I. Learning the Data

Preface:

It's hard to come up with a meaningful question before you have a general picture of the whole dataset. So, the first thing we want to do instead of coming up a question is learning the dataset.

Step:

1. Import the relevant packages

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib as mpl
import numpy as np
import seaborn as sns

%matplotlib inline
```

The pandas, numpy are using to manipulate the data.
The matplotlib. mpl_toolkits, seaborn are using to visualize the data.

2. Load the data through .csv file

```
# load the dataset
# notice: I have changed the columns name replace 'space' by '_',
pd_movies = pd.read_csv('movies.csv')
```

3. Summary statistic

```
# Have a general picture of the movie dataset
# shape
pd_movies.shape
```

executed in 15ms, finished 14:58:17 2022-11-01

```
(3201, 16)
```

```
# using df.head() to see the first 5 rows
pd_movies.head()
```

```python
# info
pd_movies.info()

# As you can see, there are lost of columns have less than 3201 row is non-null
# And there are many non-numerical columns
# So we might need to find a way to deal with the missing data
```

executed in 31ms, finished 14:58:17 2022-11-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3201 entries, 0 to 3200
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Title                   3201 non-null   object 
 1   US_Gross                3201 non-null   object 
 2   Worldwide_Gross         3201 non-null   object 
 3   US_DVD_Sales            564 non-null    float64
 4   Production_Budget       3200 non-null   float64
 5   Release_Date            3201 non-null   object 
 6   G                       2596 non-null   object 
 7   Running_Time            1209 non-null   float64
 8   Distributor             2969 non-null   object 
 9   Source                  2836 non-null   object 
 10  Major_Genre             2926 non-null   object 
 11  Creative_Type           2755 non-null   object 
 12  Director                1870 non-null   object 
 13  Rotten_Tomatoes_Rating  2321 non-null   float64
 14  IMDB_Rating             2988 non-null   float64
 15  IMDB_Votes              2988 non-null   float64
dtypes: float64(6), object(10)
memory usage: 400.2+ KB
```

```python
pd_movies.describe()
```

executed in 70ms, finished 14:58:18 2022-11-01

	US_Gross	Worldwide_Gross	US_DVD_Sales	Production_Budget	G	Running_Time	Rotten_Tomatoes_Rating	IMDB_Rating	IMDB_Votes	R
count	3.201000e+03	3.201000e+03	5.640000e+02	3.200000e+03	3201.000000	1209.000000	2321.000000	2988.000000	2988.000000	
mean	4.390586e+07	8.515677e+07	3.490155e+07	3.106917e+07	1.780069	110.193548	54.336924	6.283467	29908.644578	
std	6.252066e+07	1.498363e+08	4.589512e+07	3.558591e+07	1.191791	20.171014	28.076593	1.252290	44937.582335	
min	0.000000e+00	0.000000e+00	6.184540e+05	2.180000e+02	0.000000	46.000000	1.000000	1.400000	18.000000	
25%	5.383834e+06	8.000000e+06	9.906211e+06	6.575000e+06	1.000000	95.000000	30.000000	5.600000	4828.500000	
50%	2.193042e+07	3.088578e+07	2.033156e+07	2.000000e+07	2.000000	107.000000	55.000000	6.400000	15106.000000	
75%	5.606855e+07	9.694754e+07	3.779422e+07	4.200000e+07	3.000000	121.000000	80.000000	7.200000	35810.500000	
max	7.601676e+08	2.767891e+09	3.525821e+08	3.000000e+08	4.000000	222.000000	100.000000	9.200000	519541.000000	

II. Prepocessing Data

Preface:

It's very important to deal with your missing data in the dataset, as you can see in the previous section, there are lost of missing data in several different columns and some columns have non-numerical data type. It's hard to find the precise relationship with these data, so we need to fill up the missing data and transform the data in a way we could manipulate and analysis.

Step:

1. US_Gross & Worldwide_Gross

The US_Gross and Worldwide_Gross is oringinal object type, the reason is because some rows contain 'Unknown'. Since it just has few lines, we replace it with the columns mean. After that we can change the object type to the most suitable numerical type and displace in the plot.

PD : US_Gross & Worldwide_Gross

```
1. Something we notice in the .info() is that the Dtype of US_Gross and Worldwide_Gross is object, which wh
ile we looking at the dataset, the most of value of US_Gross and Worldwide_Gross is numercial, just few row
s appears Unknown and 0. We will replace the Unknown to 0 and keep the 0.
```

```python
# convert the US_Gross and Worldwide_Gross into numeric type
# pd.to_numeric convert non-numeric type to the most suitable numeric type
#pd_movies[['US_Gross','Worldwide_Gross']] = pd_movies[['US_Gross','Worldwide_Gross']].apply(pd.to_numeric)

# 1 .replace the 'Unknown' to 0
pd_movies['US_Gross'] = pd_movies['US_Gross'].replace('Unknown',np.mean(pd_movies['US_Gross']))
pd_movies['Worldwide_Gross'] = pd_movies['Worldwide_Gross'].replace({"Unknown":np.mean(pd_movies['Worldwide_Gross']
```

executed in 9ms, finished 17:25:09 2022-11-01

```python
pd_movies['US_Gross'][118]
```

executed in 16ms, finished 17:25:12 2022-11-01

```
0
```

```python
# 2. pd.to_numeric convert non-numeric type to the most suitable numeric type
pd_movies[['US_Gross','Worldwide_Gross']] = pd_movies[['US_Gross','Worldwide_Gross']].apply(pd.to_numeric)
```

executed in 7ms, finished 17:25:14 2022-11-01

2. Release_Date

Time is a very important data, however, the time data in this dataset is messy. Most of the format is like '15-Jan-99'. This is a format computer cannot parse. It contains a string of month rather than 01-12, the year is not complete. Also some columns is like 'Jan-1989' which didn't contains day, some columns just has year like '1999', some columns just have 'TBD'.

So we create a for loop to deal with different type data and then use pd.to_datetime() to transform the data into datetime64 type. After that we can extract the useful informal like year, month and day.

```
# create a empty list

new_ReleaseDate = []

# define a month method
def StrMonth_to_int(argument):
    switcher = {
        'Jan':'01',
        'Feb':'02',
        'Mar':'03',
        'Apr':'04',
        'May':'05',
        'Jun':'06',
        'Jul':'07',
        'Aug':'08',
        'Sep':'09',
        'Oct':'10',
        'Nov':'11',
        'Dec':'12'
    }
    return switcher.get(argument,"nothing")

# For loop pd_movies['Relase_Date']
for date in pd_movies['Release_Date']:

    # if contain '-'
    if '-' in date:
        # yes, is 31-Jan-99 format
        new_date = date.split('-') #  generate a list
        # 1 = day, 2 = month, 3 = year
        day = new_date[0]
        month = new_date[1]
        year = new_date[2]
        # change day to xx or 0x format
        if int(day) < 10:
            day = '0' + day
        # change Jan to 1
        month = StrMonth_to_int(month)
        # change 99 to 1999 if ? > 22
        if int(year) > 22:
            year = '19' + year
        else :
            year = '20' + year
        # combie and convert to 1999-01-31
        date = year + month + day
        new_ReleaseDate.append(date)
        print(date)

    elif date == 'TBD':
        date = '20000101'
        new_ReleaseDate.append(date)
        print(date)
    # not, is 1975 format
    else :
        # convert to 1975-01-01
        date = date + '01' + '01'
        new_ReleaseDate.append(date)
        print(date)

# assign the new_ReleaseDate to pd_movies['Release_Date']
pd_movies['Release_Date'] = new_ReleaseDate
```

After we transform the data in a type computer can parse, it become easier.

```
pd_movies['Release_Date'].head()
```

executed in 9ms, finished 14:58:18 2022-11-01

```
0    19980612
1    19980807
2    19980828
3    19980911
4    19981009
Name: Release_Date, dtype: object
```

```
# Using pd.to_datetime to convert the Release_Date to datetime type
pd_movies['Release_Date'] = pd.to_datetime(pd_movies['Release_Date'])
```

3. G

We use pd.unique() function to find out what different types G has.

```
pd_movies['G'].unique()
```

executed in 26ms, finished 14:58:18 2022-11-01

```
array(['R', nan, 'PG', 'Not Rated', 'PG-13', 'G', 'NC-17', 'Open'],
      dtype=object)
```

Then we create a dictionary to transform the different type into numerical numbers, in this way, it will be more easlier to find the algibra relationship.

```
0=G --> General Audiences
1=PG --> Parental Guidance Suggested
2=PG-13 --> Parents Strongly Cautioned
3=R --> Ristricted Under 17 requires accompanying parent or adult guardian
4=NC-17 --> No One 17 and Under Admitted
```

For 'nan', 'Not Rated', 'Open' and missing data, we replace them with G=0=General Audiences since it is more reasonable. In Statistic most moives are G.

```
# first fill the missing data to 0=G
pd_movies['G'] = pd_movies['G'].fillna(0)
```

executed in 7ms, finished 14:58:18 2022-11-01

```
# second replace the data
pd_movies['G'] = pd_movies['G'].replace({'G':0, 'PG':1, 'PG-13':2, 'R':3, 'NC-17':4, 'nan':0, 'Not Rated':0, 'Open'
```

executed in 19ms, finished 14:58:18 2022-11-01

```
# convert it from object type to int
pd_movies['G'] = pd_movies['G'].apply(pd.to_numeric)
pd_movies['G']
```

executed in 17ms, finished 14:58:18 2022-11-01

```
0       3
1       3
2       0
3       0
4       3
       ..
3196    3
3197    3
3198    1
3199    1
3200    2
Name: G, Length: 3201, dtype: int64
```

4. DVD_Sales

for DVD_Sales, it has around 3000 missing data, only around 500 availiable. We barely can do nothing. It's not simply some movies didn't sale the DVD because of Release_Date. The data is just missing.

5. Creative_Type

Creative_Type

```
'Contemporary Fiction':0, 'Science Fiction':1,
    'Historical Fiction':2, 'Fantasy':3, 'Dramatization':4, 'Factual':5,
    'Super Hero':6, 'Multiple Creative Types':7, 'Kids Fiction':8, nan:9
```

```
pd_movies['Creative_Type'].unique()
```

executed in 8ms, finished 19:15:04 2022-11-01

```
array([nan, 'Contemporary Fiction', 'Science Fiction',
       'Historical Fiction', 'Fantasy', 'Dramatization', 'Factual',
       'Super Hero', 'Multiple Creative Types', 'Kids Fiction'],
      dtype=object)
```

```
pd_movies['Creative_Type'] = pd_movies['Creative_Type'].fillna(9)
pd_movies['Creative_Type'] = pd_movies['Creative_Type'].replace({'Contemporary Fiction':0, 'Science Fiction':1,
    'Historical Fiction':2, 'Fantasy':3, 'Dramatization':4, 'Factual':5,
    'Super Hero':6, 'Multiple Creative Types':7, 'Kids Fiction':8, 'nan':9})
pd_movies['Creative_Type'] = pd_movies['Creative_Type'].apply(pd.to_numeric)
```

executed in 16ms, finished 20:28:28 2022-11-01

```
pd_movies['Creative_Type'].head()
```

executed in 7ms, finished 20:34:01 2022-11-01

```
0    9
1    9
2    9
3    9
4    0
Name: Creative_Type, dtype: int64
```

6. Major_Genre

Major_Genre

```
'nan':0, 'Drama':1, 'Comedy':2, 'Musical':3, 'Thriller/Suspense':4,
   'Adventure':5, 'Action':6, 'Romantic Comedy':7, 'Horror':8, 'Western':9,
   'Documentary':10, 'Black Comedy':11, 'Concert/Performance':12
```

```
pd_movies['Major_Genre'].unique()
```

executed in 9ms, finished 20:31:13 2022-11-01

```
array([nan, 'Drama', 'Comedy', 'Musical', 'Thriller/Suspense',
       'Adventure', 'Action', 'Romantic Comedy', 'Horror', 'Western',
       'Documentary', 'Black Comedy', 'Concert/Performance'], dtype=object)
```

```
pd_movies['Major_Genre'] = pd_movies['Major_Genre'].fillna(0)
pd_movies['Major_Genre'] = pd_movies['Major_Genre'].replace({'nan':0, 'Drama':1, 'Comedy':2, 'Musical':3, 'Thriller
   'Adventure':5, 'Action':6, 'Romantic Comedy':7, 'Horror':8, 'Western':9,
   'Documentary':10, 'Black Comedy':11, 'Concert/Performance':12})
pd_movies['Major_Genre'] = pd_movies['Major_Genre'].apply(pd.to_numeric)
```

executed in 25ms, finished 20:33:38 2022-11-01

```
pd_movies['Major_Genre'].head()
```

executed in 7ms, finished 20:34:19 2022-11-01

```
0    0
1    1
2    2
3    2
4    1
```

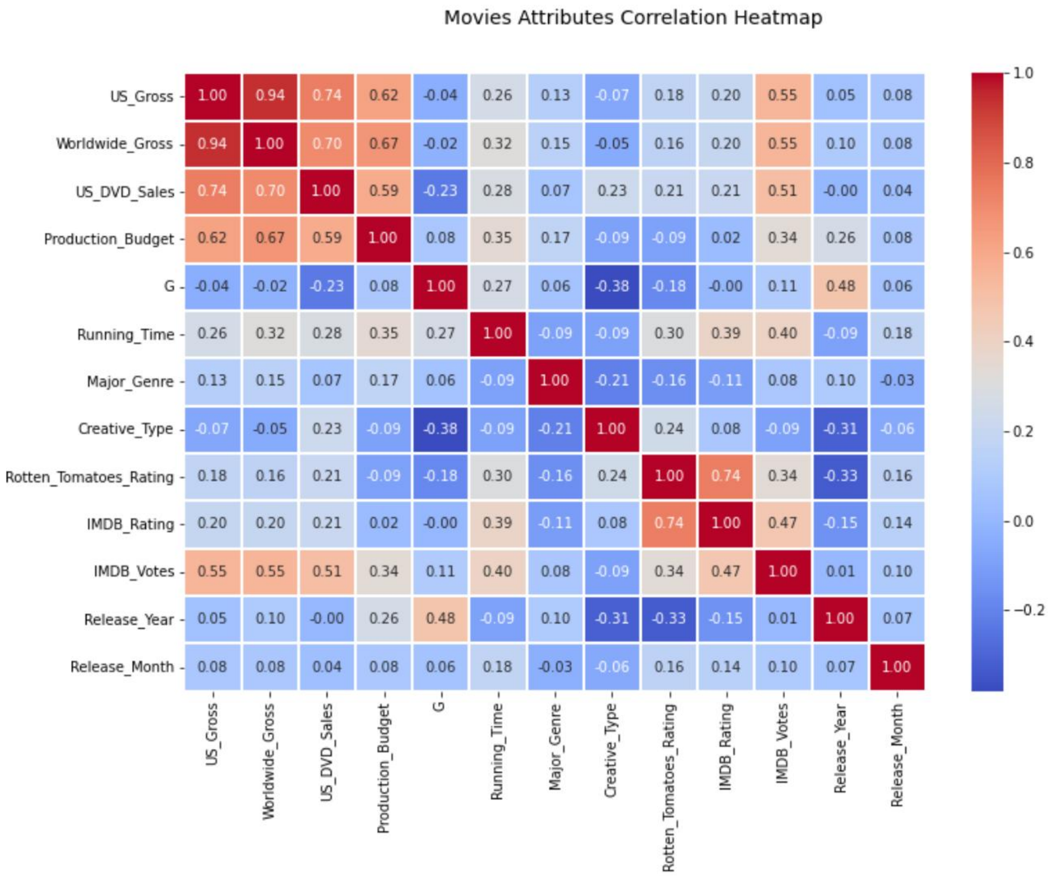
III. Questions and Visualiazation

Question 1: What is the relationship between the columns?

Heatmap:

One of the simpliest way to find out the linear relationship of different columns is the heatmap.

As you can see, after we proprocssing the data, the US_Gross, Worldwide_Gross, G, year and month can show on this heatmap.



Strong Positive Linear Relationship

1. US_Gross, Worldwide_Gross, US_DVD_Sales have very strong linear relationship. It's reasonable---- if a movie saled good in US, it probably will saled good in worldwide, and the DVD sale will be good rather bad.
2. Another strong linear relationship is Rotten_Tomatoes_Rating and IMDB_Rating. They are two different movie rating platforms, but it seems the audiences have around 70% of same feelings about the movies.

Good Positive Linear Relationship

3. The Production_Budget and {US_Gross, Worldwide_Gross, US_DVD_Sales} has a good linear relationship, if a movie has high buget, it has high possibility to become popular .
4. IMDB_Votes and {US_Gross, Worldwide_Gross, US_DVD_Sales} has good linear relationship. It means if a movie have more people rating them, this movie tend to performance good in the market.
5. IMDB_Votes and IMDB_Rating has a good positive linear relationship. It means in IMDB platform, if a movie has more people attend to vote, the rating usually will be higher.

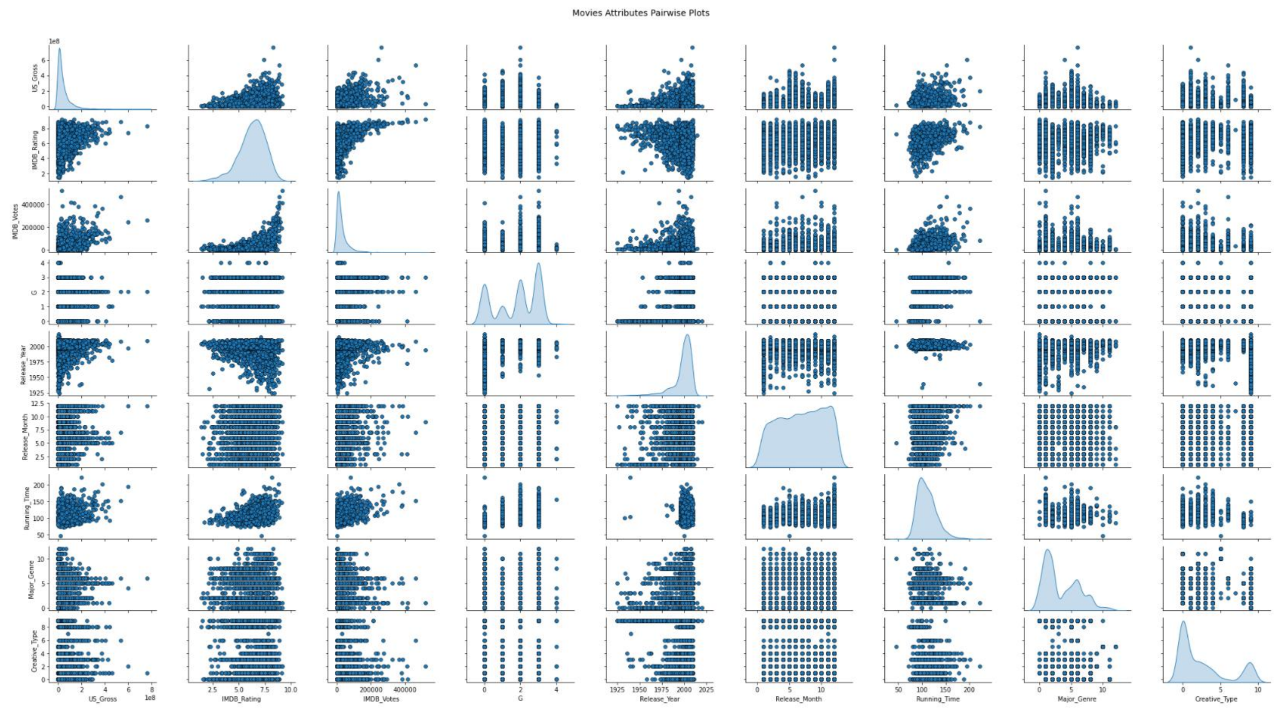
Here we come up with another question.

Question 2: Since IMDB_Votes and {US_Gross, Worldwide_Gross, US_DVD_Sales} has a good positive linear relationship and IMDB_Rating and IMDB_Votes has a good positive linear relationship, why IMDB_Rating and {US_Gross, Worldwide_Gross, US_DVD_Sales} has not positive linear relaitonship?

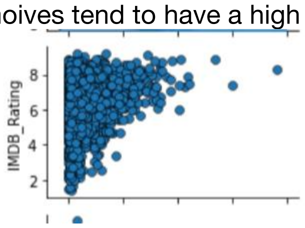
In another word, In IMDB platform, if a movie has more people to vote, the Rating, Gross tend to be high, however, if a moive rating tend to be high, that doesn't lead to a movie sales good. Which means, If a movie rating is high, the movies don't necessarly sale good.
Where this can lead to another interesting question.
Question 3: What kind of movies rating high but don't sales good.

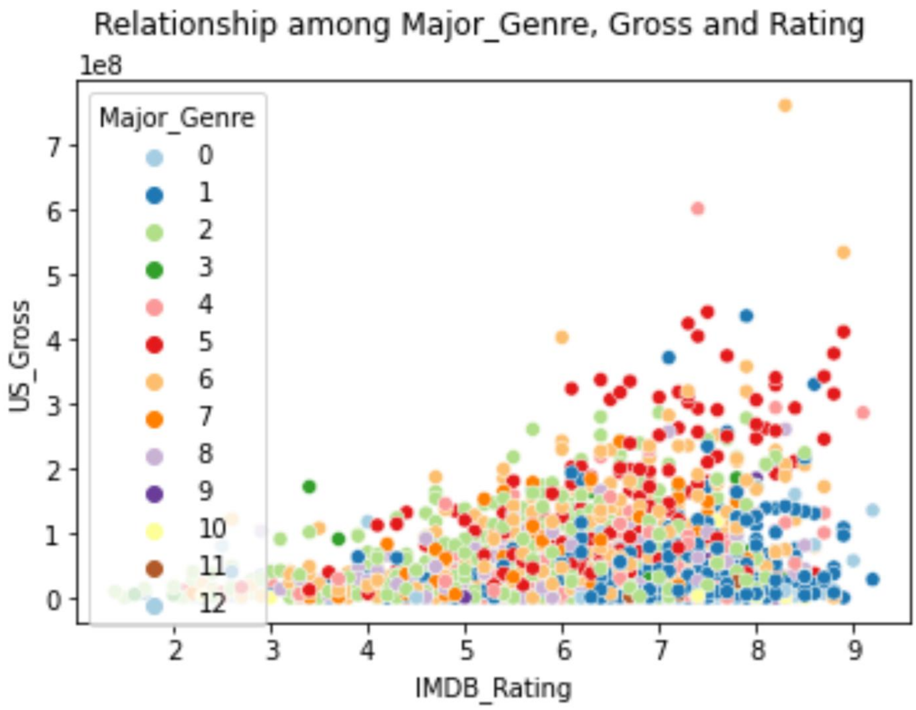
6. Release_Year and G has a good positive relationship.
This can lead to another interesting question.
Question 4: What is the detail relationship with the Release_Years and G?

Normal Positive Linear Relationship
7. Running_Time and {IMDB_Rating, IMDB_Votes} has a normal positive relationship.

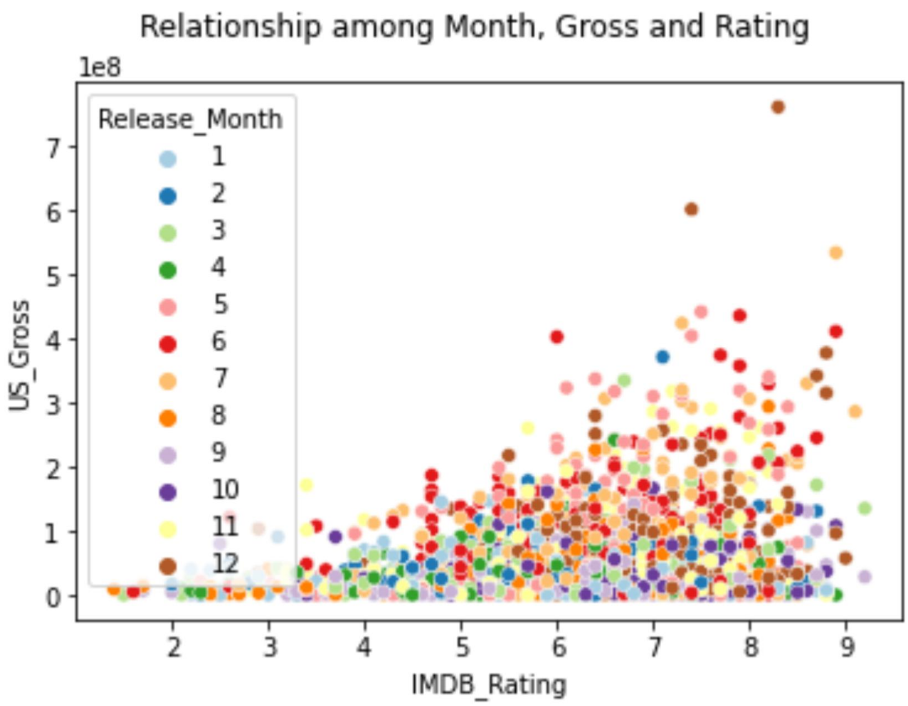


From the pair plots, we can answer Q2, it seems US_Gross and IMDB_Rating has a positive non-linear relationship. From the picture, we can tell if the rating is high, there are more moives tend to have a high gross.

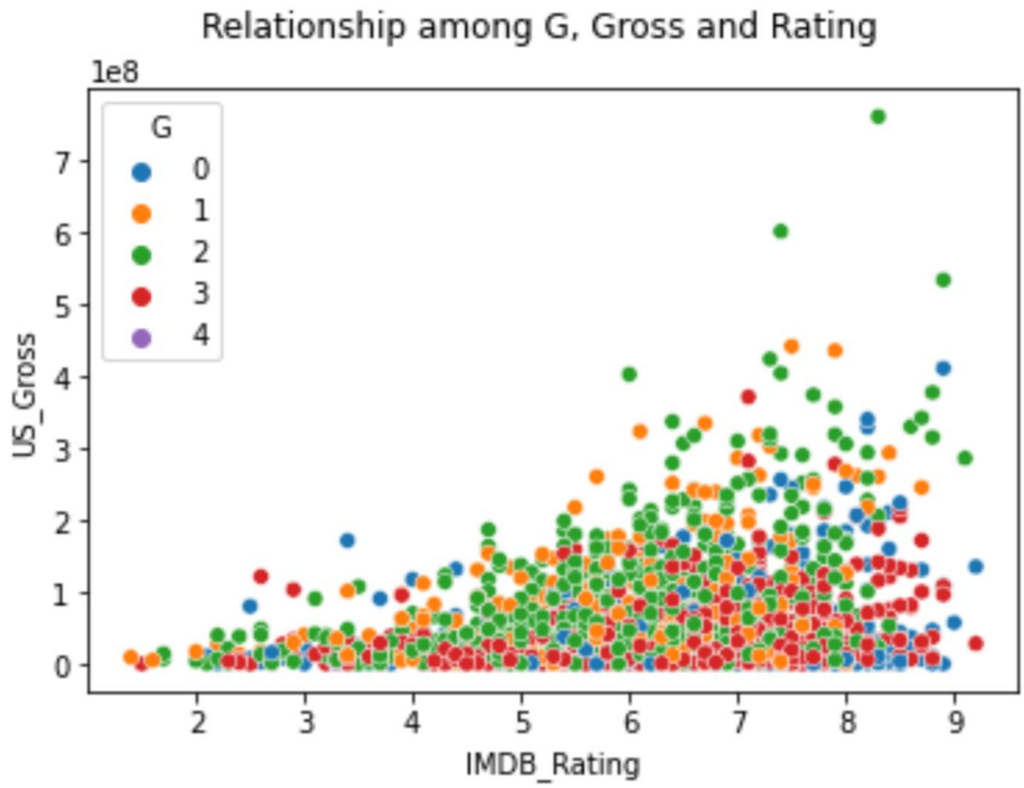




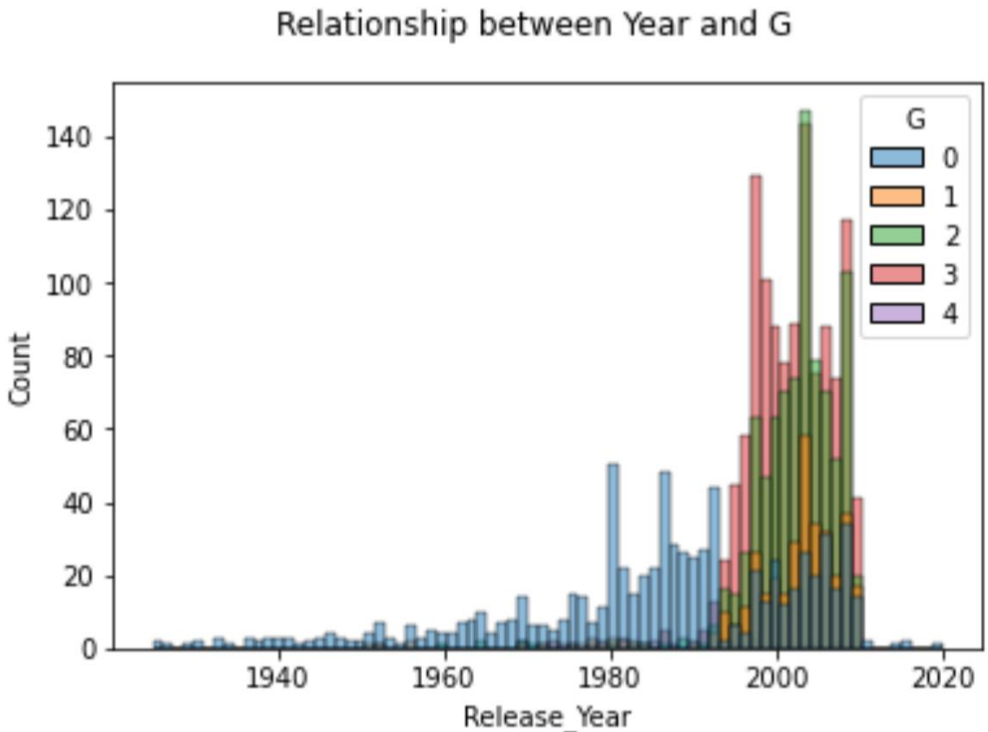
For Question3, as you can see, there are many blue dots have high Rating but low Gross. And some green, orange too. So we can conclude that there are lots of Drama rating high but sales little. Let makes sense right, drama ususally have based on very good literature, it would be easy to get a high rating score, however, no everyone like drama. green are comdey, so some comdey are rating high but sales low. Orange are actions movie, they kind of in the middle.



For month, we can not see specific relationship with Gross and Rating.



As you can see above, Red dots(NC-17) are in the bottom of the US_Gross, even lots of high rating movies. The moives that are sales high and rating high are Green dots(PG-13).



As you can see, before 1995, most movies are General Audiences. But after 1995, PG-13 and R movies surpase the G and overcome the movie domian.

VI. Summary and Chanllenge

From the visualization technology, we can more clearly tell the realationship between different columns. And we also can find some insight in the movie industrial.

We see several columns have linear and non-linear positive relationship. We see some drama has high rating but small gross, NC-17 movies are more likely to have high rating and high gross. We see movies R and PG-13 movies become booming after 1995, but before 1995, most movie are General audiences.

The main chanllenge here is dealing with the missing data. This dataset has lots of missing data. Some data we can easly fill up, but some need other method. Like Running Time, it has around 2000 missing data, you must know the whole rows is just 3201. It's not reasonable to use mean or EM algorithm to fill up the data, I think one more accurate way is using other moive database which contain the information to fill up the missing data. Same thing also apply to Director and General Type. It's not something we can use AI to fill up because there are not enough information, only thing we can do is either ignore these columns of data or fill up with correct data using other resoucres.

One of the hard part of visualize is there are so many data points in a single picture, is hard to distinguish between them. And I don't know how to do in this situation, maybe using continues plot?