investigate-a-dataset-TMDb

June 15, 2020

1 Project: TMDb Movie Data Analysis

1.1 Table of Contents

Introduction

Data Wrangling

Exploratory Data Analysis

Conclusions

Introduction

In this project we are going to analyze The Movie Database (TMDb) Dataset.This data set contains information about 10,000 movies collected from The Movie Database , including user ratings and revenue. We will try to answer the following questions:

Which genres are most popular over decades?

What properties are associated with highly profitable movies?

[254]:
```
#import packages
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.animation as animation
import seaborn as sns
from IPython.display import HTML
import matplotlib.patches as mpatches
```

Data Wrangling

In this section,we will load in the data, check for cleanliness, and then trim and clean our dataset for analysis.

[255]:
```
#Loading the dataset
df = pd.read_csv("tmdb-movies.csv")
```

```
df.head()
```

[255]:

```
      id    imdb_id  popularity     budget     revenue  \
0  135397  tt0369610   32.985763  150000000  1513528810
1   76341  tt1392190   28.419936  150000000   378436354
2  262500  tt2908446   13.112507  110000000   295238201
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360

                original_title  \
0                Jurassic World
1           Mad Max: Fury Road
2                     Insurgent
3  Star Wars: The Force Awakens
4                     Furious 7

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi…
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic…
2  Shailene Woodley|Theo James|Kate Winslet|Ansel…
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D…
4  Vin Diesel|Paul Walker|Jason Statham|Michelle …

                                      homepage          director  \
0                   http://www.jurassicworld.com/   Colin Trevorrow
1                     http://www.madmaxmovie.com/     George Miller
2   http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod…       J.J. Abrams
4                          http://www.furious7.com/         James Wan

                       tagline  …  \
0            The park is open.  …
1           What a Lovely Day.  …
2   One Choice Can Destroy You  …
3  Every generation has a story.  …
4           Vengeance Hits Home  …

                                      overview runtime  \
0  Twenty-two years after the events of Jurassic …     124
1  An apocalyptic story set in the furthest reach…     120
2  Beatrice Prior must confront her inner demons …     119
3  Thirty years after defeating the Galactic Empi…     136
4  Deckard Shaw seeks revenge against Dominic Tor…     137

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

```
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]
```

[256]:
```
#print last 5 rows fo the dataset
df.tail()
```

[256]:
```
          id    imdb_id  popularity  budget  revenue  \
10861     21  tt0060371    0.080598       0        0
10862  20379  tt0060472    0.065543       0        0
10863  39768  tt0060161    0.065141       0        0
10864  21449  tt0061177    0.064317       0        0
10865  22293  tt0060666    0.035919   19000        0

                original_title  \
10861        The Endless Summer
10862                Grand Prix
10863       Beregis Avtomobilya
10864    What's Up, Tiger Lily?
10865  Manos: The Hands of Fate

                                                cast homepage  \
10861  Michael Hynson|Robert August|Lord 'Tally Ho' B…      NaN
10862  James Garner|Eva Marie Saint|Yves Montand|Tosh…      NaN
10863  Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z…      NaN
10864  Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh…      NaN
10865  Harold P. Warren|Tom Neyman|John Reynolds|Dian…      NaN

                director                                            tagline  \
10861        Bruce Brown                                                NaN
10862  John Frankenheimer  Cinerama sweeps YOU into a drama of speed and …
```

```
10863      Eldar Ryazanov                                                NaN
10864        Woody Allen                          WOODY ALLEN STRIKES BACK!
10865    Harold P. Warren       It's Shocking! It's Beyond Your Imagination!

       …                                              overview runtime  \
10861  …  The Endless Summer, by Bruce Brown, is one of …      95
10862  …  Grand Prix driver Pete Aron is fired by his te…     176
10863  …  An insurance agent who moonlights as a carthie…      94
10864  …  In comic Woody Allen's film debut, he took the…      80
10865  …  A family gets lost on the road and stumbles up…      74

                      genres  \
10861             Documentary
10862  Action|Adventure|Drama
10863          Mystery|Comedy
10864           Action|Comedy
10865                  Horror

                                    production_companies release_date  \
10861                                  Bruce Brown Films      6/15/66
10862  Cherokee Productions|Joel Productions|Douglas …     12/21/66
10863                                          Mosfilm       1/1/66
10864                          Benedict Pictures Corp.      11/2/66
10865                                      Norm-Iris     11/15/66

      vote_count  vote_average  release_year     budget_adj  revenue_adj
10861         11           7.4          1966       0.000000          0.0
10862         20           5.7          1966       0.000000          0.0
10863         11           6.5          1966       0.000000          0.0
10864         22           5.4          1966       0.000000          0.0
10865         15           1.5          1966  127642.279154          0.0

[5 rows x 21 columns]
```

[257]:
```
#getting the shape of the dataset
df.shape
```

[257]:
```
(10866, 21)
```

[258]:
```
#getting the info of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

```
 0   id                    10866 non-null  int64
 1   imdb_id               10856 non-null  object
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64
 4   revenue               10866 non-null  int64
 5   original_title        10866 non-null  object
 6   cast                  10790 non-null  object
 7   homepage              2936 non-null   object
 8   director              10822 non-null  object
 9   tagline               8042 non-null   object
 10  keywords              9373 non-null   object
 11  overview              10862 non-null  object
 12  runtime               10866 non-null  int64
 13  genres                10843 non-null  object
 14  production_companies  9836 non-null   object
 15  release_date          10866 non-null  object
 16  vote_count            10866 non-null  int64
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

After clear observation we can drop the columns like id, imdb_id,hompage, keywords,overview,production_companies,cast, tagline which are not usefull for our analasis

1.1.1 Data Cleaning

The Cleaning Process we are going to remove the columns "id", "imdb_id","homepage","keywords","overview","tagline","cast","production_companies" to as they make no sense for my analysis. The column 'genres' is not in the first normal form which requires that in the table should not have multiple value in the same row of data. we are going to do type casting for some colums which are wrongly type casted release_date from string to date datatype. we are going to convert the columns revenue, budget, revenue_adj and budget_adj from float to int. we are going to replace the 0 in revenue, budget, revenue_adj and budget_adj with means.

[259]:
```
df.drop(columns = ["id", "imdb_id","homepage",␣
↪"keywords","overview","tagline","cast","production_companies"],inplace =␣
↪True)
df.head()
```

[259]:

	popularity	budget	revenue	original_title	\
0	32.985763	150000000	1513528810	Jurassic World	
1	28.419936	150000000	378436354	Mad Max: Fury Road	
2	13.112507	110000000	295238201	Insurgent	

```
3   11.173104  200000000  2068178225  Star Wars: The Force Awakens
4    9.335014  190000000  1506249360                  Furious 7

           director  runtime                                     genres  \
0   Colin Trevorrow      124  Action|Adventure|Science Fiction|Thriller
1     George Miller      120  Action|Adventure|Science Fiction|Thriller
2  Robert Schwentke      119         Adventure|Science Fiction|Thriller
3       J.J. Abrams      136   Action|Adventure|Science Fiction|Fantasy
4         James Wan      137                      Action|Crime|Thriller

  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
3     12/15/15        5292           7.5          2015  1.839999e+08
4       4/1/15        2947           7.3          2015  1.747999e+08

    revenue_adj
0  1.392446e+09
1  3.481613e+08
2  2.716190e+08
3  1.902723e+09
4  1.385749e+09
```

Removing null valued rows

[260]:
```
#check for null values
df.isnull().sum()
```

[260]:
```
popularity         0
budget             0
revenue            0
original_title     0
director          44
runtime            0
genres            23
release_date       0
vote_count         0
vote_average       0
release_year       0
budget_adj         0
revenue_adj        0
dtype: int64
```

[261]:
```
#drop all rows with null value
df = df.dropna(axis=0)
```

[262]:
```
#confirm for null values
df.isnull().sum()
```

[262]:
```
popularity        0
budget            0
revenue           0
original_title    0
director          0
runtime           0
genres            0
release_date      0
vote_count        0
vote_average      0
release_year      0
budget_adj        0
revenue_adj       0
dtype: int64
```

Removing duplicated rows

[263]:
```
#checking the duplicate values
df.duplicated().sum()
```

[263]:
```
1
```

[264]:
```
#droping the duplicate values
df.drop_duplicates(inplace = True)
```

[265]:
```
#checking the shape
df.shape
```

[265]:
```
(10800, 13)
```

[266]:
```
# prinitng the top 2 rows of the dataset
df.head(2)
```

[266]:

	popularity	budget	revenue	original_title	director	runtime	genres	release_date	vote_count	vote_average	release_year	budget_adj	revenue_adj
0	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	6/9/15	5562	6.5	2015	1.379999e+08	1.392446e+09
1	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	5/13/15	6185	7.1	2015	1.379999e+08	3.481613e+08

Spliting the genres data into rows

[267]:
```
#splitting the geners data seperated by ""|""
df = df.drop('genres', axis=1).join(df['genres'].str.split('|', expand=True).
 ↪stack().reset_index(level=1, drop=True).rename('genres'))
```

[268]:
```
#reseting the index of the dataset
df.reset_index(inplace = True)
```

[269]:
```
#prinitng the dataset
df.head()
```

[269]:
```
   index  popularity     budget     revenue      original_title  \
0      0   32.985763  150000000  1513528810      Jurassic World
1      0   32.985763  150000000  1513528810      Jurassic World
2      0   32.985763  150000000  1513528810      Jurassic World
3      0   32.985763  150000000  1513528810      Jurassic World
4      1   28.419936  150000000   378436354  Mad Max: Fury Road

          director  runtime release_date  vote_count  vote_average  \
0  Colin Trevorrow      124       6/9/15        5562           6.5
1  Colin Trevorrow      124       6/9/15        5562           6.5
2  Colin Trevorrow      124       6/9/15        5562           6.5
3  Colin Trevorrow      124       6/9/15        5562           6.5
4    George Miller      120      5/13/15        6185           7.1

   release_year    budget_adj   revenue_adj           genres
0          2015  1.379999e+08  1.392446e+09           Action
1          2015  1.379999e+08  1.392446e+09        Adventure
2          2015  1.379999e+08  1.392446e+09  Science Fiction
3          2015  1.379999e+08  1.392446e+09         Thriller
4          2015  1.379999e+08  3.481613e+08           Action
```

[270]:
```
df
```

[270]:
```
       index  popularity     budget     revenue            original_title  \
0          0   32.985763  150000000  1513528810            Jurassic World
1          0   32.985763  150000000  1513528810            Jurassic World
2          0   32.985763  150000000  1513528810            Jurassic World
3          0   32.985763  150000000  1513528810            Jurassic World
4          1   28.419936  150000000   378436354        Mad Max: Fury Road
…        …          …          …          …                         …
26859  10863    0.065141          0          0       Beregis Avtomobilya
26860  10863    0.065141          0          0       Beregis Avtomobilya
26861  10864    0.064317          0          0    What's Up, Tiger Lily?
26862  10864    0.064317          0          0    What's Up, Tiger Lily?
26863  10865    0.035919      19000          0  Manos: The Hands of Fate
```

```
                director  runtime release_date  vote_count  vote_average  \
0        Colin Trevorrow      124       6/9/15        5562           6.5
1        Colin Trevorrow      124       6/9/15        5562           6.5
2        Colin Trevorrow      124       6/9/15        5562           6.5
3        Colin Trevorrow      124       6/9/15        5562           6.5
4          George Miller      120      5/13/15        6185           7.1
…                     …        …           …           …             …
26859     Eldar Ryazanov       94       1/1/66          11           6.5
26860     Eldar Ryazanov       94       1/1/66          11           6.5
26861        Woody Allen       80      11/2/66          22           5.4
26862        Woody Allen       80      11/2/66          22           5.4
26863  Harold P. Warren       74     11/15/66          15           1.5

       release_year    budget_adj   revenue_adj           genres
0              2015  1.379999e+08  1.392446e+09           Action
1              2015  1.379999e+08  1.392446e+09        Adventure
2              2015  1.379999e+08  1.392446e+09  Science Fiction
3              2015  1.379999e+08  1.392446e+09         Thriller
4              2015  1.379999e+08  3.481613e+08           Action
…               …           …           …                …
26859          1966  0.000000e+00  0.000000e+00          Mystery
26860          1966  0.000000e+00  0.000000e+00           Comedy
26861          1966  0.000000e+00  0.000000e+00           Action
26862          1966  0.000000e+00  0.000000e+00           Comedy
26863          1966  1.276423e+05  0.000000e+00           Horror

[26864 rows x 14 columns]
```

Converting datatypes

[271]:
```
# Convert release_date (object datatype) to date.

df['release_date'] = pd.to_datetime(df['release_date'])


# Convert budget_adj and revenue_adj from float to int.

df['budget_adj'] = df['budget_adj'].astype(int)
df['revenue_adj'] = df['revenue_adj'].astype(int)

# convert budget and revenue from float to int.

df['budget'] = df['budget']
df['revenue'] = df['revenue'].astype(int)
```

[272]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26864 entries, 0 to 26863
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           26864 non-null  int64
 1   popularity      26864 non-null  float64
 2   budget          26864 non-null  int64
 3   revenue         26864 non-null  int64
 4   original_title  26864 non-null  object
 5   director        26864 non-null  object
 6   runtime         26864 non-null  int64
 7   release_date    26864 non-null  datetime64[ns]
 8   vote_count      26864 non-null  int64
 9   vote_average    26864 non-null  float64
 10  release_year    26864 non-null  int64
 11  budget_adj      26864 non-null  int64
 12  revenue_adj     26864 non-null  int64
 13  genres          26864 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(8), object(3)
memory usage: 2.9+ MB
```

Replacing null values with mean

[273]:
```
# replacing the 0 values with mean value
df['budget']=df['budget'].replace(0,df['budget'].mean())

df['revenue']=df['revenue'].replace(0,df['revenue'].mean())

df['budget_adj']=df['budget_adj'].replace(0,df['budget_adj'].mean())

df['revenue_adj']=df['revenue_adj'].replace(0,df['revenue_adj'].mean())
```

Exploratory Data Analysis

1.1.2 Research Question 1: Which genres are most popular over decades?

[274]:
```
# Create bin edges to decades
decades = [1960, 1970, 1980, 1990, 2000, 2010, 2020]

# Create labels
decade_names = ['1960', '1970', '1980', '1990', '2000', '2010']

# Create new column and cut into bins
df['release_decade'] = pd.cut(df['release_year'], decades, labels=decade_names)

df.head()
```

[274]:

	index	popularity	budget	revenue	original_title	director	runtime	release_date	vote_count	vote_average	release_year	budget_adj	revenue_adj	genres	release_decade
0	0	32.985763	150000000.0	1.513529e+09	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.5	2015	137999939.0	1.392446e+09	Action	2010
1	0	32.985763	150000000.0	1.513529e+09	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.5	2015	137999939.0	1.392446e+09	Adventure	2010
2	0	32.985763	150000000.0	1.513529e+09	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.5	2015	137999939.0	1.392446e+09	Science Fiction	2010
3	0	32.985763	150000000.0	1.513529e+09	Jurassic World	Colin Trevorrow	124	2015-06-09	5562	6.5	2015	137999939.0	1.392446e+09	Thriller	2010
4	1	28.419936	150000000.0	3.784364e+08	Mad Max: Fury Road	George Miller	120	2015-05-13	6185	7.1	2015	137999939.0	3.481613e+08	Action	2010

[275]:
```
df.describe()
```

[275]:

	index	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	26864.000000	26864.000000	2.686400e+04	2.686400e+04	26864.000000	26864.000000	26864.000000	26864.000000	2.686400e+04	2.686400e+04
mean	5564.420265	0.707988	2.624504e+07	7.277206e+07	102.841758	250.782720	5.954370	2000.675886	3.154200e+07	9.339738e+07
std	3131.231930	1.116378	3.119359e+07	1.252977e+08	29.800772	638.957858	0.911253	12.770128	3.367012e+07	1.521543e+08
min	0.000000	0.000188	1.000000e+00	2.000000e+00	0.000000	10.000000	1.500000	1960.000000	1.000000e+00	2.000000e+00
25%	2858.750000	0.225678	1.756557e+07	4.098266e+07	90.000000	18.000000	5.400000	1994.000000	2.110920e+07	5.899749e+07
50%	5579.000000	0.412474	1.756557e+07	4.760272e+07	100.000000	44.000000	6.000000	2005.000000	2.110920e+07	6.109446e+07
75%	8302.250000	0.777600	2.000000e+07	4.760272e+07	112.000000	174.000000	6.600000	2011.000000	2.714314e+07	6.109446e+07
max	10865.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

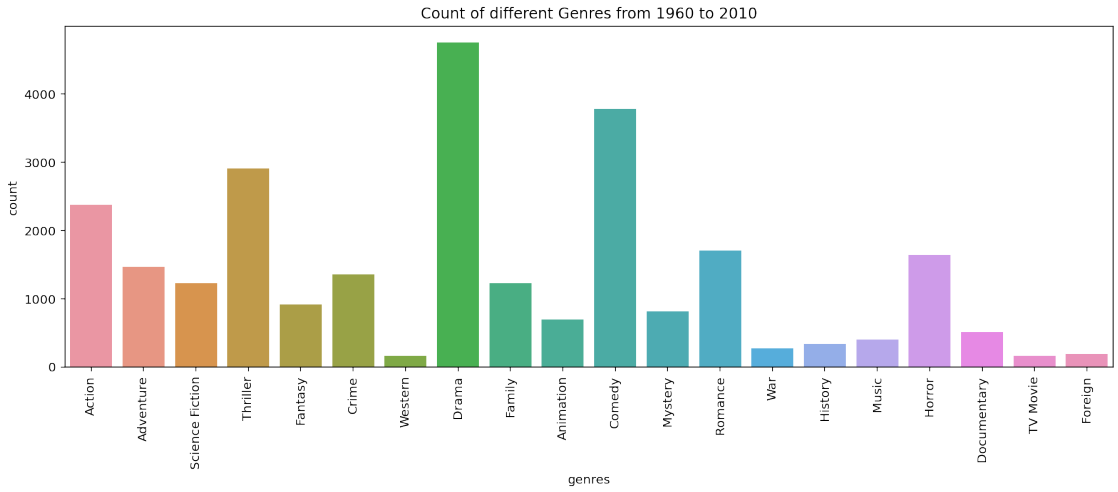
[276]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26864 entries, 0 to 26863
```

