DAND Project 2 - Investigate a Dataset

November 19, 2020

1 Project: Investigate the Movie Dataset

1.1 Project Outlines

Introduction

Introduction of the dataset and the description of two questions that I plan to explore.

Data Wrangling

In this part, I will take steps to clean the data set and convert it to usable format. Steps to clean the data includes: 1. handle missing values; 2. handle duplicated values; 3. handle complex string features and extract usable information (feature engineering); 4. handle data types and formats;

Exploratory Data Analysis

In this part, I will explore the clean data set with visualization, and try to answer two questions: 1. Which genres are most popular from year to year? 2. What kinds of properties are associated with movies that have high revenues?

Conclusions

Summary of the findings and potential improvements.

Introduction 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 two questions I will try to answer with exploratory data analysis: 1. Which genres are most popular from year to year? In this sector, I will try to find out the 5 most popular genres over 1960-2015. Also I will take a closer look at the popular genres over each decade since 1960. 2. What kinds of properties are associated with movies that have high revenues? In this section, the specific questions I will try to tackle include: * How budget, vote_count and vote_average are correlated to revenue? * In which quarter is the movie likely to have higher revenue? * What genres are more likely to associated with higher revenue? * What are the top 5 companies that produce the highest revenue movies over 1960-2015? * Who are the 10 most revenue-generating directors over 1960-2015? * Who are the 10 most revenue-generating actors/actresses over 1960-2015?

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

In [2]:
```
movies = pd.read_csv('../input/dataanalystnanodegree/tmdb-movies.csv')
movies.head(5)
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...	overview	runtime	genres
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	...	Twenty-two years after the events of Jurassic ...	124	Action|Adventure|Science Fiction|Thriller
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	...	An apocalyptic story set in the furthest reach...	120	Action|Adventure|Science Fiction|Thriller
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	...	Beatrice Prior must confront her inner demons ...	119	
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...	J.J. Abrams	Every generation has a story.	...	Thirty years after defeating the Galactic Empi...	136	
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Vengeance Hits Home	...	Deckard Shaw seeks revenge against Dominic Tor...	137	

```
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]
```

According to the description of the data field, the *revenue_adj* and *budget_adj* are better version of a movie's actual budget and revenue for time series comparision purpose. So I will drop the *revenue* and *budget* columns. Other fields I will drop include *imdb_id* (use *id* instead), original_title, homepage, runtime, tagline, and overview.

```
In [3]: movie_df = movies.drop(['imdb_id','homepage','runtime','tagline','overview','budget','re

In [4]: movie_df.shape

Out[4]: (10866, 14)

In [5]: movie_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10866 non-null  int64
 1   popularity            10866 non-null  float64
 2   original_title        10866 non-null  object
 3   cast                  10790 non-null  object
 4   director              10822 non-null  object
 5   keywords              9373 non-null   object
 6   genres                10843 non-null  object
 7   production_companies  9836 non-null   object
 8   release_date          10866 non-null  object
 9   vote_count            10866 non-null  int64
 10  vote_average          10866 non-null  float64
```

```
 11  release_year          10866 non-null  int64
 12  budget_adj            10866 non-null  float64
 13  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(3), object(7)
memory usage: 1.2+ MB
```

Data Wrangling

1.1.1 Part I: Missing Value

```
In [6]: # calculate the percentage of missing values in each column in the dataframe
        missing_value_pct = movie_df.isnull().sum()/movie_df.shape[0]
        missing_value_pct
```

```
Out[6]: id                      0.000000
        popularity              0.000000
        original_title          0.000000
        cast                    0.006994
        director                0.004049
        keywords                0.137401
        genres                  0.002117
        production_companies    0.094791
        release_date            0.000000
        vote_count              0.000000
        vote_average            0.000000
        release_year            0.000000
        budget_adj              0.000000
        revenue_adj             0.000000
        dtype: float64
```

From the above we can tell the percentage of data records with at least one missing value is around 13%, which is acceptable and won't alter the data analysis very much if we remove these data records with missing values.

```
In [7]: movie_df.dropna(inplace=True)
        # to check if all rows with missing value have been removed
        movie_df.isnull().sum()/movie_df.shape[0]
```

```
Out[7]: id                      0.0
        popularity              0.0
        original_title          0.0
        cast                    0.0
        director                0.0
        keywords                0.0
        genres                  0.0
        production_companies    0.0
        release_date            0.0
        vote_count              0.0
```

```
vote_average            0.0
release_year            0.0
budget_adj              0.0
revenue_adj             0.0
dtype: float64
```

```
In [8]: movie_df.shape
```

```
Out[8]: (8667, 14)
```

1.1.2 Part II: Duplicates

```
In [9]: # check if there are any duplicated records and how many
        movie_df.duplicated().sum()
```

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

```
In [10]: # delte the duplicated row
         movie_df.drop_duplicates(inplace=True)
```

1.1.3 Part III: Complicated String Features

```
In [11]: string_feats = ['cast','director','keywords','genres','production_companies']
         movie_df[string_feats].head(3)
```

```
Out[11]:                                                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

                                                 keywords  \
         0  monster|dna|tyrannosaurus rex|velociraptor|island
         1   future|chase|post-apocalyptic|dystopia|australia
         2  based on novel|revolution|dystopia|sequel|dyst...

                                             genres  \
         0  Action|Adventure|Science Fiction|Thriller
         1  Action|Adventure|Science Fiction|Thriller
         2         Adventure|Science Fiction|Thriller

                                       production_companies
         0  Universal Studios|Amblin Entertainment|Legenda...
         1  Village Roadshow Pictures|Kennedy Miller Produ...
         2  Summit Entertainment|Mandeville Films|Red Wago...
```

As can be seen from the above, there are usually more than one values in fields cast, keywords, genres, and production_companies. And the number of values are not necessarily the same across different rows under the same column. Additionally, the values in column director are composed of multiple words, instead of one single string value. The idea here is that, for columns with

more than one values, pick up the most important values from each column, assuming the most important values are the ones that appear first in column. For columns with one value that is composed of multiple words, combine the words into one single string so it can be used in analysis later.

First I will process column production_companies. For this column, I will only pick up the first listed production company assuming it's the primary and most important one.

```
In [12]: # Since values in production_companies are separated by pipe (|) character, we can use
         movie_df['production_companies'] = movie_df['production_companies'].apply(lambda x: x.s

In [13]: # there are 2649 unique values in the column after the process
         movie_df['production_companies'].nunique()

Out[13]: 2649
```

 # Extract top 20 production companies from the column, and save the values to a list for later use top_20 = movie_df['production_companies'].value_counts(ascending=False).head(20).index.to_list() # Only recognize the top 20 companies and tag the other companies as 'Other' movie_df['production_companies'] = movie_df['production_companies'].apply(lambda x: 'Other' if x not in top_20 else x) *# Check if this column is properly handled* movie_df['production_companies'].value_counts()

Next I will process columns cast, keywords, and genres. For these columns, I will pick top 3 values for each column and save them to a list. Unlike production company, one single value in each of the three fiends probably won't be sufficient to describe a movie. Thus we need to extract three values for each field. Again, I assume the first 3 listed values will be the most important values.

```
In [14]: # define a function get_list to extract the first 3 values in a given column
         def get_list(x):
             values = x.split('|')
             if len(values) > 3:
                 values = values[:3]
             return values

In [15]: feats = ['cast','keywords','genres']
         for feat in feats:
             movie_df[feat] = movie_df[feat].apply(get_list)
         # check if the above columns as properly processed
         movie_df[feats].head(3)

Out[15]:                                                 cast  \
         0  [Chris Pratt, Bryce Dallas Howard, Irrfan Khan]
         1   [Tom Hardy, Charlize Theron, Hugh Keays-Byrne]
         2     [Shailene Woodley, Theo James, Kate Winslet]

                                       keywords  \
         0       [monster, dna, tyrannosaurus rex]
         1       [future, chase, post-apocalyptic]
         2  [based on novel, revolution, dystopia]
```

```
                                      genres
0    [Action, Adventure, Science Fiction]
1    [Action, Adventure, Science Fiction]
2  [Adventure, Science Fiction, Thriller]
```

Next I will convert all the string values in each column to lower case. I define a function get_lower_case to convert string values. This function handles scenario when a column contains a list and when a column contaisn only a string. Additionally, this function will get rid of the space between words if these words belong to a name or a phrase. For example, it will convert 'Tom Hardy' to 'tomhardy', and convert 'Science Fiction' to 'sciencefiction'.

```
In [16]: def get_lower_case(x):
             if isinstance(x,list):
                 return [str.lower(i.replace(' ','')) for i in x]
             else:
                 if isinstance(x,str):
                     return str.lower(x.replace(' ',''))
                 else:
                     return ''
```

```
In [17]: # iterate over the column names saved in string_feats earlier, and convert string value
         for feat in string_feats:
             movie_df[feat] = movie_df[feat].apply(get_lower_case)
         movie_df.head(3)
```

```
Out[17]:       id  popularity      original_title  \
         0  135397   32.985763      Jurassic World
         1   76341   28.419936  Mad Max: Fury Road
         2  262500   13.112507           Insurgent

                                          cast         director  \
         0  [chrispratt, brycedallashoward, irrfankhan]   colintrevorrow
         1  [tomhardy, charlizetheron, hughkeays-byrne]     georgemiller
         2    [shailenewoodley, theojames, katewinslet]  robertschwentke

                                keywords  \
         0      [monster, dna, tyrannosaurusrex]
         1     [future, chase, post-apocalyptic]
         2  [basedonnovel, revolution, dystopia]

                                    genres     production_companies  \
         0    [action, adventure, sciencefiction]         universalstudios
         1    [action, adventure, sciencefiction]  villageroadshowpictures
         2  [adventure, sciencefiction, thriller]      summitentertainment

           release_date  vote_count  vote_average  release_year    budget_adj  \
         0       6/9/15        5562           6.5          2015  1.379999e+08
         1      5/13/15        6185           7.1          2015  1.379999e+08
```

```
2      3/18/15        2480           6.3          2015   1.012000e+08

    revenue_adj
0  1.392446e+09
1  3.481613e+08
2  2.716190e+08
```

1.1.4 Part IV: Data Type and Data Format

```
In [18]: movie_df.dtypes

Out[18]: id                      int64
         popularity            float64
         original_title         object
         cast                   object
         director               object
         keywords               object
         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
```

Everything else looks good except for column release_date. It should be a datatime column. However, this column is somewhat irrelevant to the questions I'm trying to answer here, so probably I can drop this column. Before I do that, I'd like to extract a new field from this column: the season or the quarter in which the movie is released. And I define the relation like this: > * Q1: if month number is 1, 2, 3; > * Q2: if month number is 4, 5, 6 > * Q3: if month number is 7, 8, 9 > * Q4: if month number is 10, 11, 12

```
In [19]: def find_quarter(x):
             month = x.split('/')[0]
             if month in ['1','2','3']:
                 quarter = 'Q1'
             elif month in ['4','5','6']:
                 quarter = 'Q2'
             elif month in ['7','8','9']:
                 quarter = 'Q3'
             else:
                 quarter = 'Q4'
             return quarter

In [20]: movie_df['release_quarter'] = movie_df['release_date'].apply(find_quarter)
         movie_df.drop(['release_date'],axis=1,inplace=True)
```

```
In [21]: movie_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 8666 entries, 0 to 10865
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    8666 non-null   int64
 1   popularity            8666 non-null   float64
 2   original_title        8666 non-null   object
 3   cast                  8666 non-null   object
 4   director              8666 non-null   object
 5   keywords              8666 non-null   object
 6   genres                8666 non-null   object
 7   production_companies  8666 non-null   object
 8   vote_count            8666 non-null   int64
 9   vote_average          8666 non-null   float64
 10  release_year          8666 non-null   int64
 11  budget_adj            8666 non-null   float64
 12  revenue_adj           8666 non-null   float64
 13  release_quarter       8666 non-null   object
dtypes: float64(4), int64(3), object(7)
memory usage: 1015.5+ KB
```

So far things look to be in good shape. I will move forward and start exploratory data analysis using the pre-processed data. ## Exploratory Data Analysis

This part I will try to answer the questions brought up during project introduction.

1.1.5 Question 1: Which genres are most popular from year to year?

Q 1.1 What are the 5 most popular genres over 1960-2015? For the genres column I extracted three different values for each row. I will count all of these values into the genres frequency. I will first create a sub-dataframe containing only columns release_year and genres, create a separate column for each of the genre value, and then melt the sub-dataframe into a 2-column dataframe again.

```
In [22]: # create a sub-dataframe containing only columns release_year, genres and revenue_adj.
         df_genre = movie_df[['release_year','genres','revenue_adj']]

         # create separate column for each genre value
         df_genre['genres1'] = df_genre['genres'].apply(lambda x: x[0])
         df_genre['genres2'] = df_genre['genres'].apply(lambda x: x[1] if len(x)>1 else None)
         df_genre['genres3'] = df_genre['genres'].apply(lambda x: x[2] if len(x)>2 else None)
         # remove original column genres, and melt 4-column dataframe into 2-column dataframe
         df_genre = df_genre.drop('genres',axis=1).melt(['release_year','revenue_adj']).drop('va
         df_genre.head(3)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
  import sys
```

Out[22]:

	release_year	revenue_adj	genres
0	2015	1.392446e+09	action
1	2015	3.481613e+08	action
2	2015	2.716190e+08	adventure

In [23]:
```
df_genre.shape
```

Out[23]:
```
(25998, 3)
```

In [24]:
```
# group the dataframe by genres and release_year, and count the frequency of different
genres_count = df_genre.groupby(['genres','release_year']).size().reset_index(name='cou
# pivot the grouped dataframe so each row represents a year and each column represents
genres_count = genres_count.pivot(index='release_year', columns='genres', values='count
genres_count.head(3)
```

Out[24]:

genres	action	adventure	animation	comedy	crime	documentary	drama
release_year							
1960	7.0	5.0	NaN	7.0	1.0	NaN	12.0
1961	7.0	5.0	1.0	7.0	2.0	NaN	16.0
1962	6.0	7.0	NaN	5.0	3.0	NaN	18.0

genres	family	fantasy	foreign	history	horror	music	mystery
release_year							
1960	3.0	2.0	NaN	3.0	7.0	1.0	NaN
1961	2.0	1.0	NaN	3.0	3.0	2.0	1.0
1962	1.0	1.0	NaN	3.0	5.0	1.0	3.0

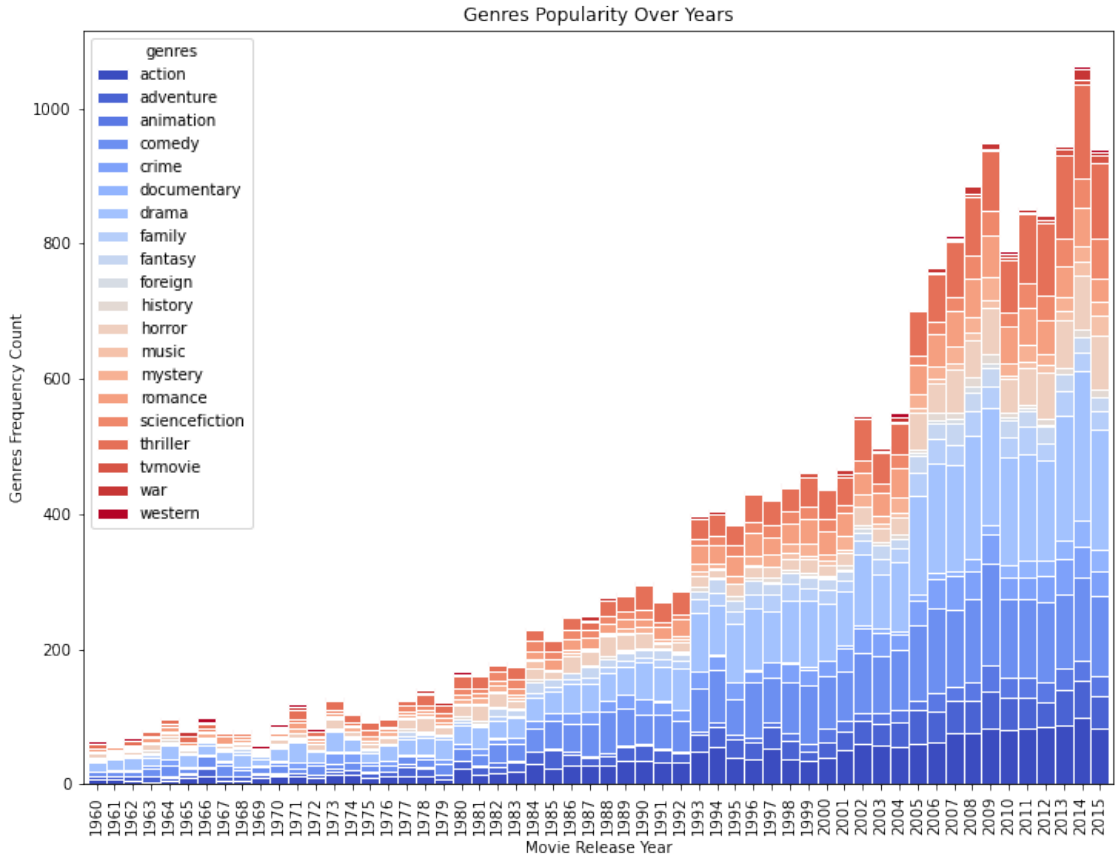
genres	romance	sciencefiction	thriller	tvmovie	war	western
release_year						

1960	4.0	2.0	6.0	NaN	NaN	4.0
1961	5.0	2.0	NaN	NaN	NaN	3.0
1962	3.0	2.0	7.0	NaN	NaN	3.0

Next I will plot a stacked boxplot to display the change of genres frequency over years.

```
In [25]: genres_count.plot(kind='bar',stacked=True,figsize=(12,9),cmap='coolwarm',edgecolor='whi
         plt.title('Genres Popularity Over Years')
         plt.xlabel('Movie Release Year')
         plt.ylabel('Genres Frequency Count')
```

```
Out[25]: Text(0, 0.5, 'Genres Frequency Count')
```

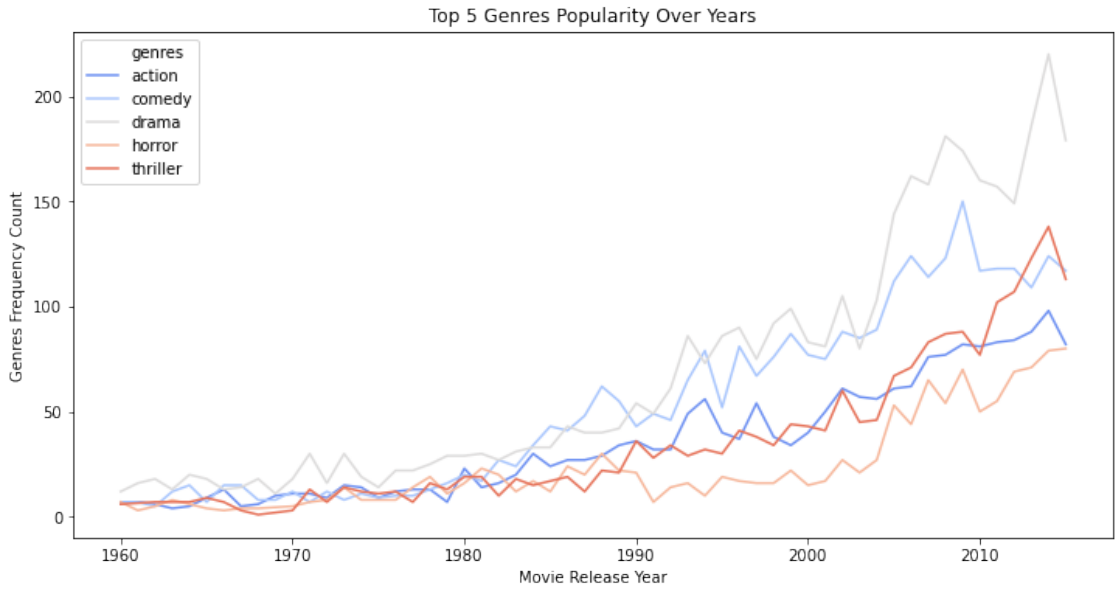


The above graph shows the over years, the 5 most popular genres are action, comedy, drama, horror, and thriller. I will pick up the 5 genren and show the trends in a line plot per below. Basically you can tell the trend shows that what's popular most likely has always been popular.

```
In [26]: plt.figure(figsize=(12,6))
         sns.lineplot('release_year','counts',hue='genres',palette='coolwarm',data=df_genre.grou

         plt.title('Top 5 Genres Popularity Over Years')
         plt.xlabel('Movie Release Year')
         plt.ylabel('Genres Frequency Count')
```

```
Out[26]: Text(0, 0.5, 'Genres Frequency Count')
```



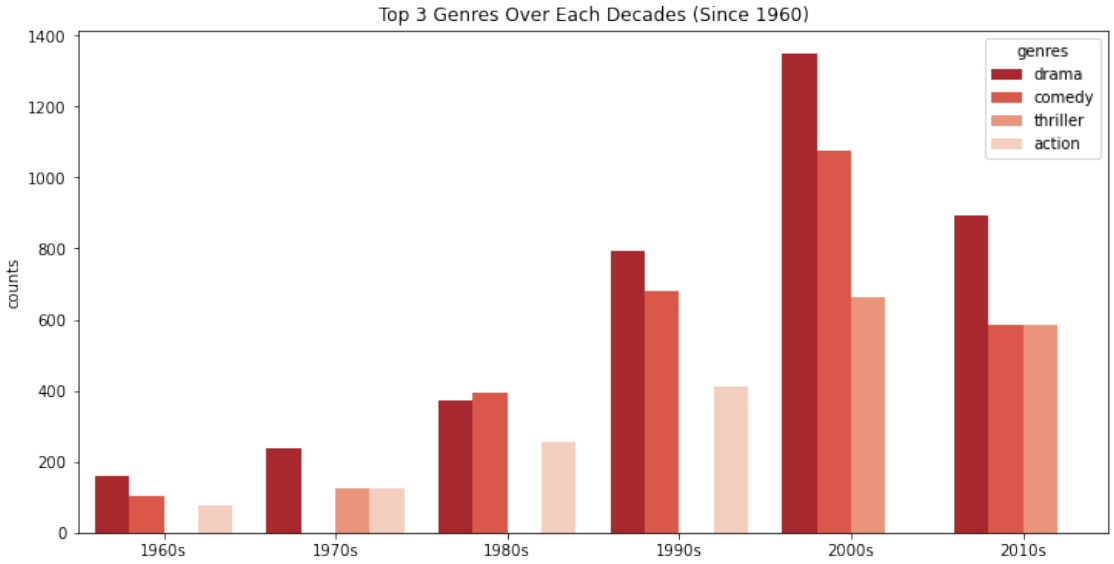
Q1.2 What are the popular genres over each decade since 1960? First I will add a new column "decades" to the dataframe df_genre. And then I will display the top 3 genres over each decade in a bar chart.

```
In [27]: # use pandas function .cut() to segregate release_year into each decade
         bin_edge = [1960, 1970, 1980, 1990, 2000, 2010, 2015]
         bin_name = ['1960s','1970s','1980s','1990s','2000s','2010s']
         df_genre['decades'] = pd.cut(df_genre['release_year'],bins=bin_edge,labels=bin_name)

In [28]: genres_decade = df_genre.groupby(['decades','genres']).size().reset_index(name='counts'
         genres_decade = genres_decade.sort_values(by=['decades','counts'],ascending=[False,Fals

         plt.figure(figsize=(12,6))
         sns.barplot(x='decades',y='counts',hue='genres',data=genres_decade,palette='Reds_r')
         plt.title('Top 3 Genres Over Each Decades (Since 1960)')
         plt.xlabel('')

Out[28]: Text(0.5, 0, '')
```



From the above we can see drama has always been the most popular genre except over 1980s, when the most popular genre became comedy.

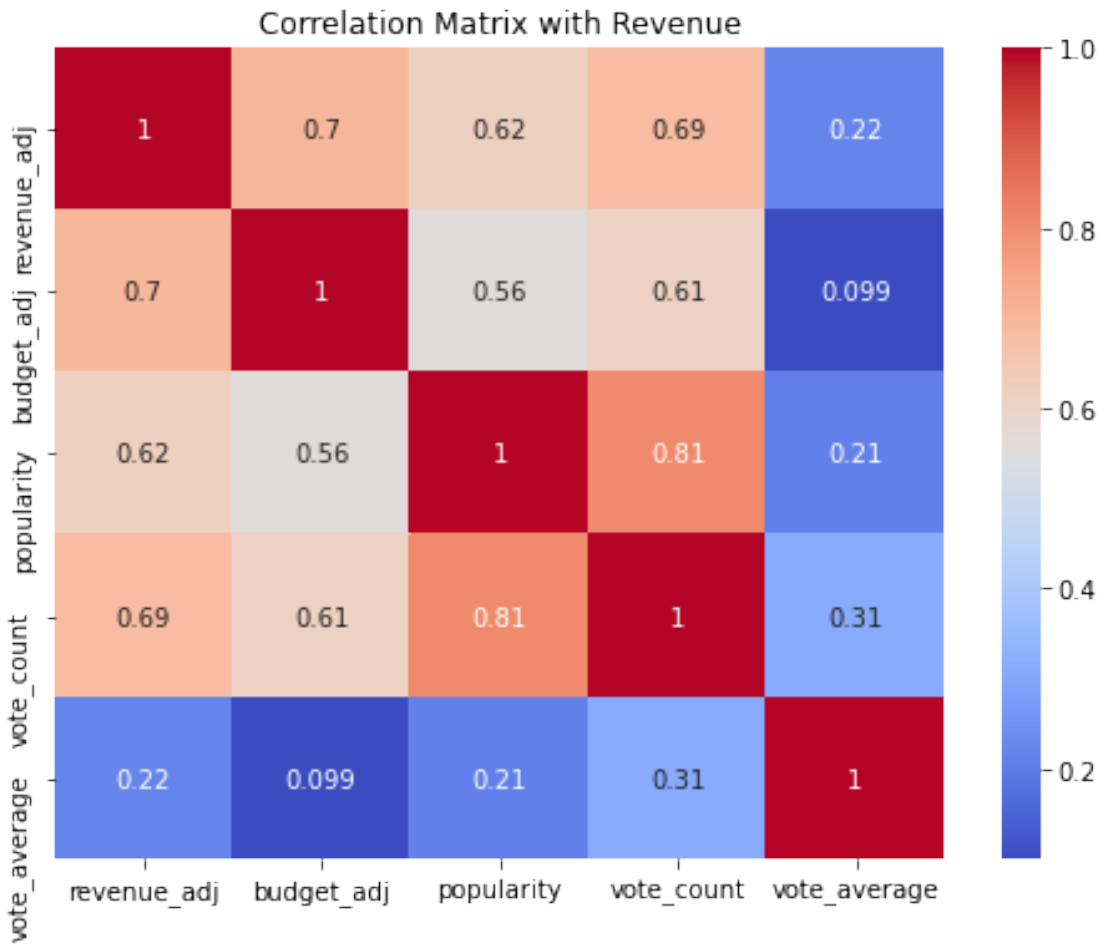
1.1.6 Question 2: What kinds of properties are associated with movies that have high revenues?

Q2.1 How budget, popularity, vote_count and vote_average are correlated to revenue?

```
In [29]: corr = movie_df[['revenue_adj','budget_adj','popularity','vote_count','vote_average']].

         plt.figure(figsize=(8,6))
         sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,annot=True,cmap='coo
         plt.title('Correlation Matrix with Revenue')

Out[29]: Text(0.5, 1.0, 'Correlation Matrix with Revenue')
```

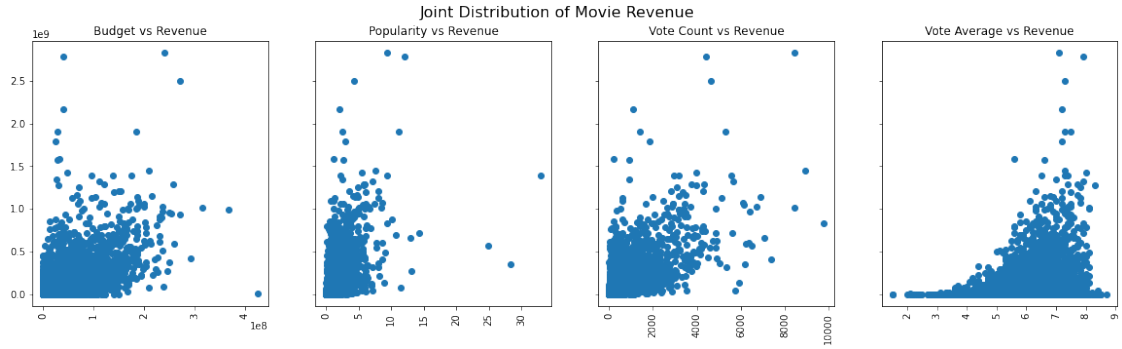


The correlation matrix shows that all four numeric features have positive correlation with revenue. Buedget, vote count and popularity have relatively strong correlation with revenue, while vote average has much weaker correlation with revenue. Below shows the joint distribution between revenue and each of the four numeric features. We can tell that a big budget movie will likely generate high revenue. If a movie is tagged as popular and many people provide voting for the movie, the revenue of the movie is also likely to be relatively high. On the other hand, if a movie has a high average voting rate, it doesn't necessarily mean that the movie will achive high revenue, maybe because sometimes the movie is only successful among a very limited of target audience with unique taste.

```
In [30]: fig,axes = plt.subplots(nrows=1,ncols=4,sharey=True,figsize=(20,5))
         fig.suptitle('Joint Distribution of Movie Revenue', fontsize=16)
         for ax in fig.axes:
             plt.sca(ax)
             plt.xticks(rotation=90)
         axes[0].scatter(x='budget_adj',y='revenue_adj',data=movie_df)
         axes[0].set_title('Budget vs Revenue')
         axes[1].scatter(x='popularity',y='revenue_adj',data=movie_df)
```

```
axes[1].set_title('Popularity vs Revenue')
axes[2].scatter(x='vote_count',y='revenue_adj',data=movie_df)
axes[2].set_title('Vote Count vs Revenue')
axes[3].scatter(x='vote_average',y='revenue_adj',data=movie_df)
axes[3].set_title('Vote Average vs Revenue')
```

```
Out[30]: Text(0.5, 1.0, 'Vote Average vs Revenue')
```



Q2.2 In which quarter is the movie likely to have higher revenue?

```
In [31]: movie_df[movie_df['revenue_adj']>0].groupby('release_quarter')['revenue_adj'].describe(
```

```
Out[31]:                  count          mean           std        min           25%  \
release_quarter
Q1                        929.0  8.664650e+07  1.671436e+08  10.000000  1.198461e+07
Q2                       1050.0  1.644706e+08  2.458958e+08  14.733479  1.713435e+07
Q3                       1316.0  9.107489e+07  1.505110e+08   6.951084  8.612733e+06
Q4                       1241.0  1.436953e+08  2.269583e+08   2.861934  1.933067e+07

                              50%           75%           max
release_quarter
Q1                   3.910921e+07  9.678616e+07  2.789712e+09
Q2                   6.450178e+07  2.072651e+08  1.907006e+09
Q3                   3.555807e+07  1.040260e+08  1.583050e+09
Q4                   6.882570e+07  1.761638e+08  2.827124e+09
```

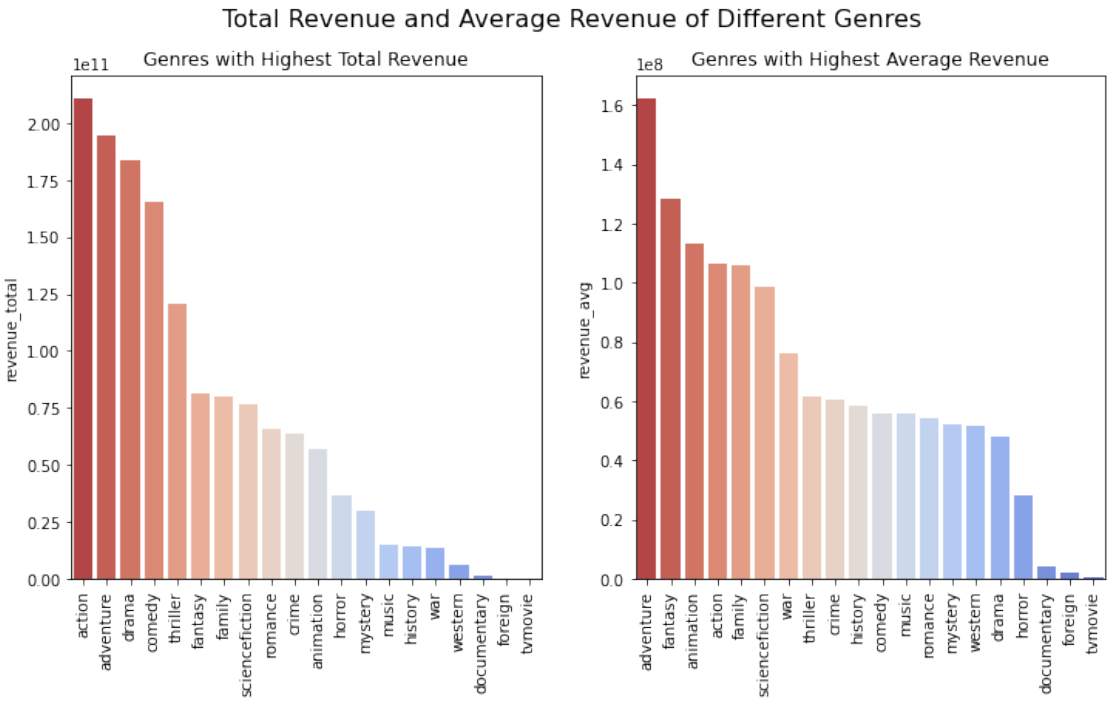
From the descriptive statistics of movie revenue data over each quarter, we can tell that Q3 and Q4 have the most number of movies released which might be due to that movie producers hope to occupy the idle time of great amount of audience during summer vacation and holiday seasons at year end. However movies released during Q3 don't necessarily have high revenue. Instead, Q1 and Q4 have the highest maximum revenue, while Q2 and Q4 have relatively high average revenue and median revenue. So to summarize, if a movie is release in Q4, statistically it's more likely to have high revenue.

Q2.3 What genres are more likely to associated with higher revenue? I will use the dataframe df_genre saved from question 1 and add two aggregation columns revenue_avg and revenue_total. Then save the new aggregation columns as well as column genres to a new dataframe df_rev.

```
In [32]: df_rev = df_genre.drop('release_year',axis=1).groupby('genres')['revenue_adj'].agg([('r

In [33]: fig,axes = plt.subplots(nrows=1,ncols=2,sharey=False,figsize=(12,6))
         fig.suptitle('Total Revenue and Average Revenue of Different Genres', fontsize=16)
         for ax in fig.axes:
             plt.sca(ax)
             plt.xticks(rotation=90)
         sns.barplot(ax=axes[0],x=df_rev.sort_values(by='revenue_total',ascending=False).index,y
         axes[0].set_title('Genres with Highest Total Revenue')
         axes[0].set_xlabel('')
         sns.barplot(ax=axes[1],x=df_rev.sort_values(by='revenue_avg',ascending=False).index,y='
         axes[1].set_title('Genres with Highest Average Revenue')
         axes[1].set_xlabel('')

Out[33]: Text(0.5, 0, '')
```



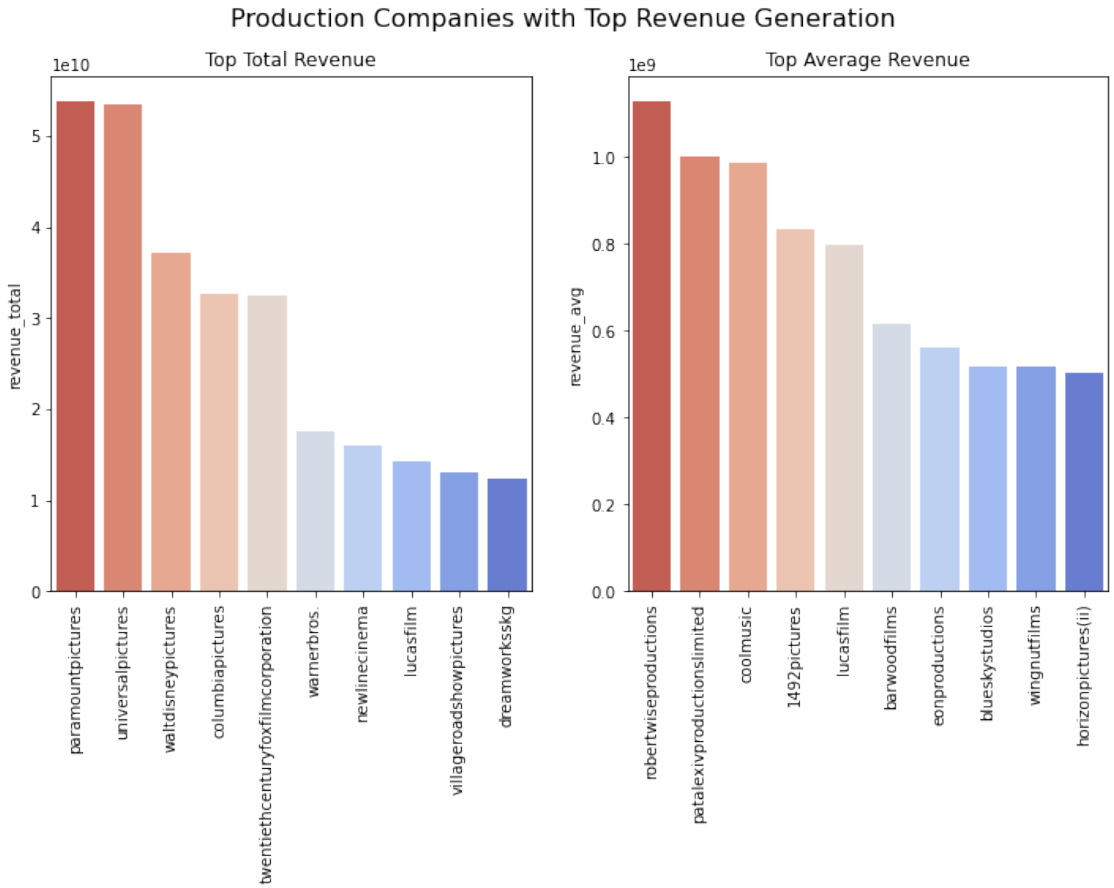
From the above we can tell that the top 5 genres with the highest total revenue contributions are action, adventure, drama, comedy and thriller. And the top 5 genres with the highest average revenue are adventure, fantasy, animation, action adn family.

Q2.4 What are the top 5 companies that produce the highest revenue movies over 1960-2015?

```
In [34]: df_prod = movie_df[['production_companies','revenue_adj']].groupby('production_companie

In [35]: fig,axes = plt.subplots(nrows=1,ncols=2,sharey=False,figsize=(12,6))
         fig.suptitle('Production Companies with Top Revenue Generation', fontsize=16)
         for ax in fig.axes:
             plt.sca(ax)
             plt.xticks(rotation=90)
         sns.barplot(ax=axes[0],x=df_prod.sort_values(by='revenue_total',ascending=False)[:10].i
         axes[0].set_title('Top Total Revenue')
         axes[0].set_xlabel('')
         sns.barplot(ax=axes[1],x=df_prod.sort_values(by='revenue_avg',ascending=False)[:10].ind
         axes[1].set_title('Top Average Revenue')
         axes[1].set_xlabel('')

Out[35]: Text(0.5, 0, '')
```



The top 5 production companies with top revenue generation ability are Paramount Pictures, Universal Pictures, Walt Disney Pictures, Columbia Pictures, and Twentieth Century Fox Film Corporation. This finding aligns with my general impression that those companies are the most

famous production and distribution companies. We can see from the visualization that these 5 companies' revenue generation ability are far more superior than the rest of the companies.

On the other hand, the top 5 production companies who have produced movies with high average revenue are Robert Wise Productions, Patalex IV Productions Limited, Cool Music, 1492 Pictures, and Lucas Film. I then took a further look at the movies produced by these 5 companies, turns out they basically are the producers of Harry Potter series and Star War series. These are among the blockbuster movies so it makes sense that these production companies have highest average revenue. The table below shows the details.

```
In [36]: movie_df.query('production_companies in ["robertwiseproductions","patalexivproductionsl
```

Out[36]:

production_companies	original_title	revenue_adj	budget_adj	vote_average
1492pictures	Harry Potter and the Philosopher's Stone	1.202518e+09	1.539360e+08	7.2
coolmusic	Harry Potter and the Order of the Phoenix	9.866889e+08	1.577503e+08	7.2
lucasfilm	Star Wars	2.789712e+09	3.957559e+07	7.9
patalexivproductionslimited	Harry Potter and the Goblet of Fire	1.000353e+09	1.674845e+08	7.3
robertwiseproductions	The Sound of Music	1.129535e+09	5.674862e+07	7.2

By the way, what's Cool Music? I never heard of such production company. So I printed the original production companies information for this movie. It turns out that this data point is somehow inserted with weird value and the weird value shows up before the actual production companies Warner Bros. and Heyday Films. So it was incorrectly picked up as the production company based on my logic of processing this column during data wrangling steps. I believe this is just an very rare misinformation and this data glitch won't affect the ranking of top 5 companies with highest total revenue. For now I will leave it as is and move on to next topic.

```
In [37]: movies[movies['original_title'] == 'Harry Potter and the Order of the Phoenix']['produc
```

```
Out[37]: array(['Cool Music|Warner Bros.|Heyday Films|Harry Potter Publishing Rights'],
               dtype=object)
```

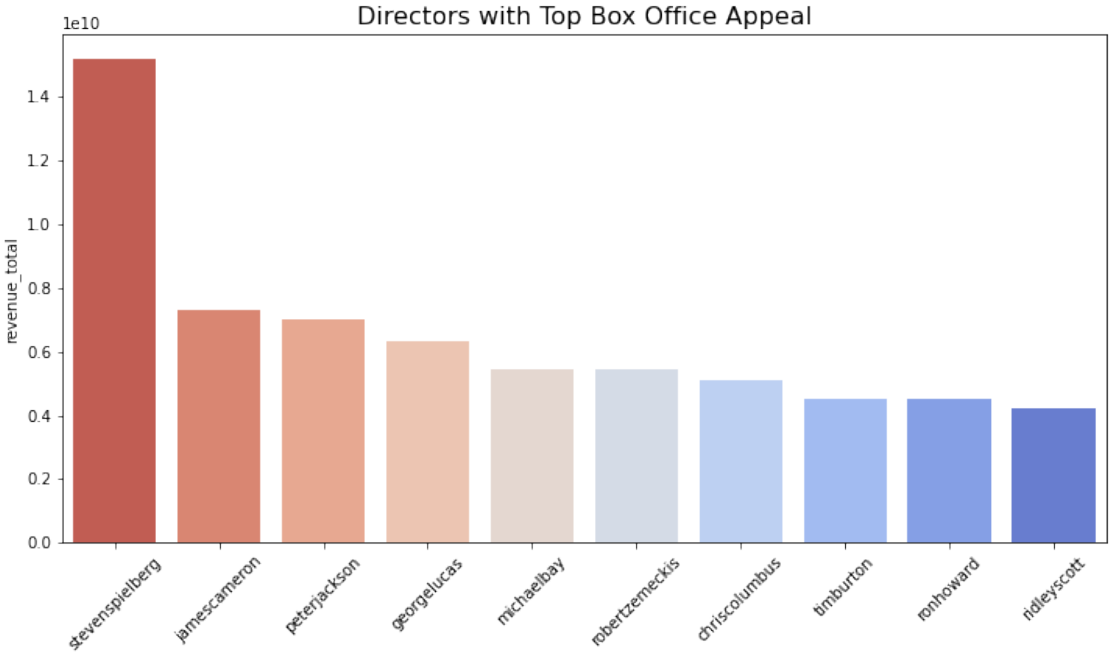
Q2.5 Who are the 10 most revenue-generating directors over 1960-2015?

```
In [38]: df_dir = movie_df[['director','revenue_adj']].groupby('director')['revenue_adj'].agg([(
```

```
In [39]: plt.figure(figsize=(12,6))
         plt.title('Directors with Top Box Office Appeal', fontsize=16)

         sns.barplot(x=df_dir.sort_values(by='revenue_total',ascending=False)[:10].index,y='reve
         plt.xlabel('')
         plt.xticks(rotation=45)
```

```
Out[39]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
          <a list of 10 Text major ticklabel objects>)
```



It shows that the top 10 directors with the greatest revenue generation ability are Steven Spielberg, James Cameron, Peter Jackson, George Lucas, Michael Bay, Robert Zemeckis, Chris Columbus, Tim Burton, Ron Howard, and Ridley Scott. Steven Spielberg's movies made a lot more money than the other directors'.

Q2.6 Who are the 10 most revenue-generating actors/actresses over 1960-2015? Previously I extracted 3 main cast for each movie, so here I will apply the similar processing on the cast column that I did to genres column.

```
In [40]: # create a sub-dataframe containing only columns release_year, genres and revenue_adj.
         df_cast = movie_df[['release_year','cast','revenue_adj']]

         # create separate column for each genre value
         df_cast['cast1'] = df_cast['cast'].apply(lambda x: x[0])
         df_cast['cast2'] = df_cast['cast'].apply(lambda x: x[1] if len(x)>1 else None)
         df_cast['cast3'] = df_cast['cast'].apply(lambda x: x[2] if len(x)>2 else None)
         # remove original column genres, and melt 4-column dataframe into 2-column dataframe
         df_cast = df_cast.drop('cast',axis=1).melt(['release_year','revenue_adj']).drop('variab
         df_cast.head(3)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
  import sys
```

Out[40]:

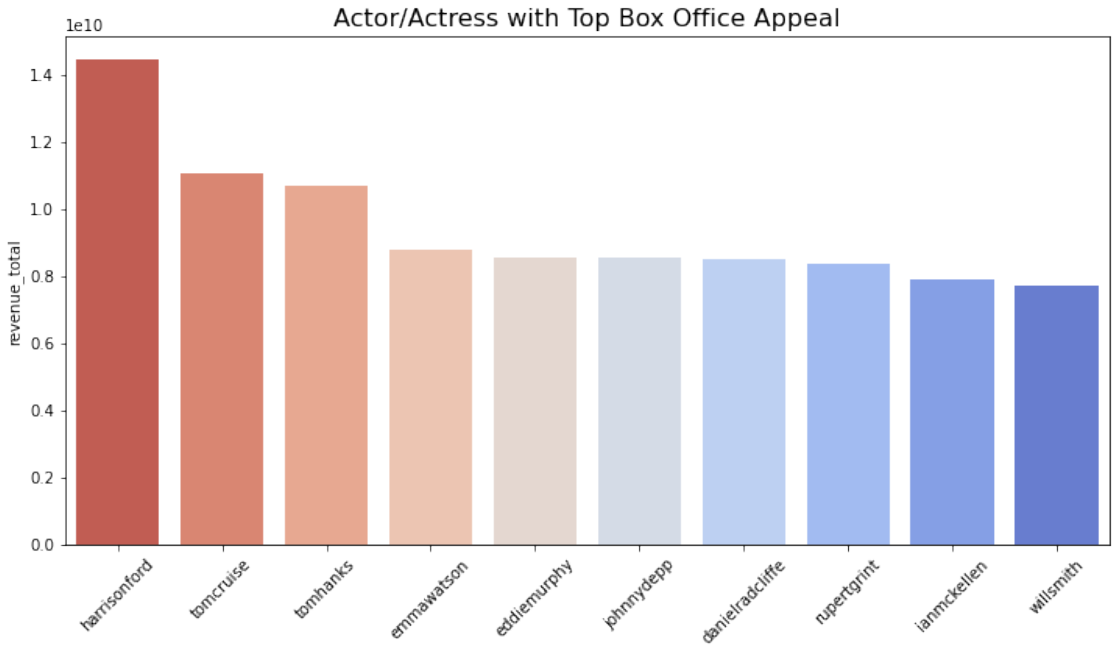
	release_year	revenue_adj	cast
0	2015	1.392446e+09	chrispratt
1	2015	3.481613e+08	tomhardy
2	2015	2.716190e+08	shailenewoodley

```
In [41]: df_cast = df_cast.drop('release_year',axis=1).groupby('cast')['revenue_adj'].agg([('rev

         plt.figure(figsize=(12,6))
         plt.title('Actor/Actress with Top Box Office Appeal', fontsize=16)

         sns.barplot(x=df_cast.sort_values(by='revenue_total',ascending=False)[:10].index,y='rev
         plt.xlabel('')
         plt.xticks(rotation=45)

Out[41]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
          <a list of 10 Text major ticklabel objects>)
```



From the above chart we can tell that the actors/actresses with the greatest box office appeal are Harrison Ford, Tom Cruise, Tom Hanks, Emma Watson, Eddie Murphy, Johnny Depp, Daniel Radcliffe, Rupert Grint, Ian McKellen, and Will Smith.

Conclusion

1.1.7 Summary of Findings

For genres popularity:

- The five most popular genres are action, comedy, drama, horror, and thriller over all years.
- When it comes down to each decade, drama has always been the winner among all genres, except in 1980s when comedy took over and became the most popular genre.

For revenue relevancy:

- The correlation matrix shows that buedget, vote count and popularity have relatively strong correlation with revenue, while vote average has much weaker correlation with revenue.
- The quarterly descriptive statistics of movie revenue data shows that Q3 and Q4 have the most number of movies released, while Q1 and Q4 have the highest maximum revenue, and Q2 and Q4 have relatively high average revenue and median revenue. If a movie is release in Q4, statistically it's more likely to have high revenue.
- The top 5 genres with the greatest total revenue contributions are action, adventure, drama, comedy and thriller. And the top 5 genres with the highest average revenue are adventure, fantasy, animation, action and family.

- The top 5 production companies with top revenue generation ability are Paramount Pictures, Universal Pictures, Walt Disney Pictures, Columbia Pictures, and Twentieth Century Fox Film Corporation. These 5 companies' revenue generation ability are far more superior than the rest of the companies.
- The top 5 production companies who have produced movies with high average revenue are Robert Wise Productions, Patalex IV Productions Limited, 1492 Pictures, Lucas Film and Barwood Films. These basically are the producers of Harry Potter series and Star War series.
- The top 10 directors with the greatest revenue generation ability are Steven Spielberg, James Cameron, Peter Jackson, George Lucas, Michael Bay, Robert Zemeckis, Chris Columbus, Tim Burton, Ron Howard, and Ridley Scott. Steven Spielberg's movies made a lot more money than the other directors'.
- From the above chart we can tell that the actors/actresses with the greatest box office appeal are Harrison Ford, Tom Cruise, Tom Hanks, Emma Watson, Eddie Murphy, Johnny Depp, Daniel Radcliffe, Rupert Grint, Ian McKellen, and Will Smith.

1.1.8 Potential Improvement

One of the most obvious improvement I could have done in this project is the way I handle column genres. I could have extracted two values (or more) from the original pipe-segregated string in column genres, so the statistics might better represent the true genres counts.

Additionally I could take logarithm values of revenue and budget when studying the correlation between various numeric features and revenue, so the charts will be easier to interpret.

One more thing I could have done is add a new feature to calculate the profitability of each movie (i.e., (revenue - budget) / budget). I could research questions like what genres tend to generate higher return on investment, how profitability has changed year from year. et cetera.

1.2 Submitting Project

```
In [42]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[42]: 255
```