Project: Investigate a Dataset - [TMDb movie data]

Table of Contents

Introduction

Dataset Description

> There are a lot of movies being produced every year. In this data set we have data about **10,000** movies in different genres and kinds wich is based on The Movie Database (TMDb) .

- movie_id - A unique identifier for each movie.
- cast - The name of lead and supporting actors.
- crew - The name of Director, Editor, Composer, Writer etc.
- budget - The budget in which the movie was made.
- genre - The genre of the movie, Action, Comedy ,Thriller etc.
- homepage - A link to the homepage of the movie.
- id - This is infact the movie_id as in the first dataset.
- keywords - The keywords or tags related to the movie.
- original_title - The title of the movie before translation or adaptation.
- overview - A brief description of the movie.
- popularity - A numeric quantity specifying the movie popularity.
- production_companies - The production house of the movie.
- production_countries - The country in which it was produced.
- release_date - The date on which it was released.
- revenue - The worldwide revenue generated by the movie.
- runtime - The running time of the movie in minutes.
- status - "Released" or "Rumored".
- title - Title of the movie.
- vote_average - average ratings the movie recieved.
- vote_count - the count of votes recieved.

Question(s) for Analysis

1. How movies production improved throw years ?
2. what is the highest movie genre type ?
3. Does movies cost more than before ?
4. Which director has most number of movies ?

In [1]:
```
# Use this cell to set up import statements for all of the packages that you
#   plan to use.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
%matplotlib inline
```

Data Wrangling

> In these section we will see how we can anwer qusetions and what we have in the dataset how we can use it

General Properties

In [2]:
```
# Load your data and print out a few lines. Perform operations to inspect data
#   types and look for instances of missing or possibly errant data.
df = pd.read_csv('tmdb-movies.csv')
#preview a few row of data
df.head(5)
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	...
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	...
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	...
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams	Every generation has a story.	...
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Vengeance Hits Home	...

5 rows × 21 columns

In [3]:
```
#print a small summary of dataset , types and if there is any NaN valus
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
```

In [4]:
```
#print summary about numiric data
df.describe()
```

Out[4]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086600e+04
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658	1.755104e+07	5.136436e+07
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941	3.430616e+07	1.446325e+08
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+00
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+00
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+00
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000	2.085325e+07	3.369710e+07
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

In [5]:
```
#there is null valus but how many and where
df.isnull().sum()
```

Out[5]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

In [6]:
```
#check for duplicated rows
df.duplicated().sum()
```

Out[6]: 1

In [7]:
```
#how many unique values
df['release_year'].unique()
```

Out[7]:
```
array([2015, 2014, 1977, 2009, 2010, 1999, 2001, 2008, 2011, 2002, 1994,
       2012, 2003, 1997, 2013, 1985, 2005, 2006, 2004, 1972, 1980, 2007,
       1979, 1984, 1983, 1995, 1992, 1981, 1996, 2000, 1982, 1998, 1989,
       1991, 1988, 1987, 1968, 1974, 1975, 1962, 1964, 1971, 1990, 1961,
       1960, 1976, 1993, 1967, 1963, 1986, 1973, 1970, 1965, 1969, 1978,
       1966])
```

In [ ]:

In [ ]:

Data Cleaning

1. Remove columns that we will not use
2. Remove any duplicated row
3. edit release date to be usable
4. dropna values from genere column
5. make new column wich contain budget without 0

In [8]:
```
#make varible refer to budget without 0
budget_v2 = df[df['budget'] > 0]
```

In [9]:
```