```
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           26864 non-null  int64
 1   popularity      26864 non-null  float64
 2   budget          26864 non-null  float64
 3   revenue         26864 non-null  float64
 4   original_title  26864 non-null  object
 5   director        26864 non-null  object
 6   runtime         26864 non-null  int64
 7   release_date    26864 non-null  datetime64[ns]
 8   vote_count      26864 non-null  int64
 9   vote_average    26864 non-null  float64
 10  release_year    26864 non-null  int64
 11  budget_adj      26864 non-null  float64
 12  revenue_adj     26864 non-null  float64
 13  genres          26864 non-null  object
 14  release_decade  26786 non-null  category
dtypes: category(1), datetime64[ns](1), float64(6), int64(4), object(3)
memory usage: 2.9+ MB
```

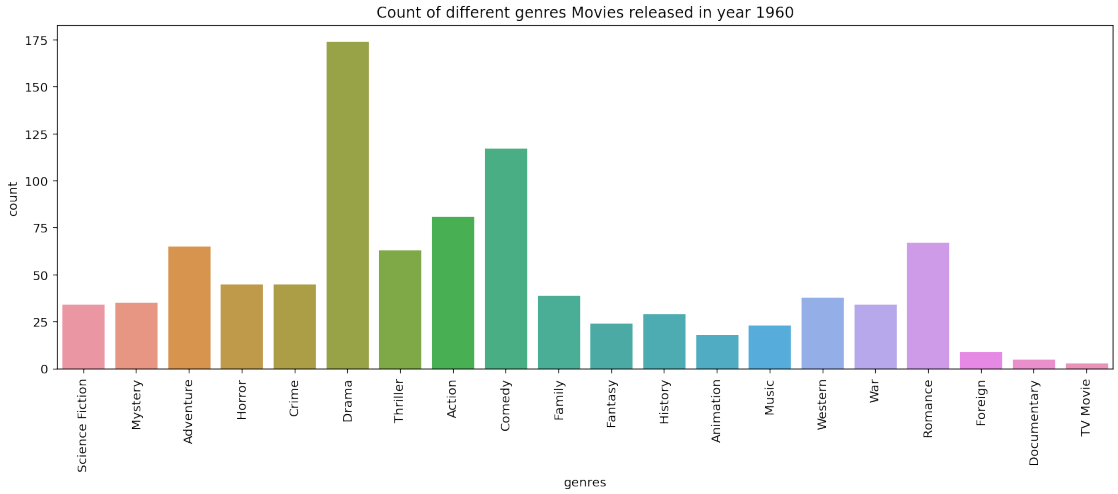
[277]:
```
plt.figure(figsize=(15,5));
sns.countplot(df["genres"]);
plt.xticks(rotation=90);
plt.title("Count of different Genres from 1960 to 2010")
```

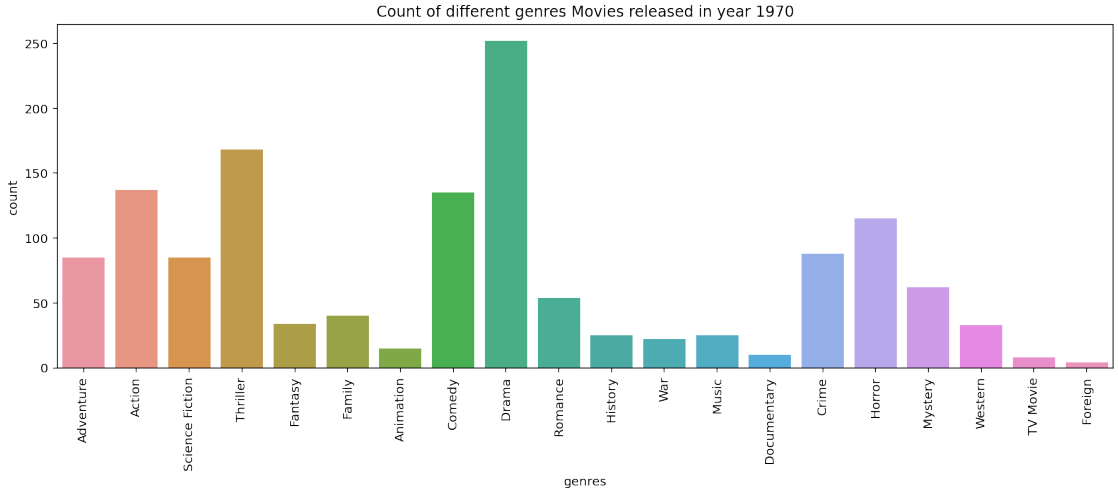
[277]: Text(0.5, 1.0, 'Count of different Genres from 1960 to 2010')

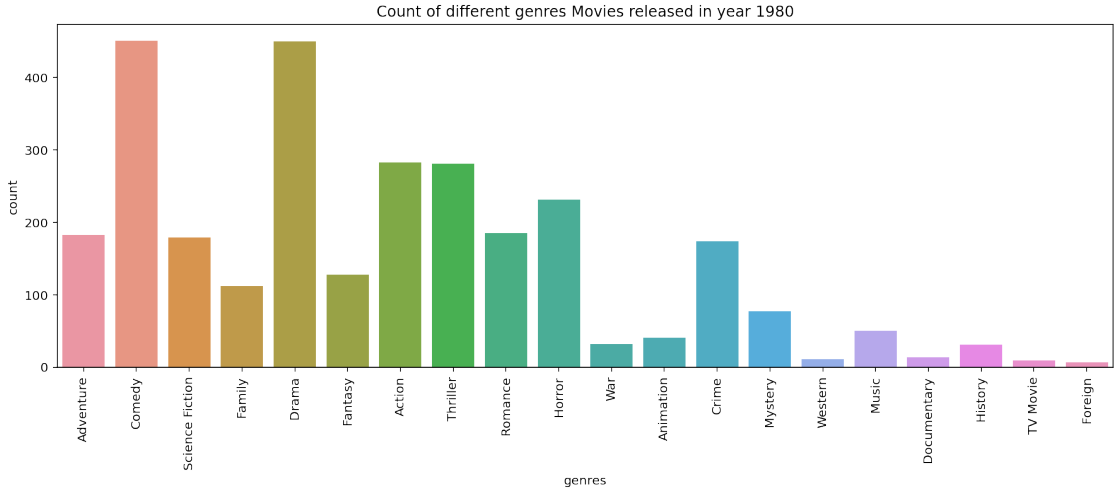


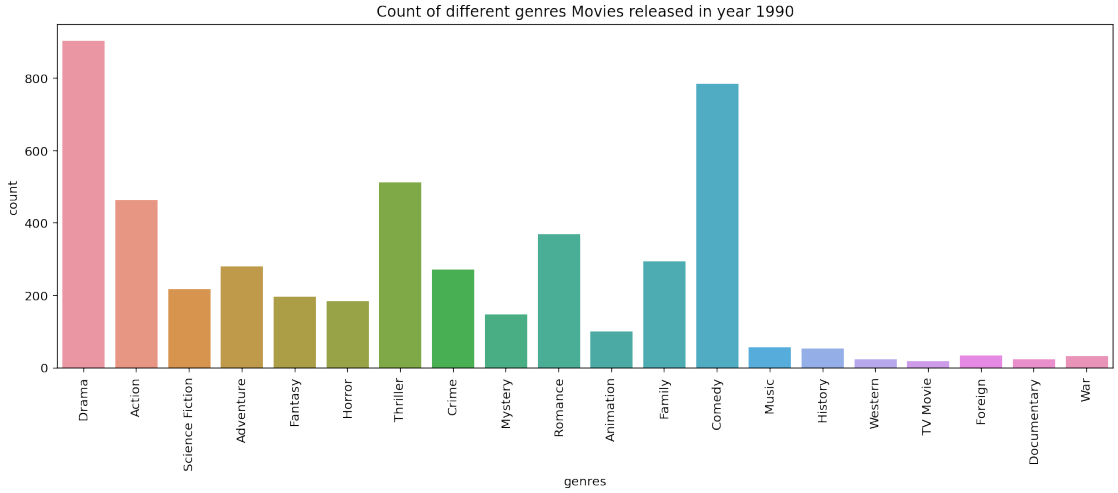
[278]:
```
lables=["1960","1970","1980","1990","2000","2010"]
for i in lables:
    plt.figure(figsize=(15,5));
```

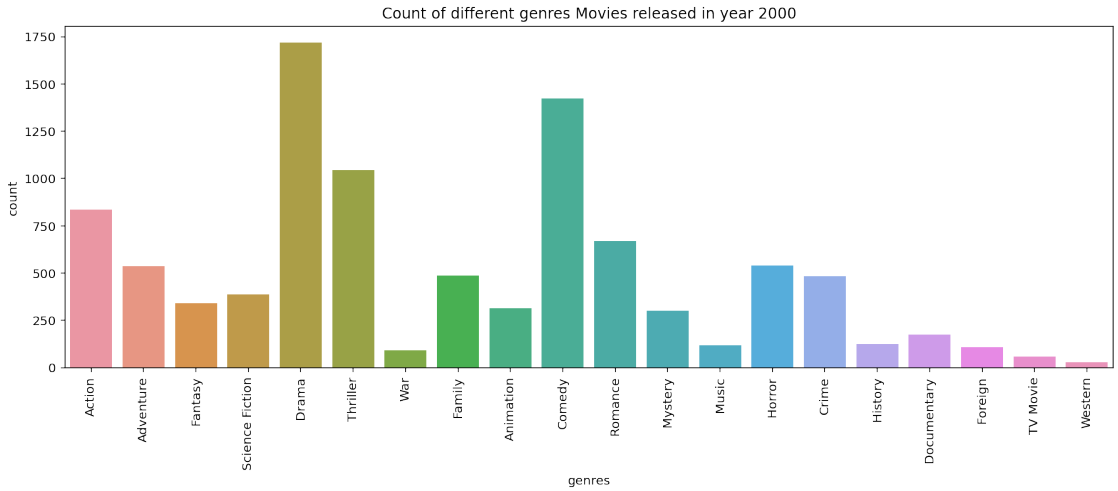
```
df_new = df[df["release_decade"]== i]
sns.countplot(df_new["genres"]);
plt.xticks(rotation=90);
plt.title("Count of different genres Movies released in year "+i)
```

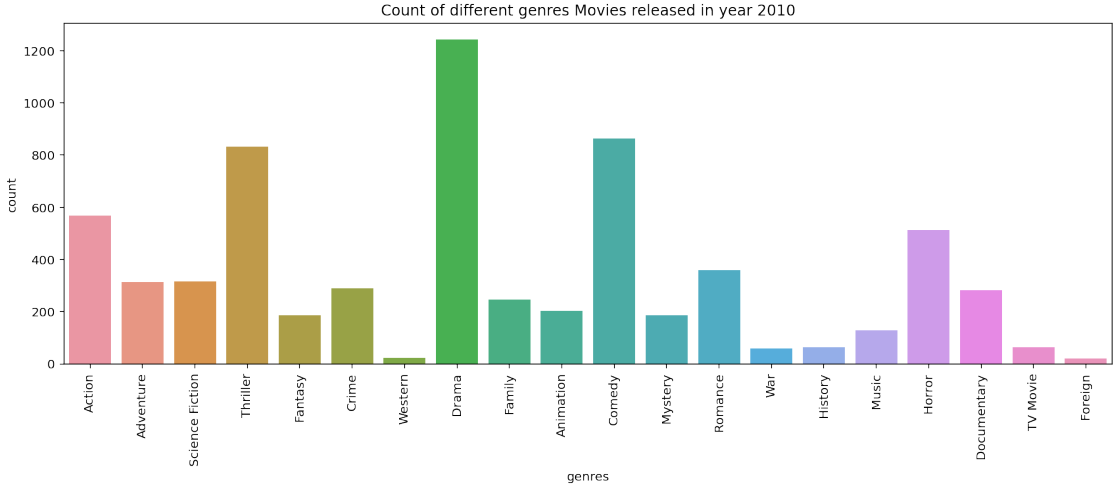





Count of different genres Movies released in year 1980
count
0
100
200
300
400
Adventure
Comedy
Science Fiction
Family
Drama
Fantasy
Action
Thriller
Romance
Horror
War
Animation
Crime
Mystery
Western
Music
Documentary
History
TV Movie
Foreign
genres


Count of different genres Movies released in year 1990
count
0
200
400
600
800
Drama
Action
Science Fiction
Adventure
Fantasy
Horror
Thriller
Crime
Mystery
Romance
Animation
Family
Comedy
Music
History
Western
TV Movie
Foreign
Documentary
War
genres


Count of different genres Movies released in year 2000
count
0
250
500
750
1000
1250
1500
1750
Action
Adventure
Fantasy
Science Fiction
Drama
Thriller
War
Family
Animation
Comedy
Romance
Mystery
Music
Horror
Crime
History
Documentary
Foreign
TV Movie
Western
genres

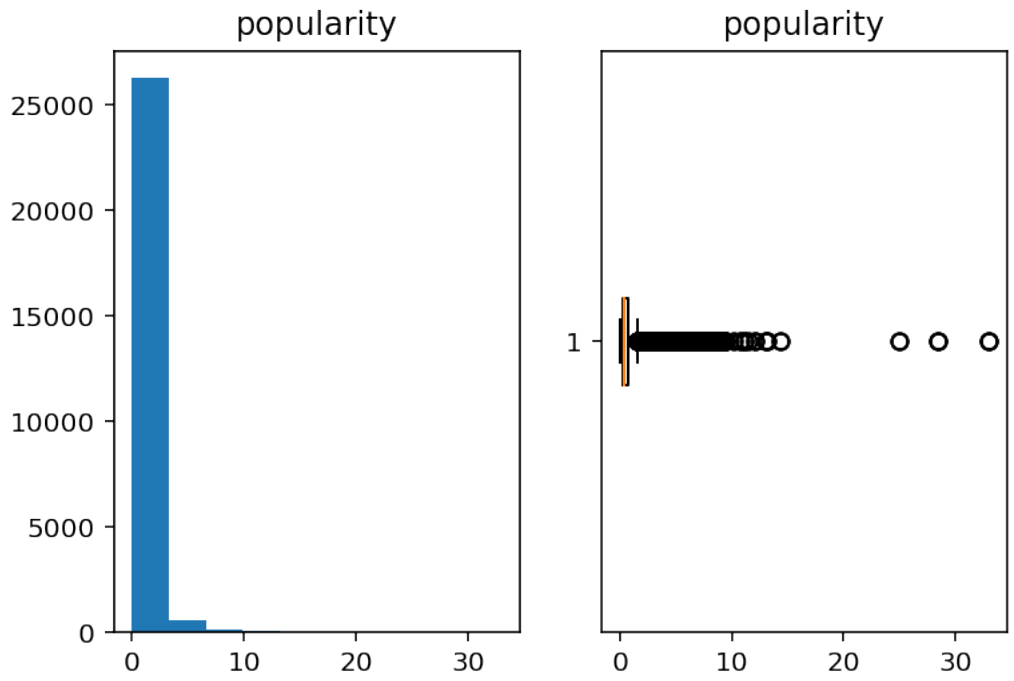


From above plots, every deacde most the movies released are drama based.

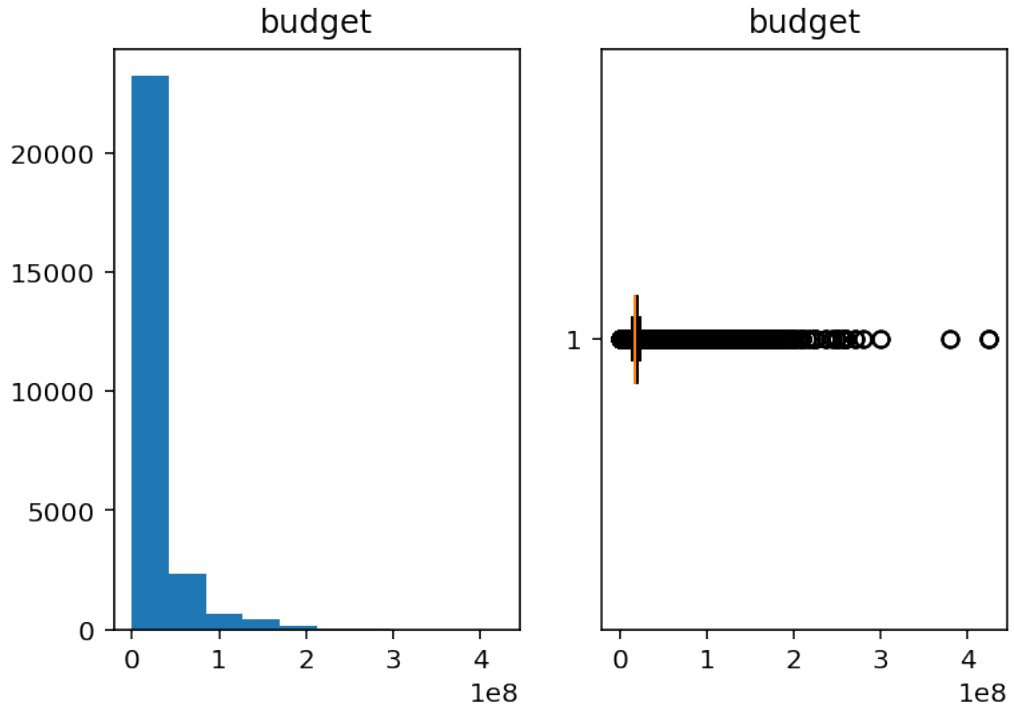
[279]:
```
def histbox(data):
    plt.subplot(1,2,1)
    plt.title(data)
    plt.hist(df[data]);
    print("Skewness of "+ str(data) +" is "+str(df[data].skew()))
    plt.subplot(1,2,2)
    plt.title(data)
    plt.boxplot(df[data], vert=False)
    plt.show()
```

[280]:
```
numerics = ['int64','float64']
columns = df.drop(columns="index").select_dtypes(include=numerics).columns
for i in columns:
    histbox(i)
```

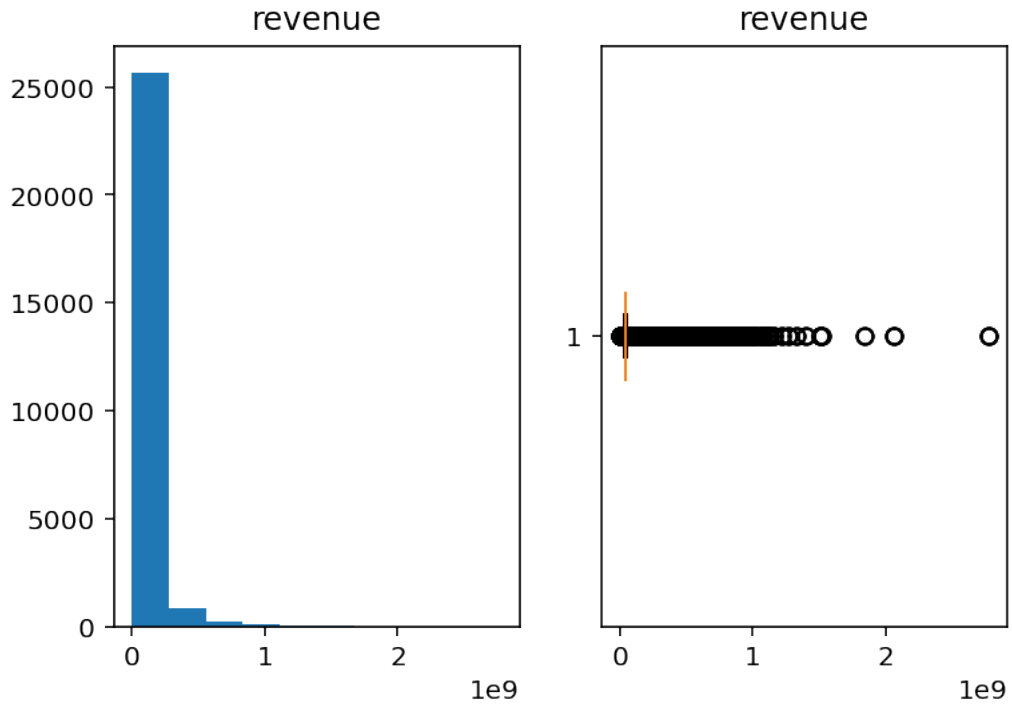
```
Skewness of popularity is 10.002785249501501
```



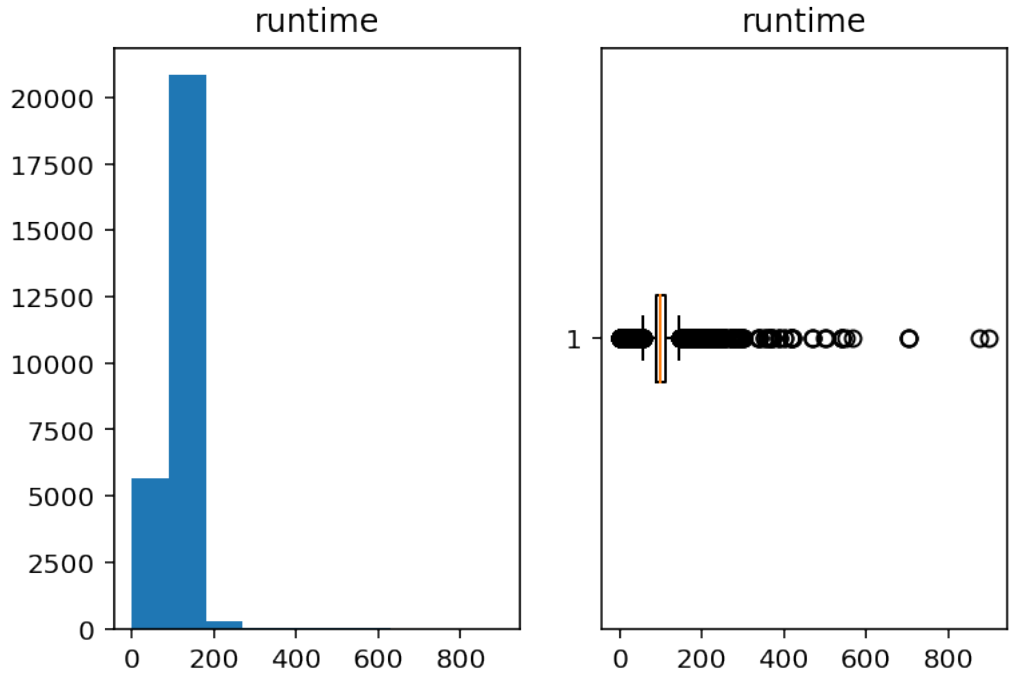

```
Skewness of budget is 3.8163788356462605
```



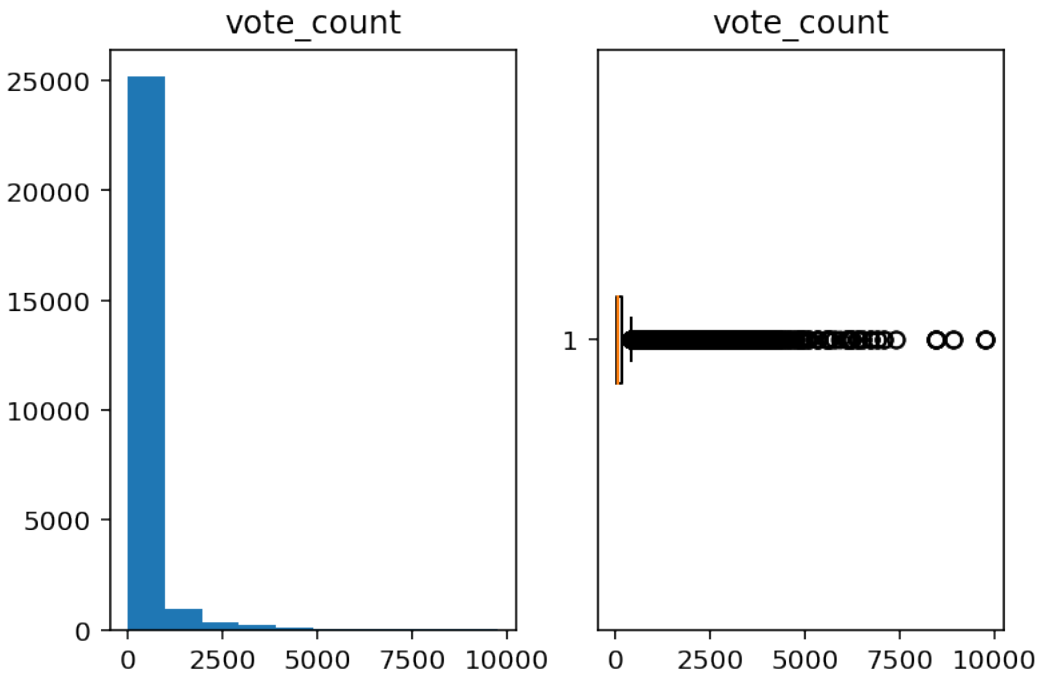
```
Skewness of revenue is 6.693638881583578
```



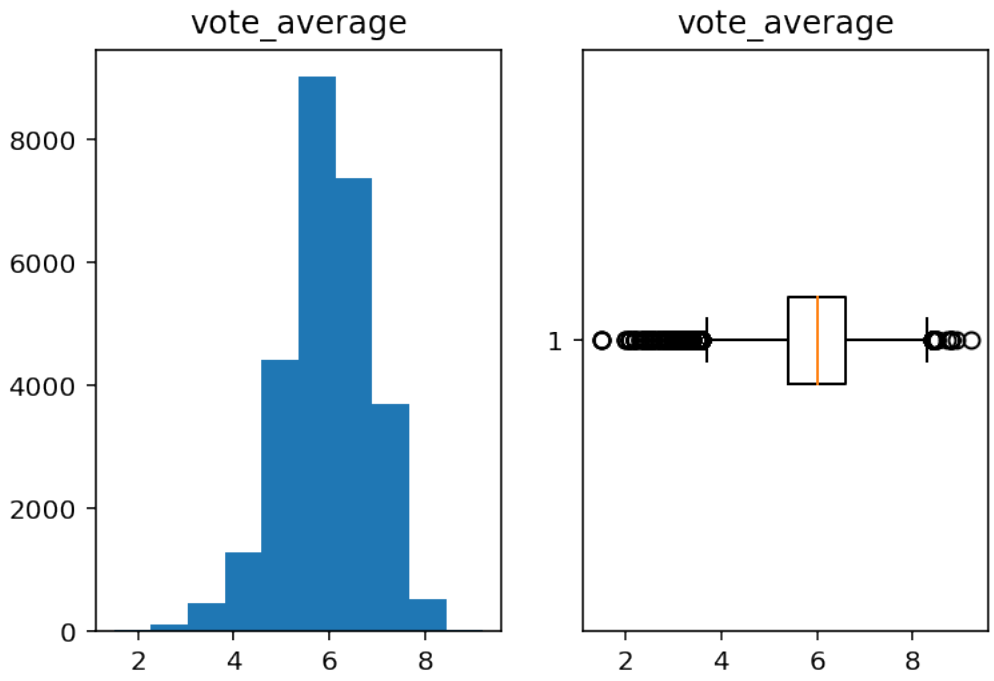

```
Skewness of runtime is 4.787848480938347
```



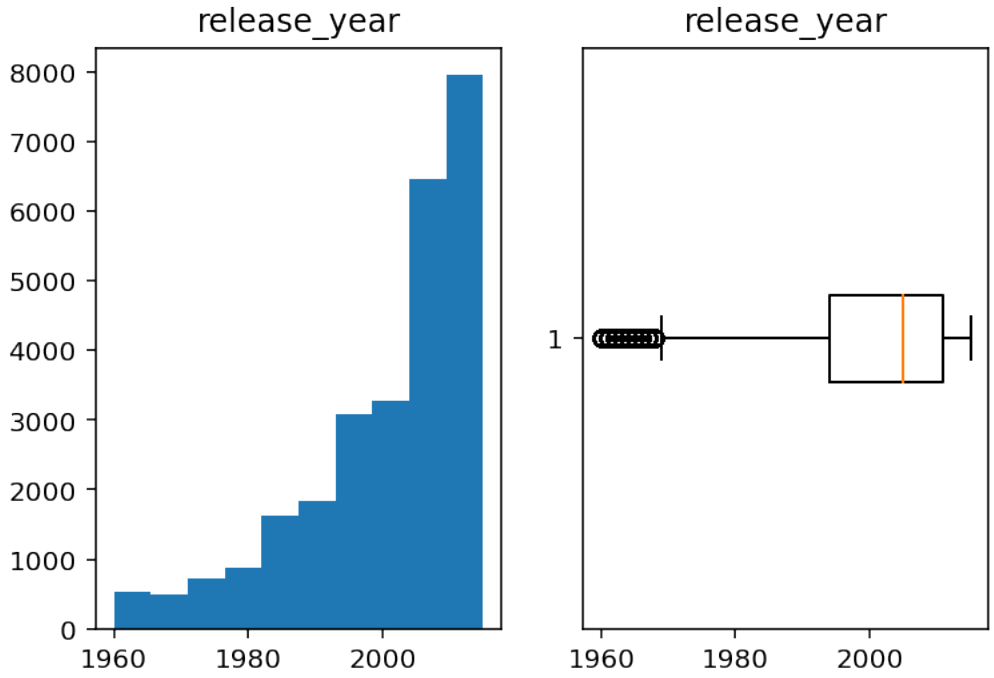
```
Skewness of vote_count is 5.760721661365619
```



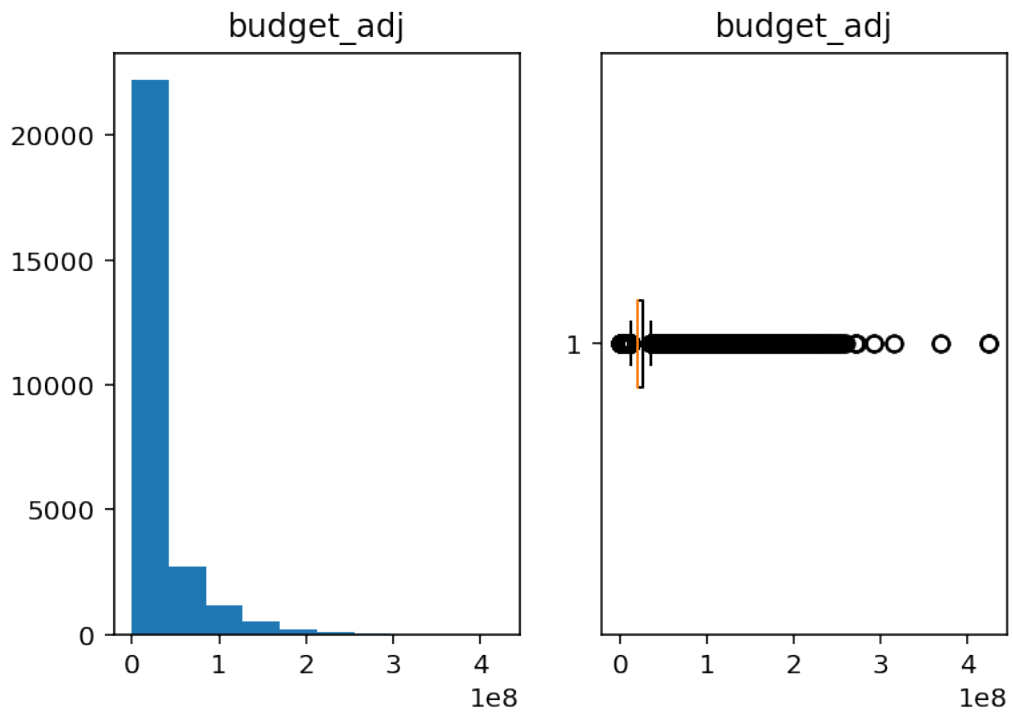

```
Skewness of vote_average is -0.4777897293969597
```



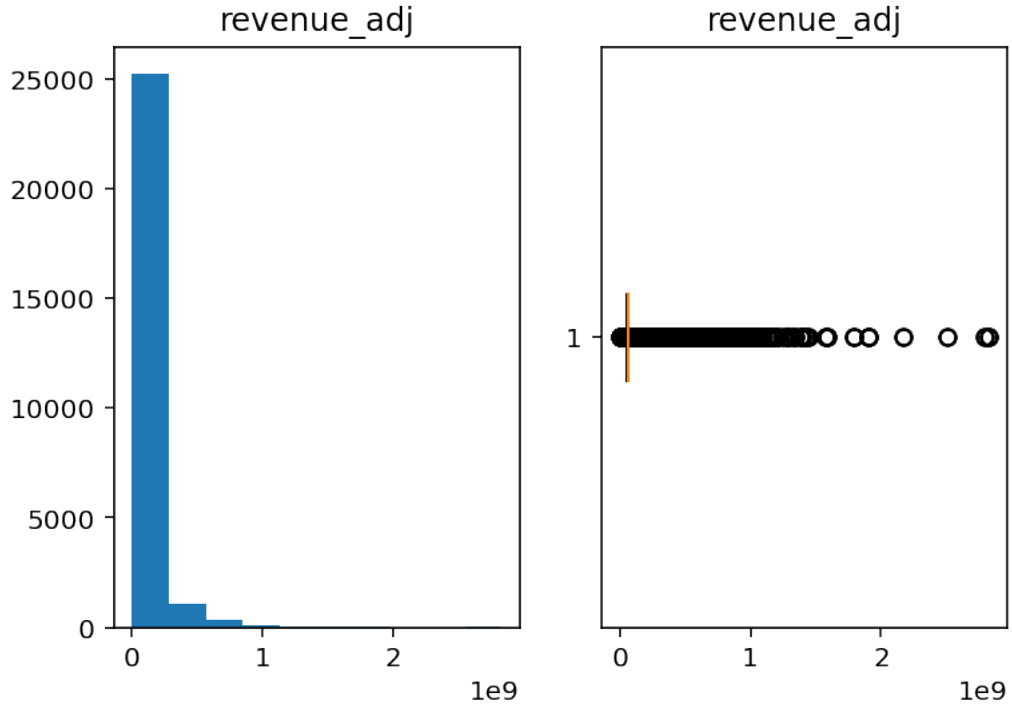

```
Skewness of release_year is -1.14140978324778
```




```
Skewness of budget_adj is 3.2316926434310718
```




```
Skewness of revenue_adj is 6.272068600557761
```



In the above distrubutions all the columns are left skewed except Vote_average. and also from the box plots we can say that there are some outliers in every columns but as these columns related to votings, budget and revenue. So, we can not say that they are errors in data. Thus, we will leave the outliers as it is.

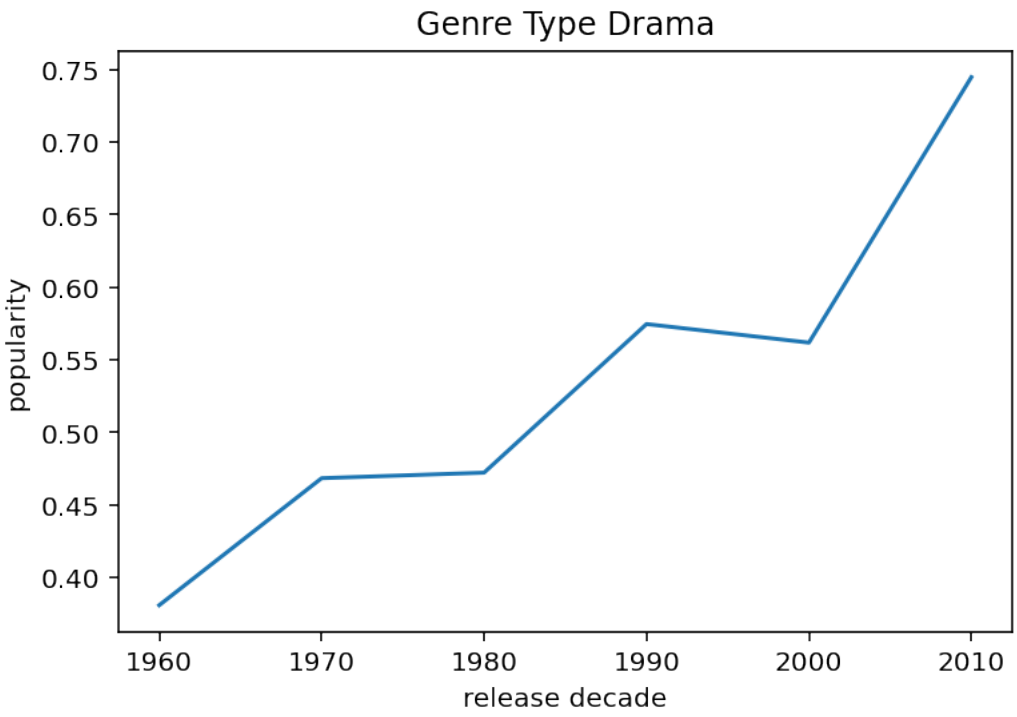
[281]:
```
df_decade=df.groupby(['release_decade','genres'], as_index =␣
 ↪False)['popularity'].mean()
df_decade
```

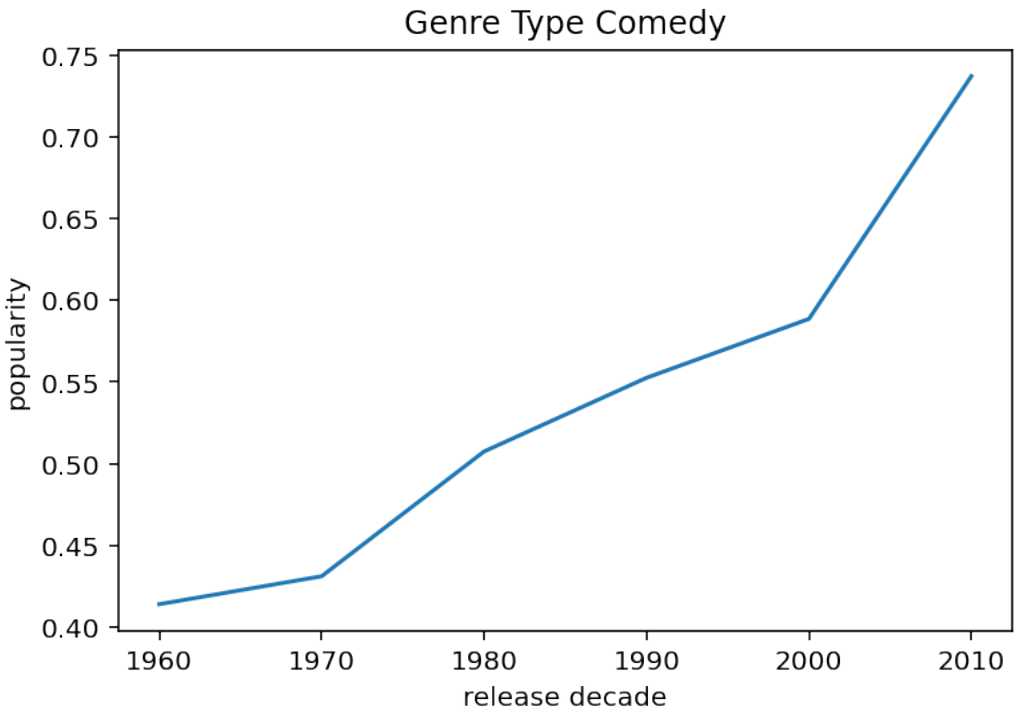
[281]:

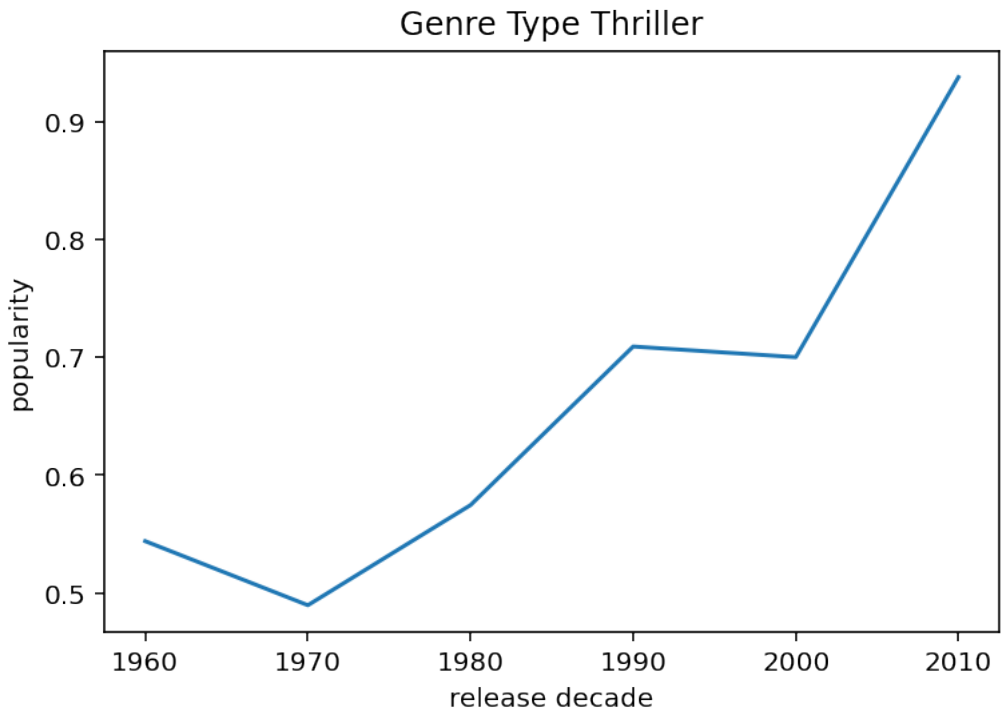
	release_decade	genres	popularity
0	1960	Action	0.483590
1	1960	Adventure	0.729040
2	1960	Animation	0.822127
3	1960	Comedy	0.413950
4	1960	Crime	0.465051
..	…	…	…
115	2010	Science Fiction	1.627444
116	2010	TV Movie	0.281422
117	2010	Thriller	0.937560
118	2010	War	1.049920
119	2010	Western	1.616852

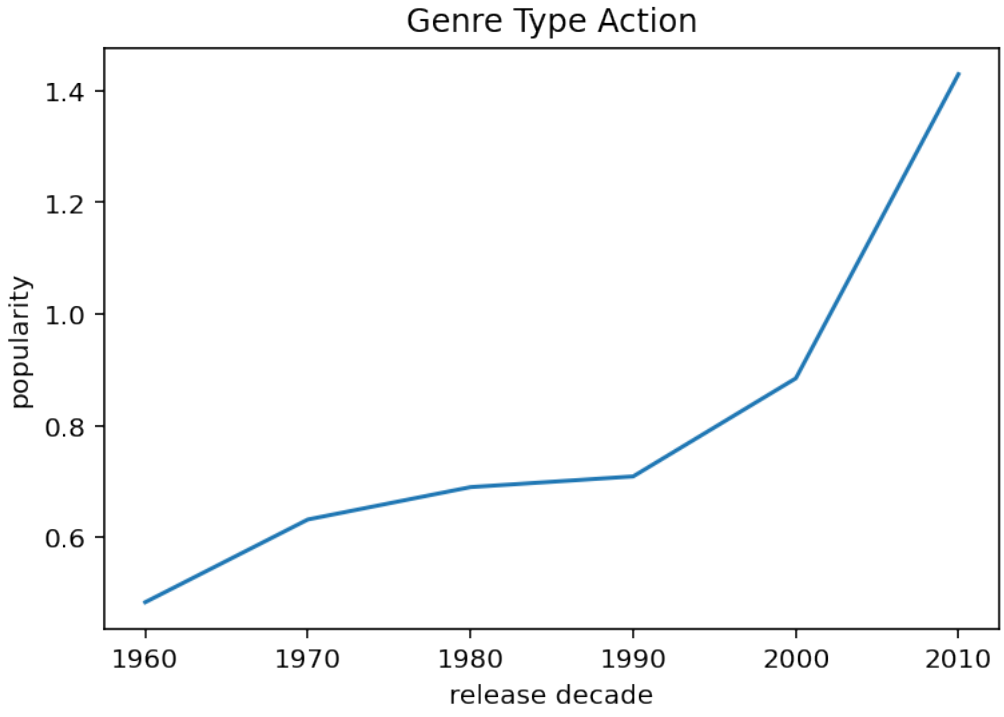
[120 rows x 3 columns]

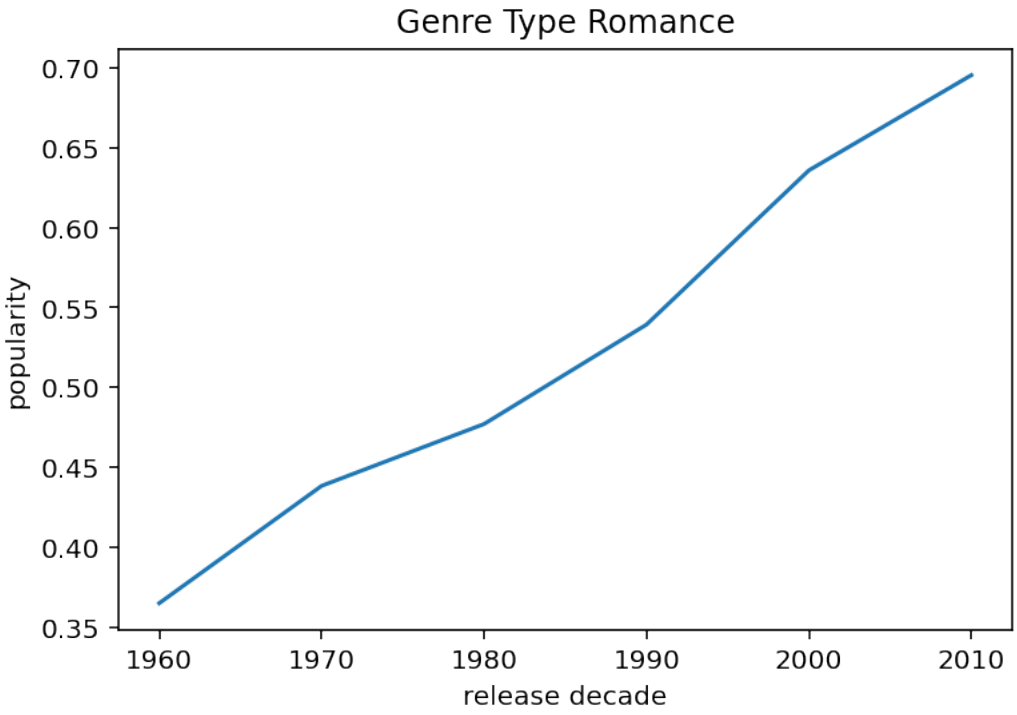
[282]:
```
for i in df["genres"].value_counts().index:
    lables=["1960","1970","1980","1990","2000","2010"]
    Data= df_decade[df_decade["genres"]==i]

    plt.plot(Data["release_decade"],Data["popularity"])
    plt.xticks(lables)
    plt.title("Genre Type "+i)
    plt.xlabel('release decade')
    plt.ylabel('popularity')
    plt.show()
```

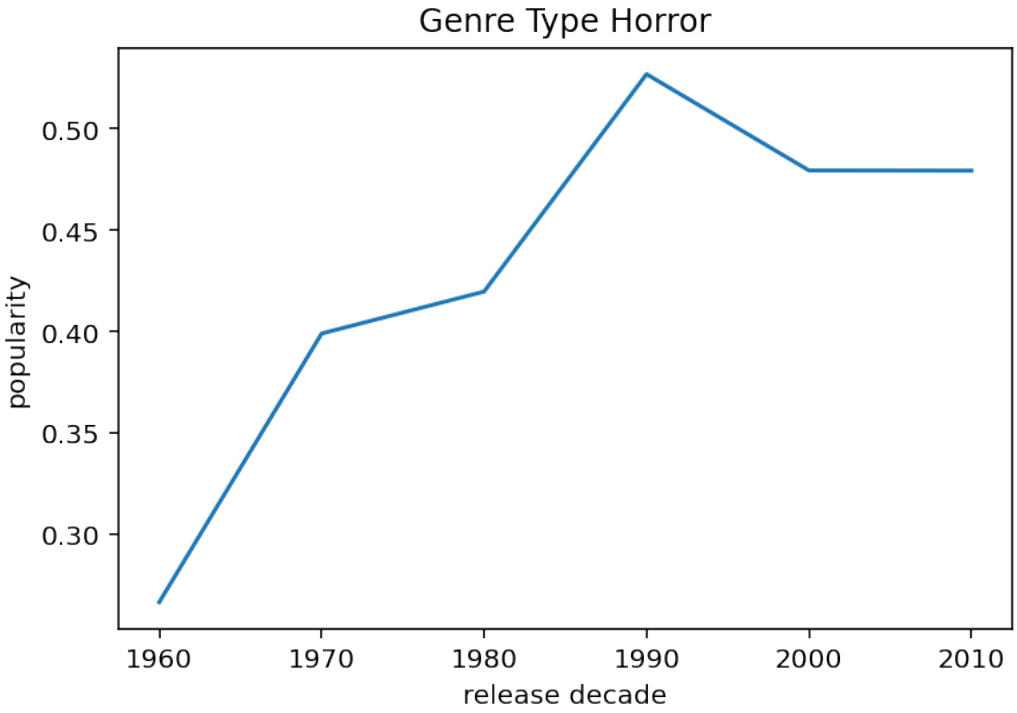

Genre Type Drama
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
popularity
1960
1970
1980
1990
2000
2010
release decade

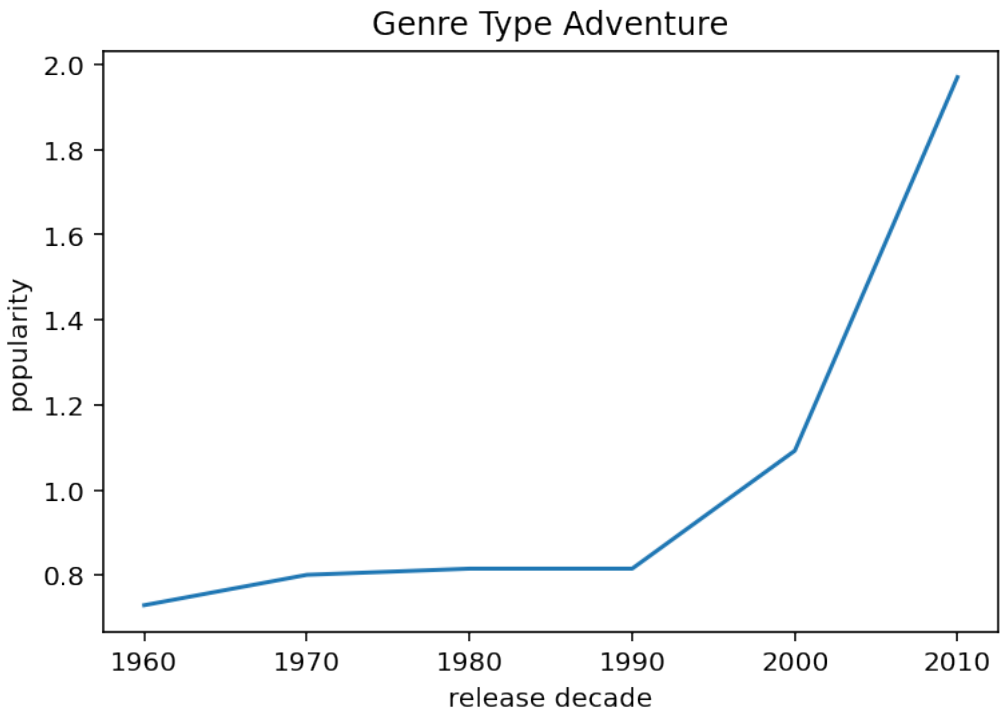

Genre Type Comedy
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
popularity
1960
1970
1980
1990
2000
2010
release decade

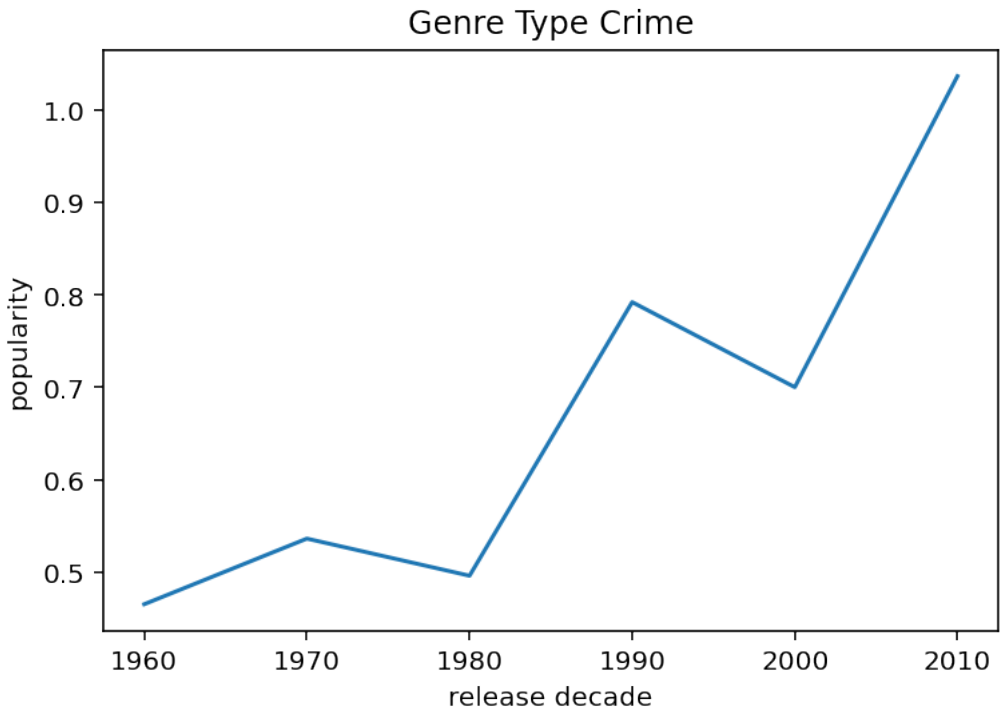

Genre Type Thriller
popularity
0.9
0.8
0.7
0.6
0.5
1960
1970
1980
1990
2000
2010
release decade

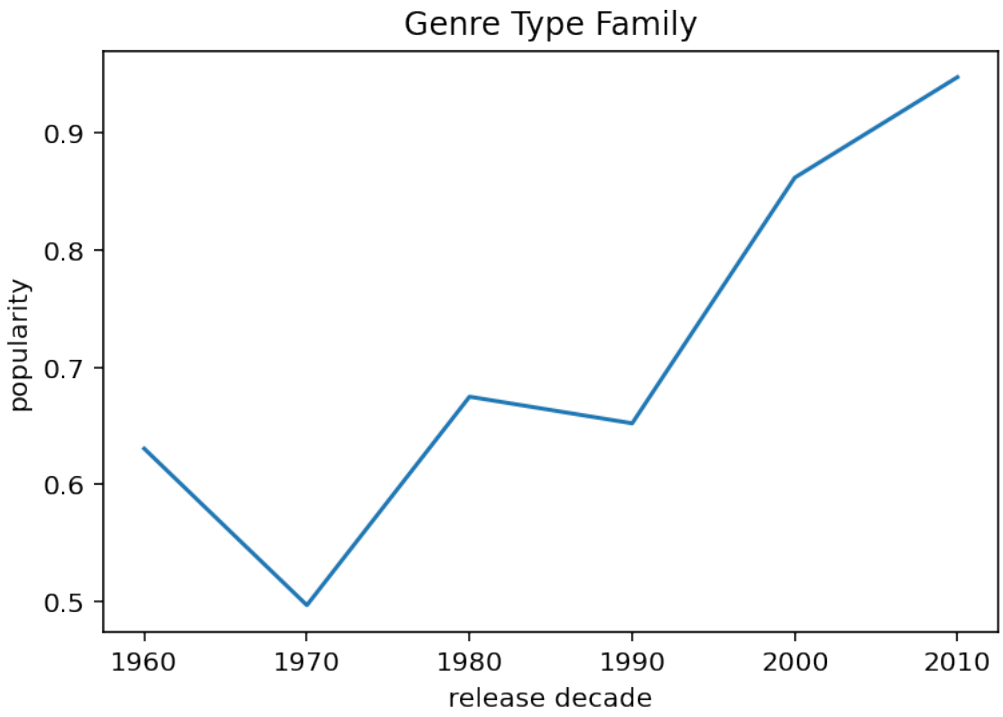

Genre Type Action
popularity
1.4
1.2
1.0
0.8
0.6
1960
1970
1980
1990
2000
2010
release decade

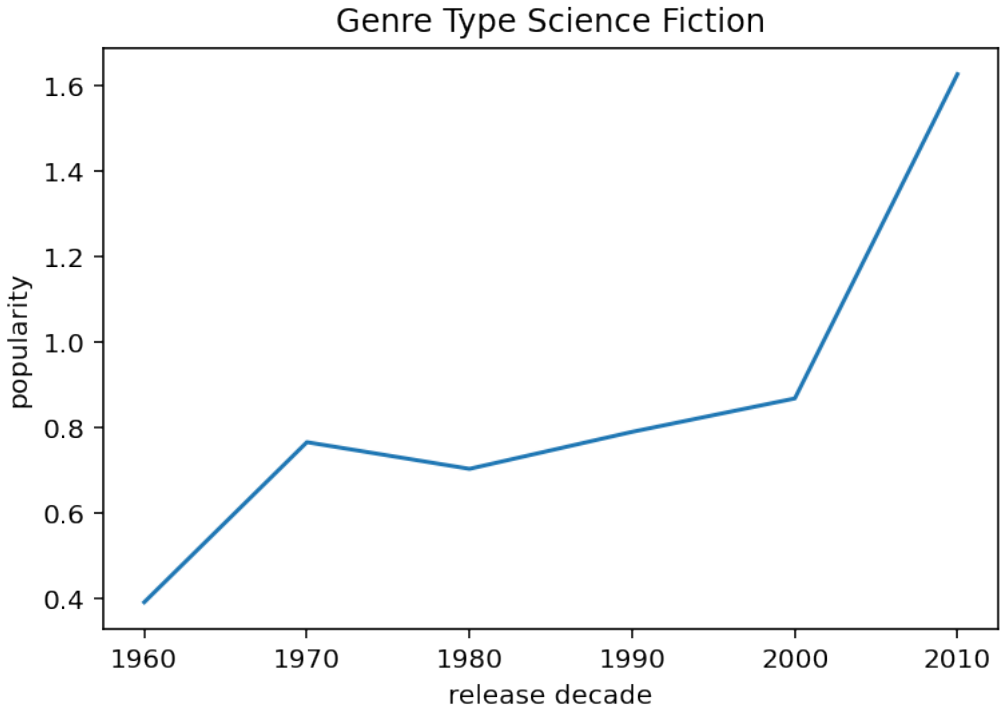

Genre Type Romance
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
popularity
1960
1970
1980
1990
2000
2010
release decade

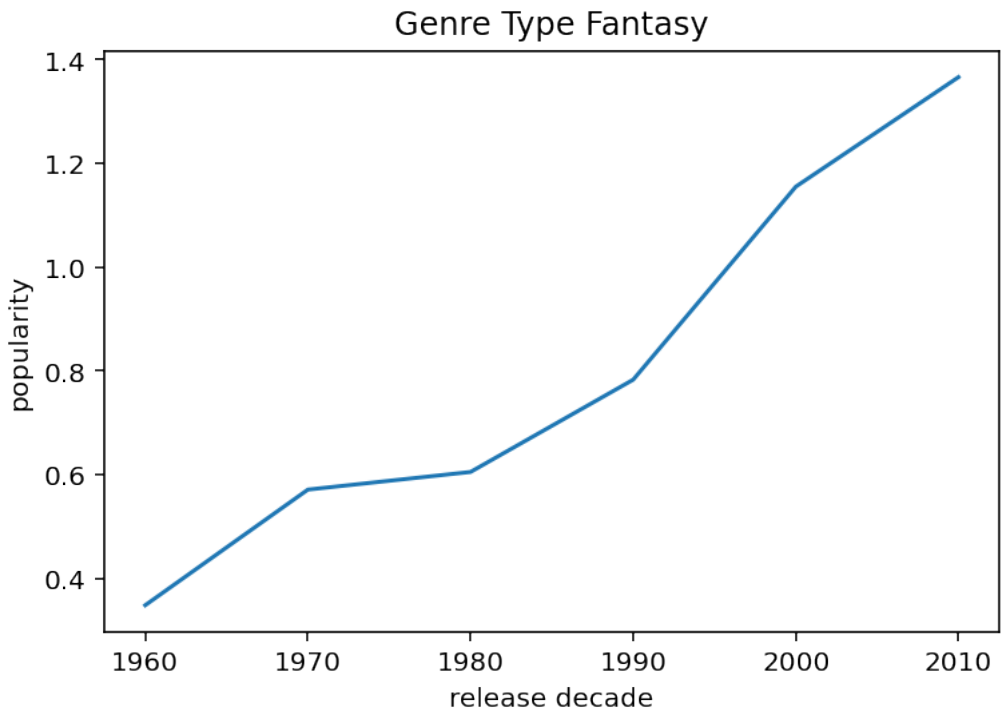

Genre Type Horror
0.50
0.45
0.40
0.35
0.30
popularity
1960
1970
1980
1990
2000
2010
release decade

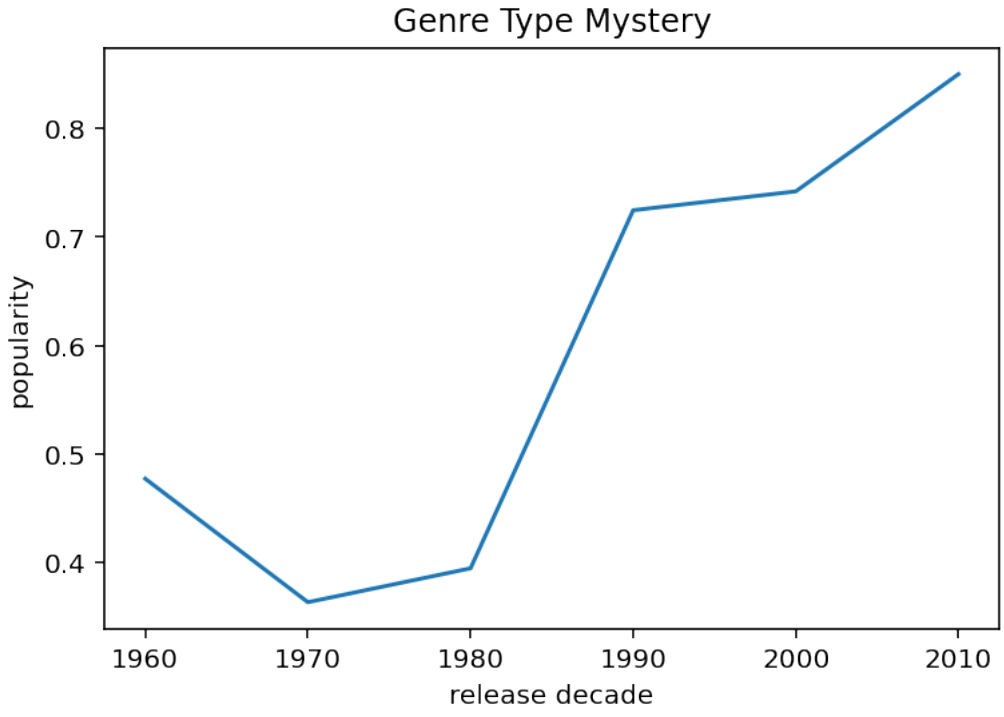

Genre Type Adventure
2.0
1.8
1.6
1.4
1.2
1.0
0.8
popularity
1960
1970
1980
1990
2000
2010
release decade

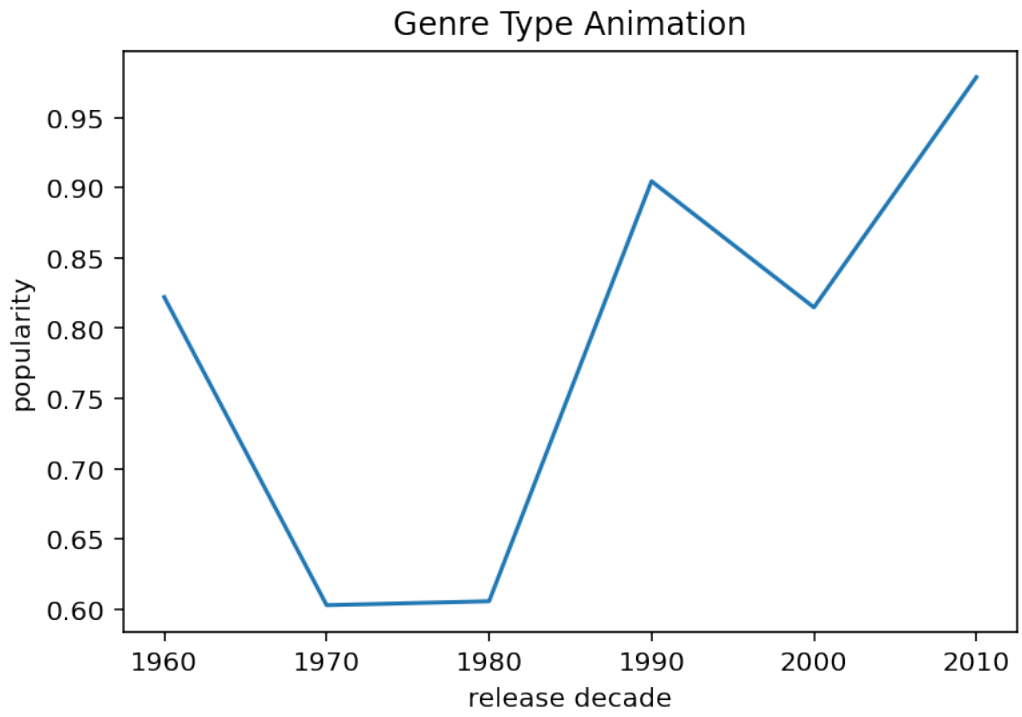

Genre Type Crime
1.0
0.9
0.8
0.7
0.6
0.5
popularity
1960
1970
1980
1990
2000
2010
release decade

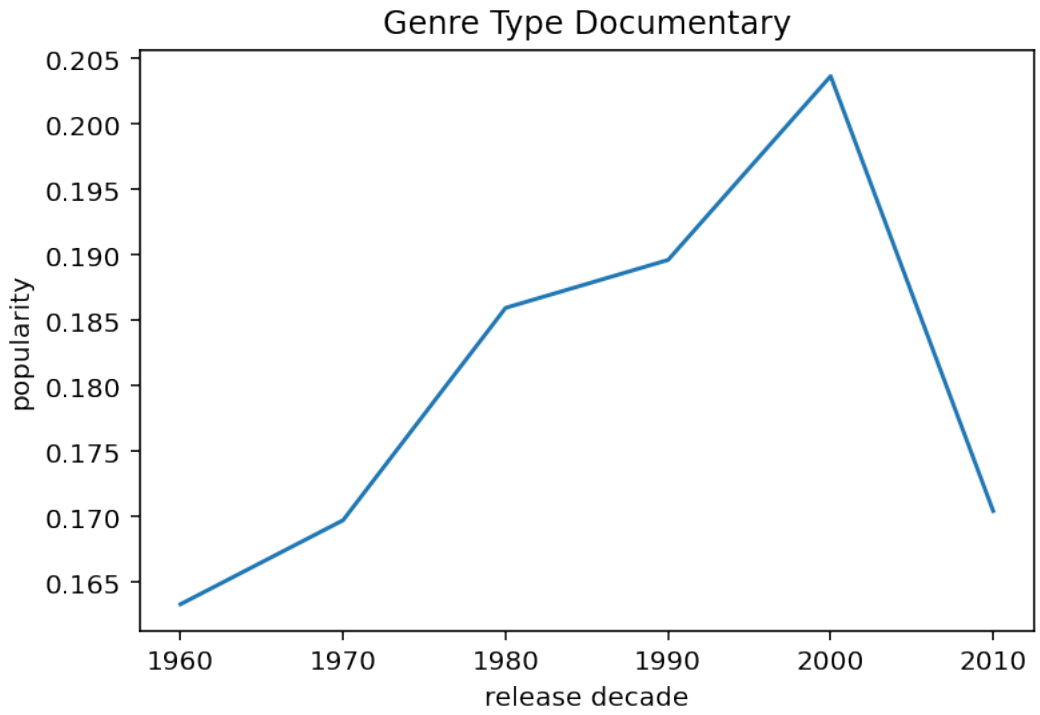

Genre Type Family
popularity
0.9
0.8
0.7
0.6
0.5
1960
1970
1980
1990
2000
2010
release decade

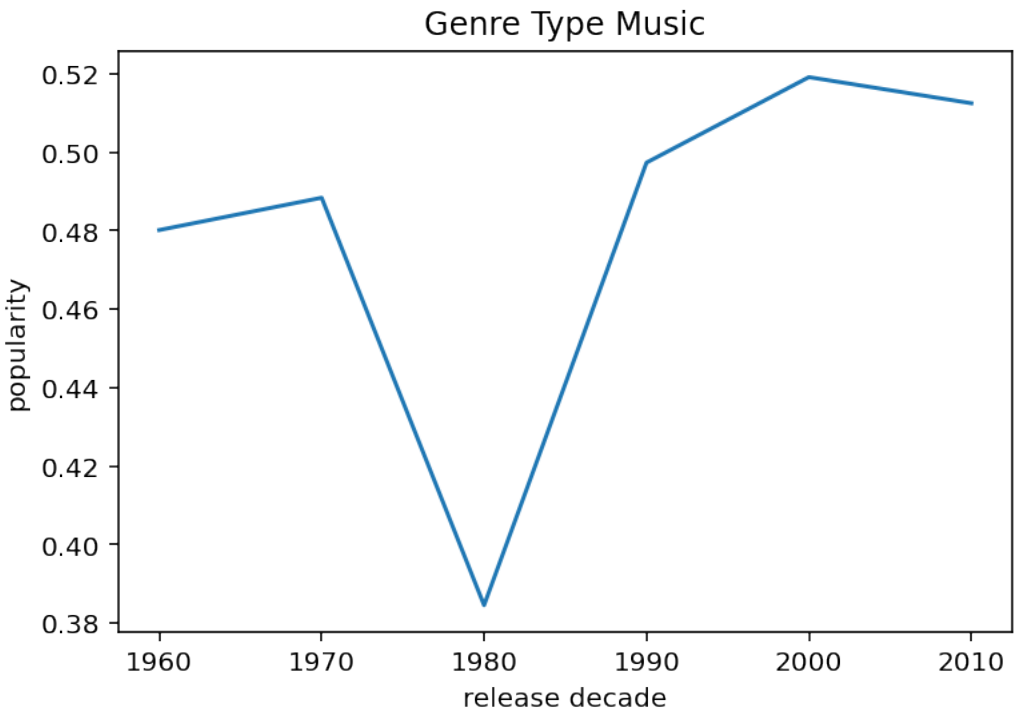

Genre Type Science Fiction
popularity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
1960
1970
1980
1990
2000
2010
release decade

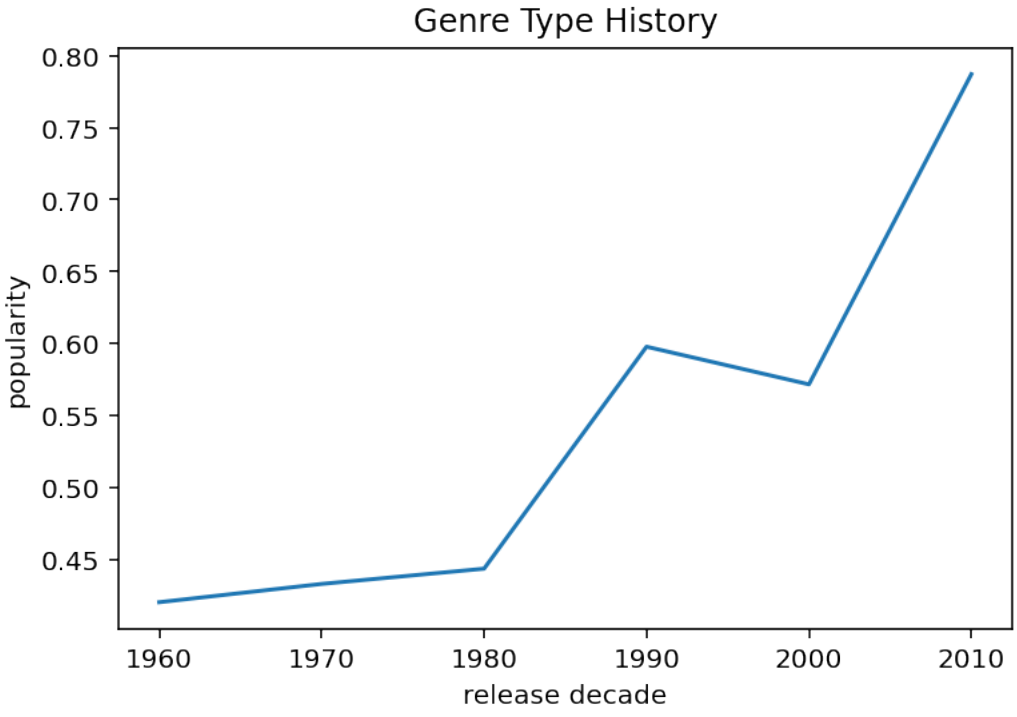

Genre Type Fantasy
1.4
1.2
1.0
0.8
0.6
0.4
popularity
1960
1970
1980
1990
2000
2010
release decade

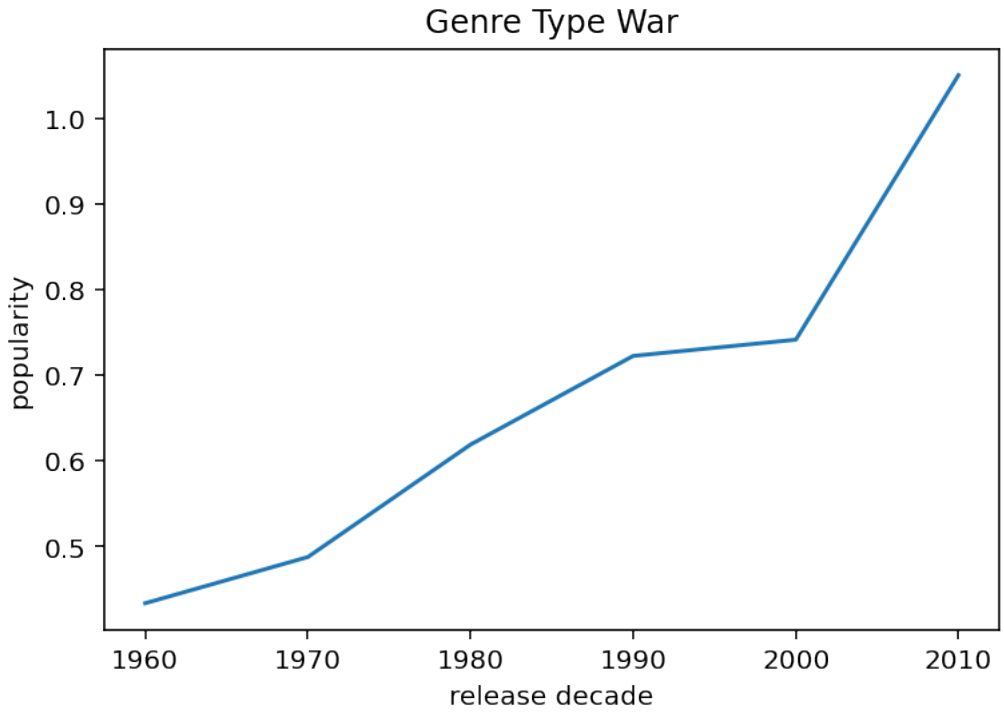

Genre Type Mystery
0.8
0.7
0.6
0.5
0.4
popularity
1960
1970
1980
1990
2000
2010
release decade

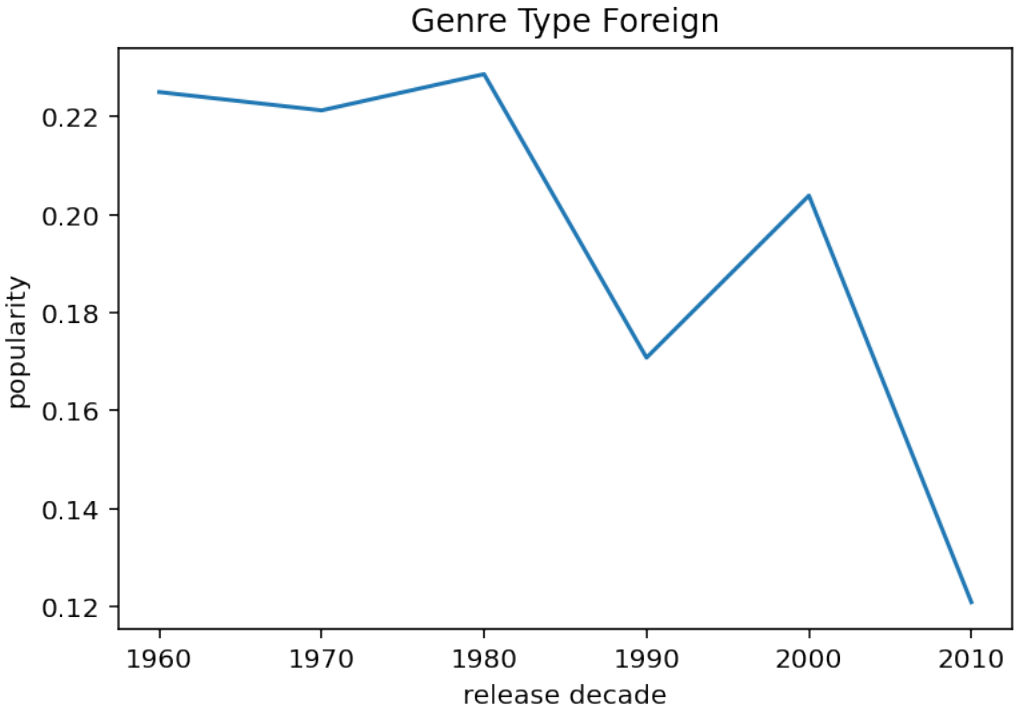

Genre Type Animation
popularity
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
1960
1970
1980
1990
2000
2010
release decade

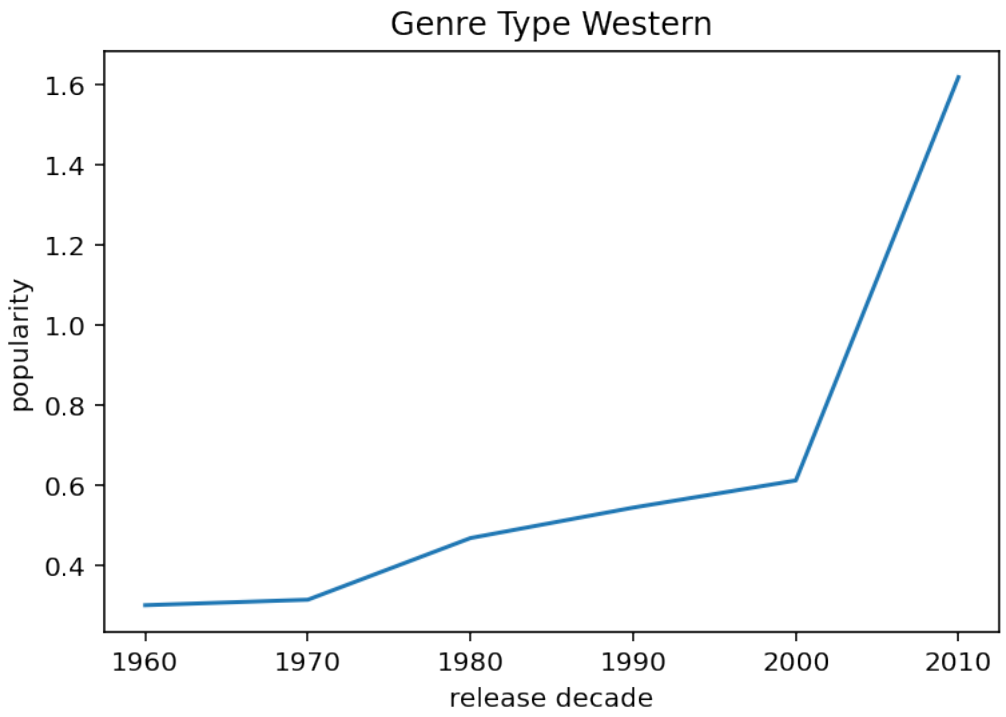

Genre Type Documentary
popularity
0.205
0.200
0.195
0.190
0.185
0.180
0.175
0.170
0.165
1960
1970
1980
1990
2000
2010
release decade

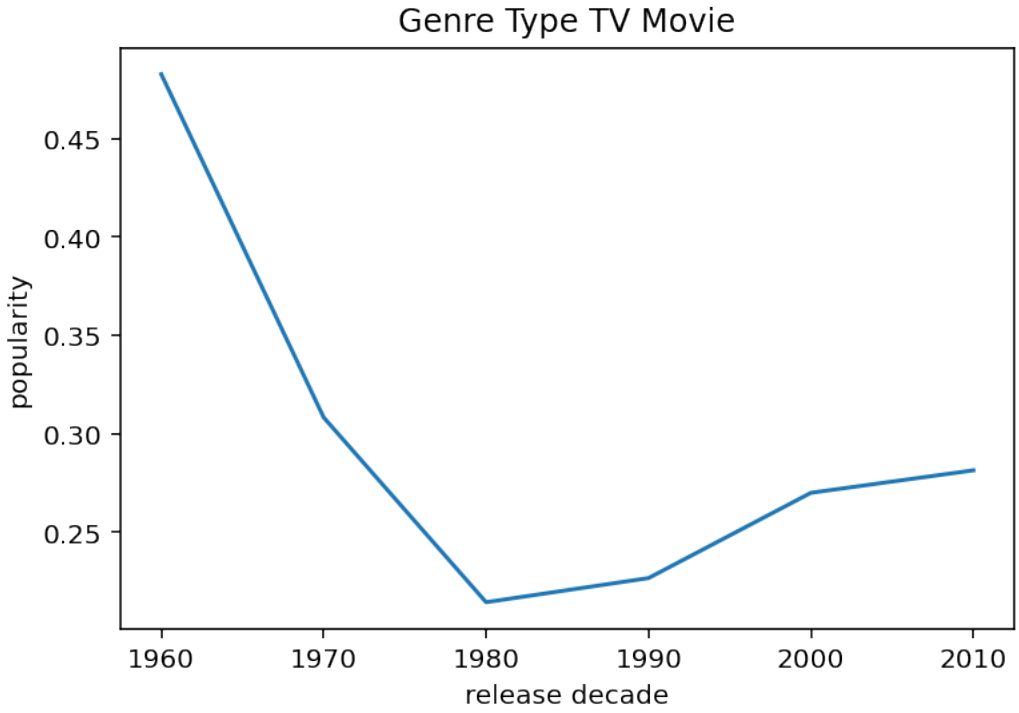

Genre Type Music
0.52
0.50
0.48
0.46
0.44
0.42
0.40
0.38
popularity
1960
1970
1980
1990
2000
2010
release decade


Genre Type History
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
popularity
1960
1970
1980
1990
2000
2010
release decade

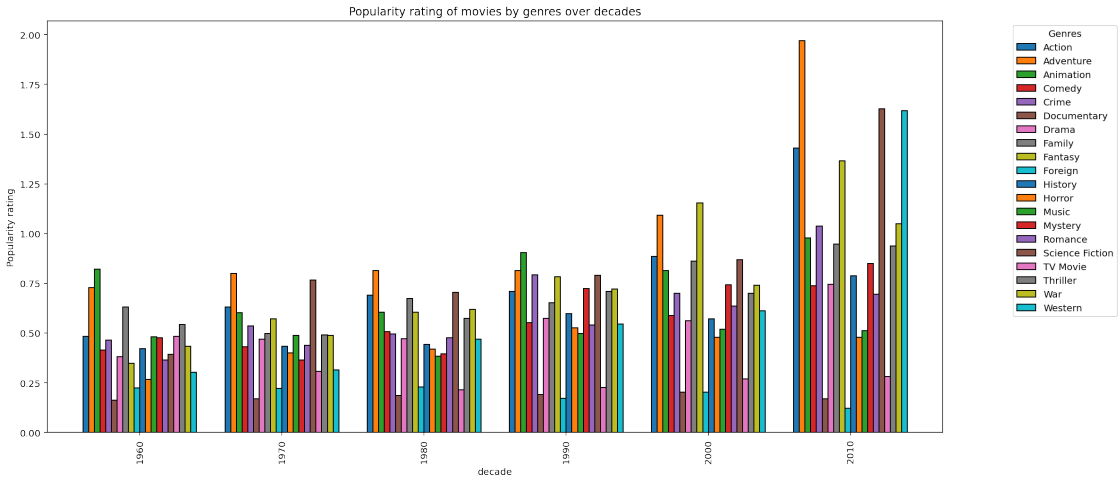

Genre Type War
popularity
1.0
0.9
0.8
0.7
0.6
0.5
1960
1970
1980
1990
2000
2010
release decade


Genre Type Foreign
popularity
0.22
0.20
0.18
0.16
0.14
0.12
1960
1970
1980
1990
2000
2010
release decade


Genre Type Western
popularity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
1960
1970
1980
1990
2000
2010
release decade


Genre Type TV Movie
popularity
0.45
0.40
0.35
0.30
0.25
1960
1970
1980
1990
2000
2010
release decade

[283]:
```
df_genres = df_decade.pivot("release_decade", "genres", "popularity")
X = np.arange(4)
df_genres.plot(kind='bar',width=0.8,figsize=(17,8), edgecolor = "black")
plt.legend(bbox_to_anchor=(1.2,1), loc='upper right', ncol=1,title="Genres")
plt.title("Popularity rating of movies by genres over decades")
plt.xlabel("decade")
plt.ylabel("Popularity rating")
plt.show()
```

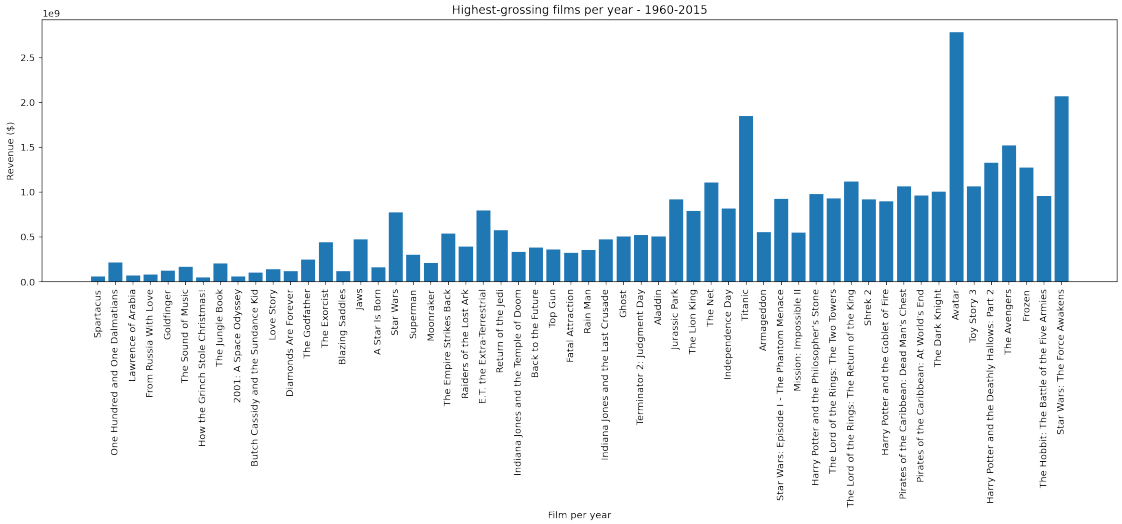


According to the plots in the above cells Drama,Comedy,Thriller,Action,Romance,Horror,Adventure,Crime,Science Fiction,Family,Fantasy,History,War,Western genre types have positive slope that means popularity is increased.For Mystery,Animation type genre movies in first decade popularity got decreased but after 1970 it gained its popularity and continued to increase every decade.Documentary genre type movies popularity increased every decade from 1960 to 2000 but in 2010 its popularity decreased.For Music type movies popularity is fluctuating every decade.Foreign,TV Movie popularity got decreased every decade.

1.1.3 Research Question 2: What properties are associated with highly profitable movies?

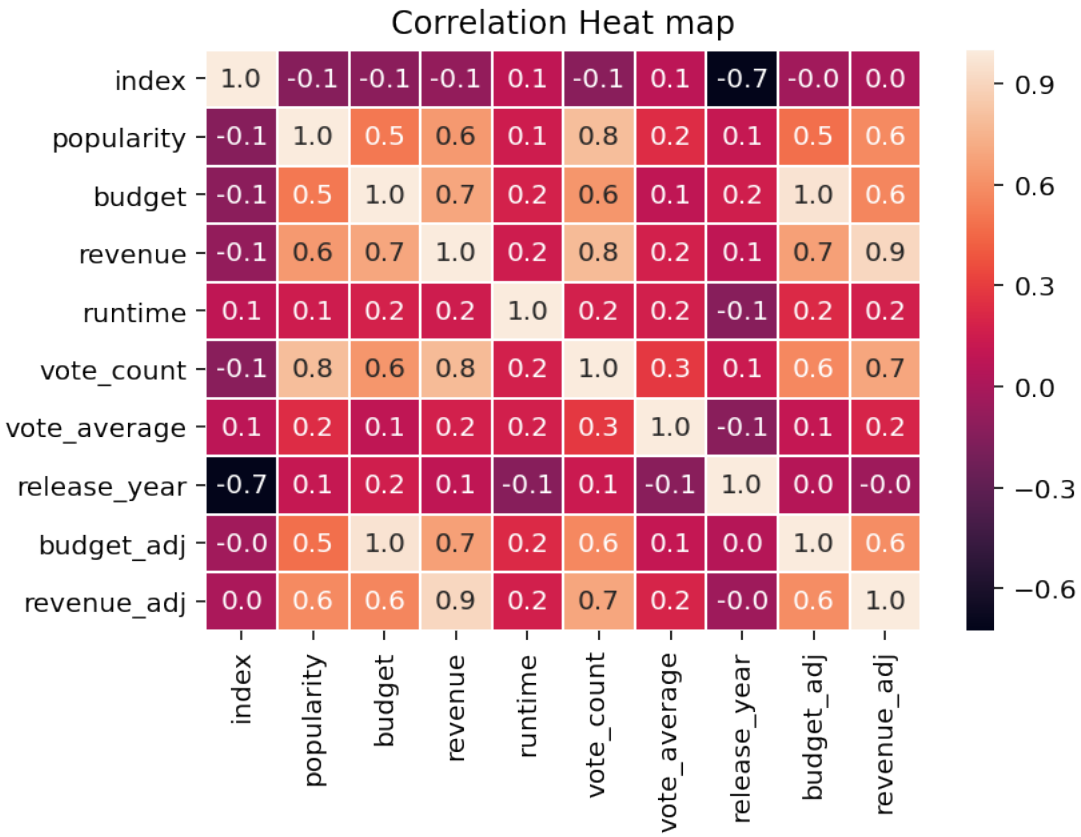
[284]:
```
df_film_max = df.loc[df.groupby('release_year')['revenue'].idxmax()]
```

[285]:
```
plt.figure(figsize=(20,5))
plt.bar(df_film_max.original_title, df_film_max.revenue)
plt.xticks(df_film_max.original_title, rotation = 90)
plt.xlabel('Film per year')
plt.ylabel('Revenue ($)')
plt.title('Highest-grossing films per year - 1960-2015')
plt.show()
```

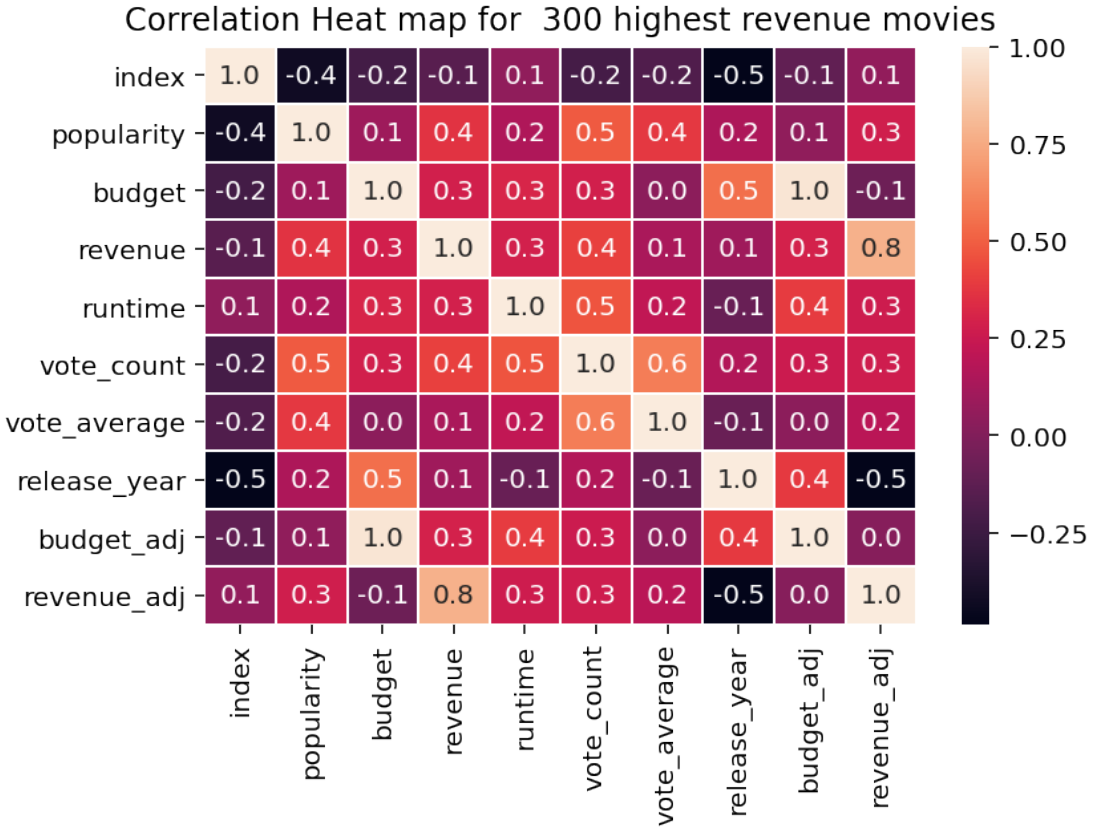


Above plot shows the HIghest grossing films over the years. Avatar stood first in collecting revenue, second is Starwars and third is Titanic.

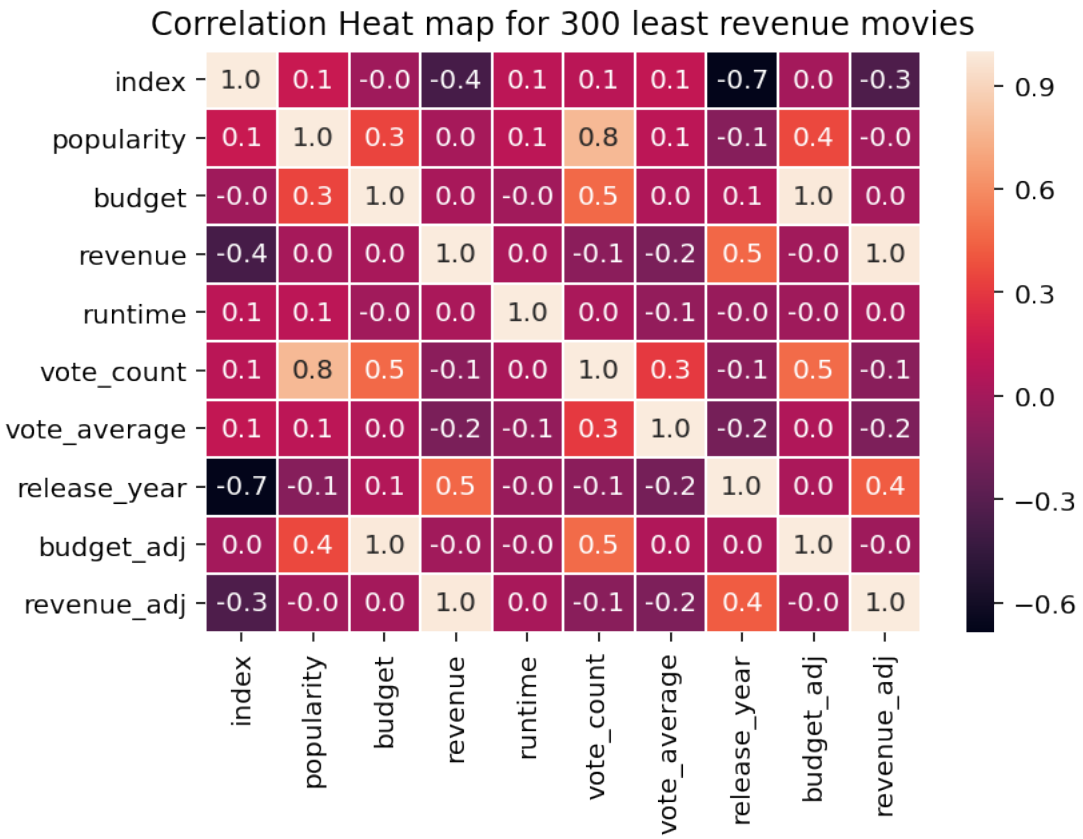
[286]:
```
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt='.1f');
plt.title("Correlation Heat map");
```



[287]:
```
sns.heatmap(df.sort_values(by = 'revenue', ascending = False).head(300).
 ↪corr(),annot=True, linewidths=.5, fmt='.1f');
plt.title("Correlation Heat map for  300 highest revenue movies");
```



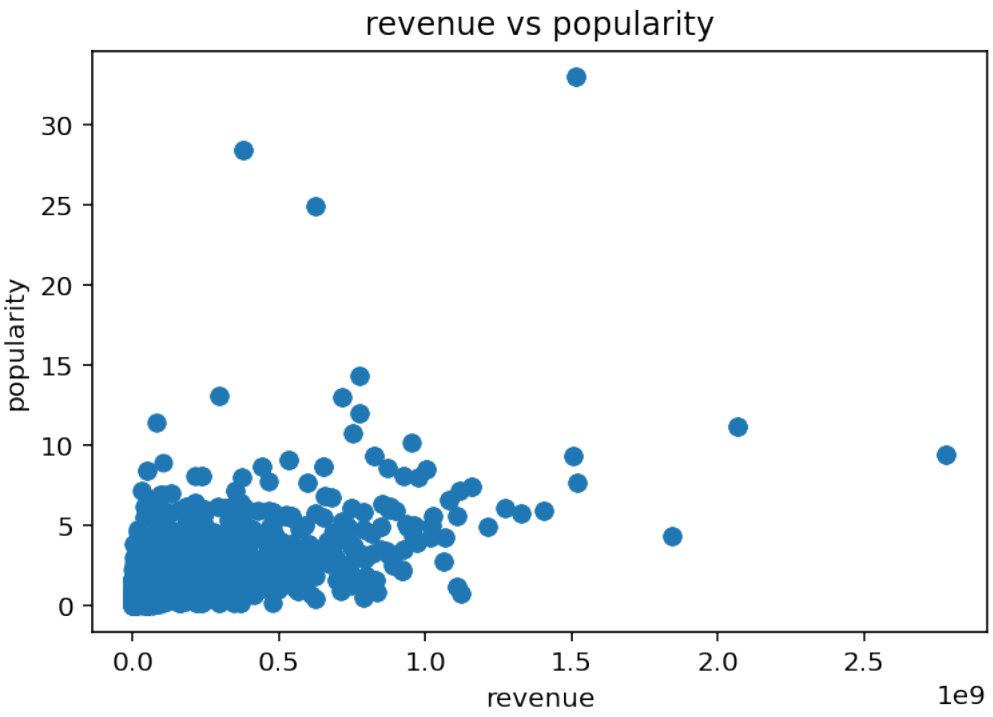
[288]:
```
sns.heatmap(df.sort_values(by = 'revenue', ascending = True).head(300).
 ↪corr(),annot=True, linewidths=.5, fmt='.1f');
plt.title("Correlation Heat map for 300 least revenue movies");
```

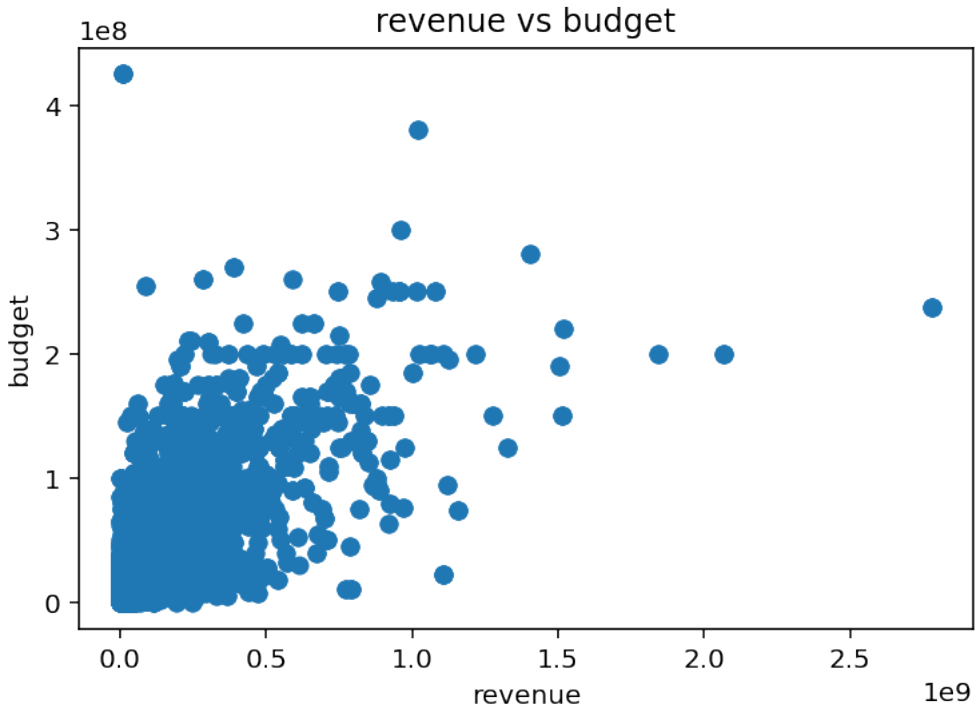


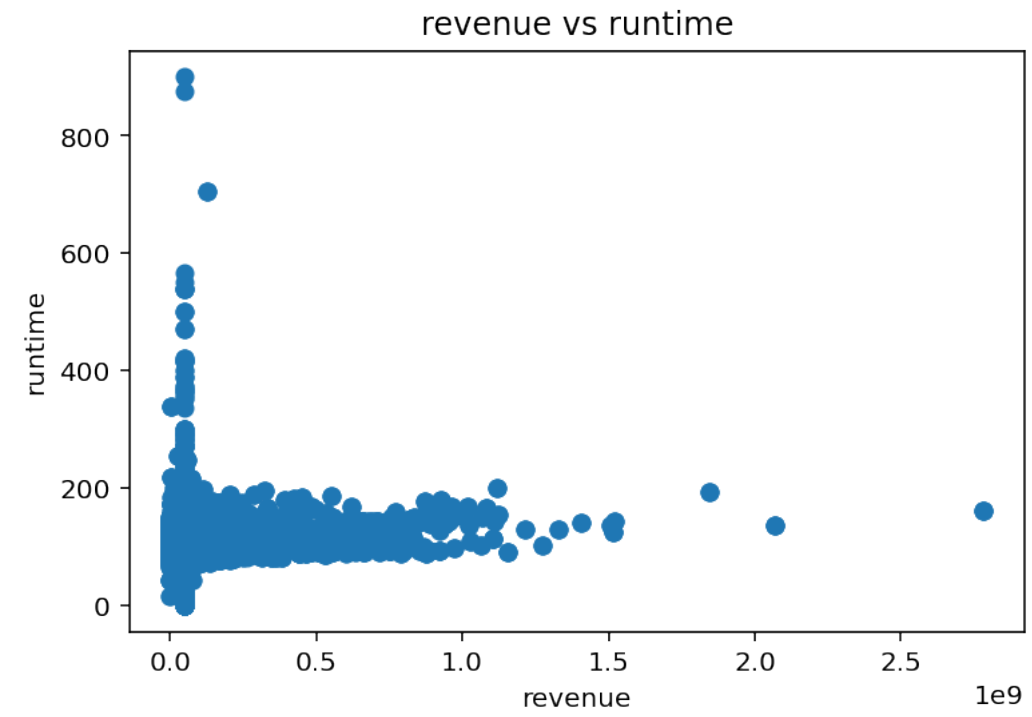
The Above three Plots Shows the corealtion between the columns. 1st plot shows overall correlation 2nd plot shows the correaction for 300 heighest revenue movies. 3rd plot shows the correaction for 300 lowest revenue movies.

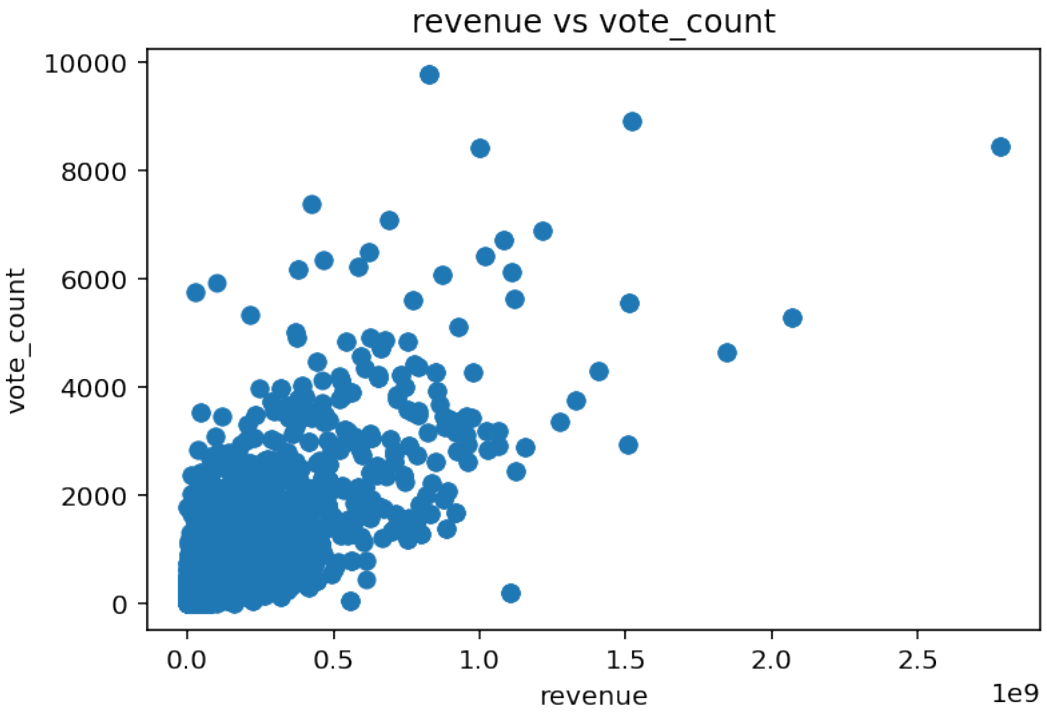
[289]:
```
def scat(data):
    x = df["revenue"]
    y = df[data]
    plt.scatter(x, y)
    plt.title("revenue vs "+data)
    plt.xlabel("revenue")
    plt.ylabel(data)
    plt.show()
```

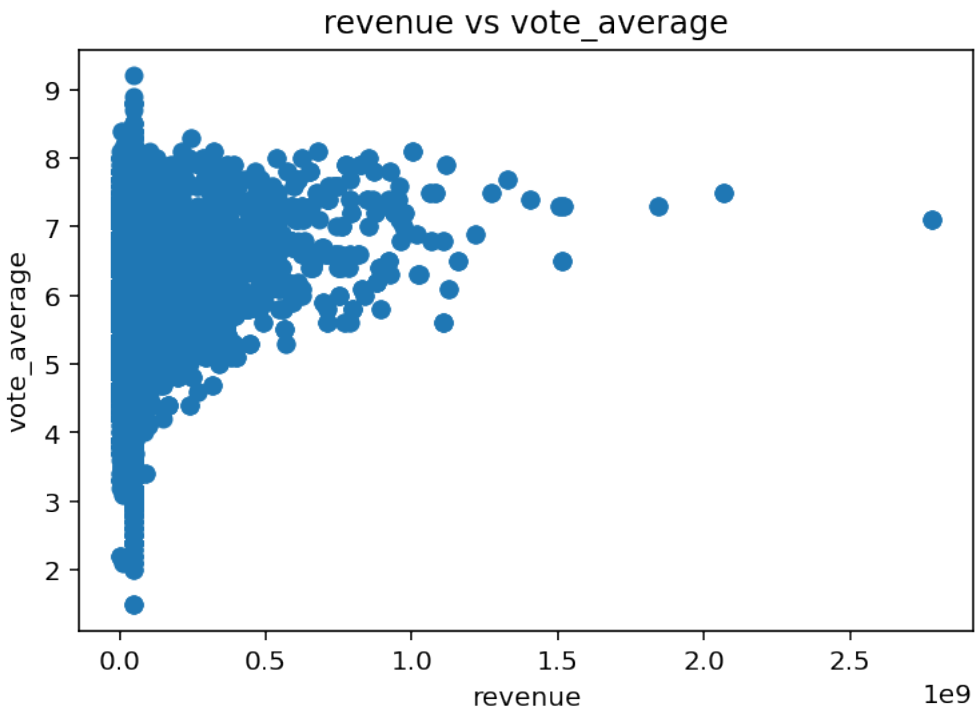
[290]:
```
numerics = ['int64','float64']
columns = df.drop(columns=["index","revenue"]).select_dtypes(include=numerics).
 ↪columns
j = 1
for i in columns:
    scat(i)
```

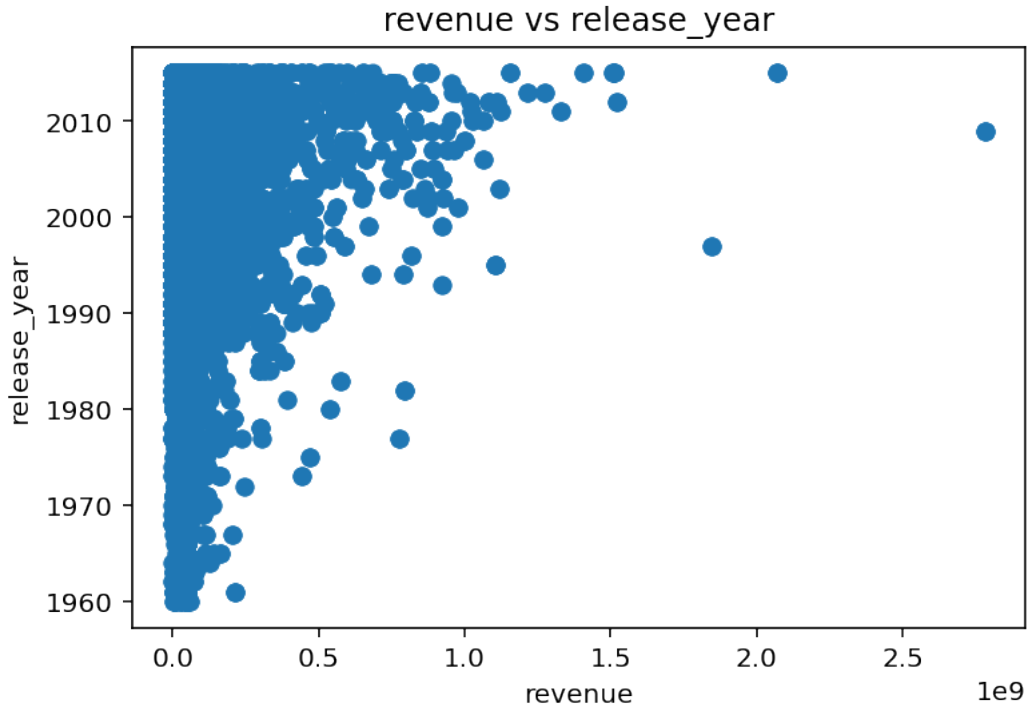

revenue vs popularity
popularity
30
25
20
15
10
5
0
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9

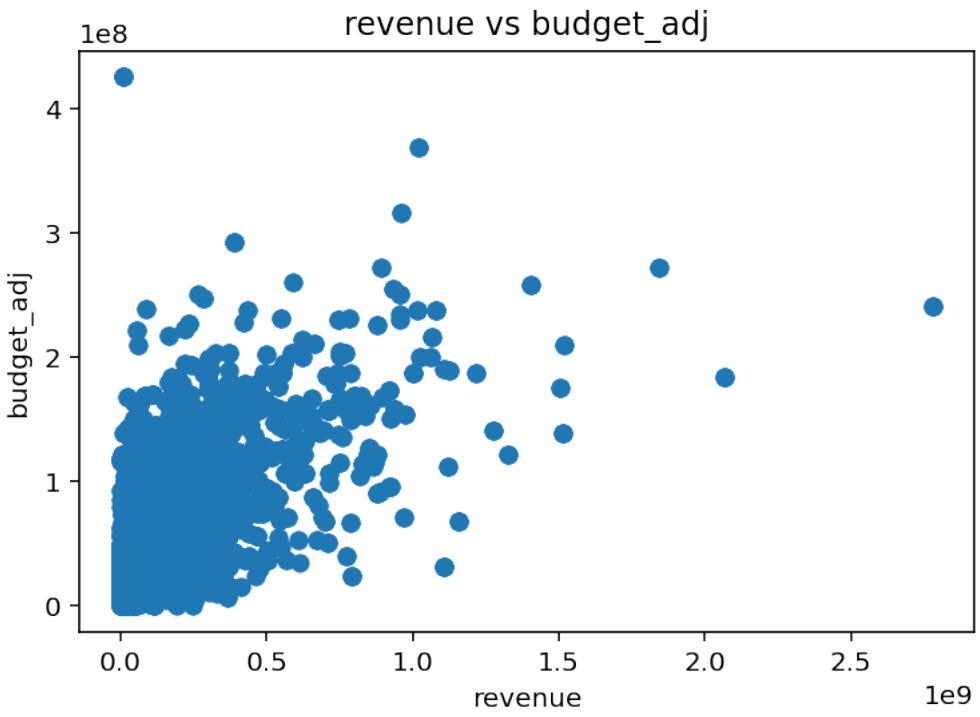

revenue vs budget
1e8
budget
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9

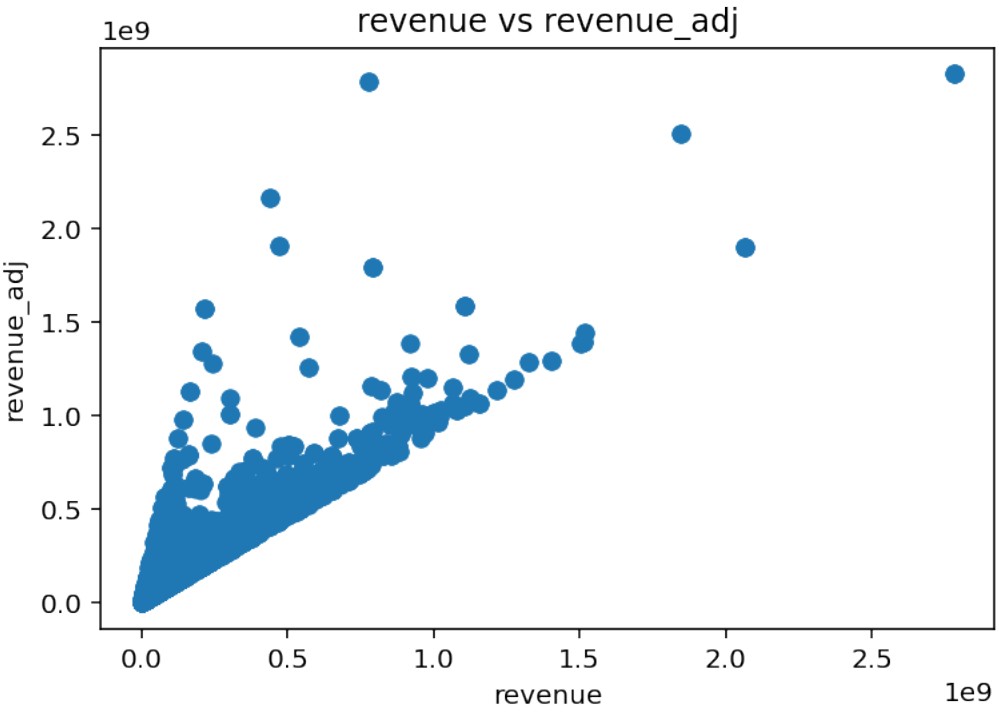

revenue vs runtime
runtime
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9


revenue vs vote_count
vote_count
10000
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9


revenue vs vote_average
vote_average
9
8
7
6
5
4
3
2
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9


revenue vs release_year
release_year
2010
2000
1990
1980
1970
1960
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9

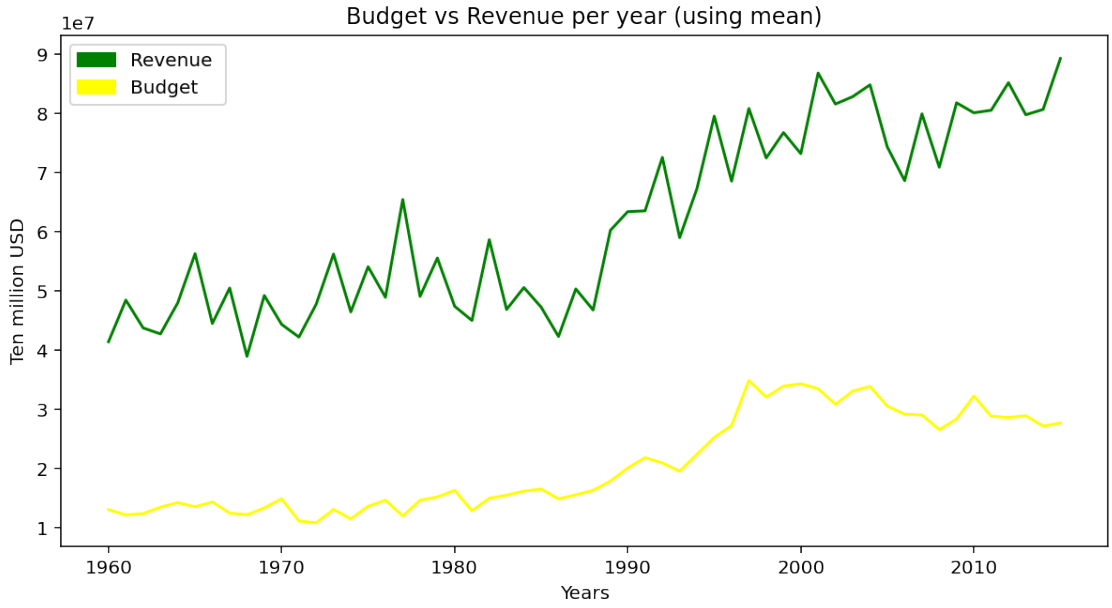

revenue vs budget_adj
1e8
4
3
2
1
0
budget_adj
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9


revenue vs revenue_adj
1e9
2.5
2.0
1.5
1.0
0.5
0.0
revenue_adj
0.0
0.5
1.0
1.5
2.0
2.5
revenue
1e9

The above scatter plots shows the correlation between revenue and other columns. columns Popularity, budget, runtime, vote_count, vote average,release_year have positive correlation with revenue.

[291]:
```
df_budgetrev = df.groupby(['release_year'])['budget','revenue'].mean()
plt.figure(figsize=(10,5))

p1,=plt.plot(df_budgetrev.index, df_budgetrev.budget, color='yellow')
p2, = plt.plot(df_budgetrev.index, df_budgetrev.revenue, color='green')
plt.xlabel('Years')
plt.ylabel('Ten million USD')
plt.title('Budget vs Revenue per year (using mean)')
#plt.legend([p1, p2], ['budget','Revenue'])
p = mpatches.Patch(color='yellow', label='Budget')
q = mpatches.Patch(color='green', label='Revenue ')
plt.legend(handles=[q,p])
plt.show()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```



columns Popularity, budget, runtime, vote_count, vote average,release_year have positive correlation with revenue.

Conclusions In the first section I examined the popularity over the decades. I made my analysis based on the values of 'released_year' and 'popularity' for different genres.

After that I analyzed the ratings of the most and least revenue movies and I found out that the movies with higher revenue got higher votes than the cheaper ones.

1.2 Limitations

Some categorical level have insufficient data to support insights, for example, the collected data amount from different genres and years is not equally distributed. Our insights are likely to be biased based on such small number of records. To safely handle outliers or missing values, we should first have sufficient domain knowledge or evidence to understand the reason behind these mistakes. For example, if data provider clarifies that high revenue,budget are not mistake, we cannot just discard them from our dataset.

[ ]: