variables
df.hist(figsize=(15,15));
```

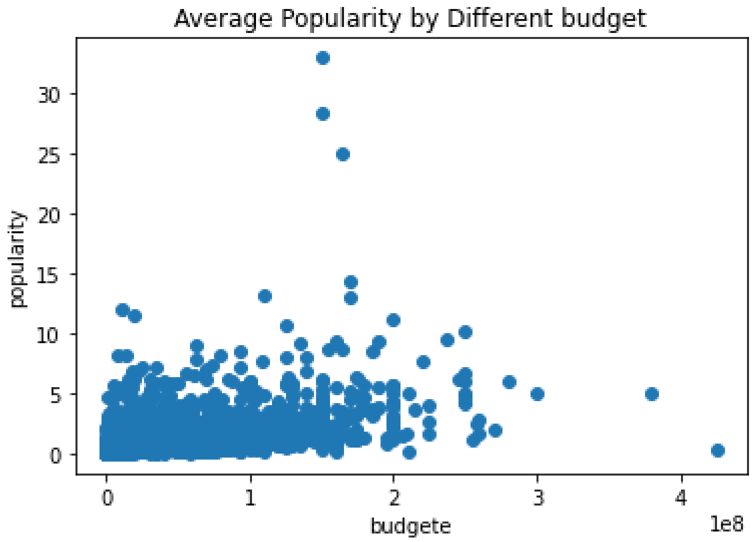


Exploration with Visuals and Conclusions

Question 1. Does higher budget mean higher popularity ? Is there a coefficent relationship ?

```
# plot the relation between budget and popularity
x = df['budget']
y = df['popularity']

plt.scatter(x,y)
plt.title('Average Popularity by Different budget',fontsize=12)
plt.xlabel('budgete',fontsize=10)
plt.ylabel('popularity',fontsize=10)
```

```
Text(0, 0.5, 'popularity')
```

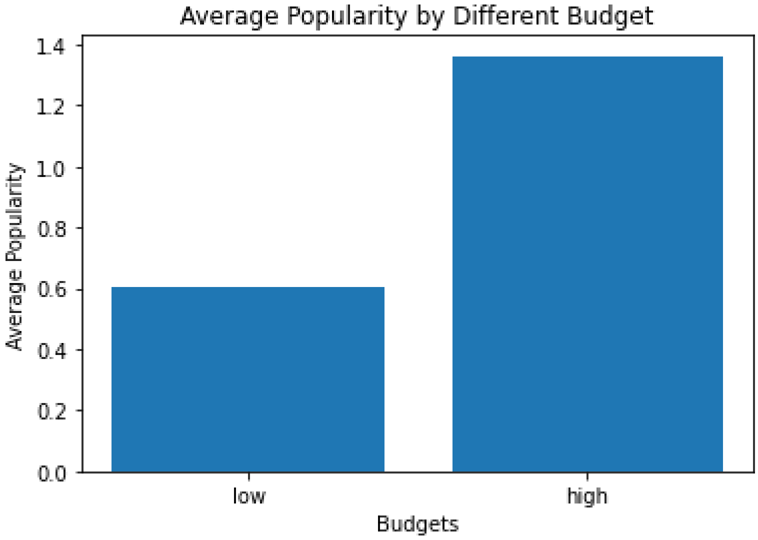


We can not see very strong relatioship between the budget and the popularity from above plot. Let's try to compare the data in another way: create two groups based on median value of budget

```
# based on median budget value to divide the budget into two groups : low and high
m = df['budget'].median()
low_budg =  df.query('budget < {}'.format(m))
high_budg =  df.query('budget >= {}'.format(m))


# check low budget and high budget mean values respecively
mean_popularity_of_low_budget = low_budg['popularity'].mean()
mean_popularity_of_high_budget = high_budg['popularity'].mean()


# create a bar chart with the values we get above
locations = [1,2]
heights = [mean_popularity_of_low_budget , mean_popularity_of_high_budget]
labels=['low','high']
plt.bar(locations, heights, tick_label = labels)
plt.title('Average Popularity by Different Budget')
```

```
plt.xlabel('Budgets')
plt.ylabel('Average Popularity')
```

```
Text(0, 0.5, 'Average Popularity')
```




```
increase_percentage = (mean_popularity_of_high_budget - mean_popularity_of_low_bud
increase_percentage
```

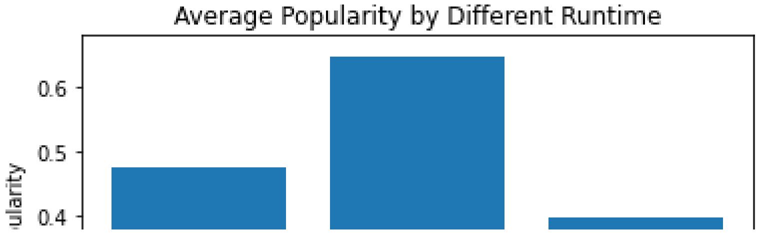
```
55.50933772947093
```

```
# here I will create 3 groups with query().  <60 min: short   , 60 min <=  <= - 12
short =  df.query('runtime < {}'.format(100))
medium =  df.query('runtime < {}'.format(200))
long = df.query('runtime > {}'.format(200))
```

```
# check mean popularity of different movie lengths
mean_popularity_of_short = short['popularity'].mean()
mean_popularity_of_medium = medium['popularity'].mean()
mean_popularity_of_long = long['popularity'].mean()
```

```
locations = [1,2,3]
heights = [mean_popularity_of_short, mean_popularity_of_medium, mean_popularity_of_
labels=['low','medium','high']
plt.bar(locations, heights, tick_label = labels)
plt.title('Average Popularity by Different Runtime')
plt.xlabel('Runtime')
plt.ylabel('Average Popularity')
```

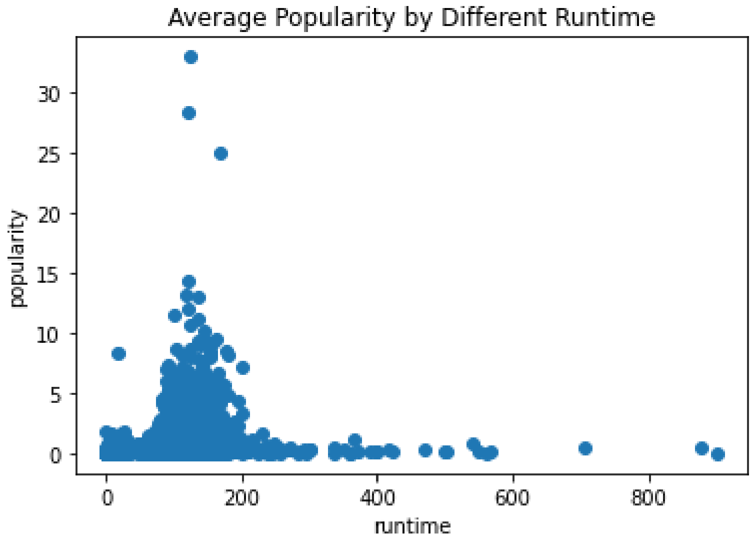
```
Text(0, 0.5, 'Average Popularity')
```



The movies should not be too long or too short. Medium length is better to gain higher popularity. But above bar chart is hard to tell the best length of runtime. Scatter plot may be a better choice.

```
# plot the relation between runtime and popularity
x = df['runtime']
y = df['popularity']

plt.scatter(x,y)

plt.title('Average Popularity by Different Runtime',fontsize=12)
plt.xlabel('runtime',fontsize=10)
plt.ylabel('popularity',fontsize=10)
```

```
Text(0, 0.5, 'popularity')
```



Conclusion Q2:

Combine two plots above, we can not simply say , the longer runtime, the more popular the movies are.

If the movies are within 200 minutes,it will be more popular. Once the movies run over 200 minutes, it's hard for them to gain high popularity

Q3 : Higher popularity means higher profits ?

```
# we need to get the mean of popularity
```

```
m_popularity = df['popularity'].median()
lower_popularity =  df.query('popularity < {}'.format(m_popularity))
higher_popularity =  df.query('popularity >= {}'.format(m_popularity))
```

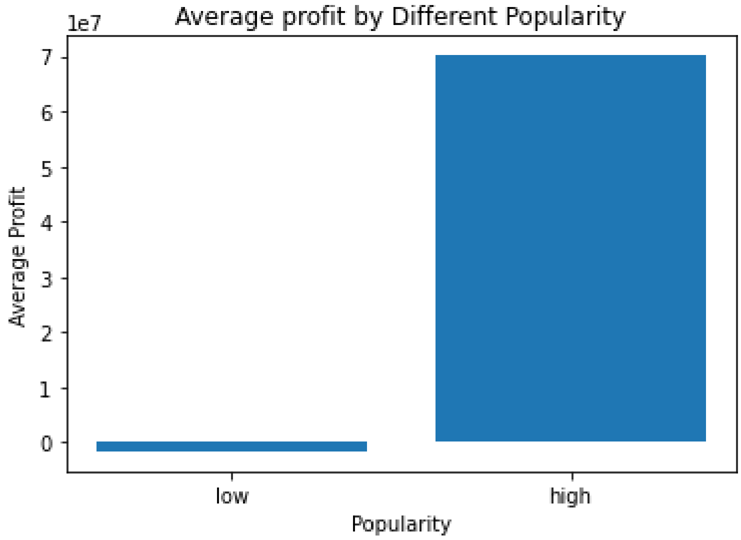
Double-click (or enter) to edit

```
# create a new column called profit. profit = Revenue - budget
df['profit'] = df['revenue'] - df['budget']
#df['profit'].head(20)
#df.head()
```

```
# average net profit for low_popularity and high_popularity
mean_profit_of_low_popularity = lower_popularity['profit'].mean()
mean_profit_of_high_popularity = higher_popularity['profit'].mean()
# df.head()
```

```
# create a bar chart with the values we get above
locations = [1,2]
heights = [mean_profit_of_low_popularity, mean_profit_of_high_popularity]
labels=['low','high']
plt.bar(locations, heights, tick_label = labels)
plt.title('Average profit by Different Popularity')
plt.xlabel('Popularity')
plt.ylabel('Average Profit')
```

```
Text(0, 0.5, 'Average Profit')
```



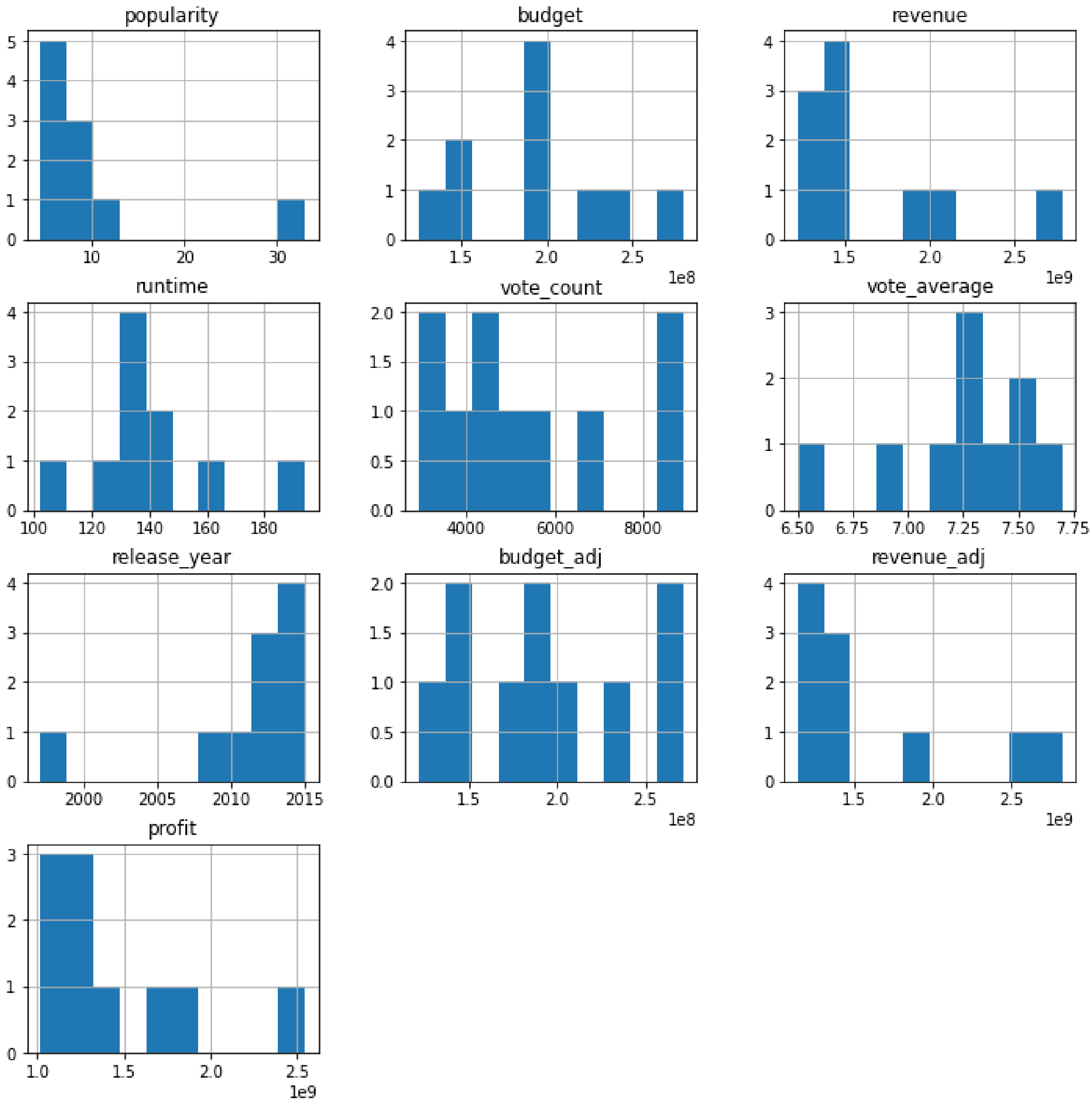
Conclusion for Question 3: as we can see above, higher popularity does make much higher average profits.

4. What Features are Associate with Top 10 Revenue Movies ?

```
top10_revenue = df.nlargest(10,'revenue')
```

```
top10_revenue.hist(figsize=(12,12))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f1aae1811d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad258cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad2253d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad1dc790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad191bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad156110>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad10c650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad0c3a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad0c3ad0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad088110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1aad075a10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1abd5bde90>]],
      dtype=object)
```



Double-click (or enter) to edit

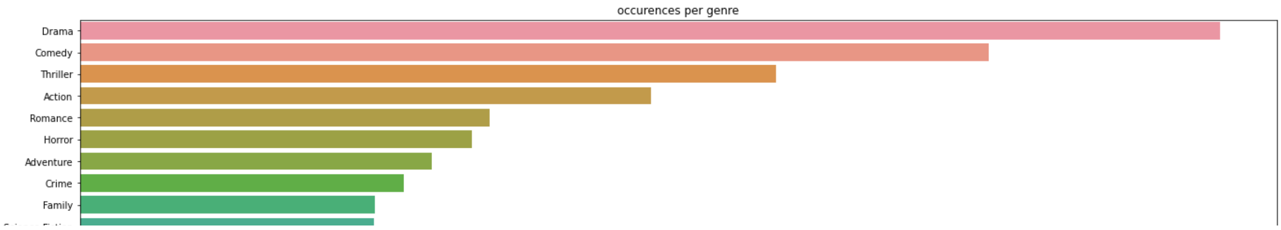
Conclusion for question 4:

There are some characteristics we can conclude from the top 10 movies. Runtime ranges from 100 mins to 200 mins. The released year are between 1995 to 2015.

5. Which genres are most popular from year to year?

```
#The following function can give all the counts for per category
def extract_data(column_name):
    data = df[column_name].str.cat(sep = '|') # put all the genres into a long str
    # Create pandas series and store the values separately
    data = pd.Series(data.split('|'))   # split the genres by |
    # Display value count in descending order
    count = data.value_counts(ascending = False) # count the occurrence of each ger
    return count
```

```
# use the function created above to split genres and count the occurrence of each
genre_count = extract_data('genres')

#create a separate dataframe to plot
df_genre_counts = pd.DataFrame({'genres': genre_count.index, 'count': genre_count.
#df_genre_counts

f, ax = plt.subplots(figsize=(23, 9))
# use the dataframe just created as the input data
sns.barplot(x = 'count', y = 'genres', data=df_genre_counts) # how to get the data
ax.set_title(' occurences per genre ')
ax.set_xlabel('occurrences')
ax.set_ylabel('genres')
plt.show()
```

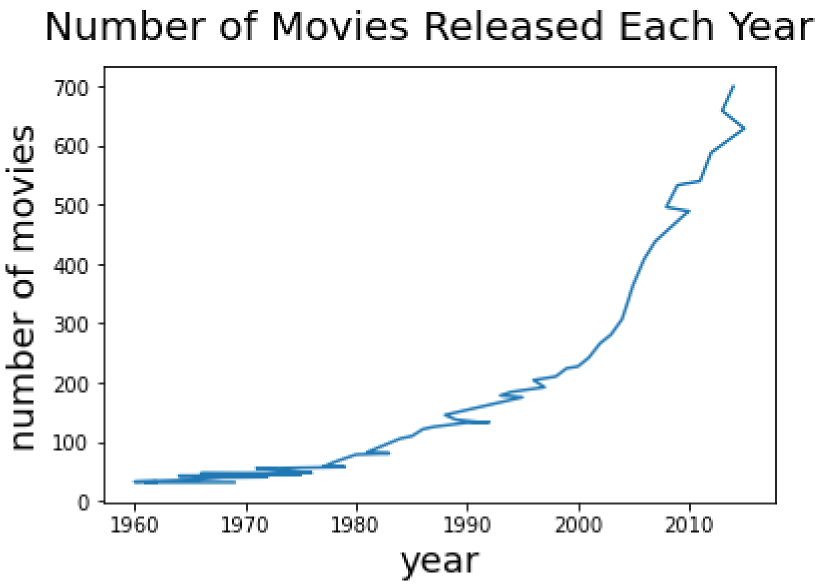



```
director_count = extract_data('director')
director_count
```

```
Woody Allen         46
missing             44
Clint Eastwood      34
Martin Scorsese     31
Steven Spielberg    30
                    ..
Mike Maguire         1
Tom Kuntz            1
John Simpson         1
Simon Hunter         1
Harold P. Warren     1
Length: 5363, dtype: int64
```

```
movie_count = df['release_year'].value_counts()
# movie_count.plot(xlabel='year',ylabel='number of movies',title='Number of Movies
fig = plt.figure()
plt.plot(movie_count)
fig.suptitle('Number of Movies Released Each Year',fontsize=20)
plt.xlabel('year',fontsize=18)
plt.ylabel('number of movies',fontsize=18)
```

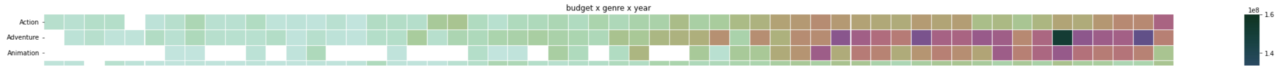
```
Text(0, 0.5, 'number of movies')
```



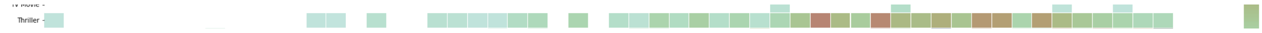
Throught above two plots, we can see The top 5 genres are Drama, Comedy, Action, Horror and Adventrue The number of movies increased along the time.

```
# The following is a really comprehensive plot. It shows the revenue and budget fo
# genres are so specific, I will just take the first genre for each movie instead
```

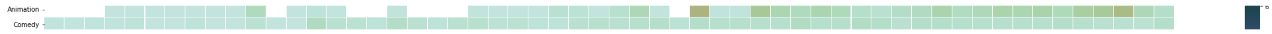
```
df['genre'] =  df['genres'].apply(lambda x: x.split('|')[0])

# plot all the genre types for each year with the budget and revenue
genre_year = df.groupby(['genre', 'release_year']).mean().sort_index()
df_gyBudget = genre_year.pivot_table(index=['genre'], columns=['release_year'], va
df_gyBudget = genre_year.pivot_table(index=['genre'], columns=['release_year'], va


df_gyGross = genre_year.pivot_table(index=['genre'], columns=['release_year'], valu
f, [axA, axB] = plt.subplots(figsize=(40, 20), nrows=2)
cmap = sns.cubehelix_palette(start=1.5, rot=1.5, as_cmap=True)
sns.heatmap(df_gyBudget, xticklabels=3, cmap=cmap, linewidths=0.05, ax=axA)
sns.heatmap(df_gyGross, xticklabels=3, cmap=cmap, linewidths=0.05, ax=axB)
axA.set_title('budget x genre x year')
axA.set_xlabel('release_years')
axA.set_ylabel('genres')


axB.set_title('revenue x genre x year')
axB.set_xlabel('release_years')
axB.set_ylabel('genres')
plt.show()
```



Conclusion for Question 5: As the time goes, we have a wider range of movies and genres to choose from. From 1984 to 2014, there are more and more high budget, high revenue movies. But compared to the budgets,



Conclusion:



Based on the analysis I did above, we can make the following summarizations:

1. The quantity and range of movie gets larger.We have more choices to choose from as an audience.
2. We can not say high budget guarantees high popularity. But for movies with higher budgets do produce higher average popularity.
3. To produce a more popular movie, the runtime should be best around 150 mins; Drama, Comedy, Action, these genres would be preferable.

Limitations:

1.These are factors that makes the movies become popular and successful. But we should also notice the limitations. There are some missing data and many erroreous zeros which may affect the analysis.

2. It's hard for us to know how the vote_counts and popularity are measured.

3. For foreign movies,currecy is not indicated. inflation over the years should also be taken into consideration.

Reference:

1. https://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.nlargest.html
2. https://www.kaggle.com/diegoinacio/imdb-genre-based-analysis
3. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.apply.html
4. https://pandas.pydata.org/pandas-docs/stable/visualization.html

0s completed at 10:22 AM