#dropna values from genere column
df.dropna(how='any', subset=['genres'], inplace=True)
```

In [10]:
```
# After discussing the structure of the data and any problems that need to be
#   cleaned, perform those cleaning steps in the second part of this section.
#using drop to remove columns that we dont need
df.drop(['imdb_id','homepage','overview','tagline','cast','keywords','production_companies'],axis =1,inplace = True)
```

In [11]:
```
#remove any duplicated row
df.drop_duplicates(inplace = True)
```

In [12]:
```
#using to_datetime to change type of release_date column
df['release_date'] = pd.to_datetime(df['release_date'])
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10842 entries, 0 to 10865
Data columns (total 14 columns):
id                10842 non-null int64
popularity        10842 non-null float64
budget            10842 non-null int64
revenue           10842 non-null int64
original_title    10842 non-null object
director          10800 non-null object
runtime           10842 non-null int64
genres            10842 non-null object
release_date      10842 non-null datetime64[ns]
vote_count        10842 non-null int64
vote_average      10842 non-null float64
release_year      10842 non-null int64
budget_adj        10842 non-null float64
revenue_adj       10842 non-null float64
dtypes: datetime64[ns](1), float64(4), int64(6), object(3)
memory usage: 1.2+ MB
```

In [14]:
```
df.describe()
```

Out[14]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10842.000000	10842.000000	1.084200e+04	1.084200e+04	10842.000000	10842.000000	10842.000000	10842.000000	1.084200e+04	1.084200e+04
mean	65870.675521	0.647461	1.465531e+07	3.991138e+07	102.138443	217.823649	5.974064	2001.314794	1.758712e+07	5.147797e+07
std	91981.355752	1.001032	3.093971e+07	1.171179e+08	31.294612	576.180993	0.934257	12.813617	3.433437e+07	1.447723e+08
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+00
25%	10589.250000	0.208210	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+00
50%	20557.000000	0.384532	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+00
75%	75186.000000	0.715393	1.500000e+07	2.414118e+07	111.000000	146.000000	6.600000	2011.000000	2.092507e+07	3.387838e+07
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

In [15]:
```
#counting how many rows with 0 value
df[df['budget'] == 0].count()[0]
```

Out[15]: 5674

In [21]:
```
#creating copy
df1 = df.copy()
#set list with columns with zero values
zero_cols=['budget','runtime','revenue','budget_adj','revenue_adj']
#replace zeros with nan
df1[zero_cols]=df1[zero_cols].apply(lambda cols:cols.replace(0,np.nan),axis=1)
df1.describe()
```

Out[21]:

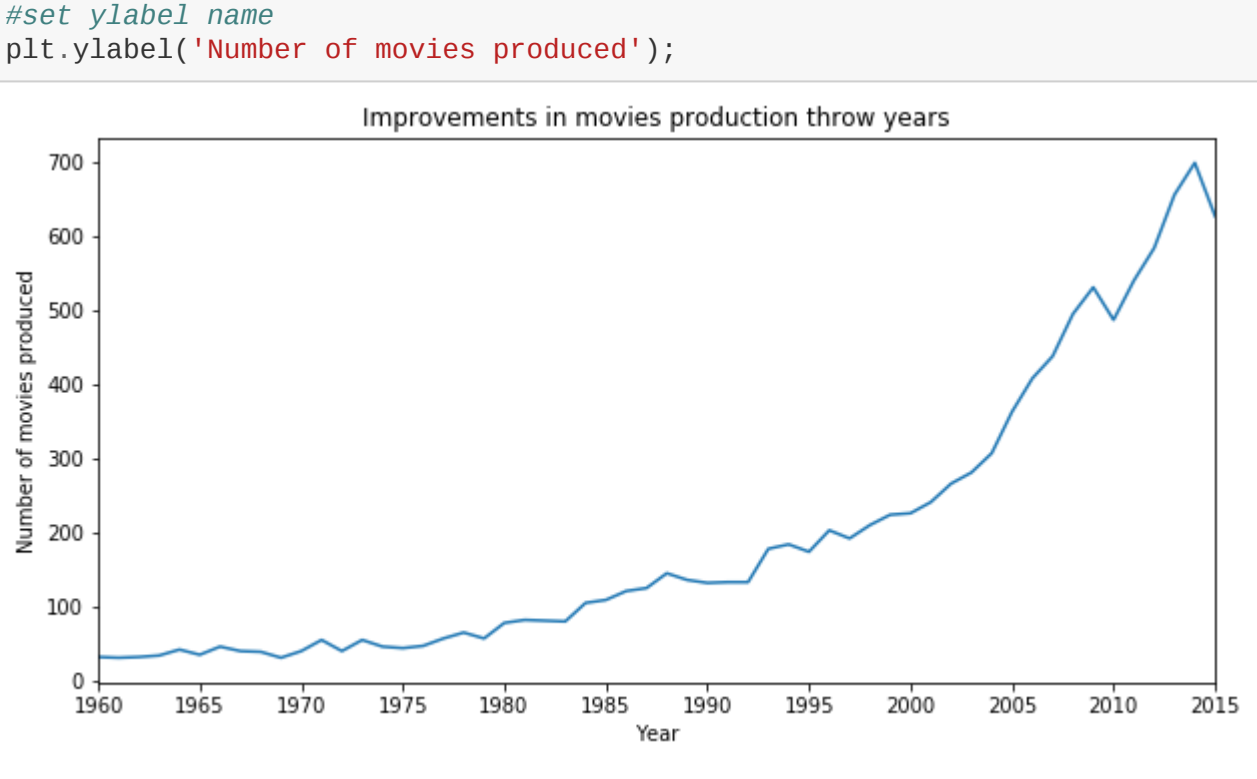
	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10842.000000	10842.000000	5.168000e+03	4.849000e+03	10812.000000	10842.000000	10842.000000	10842.000000	5.168000e+03	4.849000e+03
mean	65870.675521	0.647461	3.074552e+07	8.923886e+07	102.421846	217.823649	5.974064	2001.314794	3.689620e+07	1.151009e+08
std	91981.355752	1.001032	3.890583e+07	1.620801e+08	30.871363	576.180993	0.934257	12.813617	4.196188e+07	1.988557e+08
min	5.000000	0.000065	1.000000e+00	2.000000e+00	2.000000	10.000000	1.500000	1960.000000	9.210911e-01	2.370705e+00
25%	10589.250000	0.208210	6.000000e+06	7.732325e+06	90.000000	17.000000	5.400000	1995.000000	8.102293e+06	1.046585e+07
50%	20557.000000	0.384532	1.700000e+07	3.185308e+07	99.000000	38.000000	6.000000	2006.000000	2.272271e+07	4.395666e+07
75%	75186.000000	0.715393	4.000000e+07	9.996575e+07	112.000000	146.000000	6.600000	2011.000000	5.008384e+07	1.316482e+08
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+09

Exploratory Data Analysis

1- Which year has the highst number of movies produced ?

In [22]:
```
# Use this, and more code cells, to explore your data. Don't forget to add
#   Markdown cells to document your observations and findings.
#set varrible which contain number of movies per year
num_movies = df.groupby('release_year').count()['id']
```

In [23]:
```
num_movies.plot(xticks = np.arange(1960,2016,5),figsize=(10,5))
#set plot title
plt.title('Improvements in movies production throw years')
#set xlabel name
plt.xlabel('Year')
#set ylabel name
plt.ylabel('Number of movies produced');
```

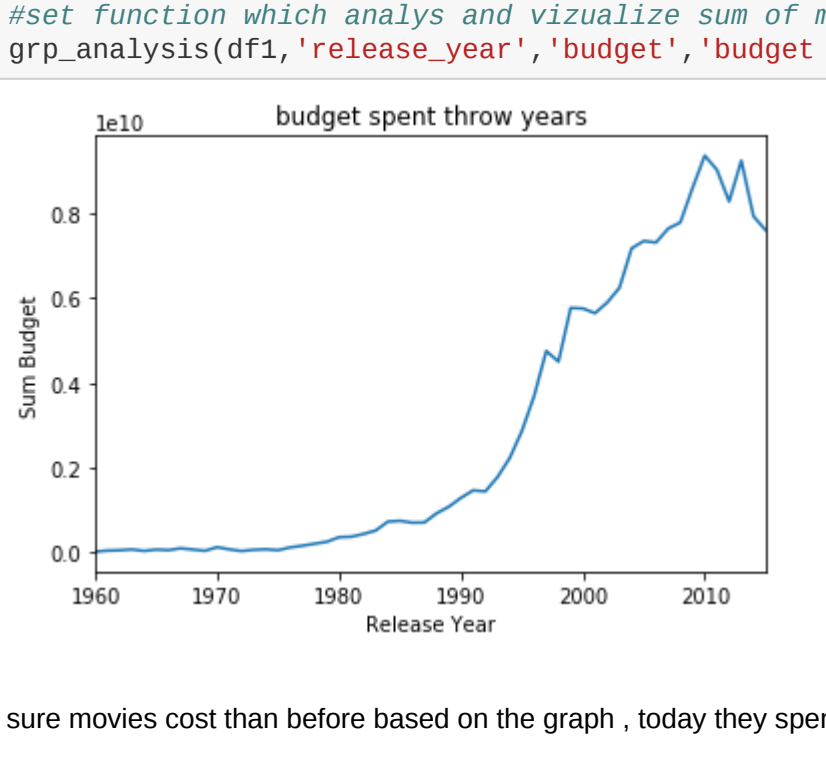


Obviously there is a huge diiferance between nowdays and the past there produced between 0-100 movies from 1960 to 1980 after they produced more until 2016 almost 700 movies per year

2- Does movies cost more than before ?

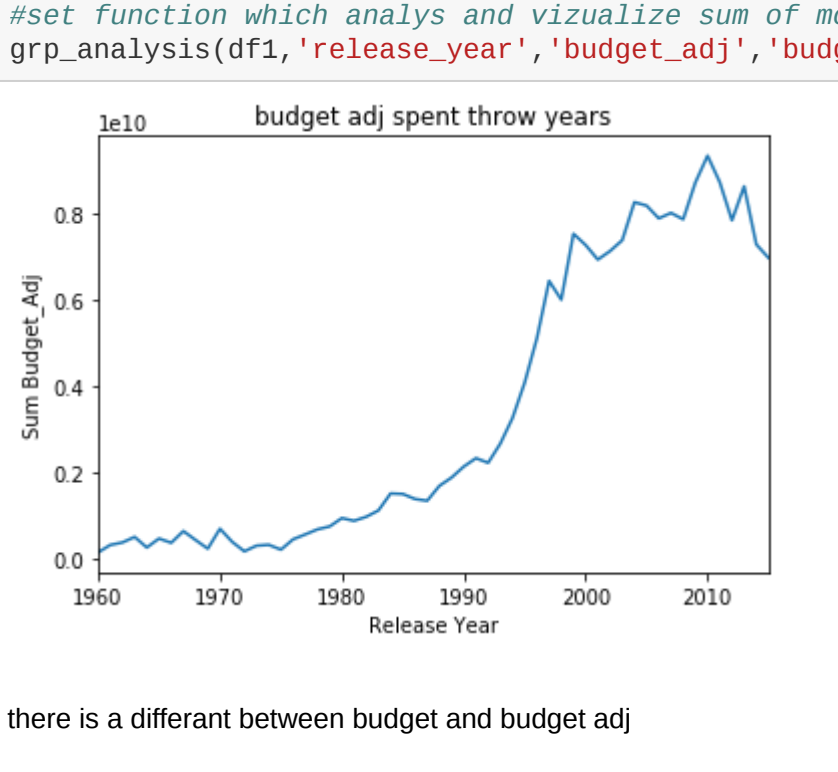
In [24]:
```
#creating fucation to combare bettwen budget_adj and budget
def grp_analysis (df,x,y,t,stat='count'):
    '''
docstring:Thise funcation used to code and vizualise varibls which in dataframe.

df : dataframe name
x : column to gruopby and xlabel
y : column to calculate mean,sum,count....
stat : calculate method
t : refet to title of the plot name

    '''
    #coding section useing groupby function and method to calculate
    df.groupby([x])[y].agg(stat).plot()
    #vizualize out calculation set title of the plot
    plt.title(f'{t}')
    #set x label
    plt.xlabel(x.title().replace('_',' '))
    #set y lablel
    plt.ylabel(f'{stat} {y}'.title());
```

In [25]:
```
#set function which analys and vizualize sum of money has been spent per year
grp_analysis(df1,'release_year','budget','budget spent throw years',stat='sum')
```



sure movies cost than before based on the graph , today they spent more than 90's

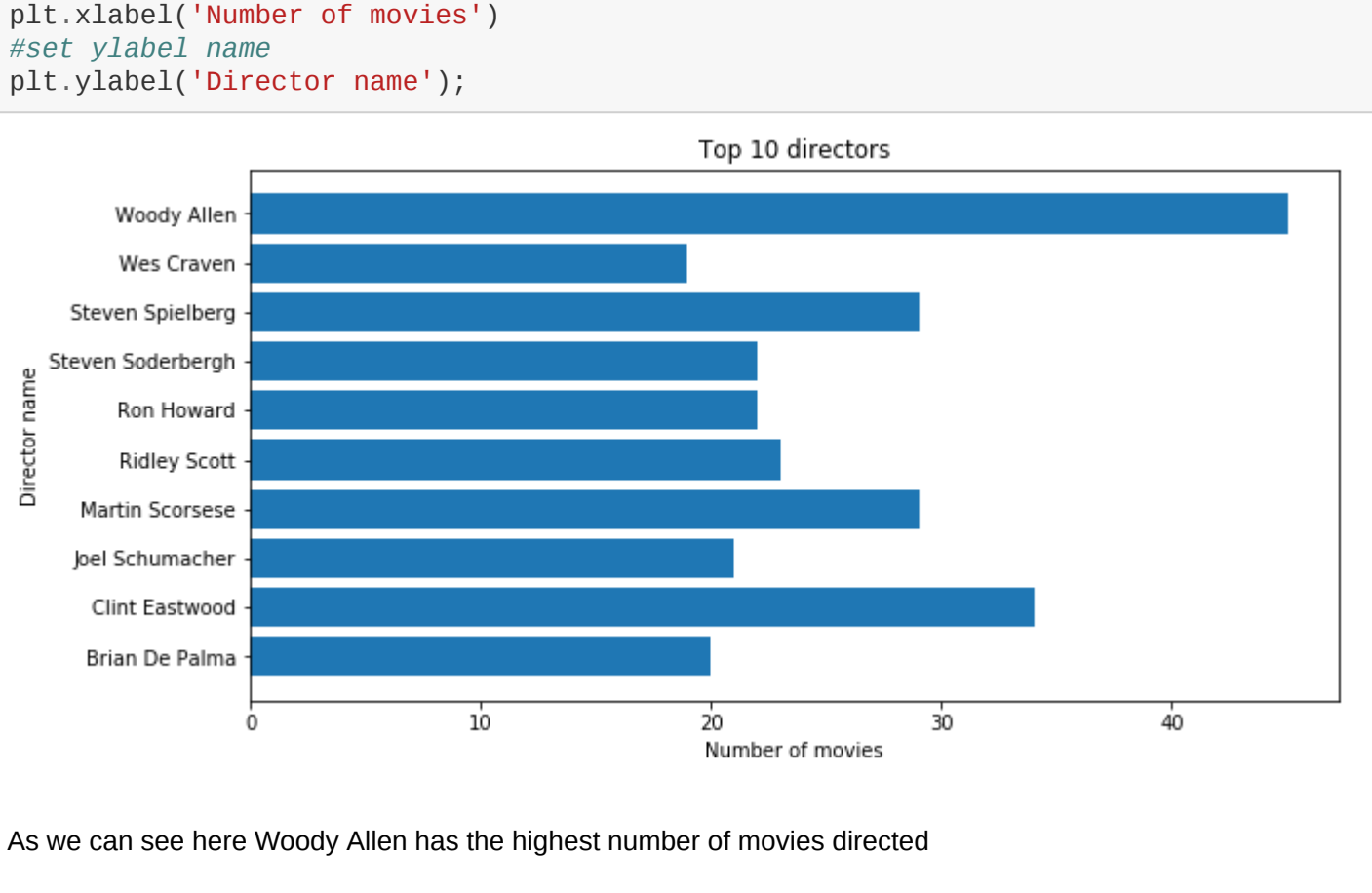
In [26]:
```
#set function which analys and vizualize sum of money has been spent per year
grp_analysis(df1,'release_year','budget_adj','budget adj spent throw years',stat='sum')
```



there is a differant between budget and budget adj

3- Which director has most number of movies ?

In [27]:
```
#set varrible which contain num of movies per director
hight_dic = df.groupby('director').count()['id']
#prepare and get top 10 directors
hight_dic = hight_dic.sort_values(ascending=False)[:10]
fig, ax = plt.subplots(figsize =(10, 5))
ax.barh(hight_dic.index,hight_dic)
#set plot title
plt.title('Top 10 directors')
#set xlabel name
plt.xlabel('Number of movies')
#set ylabel name
plt.ylabel('Director name');
```

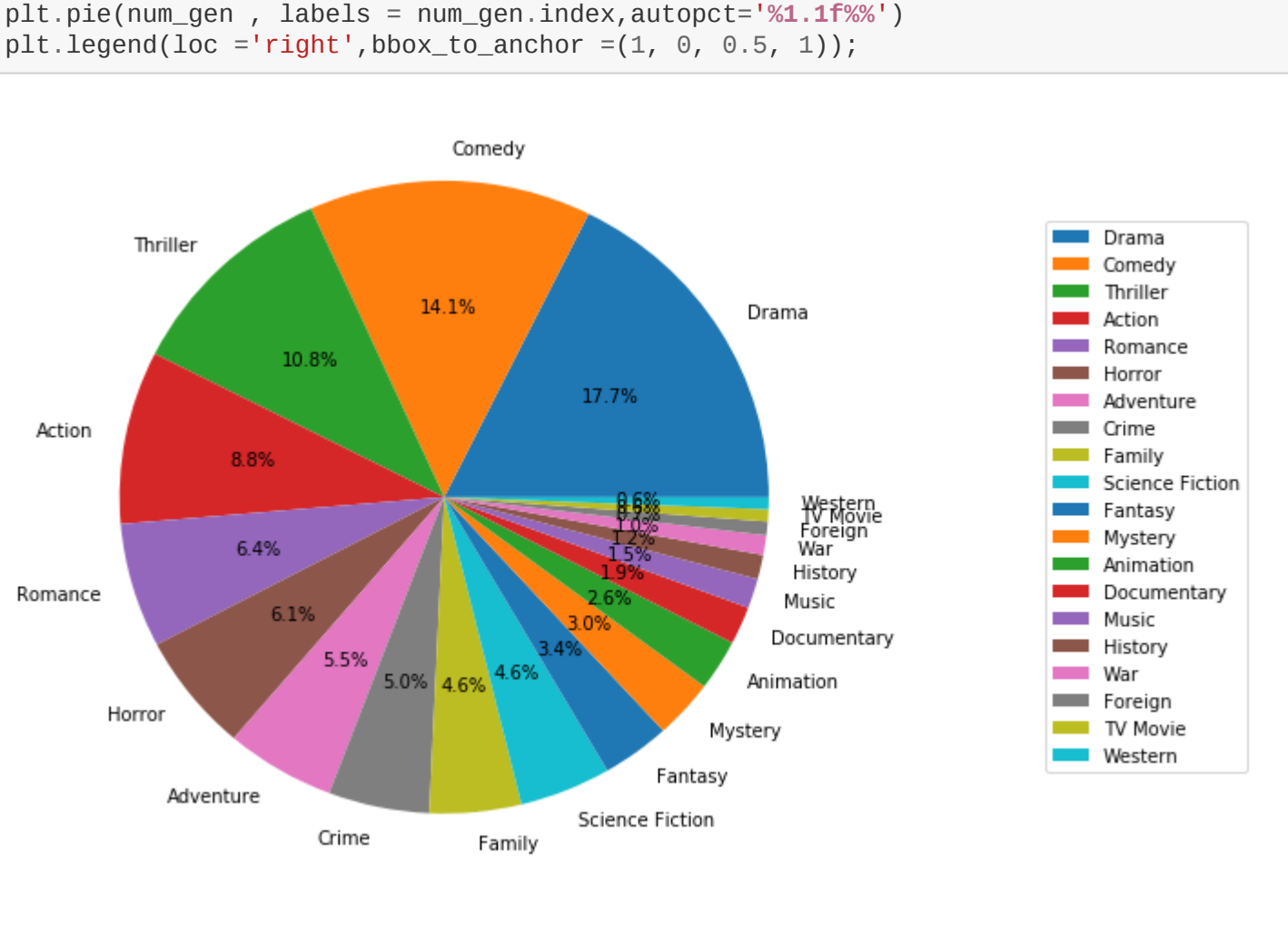


As we can see here Woody Allen has the highest number of movies directed

4- what is the highest movie genre type ?

In [28]:
```
# create a copy of the original dataset
df1 = df.copy()
#split each genre
df1.genres = df1.genres.str.split('|')
#using explode to create new rows which contain 1 genre
df1 = df1.explode('genres')
#set varibale contain number of each genre by percentage
num_gen = df1.genres.value_counts(normalize= True)
num_gen *=100
```

In [30]:
```
# Pie chart
plt.figure(figsize =(8, 8))
#set pie chart values
plt.pie(num_gen , labels = num_gen.index,autopct='%1.1f%%')
plt.legend(loc ='right',bbox_to_anchor =(1, 0, 0.5, 1));
```



based on the chart we found that the highest movies genres is drama by 17%

Conclusions

> After analysis and answering qustion , I've found that movies industry is growing every year also investement in these industry is very effective and profitable , Woody Allen has directed 45 movies also found that producer focusing on drama and comedy genre .

limitations

> > there are inaccurate data , Nan values there are more than 50% of rows contain 0 values that will affect on analysis
> >
> > results wont be accurate in budget column, revenue,budget adj and revenue adj made me remove alot of rows.

Submitting your Project

In [ ]:

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

In [ ]: