Project 2: TMDb Movie Data Analysis

Table of Contents

Introduction

Questions to Explore:

> - **Q0: Fun Facts**
> - **Q1: Genre Trends from 1960 to 2015**
> - **Q2: Properties Associated With Higher Profits**

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
```

Data Wrangling

General Properties

In [2]:
```
# Load data and print out a few lines.
df = pd.read_csv("tmdb_movies.csv")
df.head()
```

Out[2]:

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

5 rows × 21 columns

In [3]:
```
# Assess number of rows and columns of dataset
df.shape
```

Out[3]:
```
(10866, 21)
```

In [4]:
```
# Assess dataset, including datatypes, and check for missing data.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

- **Many columns have missing row data.**
- **First select columns for analysis and drop non-useful columns and then deal with missing data.**

In [5]:
```
df.describe()
```

Out[5]:

	id	popularity	budget	revenue	runtime	vote_count	vc
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

- **There are too much information and will drop some colummns when perform data analysis.**
- **Could add profit = (revenue - budget) and profit_ratio = (profit/budget) columns to investigate profitability**
- **Columns 'cast', 'director', 'keywords', 'genres', 'production_companies' contain multiple values separated by pipe (|) that need to be seperated out.**

Data Cleaning

First I'll remove extraneous columns that aren't relevant to my analysis and duplicates rows. Then I'll add and/or replace information to ensure my dataset is clean for analysis.

Need to drop columns:

- **'imdb_id' : Already have 'id'**
- **'homepage' : Not relevant**
- **'tagline' : Not relevant**
- **'overview' : Not relevant**
- **'release_year' : Already have 'release_date'**

In [6]:
```
# Drop Columns
df.drop(['imdb_id','homepage','tagline','overview','release_date'], axis = 1,
inplace = True)
df.head(1)
```

Out[6]:

	id	popularity	budget	revenue	original_title	cast	director	
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v

- **Add 'profit' and 'profit_adj' columns**

In [7]:
```
df['profit'] = df['revenue'] - df['budget']
df['profit_adj'] = df['revenue_adj'] - df['budget_adj']
```

- **Add 'profit_ratio' and 'profit_ratio_adj' columns**
- **Add 0.000001 to revenue to prevent NaN in ratios**

In [8]:
```
df['profit_ratio'] = df['profit']/(0.000001+df['budget'])
df['profit_ratio_adj'] = df['profit_adj']/(0.000001+df['budget_adj'])
```

Check/Drop for duplicates

In [9]:
```
df[df.duplicated()]
```

Out[9]:

	id	popularity	budget	revenue	original_title	cast	director	keyw
2090	42194	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	Dwight H. Little	m arts|dystopia|b on \ game|m

In [10]:
```
df.query('original_title == "TEKKEN"')
```

Out[10]:

	id	popularity	budget	revenue	original_title	cast	director	keyw
2089	42194	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	Dwight H. Little	m arts|dystopia|b on \ game|m
2090	42194	0.59643	30000000	967000	TEKKEN	Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...	Dwight H. Little	m arts|dystopia|b on \ game|m

In [11]:
```
df.drop_duplicates(inplace = True)
print(sum(df.duplicated()))
print(df.shape)
```

```
0
(10865, 20)
```

Check for missing values

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 20 columns):
id                      10865 non-null int64
popularity              10865 non-null float64
budget                  10865 non-null int64
revenue                 10865 non-null int64
original_title          10865 non-null object
cast                    10789 non-null object
director                10821 non-null object
keywords                9372 non-null object
runtime                 10865 non-null int64
genres                  10842 non-null object
production_companies    9835 non-null object
vote_count              10865 non-null int64
vote_average            10865 non-null float64
release_year            10865 non-null int64
budget_adj              10865 non-null float64
revenue_adj             10865 non-null float64
profit                  10865 non-null int64
profit_adj              10865 non-null float64
profit_ratio            10865 non-null float64
profit_ratio_adj        10865 non-null float64
dtypes: float64(7), int64(7), object(6)
memory usage: 1.7+ MB
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
id                         0
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
director                  44
keywords                1493
runtime                    0
genres                    23
production_companies    1030
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
profit                     0
profit_adj                 0
profit_ratio               0
profit_ratio_adj           0
dtype: int64
```

Drop the rows with missing values

- **First drop rows with missing 'cast', 'director' and 'genres' informations**

In [14]:
```
df.dropna(subset = ['cast', 'director', 'genres'], inplace = True)
df.isnull().sum()
```

Out[14]:
```
id                        0
popularity                0
budget                    0
revenue                   0
original_title            0
cast                      0
director                  0
keywords               1425
runtime                   0
genres                    0
production_companies    959
vote_count                0
vote_average              0
release_year              0
budget_adj                0
revenue_adj               0
profit                    0
profit_adj                0
profit_ratio              0
profit_ratio_adj          0
dtype: int64
```

NOTE:
df is for answering general questions.
Also need seperate df_keywords, df_production, df_cast and df_director.

In [15]:

```
df_keywords = df.copy()
df_keywords.dropna(subset = ['keywords'], inplace = True)
df_keywords.isnull().sum()
```

Out[15]:

```
id                      0
popularity              0
budget                  0
revenue                 0
original_title          0
cast                    0
director                0
keywords                0
runtime                 0
genres                  0
production_companies  640
vote_count              0
vote_average            0
release_year            0
budget_adj              0
revenue_adj             0
profit                  0
profit_adj              0
profit_ratio            0
profit_ratio_adj        0
dtype: int64
```

In [16]:

```
df_production = df.copy()
df_production.dropna(subset = ['production_companies'], inplace = True)
df_production.isnull().sum()
```

Out[16]:

```
id                         0
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                       0
director                   0
keywords                1106
runtime                    0
genres                     0
production_companies       0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
profit                     0
profit_adj                 0
profit_ratio               0
profit_ratio_adj           0
dtype: int64
```

Now have df is for answering general questions not related to keywords and production companies. Also need to make seperate df_keywords and df_production for keywords and production company related questions.

Split up 'genres' columns

In [17]:
```
df_split_genre = df.copy()
split_genre = df_split_genre['genres'].str.split('|').apply(pd.Series,1).stack
().reset_index(level=1, drop=True)
split_genre.name = 'genre_split'
df_split_genre = df_split_genre.drop(['genres'], axis=1).join(split_genre)
df_split_genre.head(3)
```

Out[17]:

	id	popularity	budget	revenue	original_title	cast	director	
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v

In [18]:
```
df_split_genre.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26753 entries, 0 to 10865
Data columns (total 20 columns):
id                      26753 non-null int64
popularity              26753 non-null float64
budget                  26753 non-null int64
revenue                 26753 non-null int64
original_title          26753 non-null object
cast                    26753 non-null object
director                26753 non-null object
keywords                23523 non-null object
runtime                 26753 non-null int64
production_companies    24650 non-null object
vote_count              26753 non-null int64
vote_average            26753 non-null float64
release_year            26753 non-null int64
budget_adj              26753 non-null float64
revenue_adj             26753 non-null float64
profit                  26753 non-null int64
profit_adj              26753 non-null float64
profit_ratio            26753 non-null float64
profit_ratio_adj        26753 non-null float64
genre_split             26753 non-null object
dtypes: float64(7), int64(7), object(6)
memory usage: 4.3+ MB
```

- **'keywords' and 'production_companies' have null value but not affect the analysis that not related to them.**

In [19]:
```
df_split_genre.shape
```

Out[19]:
```
(26753, 20)
```

In [20]:
```
df_split_genre.describe()
```

Out[20]:

	id	popularity	budget	revenue	runtime	vote_count	vo
count	26753.000000	26753.000000	2.675300e+04	2.675300e+04	26753.000000	26753.000000	26
mean	58236.098045	0.710244	1.763665e+07	4.779885e+07	103.048892	251.691436	
std	86350.207583	1.118093	3.470727e+07	1.326446e+08	29.560855	640.123565	
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10184.000000	0.226575	0.000000e+00	0.000000e+00	90.000000	18.000000	
50%	18065.000000	0.414311	2.500000e+04	0.000000e+00	100.000000	44.000000	
75%	57718.000000	0.779596	2.000000e+07	3.132790e+07	112.000000	176.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

Split up 'keywords' columns

In [21]:
```
df_split_keywords = df_keywords.copy()
split_keywords = df_split_keywords['keywords'].str.split('|').apply(pd.Series, 1).stack().reset_index(level=1, drop=True)
split_keywords.name = 'keywords_split'
df_split_keywords = df_split_keywords.drop(['keywords'], axis=1).join(split_keywords)
df_split_keywords.head(3)
```

Out[21]:

	id	popularity	budget	revenue	original_title	cast	director	runtime
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	124

In [22]:
```
df_split_keywords.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37235 entries, 0 to 10865
Data columns (total 20 columns):
id                      37235 non-null int64
popularity              37235 non-null float64
budget                  37235 non-null int64
revenue                 37235 non-null int64
original_title          37235 non-null object
cast                    37235 non-null object
director                37235 non-null object
runtime                 37235 non-null int64
genres                  37235 non-null object
production_companies    35234 non-null object
vote_count              37235 non-null int64
vote_average            37235 non-null float64
release_year            37235 non-null int64
budget_adj              37235 non-null float64
revenue_adj             37235 non-null float64
profit                  37235 non-null int64
profit_adj              37235 non-null float64
profit_ratio            37235 non-null float64
profit_ratio_adj        37235 non-null float64
keywords_split          37235 non-null object
dtypes: float64(7), int64(7), object(6)
memory usage: 6.0+ MB
```

- **'production_companies' has null value but not affect the analysis that not related to it.**

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

Out[23]:
```
(37235, 20)
```

In [24]:
```
df_split_keywords.describe()
```

Out[24]:

	id	popularity	budget	revenue	runtime	vote_count	vc
count	37235.000000	37235.000000	3.723500e+04	3.723500e+04	37235.000000	37235.000000	37
mean	52963.982463	0.783637	1.879837e+07	5.387756e+07	104.437787	289.243776	
std	83038.898888	1.159073	3.494342e+07	1.364298e+08	27.254729	675.433227	
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	9562.000000	0.255858	0.000000e+00	0.000000e+00	91.000000	21.000000	
50%	14459.000000	0.467556	2.200000e+06	7.707060e+05	101.000000	60.000000	
75%	49010.000000	0.890557	2.360000e+07	4.400869e+07	114.000000	227.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

Split up 'production_companies' columns

In [25]:
```
df_split_production = df_production.copy()
split_production = df_split_production['production_companies'].str.split('|').
apply(pd.Series,1).stack().reset_index(level=1, drop=True)
split_production.name = 'production_split'
df_split_production = df_split_production.drop(['production_companies'], axis=
1).join(split_production)
df_split_production.head(3)
```

Out[25]:

	id	popularity	budget	revenue	original_title	cast	director	
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	monster|d rex|v

In [26]:
```
df_split_production.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23143 entries, 0 to 10865
Data columns (total 20 columns):
id                  23143 non-null int64
popularity          23143 non-null float64
budget              23143 non-null int64
revenue             23143 non-null int64
original_title      23143 non-null object
cast                23143 non-null object
director            23143 non-null object
keywords            20804 non-null object
runtime             23143 non-null int64
genres              23143 non-null object
vote_count          23143 non-null int64
vote_average        23143 non-null float64
release_year        23143 non-null int64
budget_adj          23143 non-null float64
revenue_adj         23143 non-null float64
profit              23143 non-null int64
profit_adj          23143 non-null float64
profit_ratio        23143 non-null float64
profit_ratio_adj    23143 non-null float64
production_split    23143 non-null object
dtypes: float64(7), int64(7), object(6)
memory usage: 3.7+ MB
```

- **'keywords' has null value but not affect the analysis that not related to it.**

In [27]:
```
df_split_production.shape
```

Out[27]:
```
(23143, 20)
```

In [28]:
```
df_split_production.describe()
```

Out[28]:

	id	popularity	budget	revenue	runtime	vote_count	vo
count	23143.000000	23143.000000	2.314300e+04	2.314300e+04	23143.000000	23143.000000	23
mean	63782.611546	0.816030	2.084325e+07	5.533152e+07	104.915482	308.415158	
std	90008.832896	1.209748	3.624897e+07	1.365742e+08	25.835334	681.717341	
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	9905.000000	0.267255	0.000000e+00	0.000000e+00	92.000000	22.000000	
50%	18228.000000	0.484139	4.000000e+06	7.918300e+05	101.000000	65.000000	
75%	74751.500000	0.937272	2.600000e+07	4.819070e+07	114.000000	257.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	877.000000	9767.000000	

Split up 'cast' columns

In [29]:
```
df_split_cast = df.copy()
split_cast = df_split_cast['cast'].str.split('|').apply(pd.Series,1).stack().r
eset_index(level=1, drop=True)
split_cast.name = 'cast_split'
df_split_cast = df_split_cast.drop(['cast'], axis=1).join(split_cast)
df_split_cast.head(3)
```

Out[29]:

	id	popularity	budget	revenue	original_title	director	keyworc
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	monster|dna|tyrannosauru rex|velociraptor|islan
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	monster|dna|tyrannosauru rex|velociraptor|islan
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	monster|dna|tyrannosauru rex|velociraptor|islan

In [30]:
```
df_split_cast.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 52334 entries, 0 to 10865
Data columns (total 20 columns):
id                        52334 non-null int64
popularity                52334 non-null float64
budget                    52334 non-null int64
revenue                   52334 non-null int64
original_title            52334 non-null object
director                  52334 non-null object
keywords                  45600 non-null object
runtime                   52334 non-null int64
genres                    52334 non-null object
production_companies      48020 non-null object
vote_count                52334 non-null int64
vote_average              52334 non-null float64
release_year              52334 non-null int64
budget_adj                52334 non-null float64
revenue_adj               52334 non-null float64
profit                    52334 non-null int64
profit_adj                52334 non-null float64
profit_ratio              52334 non-null float64
profit_ratio_adj          52334 non-null float64
cast_split                52334 non-null object
dtypes: float64(7), int64(7), object(6)
memory usage: 8.4+ MB
```

- **'keywords' and 'production_companies' have null value but not affect the analysis that not related to them.**

In [31]:
```
df_split_cast.shape
```

Out[31]: (52334, 20)

In [32]:
```
df_split_cast.describe()
```

Out[32]:

	id	popularity	budget	revenue	runtime	vote_count	vo
count	52334.000000	52334.000000	5.233400e+04	5.233400e+04	52334.000000	52334.000000	52
mean	63949.016949	0.663357	1.516689e+07	4.132668e+07	102.954618	224.558490	
std	90563.719104	1.013682	3.136560e+07	1.189496e+08	28.888852	585.220224	
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10448.000000	0.216906	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	19621.000000	0.394209	0.000000e+00	0.000000e+00	99.000000	40.000000	
75%	71866.000000	0.733947	1.700000e+07	2.681011e+07	112.000000	154.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

Split up 'director' columns

In [33]:
```
df_split_director = df.copy()
split_director = df_split_director['director'].str.split('|').apply(pd.Series, 1).stack().reset_index(level=1, drop=True)
split_director.name = 'director_split'
df_split_director = df_split_director.drop(['director'], axis=1).join(split_director)
df_split_director.head(3)
```

Out[33]:

	id	popularity	budget	revenue	original_title	cast	
0	135397	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	monster|(rex|'
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	apocalyptic
2	262500	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	novel|revolution|dyst

In [34]:
```
df_split_director.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11774 entries, 0 to 10865
Data columns (total 20 columns):
id                      11774 non-null int64
popularity              11774 non-null float64
budget                  11774 non-null int64
revenue                 11774 non-null int64
original_title          11774 non-null object
cast                    11774 non-null object
keywords                10209 non-null object
runtime                 11774 non-null int64
genres                  11774 non-null object
production_companies    10708 non-null object
vote_count              11774 non-null int64
vote_average            11774 non-null float64
release_year            11774 non-null int64
budget_adj              11774 non-null float64
revenue_adj             11774 non-null float64
profit                  11774 non-null int64
profit_adj              11774 non-null float64
profit_ratio            11774 non-null float64
profit_ratio_adj        11774 non-null float64
director_split          11774 non-null object
dtypes: float64(7), int64(7), object(6)
memory usage: 1.9+ MB
```

- **'keywords' and 'production_companies' have null value but not affect the analysis that not related to them.**

In [35]:
```
df_split_director.shape
```

Out[35]: (11774, 20)

In [36]:
```
df_split_director.describe()
```

Out[36]:

	id	popularity	budget	revenue	runtime	vote_count	vc
count	11774.000000	11774.000000	1.177400e+04	1.177400e+04	11774.000000	11774.000000	11
mean	67035.732631	0.655070	1.478524e+07	4.080166e+07	103.047138	221.918379	
std	92428.824638	1.005885	3.134590e+07	1.195286e+08	41.075401	580.822606	
min	5.000000	0.000188	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10705.250000	0.209859	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	21330.000000	0.386192	0.000000e+00	0.000000e+00	98.000000	39.000000	
75%	78382.500000	0.722796	1.500000e+07	2.417932e+07	111.000000	149.000000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

- **'keywords' and 'production_companies' have null value but not affect the analysis that not related to them.**

Now we have 5 clean dataframes:

- **df**
- **df_keywords**
- **df_production**
- **df_cast**
- **df_director**

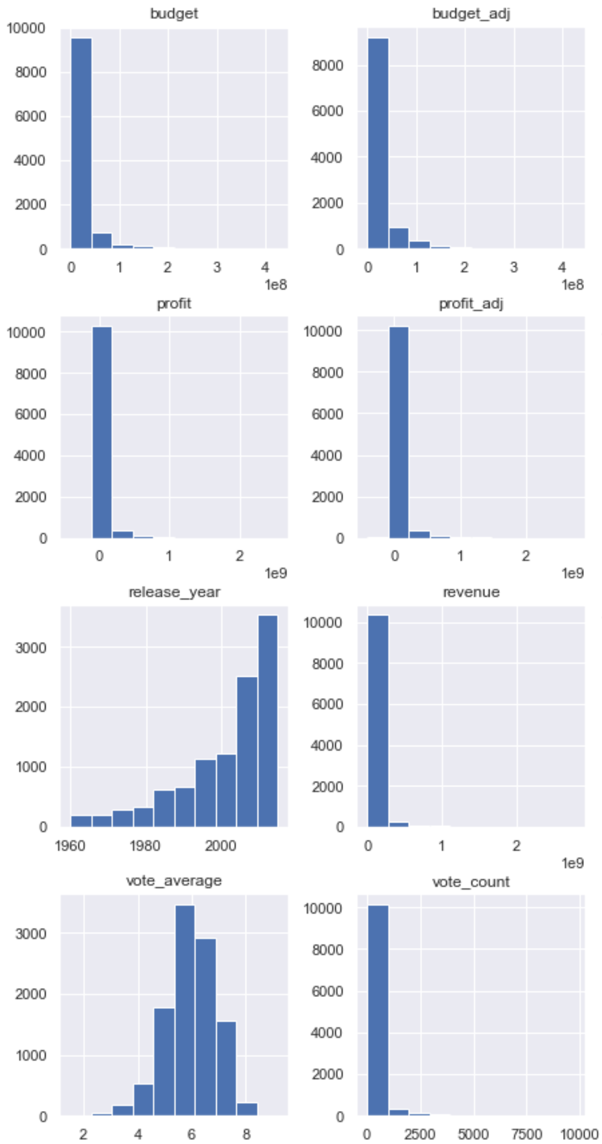
Exploratory Data Analysis

In [37]:
```
df.corr()
```

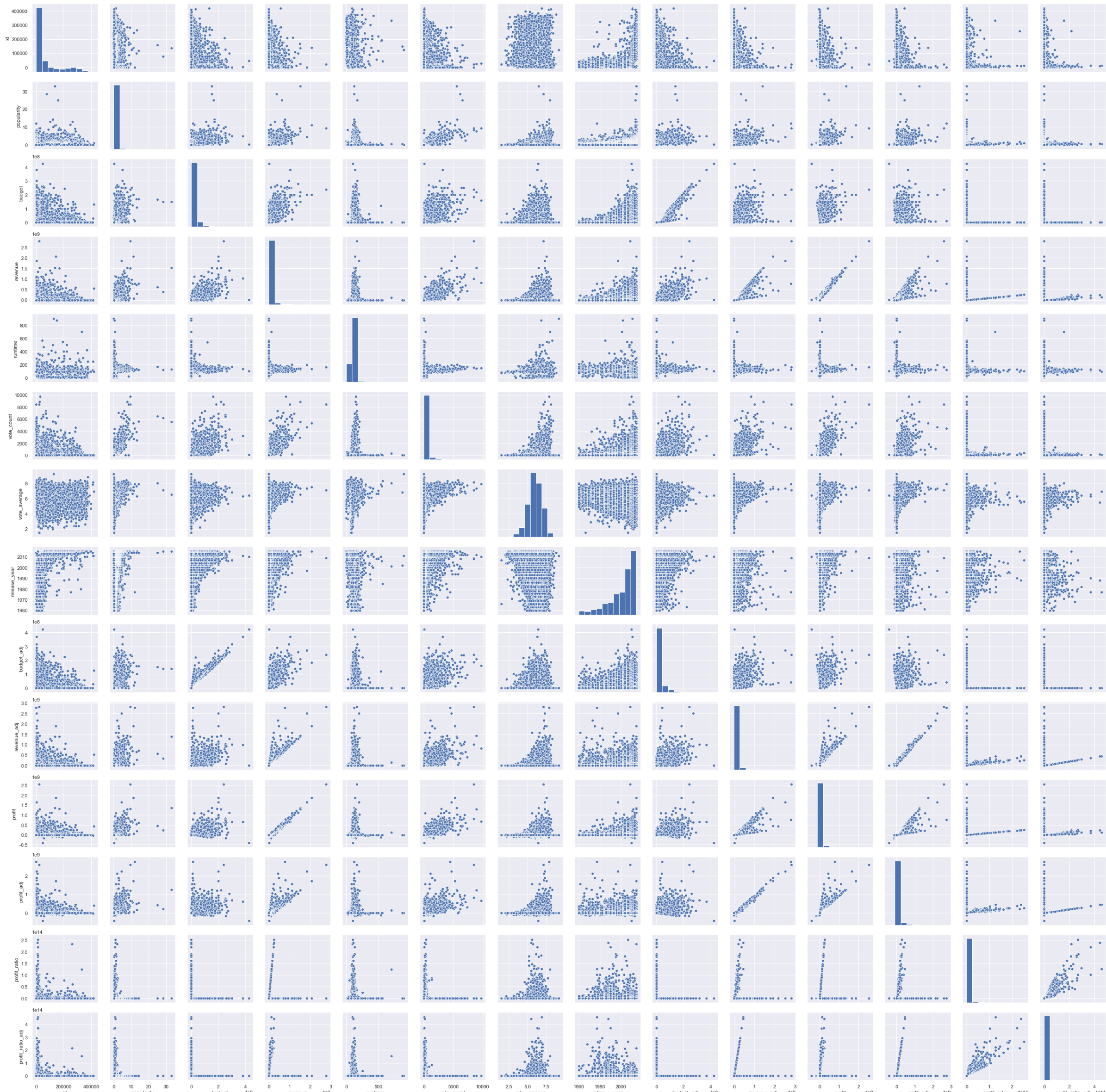
Out[37]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average
id	1.000000	-0.009464	-0.138935	-0.097424	-0.083996	-0.032767	-0.071896
popularity	-0.009464	1.000000	0.544240	0.662843	0.138278	0.800619	0.217906
budget	-0.138935	0.544240	1.000000	0.734487	0.192168	0.632074	0.087318
revenue	-0.097424	0.662843	0.734487	1.000000	0.164276	0.790889	0.178477
runtime	-0.083996	0.138278	0.192168	0.164276	1.000000	0.164966	0.177276
vote_count	-0.032767	0.800619	0.632074	0.790889	0.164966	1.000000	0.260554
vote_average	-0.071896	0.217906	0.087318	0.178477	0.177276	0.260554	1.000000
release_year	0.510385	0.093044	0.119004	0.059072	-0.119286	0.110317	-0.127746
budget_adj	-0.186980	0.512098	0.968881	0.705949	0.222645	0.586298	0.099925
revenue_adj	-0.137099	0.608384	0.621809	0.918990	0.177397	0.707517	0.199418
profit	-0.073557	0.628699	0.569730	0.976173	0.137497	0.755681	0.188220
profit_adj	-0.107138	0.562359	0.452854	0.867967	0.143911	0.656523	0.202922
profit_ratio	-0.058220	-0.004752	-0.075339	0.036690	0.038097	-0.023790	0.022227
profit_ratio_adj	-0.070609	-0.013727	-0.073365	0.030528	0.037407	-0.029740	0.029567

In [38]:
```
df.hist(figsize=(15,15));
```



In [39]:
```
# Scatter plot of df using seaborn
sns_scatterplt = sns.pairplot(df, height = 2.5)
```



From scatter plots:

- **Revenue and Profits are positively correlated**

Research Question 0: Fun Facts

> **This section I will investigate various fun facts**

- **1. Words with highest frequency appeared in the movie titles:**

In [40]:
```
# Wordcloud for title visualization:

from wordcloud import WordCloud, STOPWORDS
text = (str(df['original_title']))
plt.subplots(figsize=(15,15))
wordcloud = WordCloud(stopwords=STOPWORDS, background_color='white', width=150
0, height=1200).generate(text)
plt.imshow(wordcloud)
plt.title('Title')
plt.axis('off');
```

- **Title analysis just for fun. ^^**

- **2. Top 10 Most Popular Movies:**

In [41]:
```
# Top 10 Most Popular Movies:

df[['popularity', 'original_title']].sort_values(by='popularity', ascending=False).head(10)
```

Out[41]:

	popularity	original_title
0	32.985763	Jurassic World
1	28.419936	Mad Max: Fury Road
629	24.949134	Interstellar
630	14.311205	Guardians of the Galaxy
2	13.112507	Insurgent
631	12.971027	Captain America: The Winter Soldier
1329	12.037933	Star Wars
632	11.422751	John Wick
3	11.173104	Star Wars: The Force Awakens
633	10.739009	The Hunger Games: Mockingjay - Part 1

- **3. Top 10 Highest Rating Movies:**

In [42]:
```
# Top 10 Highest Rating Movies:
df[['vote_average', 'original_title']].sort_values(by='vote_average', ascending=False).head(10)
```

Out[42]:

	vote_average	original_title
3894	9.2	The Story of Film: An Odyssey
1200	8.8	Black Mirror: White Christmas
6911	8.7	Pink Floyd: Pulse
3690	8.5	The Art of Flight
8221	8.5	A Personal Journey with Martin Scorsese Throug...
8839	8.5	Dave Chappelle: Killin' Them Softly
8411	8.5	Queen - Rock Montreal
4178	8.4	The Shawshank Redemption
2334	8.4	Rush: Beyond the Lighted Stage
609	8.4	The Jinx: The Life and Deaths of Robert Durst

- **4.Top 10 Most Profitable Movies (sorted by adjusted profit and profit):**

In [43]:
```
df[['profit_adj', 'original_title']].sort_values(by='profit_adj', ascending=False).head(10)
```

Out[43]:

	profit_adj	original_title
1329	2.750137e+09	Star Wars
1386	2.586237e+09	Avatar
5231	2.234714e+09	Titanic
10594	2.128036e+09	The Exorcist
9806	1.878643e+09	Jaws
8889	1.767968e+09	E.T. the Extra-Terrestrial
3	1.718723e+09	Star Wars: The Force Awakens
8094	1.551568e+09	The Net
10110	1.545635e+09	One Hundred and One Dalmatians
7309	1.376998e+09	The Empire Strikes Back

In [44]:
```
df[['profit', 'original_title']].sort_values(by='profit', ascending=False).head(10)
```

Out[44]:

	profit	original_title
1386	2544505847	Avatar
3	1868178225	Star Wars: The Force Awakens
5231	1645034188	Titanic
0	1363528810	Jurassic World
4	1316249360	Furious 7
4361	1299557910	The Avengers
3374	1202817822	Harry Potter and the Deathly Hallows: Part 2
14	1125035767	Avengers: Age of Ultron
5422	1124219009	Frozen
8094	1084279658	The Net

> **Profitability should be evaluated by profit rather than ratios. Some movies have "0" budget so we add 0.000001 in previous steps, and this will make profit_ratio super high.**

- **5. Top 10 Actors Starred In Most Movies:**

In [45]:
```
df_split_cast['cast_split'].value_counts().head(10)
```

Out[45]:
```
Robert De Niro       72
Samuel L. Jackson    71
Bruce Willis         62
Nicolas Cage         61
Michael Caine        53
Robin Williams       51
John Cusack          50
Morgan Freeman       49
John Goodman         49
Liam Neeson          48
Name: cast_split, dtype: int64
```

- **6. Top 10 Keywords In Most Movies:**

In [46]:
```
df_split_keywords['keywords_split'].value_counts().head(10)
```

Out[46]:
```
woman director      408
independent film    393
based on novel      278
sex                 272
sport               215
murder              204
musical             169
biography           168
new york            162
suspense            159
Name: keywords_split, dtype: int64
```

- **7. Top 10 Production Companies:**

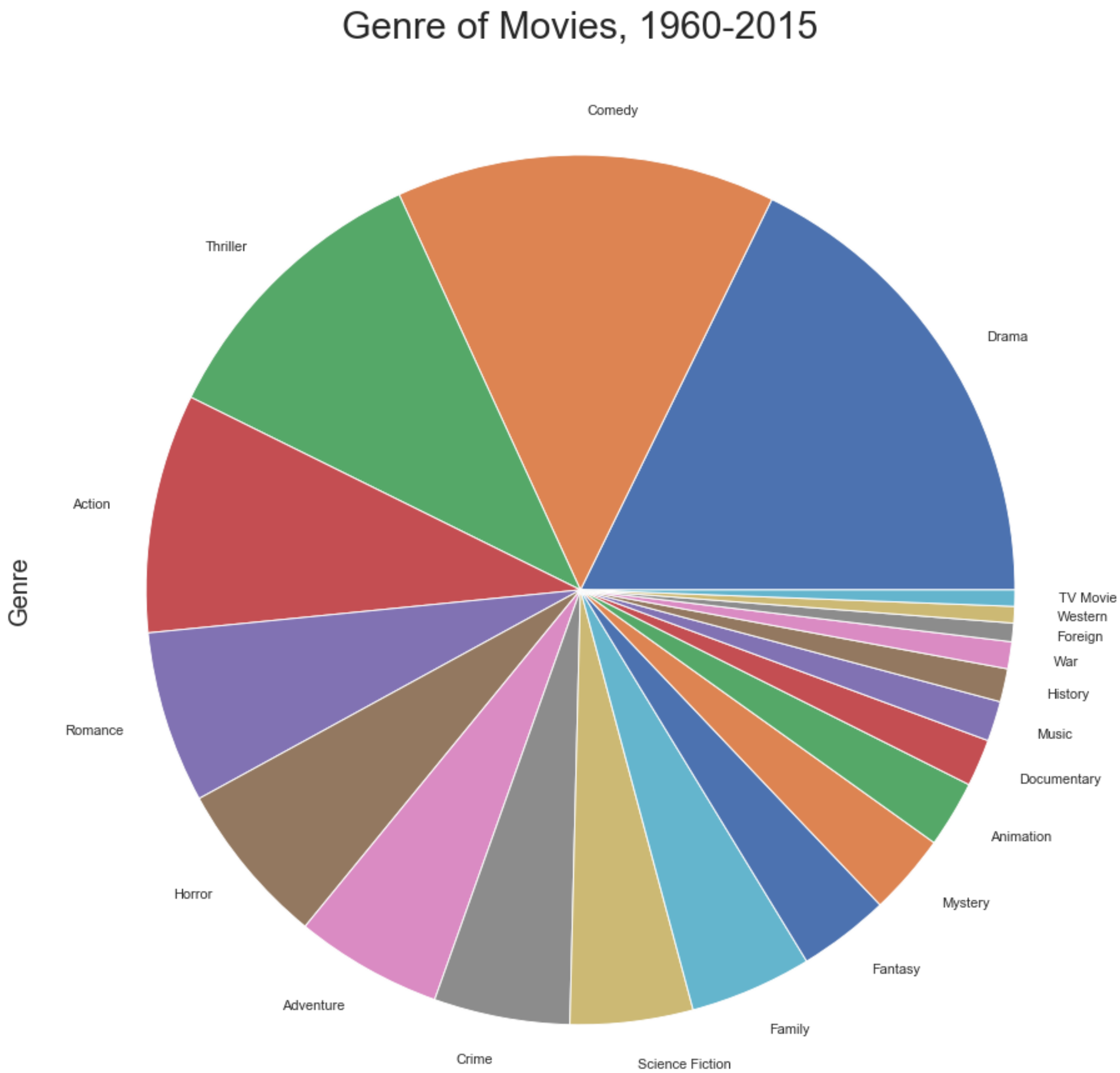
In [47]:
```
df_split_production['production_split'].value_counts().head(10)
```

Out[47]:
```
Universal Pictures                        522
Warner Bros.                              509
Paramount Pictures                        431
Twentieth Century Fox Film Corporation    282
Columbia Pictures                         272
New Line Cinema                           219
Metro-Goldwyn-Mayer (MGM)                 218
Walt Disney Pictures                      213
Touchstone Pictures                       178
Columbia Pictures Corporation             160
Name: production_split, dtype: int64
```

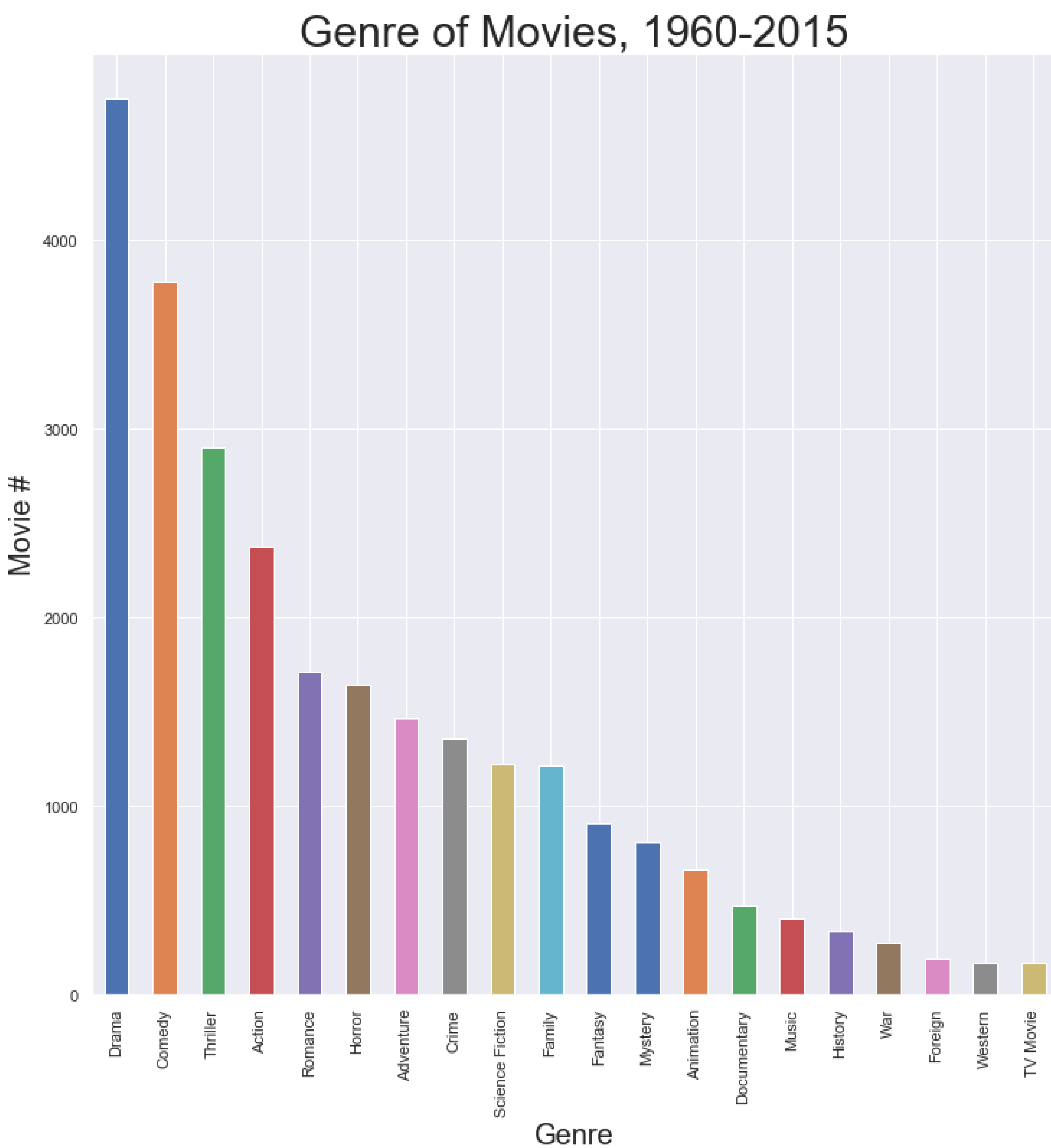
Research Question 1: Genre Trends from 1960 to 2015

This section I will investigate different genres

In [48]:
```
# Plot pie chart to visualize genre distribution
df_split_genre['genre_split'].value_counts().plot(kind = 'pie',figsize = (16,1
6));
plt.title('Genre of Movies, 1960-2015', size=30)
plt.ylabel('Genre', size=20);
```

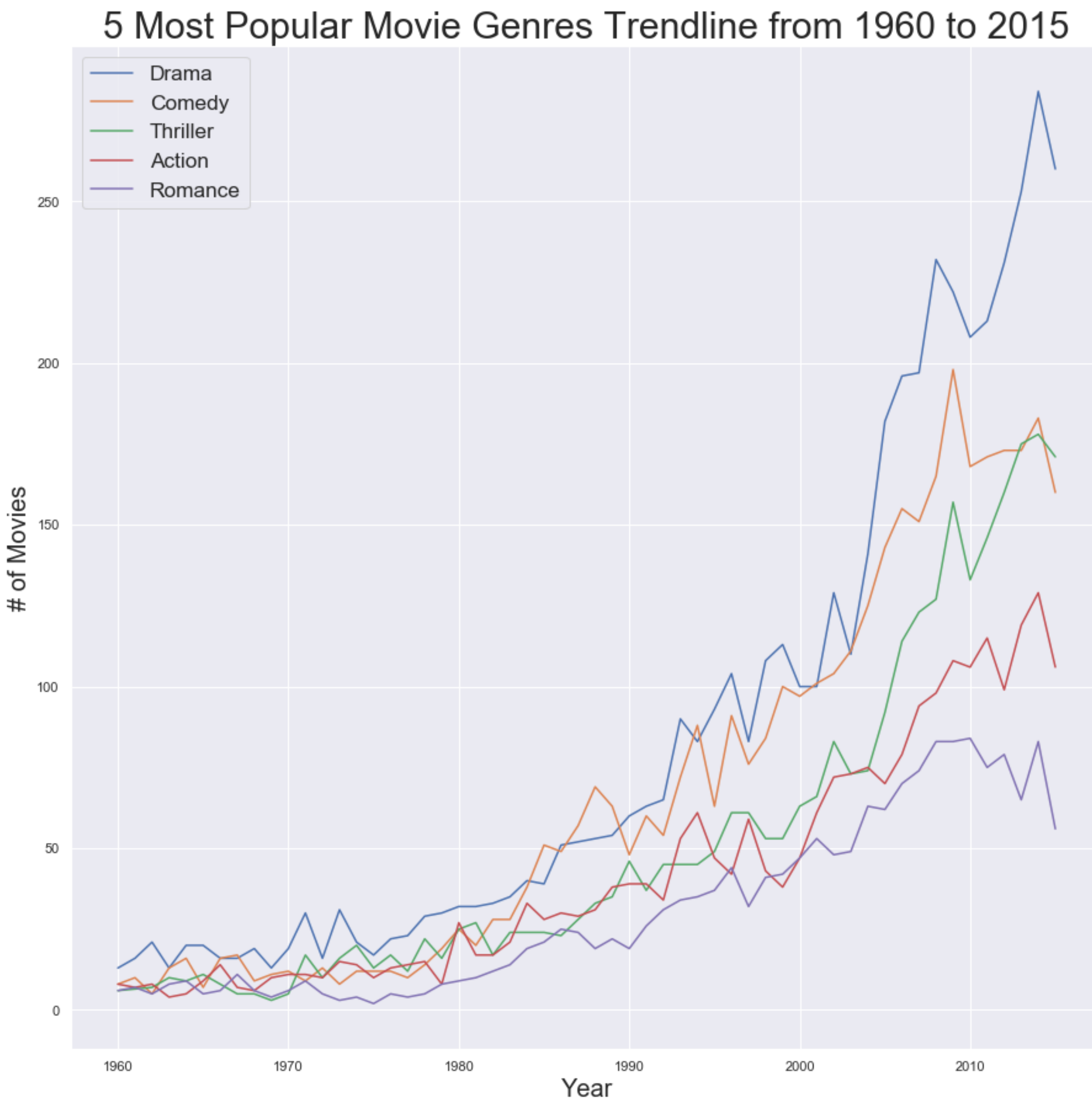


In [49]:
```
#Plot bar chart to visualize genre distribution
df_split_genre['genre_split'].value_counts().plot(kind='bar',figsize = (12,12
));
plt.title('Genre of Movies, 1960-2015', size=30)
plt.xlabel('Genre', size=20)
plt.ylabel('Movie #', size=20);
```

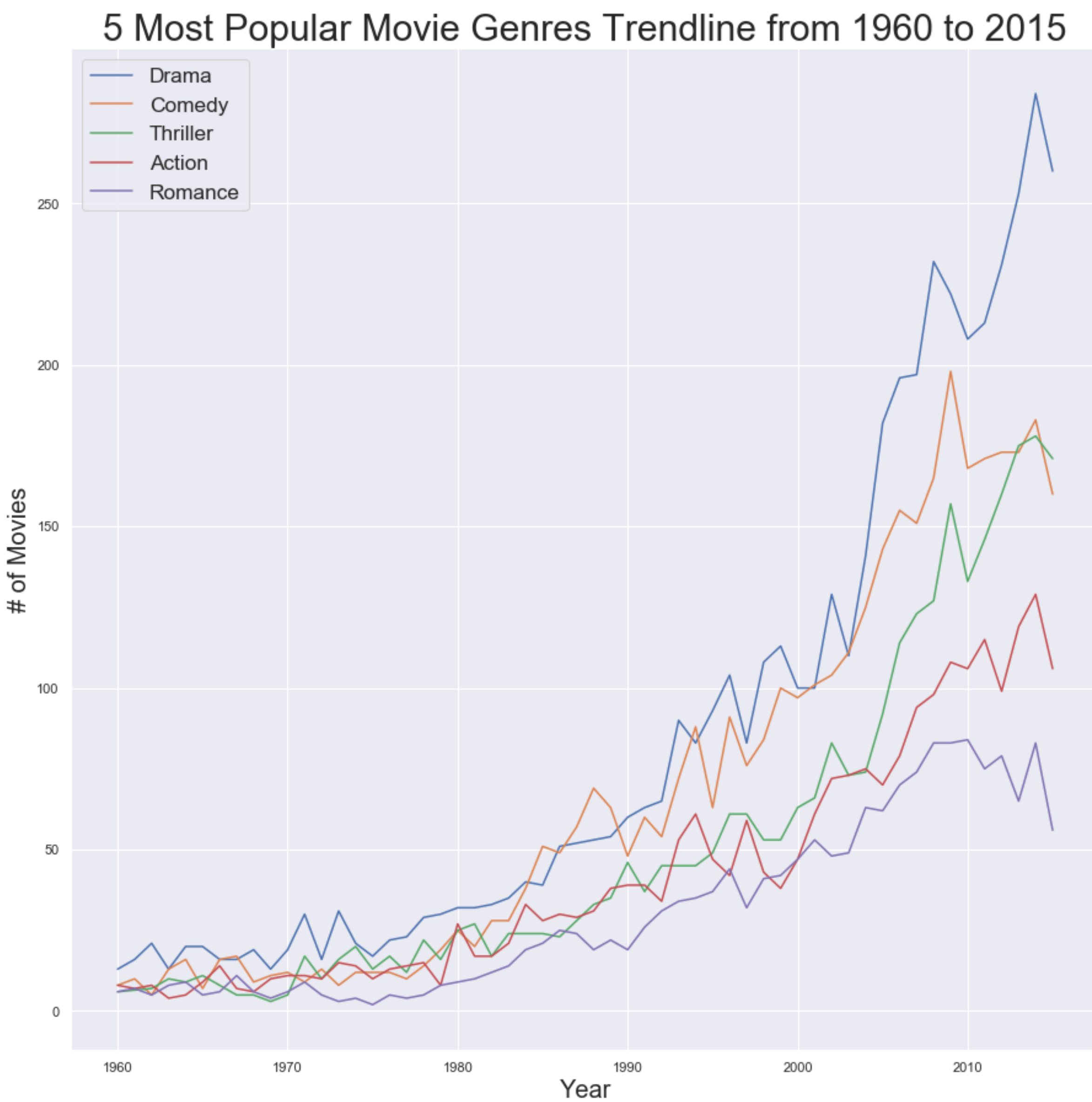


In [50]:

```
# Select data from df for 5 most popular genres: Drama, Comedy, Thriller, Acti
on, Romance.
# Then plot the total counts of different genres for each year from 1960 to 20
15
drama = df_split_genre.genre_split == 'Drama'
df_drama = df_split_genre[drama]
df_drama.groupby('release_year')['genre_split'].count().plot(figsize=(15,15),l
abel='Drama')


comedy = df_split_genre.genre_split == 'Comedy'
df_comedy = df_split_genre[comedy]
df_comedy.groupby('release_year')['genre_split'].count().plot(label='Comedy')

thriller = df_split_genre.genre_split == 'Thriller'
df_thriller = df_split_genre[thriller]
df_thriller.groupby('release_year')['genre_split'].count().plot(label='Thrille
r')


action = df_split_genre.genre_split == 'Action'
df_action = df_split_genre[action]
df_action.groupby('release_year')['genre_split'].count().plot(label='Action')


romance = df_split_genre.genre_split == 'Romance'
df_romance = df_split_genre[romance]
df_romance.groupby('release_year')['genre_split'].count().plot(label='Romance'
)

plt.title('5 Most Popular Movie Genres Trendline from 1960 to 2015',size=30)
plt.xlabel('Year',size=20)
plt.ylabel('# of Movies',size=20)
plt.legend(fontsize = 'xx-large');
```


5 Most Popular Movie Genres Trendline from 1960 to 2015
Drama
Comedy
Thriller
Action
Romance
# of Movies
0
50
100
150
200
250
Year
1960
1970
1980
1990
2000
2010

In [51]:
```
# Select data from df for 5 most popular genres: Drama, Comedy, Thriller, Acti
on, Romance.
# Then plot the total counts of different genres for each year from 1960 to 20
15

top_genres = ['Drama', 'Comedy', 'Thriller', 'Action', 'Romance']

######### Code below is hard to read but more concise, I don't know which one
 is better according to rubric ########
for i in range(len(top_genres)):
    df_split_genre[df_split_genre.genre_split == top_genres[i]].groupby('relea
se_year')['genre_split'].count().plot(figsize=(15,15),label=top_genres[i])

plt.title('5 Most Popular Movie Genres Trendline from 1960 to 2015',size=30)
plt.xlabel('Year',size=20)
plt.ylabel('# of Movies',size=20)
plt.legend(fontsize = 'xx-large');
```



From the line chart above:

- **All types of movies are increasing from 1960 to 2015**
- **The largest growth rate occurred during 2000-2010**
- **Drama is the most popular genre through the years except being exceeded by Comedy during the late 80's**
- **Romance is the least popular genre among top 5 genres. The growth rate of romance movies is the slowest.**

Research Question 2: Properties Associated With Higher Profits

> **This section I will investigate which movie factors are related to higher profits**

- **1. Profit vs. Genre:**

In [52]:
```
# Find all genres:

df_split_genre.genre_split.unique()
```

Out[52]:
```
array(['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy',
       'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',
       'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror',
       'Documentary', 'TV Movie', 'Foreign'], dtype=object)
```

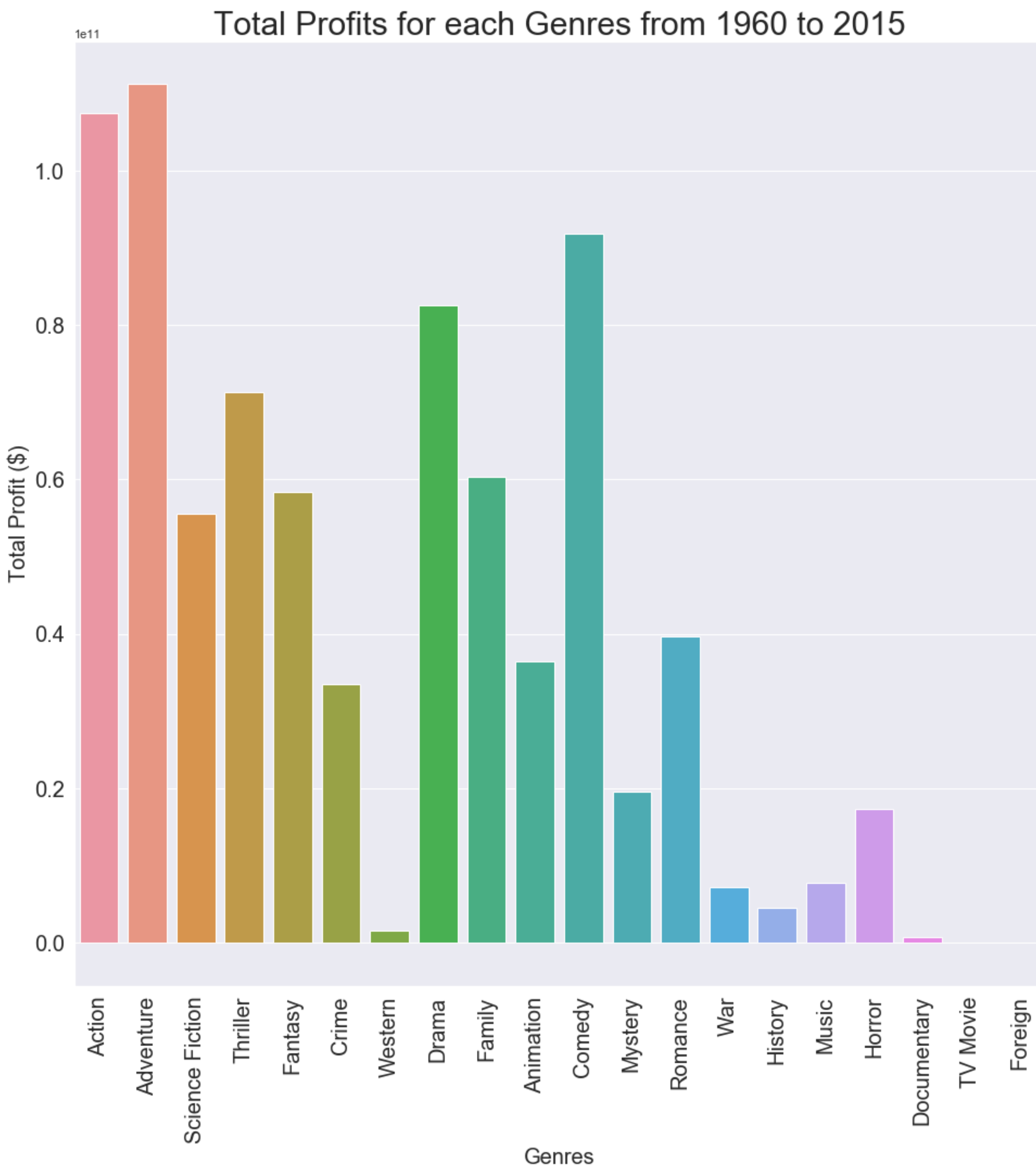
In [53]:
```
# Get total number of genres for later avg_profit calculation:

num_g = df_split_genre.genre_split.value_counts()
genres_total_num = list(num_g)
genres_total_num
```

Out[53]:
```
[4746,
 3775,
 2902,
 2376,
 1708,
 1636,
 1465,
 1353,
 1221,
 1214,
 908,
 808,
 664,
 470,
 399,
 330,
 268,
 184,
 164,
 162]
```

In [54]:
```
# Calculate sums of profit of each genres:

genres_labels = ['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantas
y',
       'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',
       'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror',
       'Documentary', 'TV Movie', 'Foreign']

profit_by_genres = []

for i in range(len(genres_labels)):
    profit_by_genres.append(df_split_genre[df_split_genre.genre_split == genre
s_labels[i]].profit.sum())

profit_by_genres
```

Out[54]:
```
[107439517424,
 111199018978,
 55511321460,
 71284730705,
 58355181708,
 33450381145,
 1583109216,
 82594648101,
 60420445751,
 36417750351,
 91896372240,
 19519620245,
 39644299221,
 7212590243,
 4488235887,
 7813519034,
 17346109400,
 745802029,
 -2700000,
 9406683]
```

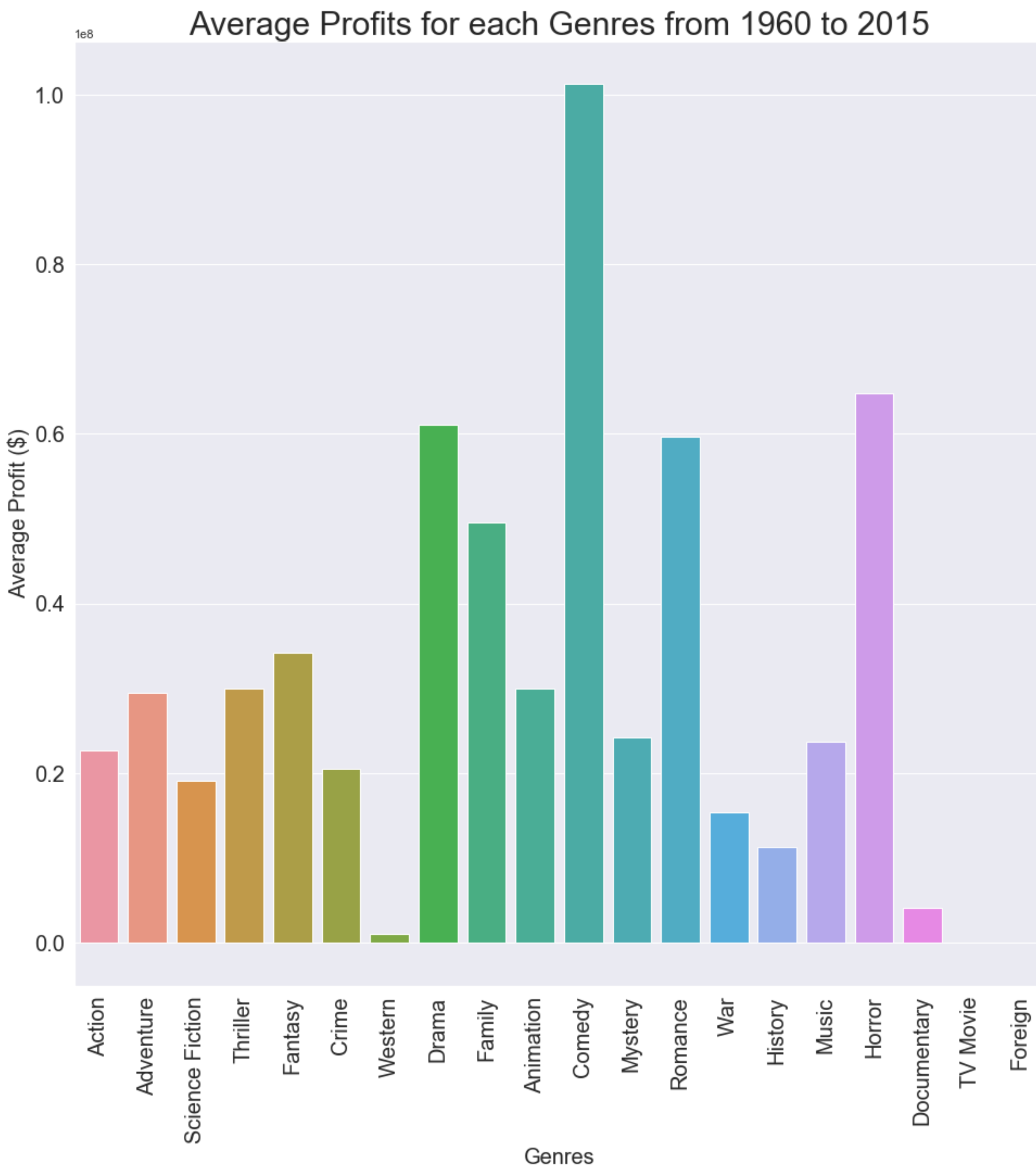
In [55]:
```
# Visualization for total profits of each genres:

plt.figure(figsize=(16, 16))
sns.barplot(x=genres_labels, y=profit_by_genres)
plt.title('Total Profits for each Genres from 1960 to 2015',size=30)
plt.xlabel('Genres',size=20)
plt.ylabel('Total Profit ($)',size=20)
plt.xticks(fontsize=20,rotation=90)
plt.yticks(fontsize=20);
```



In [56]:
```
# Calculate average profits for each genres:

avg_profit = np.divide(profit_by_genres,genres_total_num)
avg_profit = avg_profit.astype(int)
avg_profit
```

Out[56]:
```
array([ 22637909,  29456693,  19128642,  30001991,  34165797,  20446443,
         1080620,  61045564,  49484394,  29998146, 101207458,  24157945,
        59705269,  15345936,  11248711,  23677330,  64724288,   4053271,
          -16463,     58065])
```

In [57]:
```python
# Visualization for average profits of each genres:

plt.figure(figsize=(16, 16))
sns.barplot(x=genres_labels, y=avg_profit)
plt.title('Average Profits for each Genres from 1960 to 2015',size=30)
plt.xlabel('Genres',size=20)
plt.ylabel('Average Profit ($)',size=20)
plt.xticks(fontsize=20,rotation=90)
plt.yticks(fontsize=20);
```

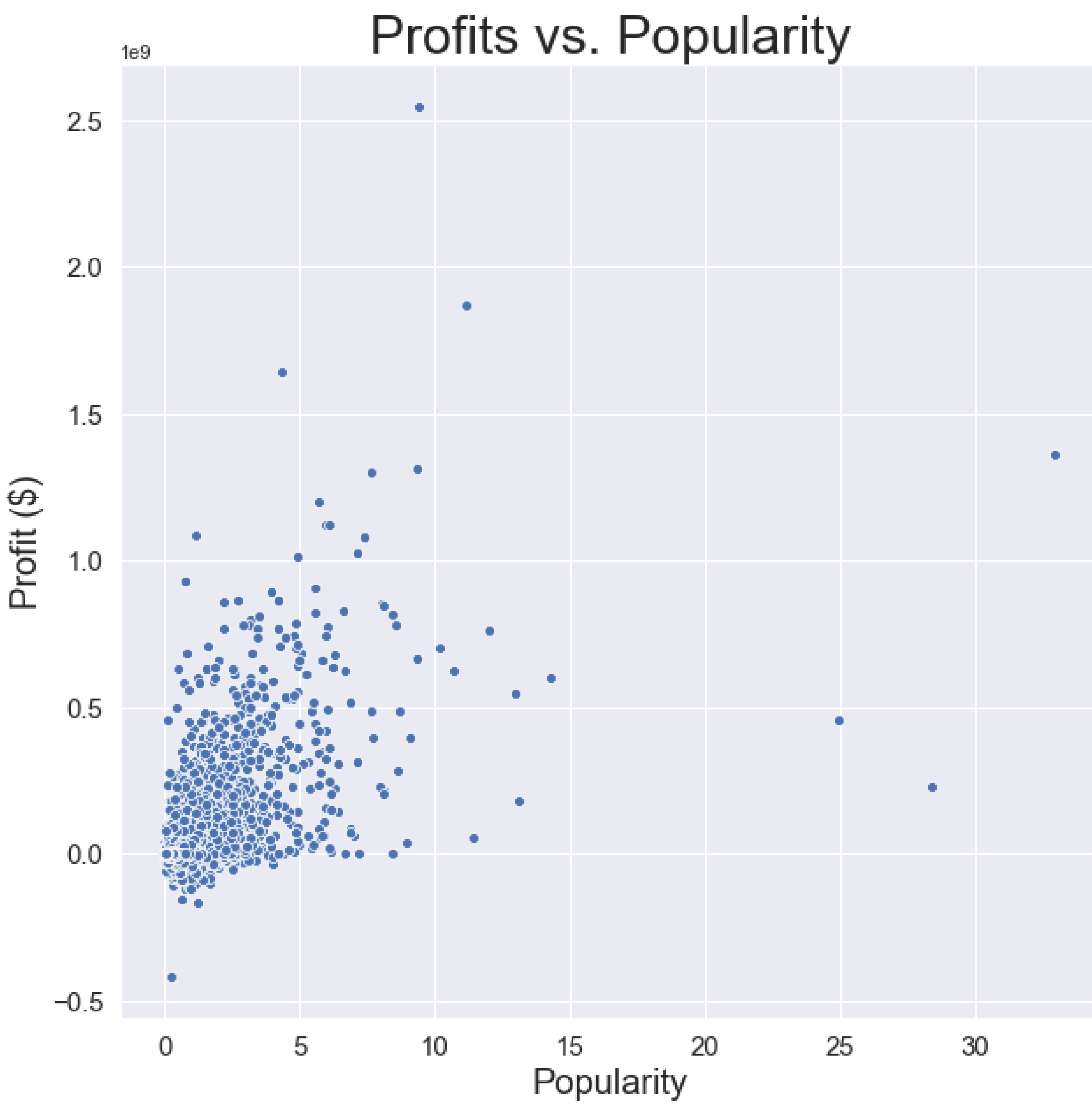


From bar chart above:

- **Top 5 most profitable genres are: 1.Comedy, 2.Horror, 3.Drama, 4.Romance, 5.Family.**

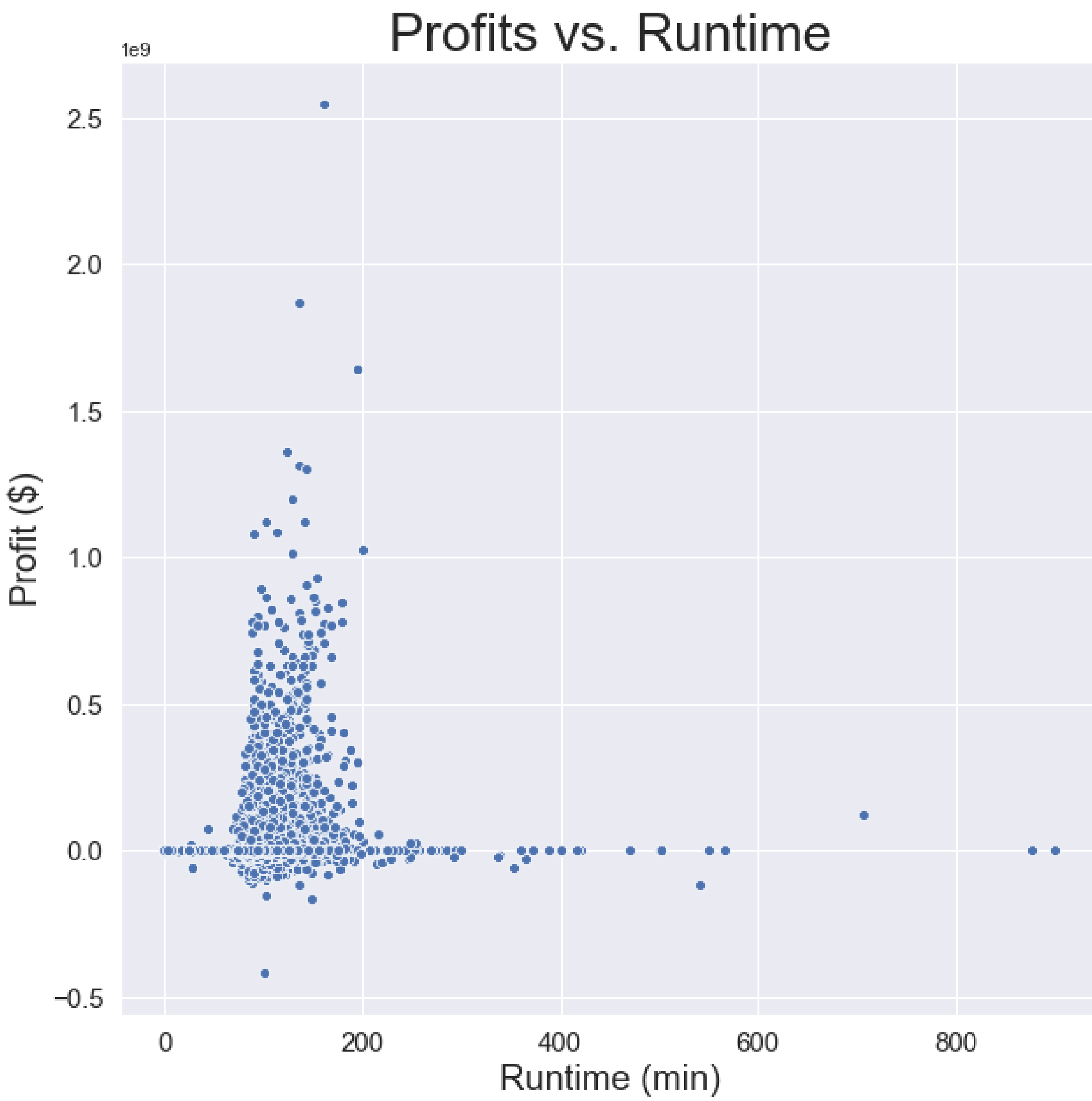
- **2. Profit vs. Popularity:**

In [58]:
```
# Plot Profit vs. Popularity:

plt.figure(figsize=(10, 10))
sns.scatterplot(x=df['popularity'],y=df['profit'])
plt.title('Profits vs. Popularity',size=30)
plt.xlabel('Popularity',size=20)
plt.ylabel('Profit ($)',size=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```



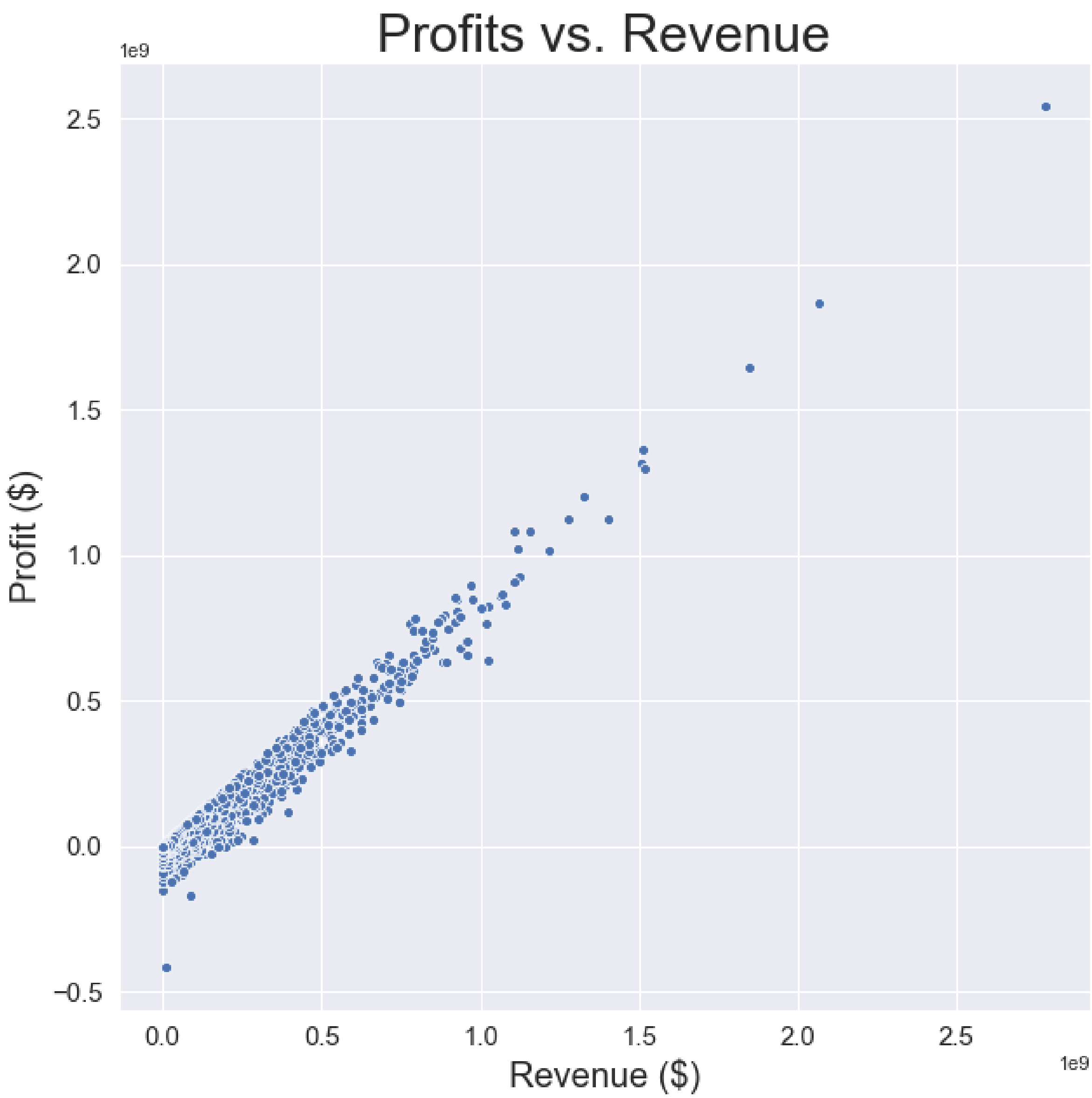
- **3. Profit vs. Runtime:**

In [59]:
```
# Plot Profit vs. Runtime:

plt.figure(figsize=(10, 10))
sns.scatterplot(x=df['runtime'],y=df['profit'])
plt.title('Profits vs. Runtime',size=30)
plt.xlabel('Runtime (min)',size=20)
plt.ylabel('Profit ($)',size=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```



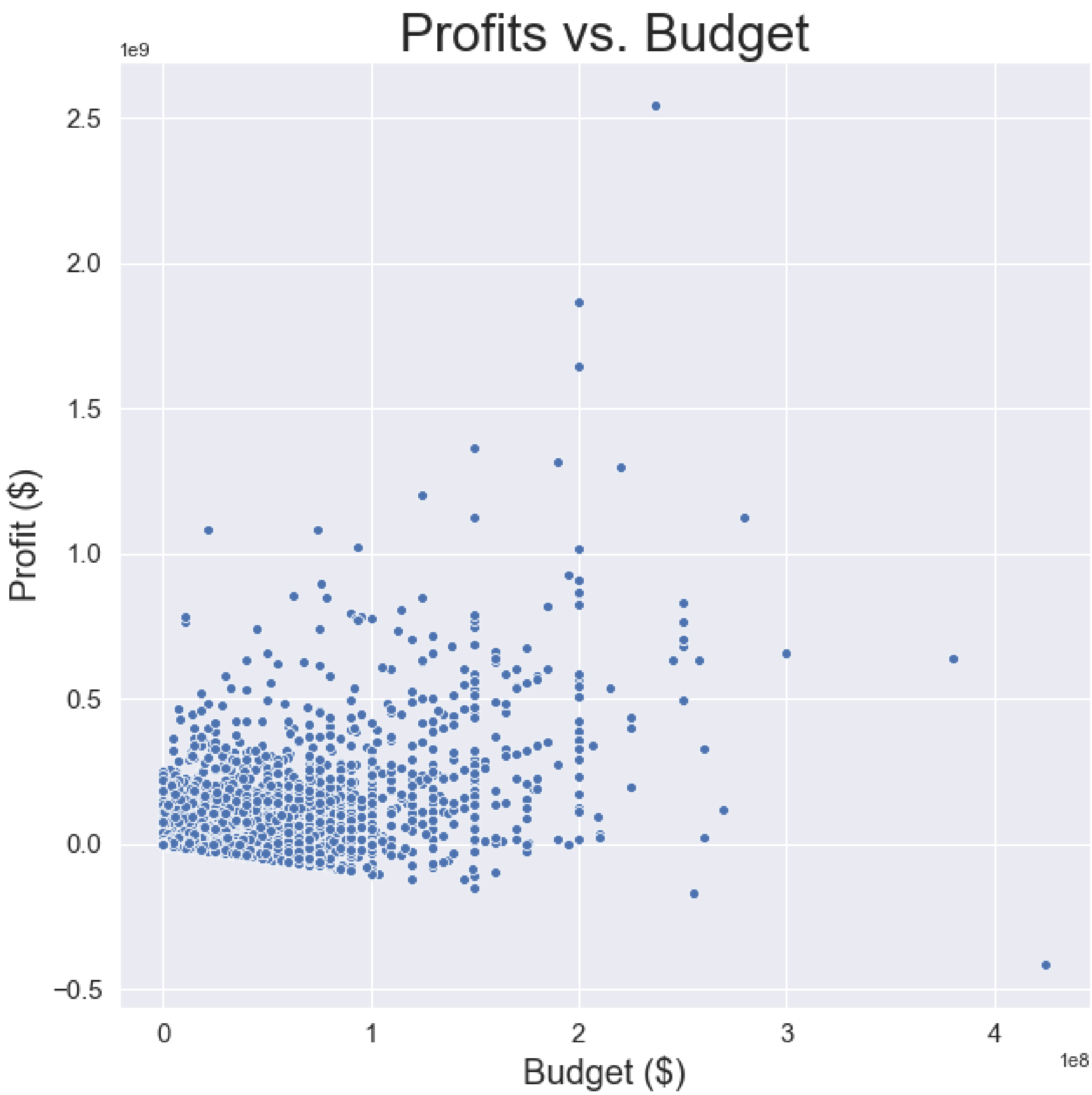
- **4. Profit vs. Revenue:**

In [60]:
```
# Plot Profit vs. Revenue:

plt.figure(figsize=(10, 10))
sns.scatterplot(x=df['revenue'],y=df['profit'])
plt.title('Profits vs. Revenue',size=30)
plt.xlabel('Revenue ($)',size=20)
plt.ylabel('Profit ($)',size=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```



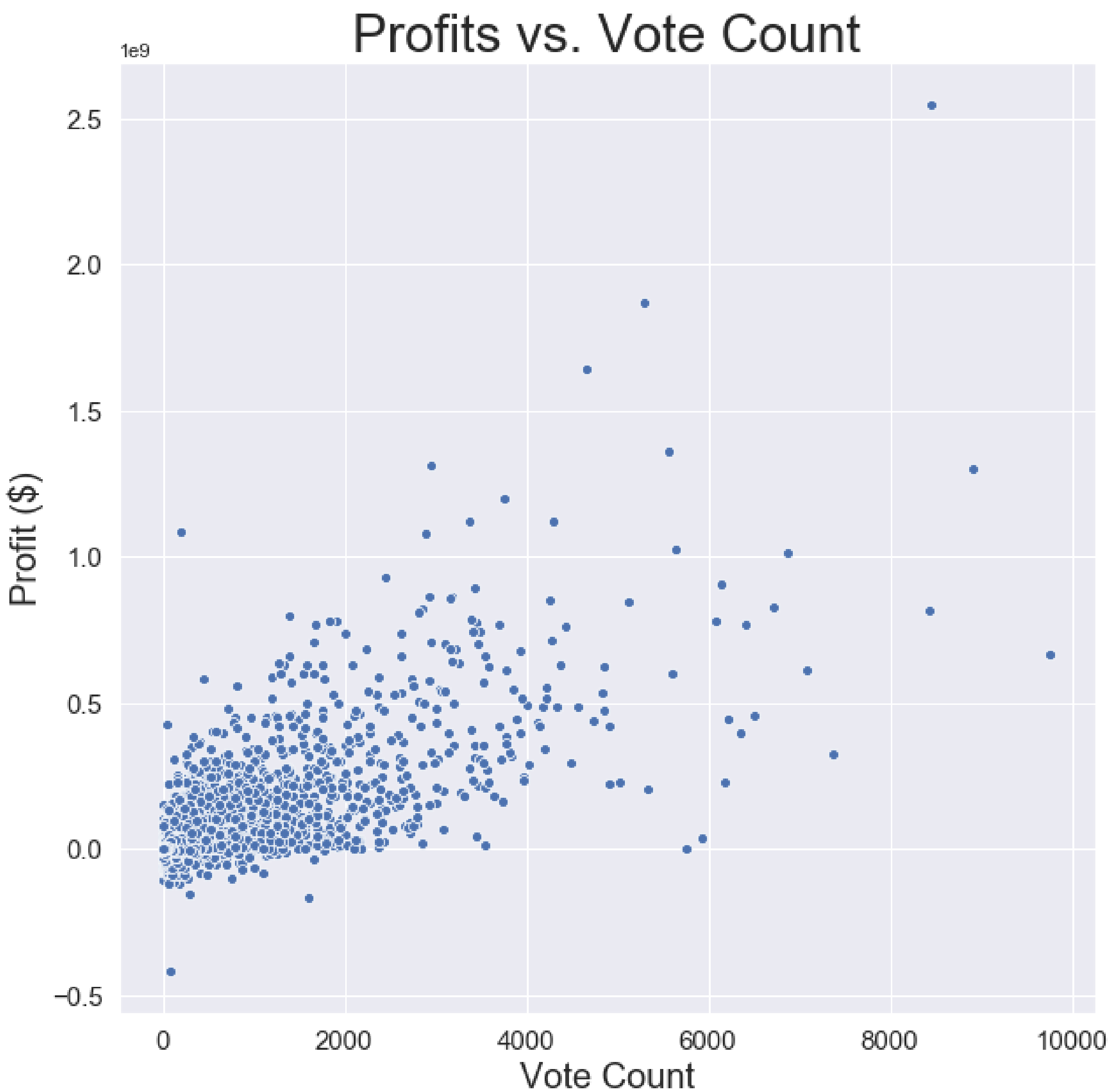
- **5. Profit vs. Budget:**

In [61]:
```
# Plot Profit vs. Budget:

plt.figure(figsize=(10, 10))
sns.scatterplot(x=df['budget'],y=df['profit'])
plt.title('Profits vs. Budget',size=30)
plt.xlabel('Budget ($)',size=20)
plt.ylabel('Profit ($)',size=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```



- **6. Profit vs. Vote_Count:**

In [62]:
```
# Plot Profit vs. Vote_Count:

plt.figure(figsize=(10, 10))
sns.scatterplot(x=df['vote_count'],y=df['profit'])
plt.title('Profits vs. Vote Count',size=30)
plt.xlabel('Vote Count',size=20)
plt.ylabel('Profit ($)',size=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```



Conclusions

Summary of Data

Question 0: Fun Facts

- **Top 5 most common words in movie titles:** ***Max, Man, Big, Carry, Hand.***
- **Top 10 Most Popular Movies:** ***Jurassic World, Mad Max: Fury Road, Interstellar, Guardians of the Galaxy, Insurgent, Captain America: The Winter Soldier, Star Wars, John Wick, Star Wars: The Force Awakens, The Hunger Games: Mockingjay - Part 1.***
- **Top 10 Highest Rating Movies:** ***The Story of Film: An Odyssey, Black Mirror: White Christmas, Pink Floyd: Pulse, The Art of Flight, A Personal Journey With Martin Scorsese Through American Movies, Dave Chappelle: Killin' Them Softly, Queen - Rock Montreal, The Shawshank Redemption, Rush: Beyond the Lighted Stage, The Jinx: The Life and Deaths of Robert Durst.***
- **Top 10 Most Profitable Movies:** ***Avatar, Star Wars: The Force Awakens, Titanic, Jurassic World, Furious 7, The Avengers, Harry Potter and the Deathly Hallows: Part 2, Avengers: Age of Ultron, Frozen, The Net.***
- **Top 10 Most Profitable Movies (sorted by adjusted profit):** ***Star Wars, Avatar, Titanic, The Exorcist, Jaws, E.T. the Extra-Terrestrial, Star Wars: The Force Awakens, The Net, One Hundred and One Dalmatians, The Empire Strikes Back.***
- **Top 10 Actors Starred Most Movies:** ***Robert De Niro, Samuel L. Jackson, Bruce Willis, Nicolas Cage, Michael Caine, Robin Williams, John Cusack, Morgan Freeman, John Goodman, Susan Sarandon.***
- **Top 10 Keywords:** ***woman director, independent film, based on novel, sex, sport, murder, musical, biography, new york, suspense.***
- **Top 10 Production Companies:** ***Universal Pictures, Warner Bros., Paramount Pictures, Twentieth Century Fox Film Corporation, Columbia Pictures, New Line Cinema, Metro-Goldwyn-Mayer (MGM), Walt Disney Pictures, Touchstone Pictures, Columbia Pictures Corporation.***

> **NOTE： I'm not sure if *Columbia Pictures* and *Columbia Pictures Corporation* are the same company. If so, I should replace one name to the other to merge those two.**

Question 1: Genre Trends from 1960 to 2015

- **Drama, Comedy, Thriller, Action and Romance are the most popular genres and make up over 50% of all movies made from 1960-2015. TV Movie, Western, and Foreign are the least popular.**
- **All types of movies are increasing from 1960 to 2015.**
- **The largest growth rate occurred during 2000-2010.**
- **Drama is the most popular genre through the years except being exceeded by Comedy during the late 80’s.**
- **Romance is the least popular genre among top 5 genres. The growth rate of romance movies is the slowest.**

Question 2: Properties Associated with Higher Profits

- **Top 5 Genres with Highest Total Profits:** ***Adventure, Action, Comedy, Drama, Thriller.***
- **Top 5 Genres with Highest Average Profits:** ***Comedy, Horror, Drama, Romance, Family.***
- **Profit vs. Popularity: Correlation is 0.628699, moderate positive correlation. Higher popularity can somewhat lead to higher profit for a movie.**

- **Profit vs. Runtime: Correlation is 0.137497, weak positive correlation. Runtime is not related to profit for a movie.**
- **Profit vs. Revenue: Correlation is 0.976173, strong positive correlation which is obvious.**
- **Profit vs. Budget: Correlation is 0.569730, moderate positive correlation. Higher budget cannot guarantee higher profit for a movie.**
- **Profit vs. Vote Count: Correlation is 0.755681, medium-strong positive correlation. Since correlation between vote count and popularity is 0.800619 and has strong positive correlation, more vote counts indicate the movie will have higher profit.**

Notes & Limitations

Raw Data:

Original data was collected only from *The Movie Database (TMDB)*, so sample bias may exist for data such as 'popularity', 'vote_count' and 'vote_average'. For more accurate results, data from other movie database (eg. IMDb) should be joined into our dataframe.

Data Cleaning:

Columns dropped and reasons:

> **'imdb_id' : Already have 'id'**
> **'homepage' : Not relevant**
> **'tagline' : Already have keywords**
> **'overview' : Already have keywords**
> **'release_year' : Already have 'release_date'**

Added profit = (revenue - budget) and profit_ratio = (profit/budget) columns to investigate profitability.

Split columns 'cast', 'director', 'keywords', 'genres', 'production_companies' that contain multiple values separated by pipe (|).

For the purpose of preserving more data, I did not remove rows with null values in 'keywords' and/or 'production_companies' columns. After Data Cleaning, I have 5 clean dataframes:

> - **df**
> - **df_keywords**
> - **df_production**
> - **df_cast**
> - **df_director**

All of them have null value in either 'keywords' or 'production_companies' or both columns, but this will not affect our analysis because each dataframe will only be using to answer specific questions that not related to the column(s) with null values.