In [2]:

```
## importing numpy and pandas
import numpy as np
import pandas as pd
```

In [46]:

```
##reading csv file:
df=pd.read_csv("Movie-Rating.csv")
df.head()
```

Out[46]:

| | Film | Genre | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|---|---|
| **0** | (500) Days of Summer | Comedy | 87 | 81 | 8 | 2009 |
| **1** | 10,000 B.C. | Adventure | 9 | 44 | 105 | 2008 |
| **2** | 12 Rounds | Action | 30 | 52 | 20 | 2009 |
| **3** | 127 Hours | Adventure | 93 | 84 | 18 | 2010 |
| **4** | 17 Again | Comedy | 55 | 70 | 20 | 2009 |

In [64]:

```
# checking last 5 valuesd from the given data:
df.tail()
```

Out[64]:

| | Film | Genre | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|---|---|
| **554** | Your Highness | Comedy | 26 | 36 | 50 | 2011 |
| **555** | Youth in Revolt | Comedy | 68 | 52 | 18 | 2009 |
| **556** | Zodiac | Thriller | 89 | 73 | 65 | 2007 |
| **557** | Zombieland | Action | 90 | 87 | 24 | 2009 |
| **558** | Zookeeper | Comedy | 14 | 42 | 80 | 2011 |

In [6]:

```
# Checking rows and column
df.shape
```

Out[6]:

```
(559, 6)
```

In [7]:

```
# cheking the size or total entries in the given dataset:
df.size
```

Out[7]:

```
3354
```

In [8]:

```
# checking detaied info of the given dataset:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Film                       559 non-null    object
 1   Genre                      559 non-null    object
 2   Rotten Tomatoes Ratings %  559 non-null    int64 
 3   Audience Ratings %         559 non-null    int64 
 4   Budget (million $)         559 non-null    int64 
 5   Year of release            559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

In [9]:

```
# checking total number of missing values
df.isnull().sum()
```

Out[9]:

```
Film                         0
Genre                        0
Rotten Tomatoes Ratings %    0
Audience Ratings %           0
Budget (million $)           0
Year of release              0
dtype: int64
```

- in a above case there is no missing val
- if there is any missing values in the case then we xcan handle it like
- by using s.isnull() we can see the there is any missing values are present or not
- then we can use s.isnul().sum() to count the how many missing val are there.
- then we can find their missing values by col or by features name
- then by using s.filna() methodwe can fill he missing values by backword and forword(s.fillna(method="ffill"))
- then we can fill tha all values by zero or we can use catagorical values
- or we can drop the missing values col by using s.dropna method

In [11]:

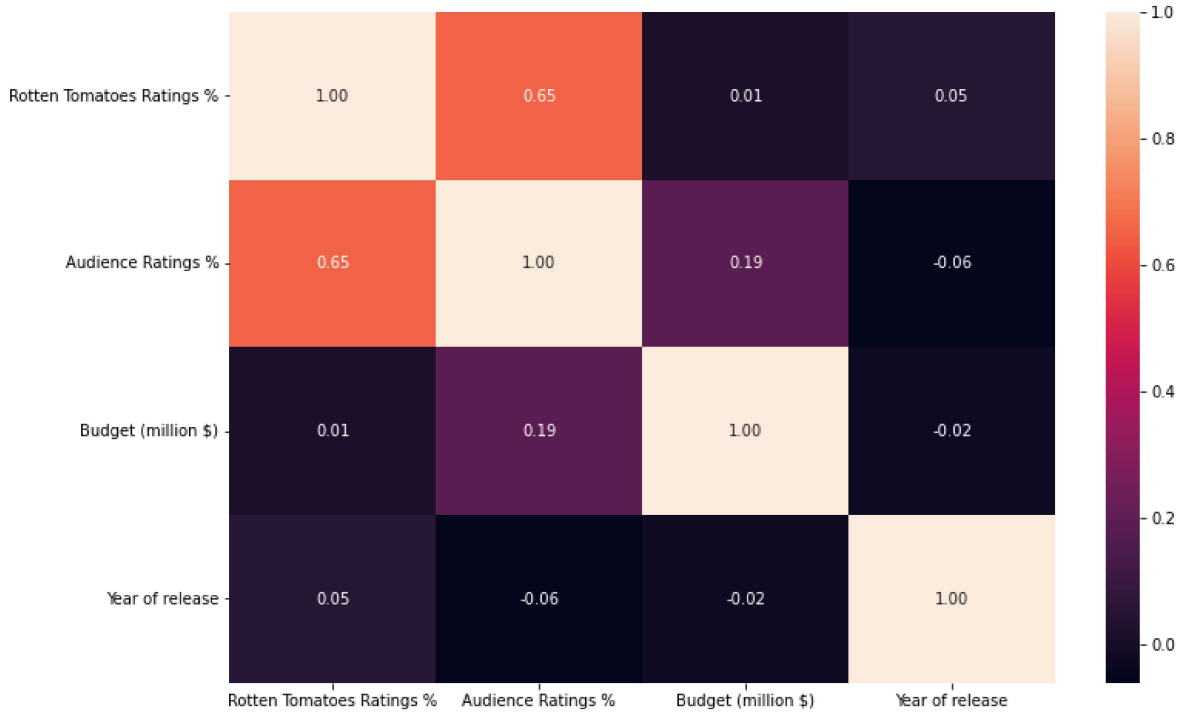
```
# checking corelation by using .corr():
df.corr(method='pearson')
```

Out[11]:

| | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|
| **Rotten Tomatoes Ratings %** | 1.000000 | 0.654803 | 0.014071 | 0.050674 |
| **Audience Ratings %** | 0.654803 | 1.000000 | 0.191108 | -0.060985 |
| **Budget (million $)** | 0.014071 | 0.191108 | 1.000000 | -0.019622 |
| **Year of release** | 0.050674 | -0.060985 | -0.019622 | 1.000000 |

In [49]:

```
fig=plt.figure(figsize=(12,8))
ax = sns.heatmap(df.corr(), annot=True, fmt=".2f")
```



In [12]:

```
df.describe()
```

Out[12]:

| | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|
| **count** | 559.000000 | 559.000000 | 559.000000 | 559.000000 |
| **mean** | 47.309481 | 58.744186 | 50.236136 | 2009.152057 |
| **std** | 26.413091 | 16.826887 | 48.731817 | 1.362632 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 2007.000000 |
| **25%** | 25.000000 | 47.000000 | 20.000000 | 2008.000000 |
| **50%** | 46.000000 | 58.000000 | 35.000000 | 2009.000000 |
| **75%** | 70.000000 | 72.000000 | 65.000000 | 2010.000000 |
| **max** | 97.000000 | 96.000000 | 300.000000 | 2011.000000 |

In [11]:

```
# cheking mean value from the given dataset:
df.mean()
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_11180/3698961737.py:1: FutureWar
ning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Sel
ect only valid columns before calling the reduction.
  df.mean()
```

Out[11]:

```
Rotten Tomatoes Ratings %      47.309481
Audience Ratings %             58.744186
Budget (million $)             50.236136
Year of release              2009.152057
dtype: float64
```

In [12]:

```
# cheking median values values fro the given dataset:
df.median()
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_11180/530051474.py:1: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_onl
y=None') is deprecated; in a future version this will raise TypeError.  Sele
ct only valid columns before calling the reduction.
  df.median()
```

Out[12]:

```
Rotten Tomatoes Ratings %      46.0
Audience Ratings %             58.0
Budget (million $)             35.0
Year of release              2009.0
dtype: float64
```

In [52]:

```
df.groupby('Year of release').sum()
```

Out[52]:

| Year of release | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) |
|---|---|---|---|
| **2007** | 3931 | 5247 | 4571 |
| **2008** | 5415 | 7005 | 5726 |
| **2009** | 5231 | 6430 | 5581 |
| **2010** | 5422 | 6600 | 5997 |
| **2011** | 6447 | 7556 | 6207 |

In [56]:

```
df.groupby('Genre').sum()
```

Out[56]:

| Genre | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|
| **Action** | 6838 | 9043 | 13033 | 309418 |
| **Adventure** | 1540 | 1819 | 2363 | 58256 |
| **Comedy** | 7726 | 9702 | 6211 | 345560 |
| **Drama** | 5704 | 6507 | 2813 | 202937 |
| **Horror** | 1694 | 2322 | 1062 | 98447 |
| **Romance** | 817 | 1084 | 632 | 36175 |
| **Thriller** | 2127 | 2361 | 1968 | 72323 |

In [15]:

```
## cheking the total values counts:
df["Genre"].value_counts()
```

Out[15]:

```
Comedy       172
Action       154
Drama        101
Horror        49
Thriller      36
Adventure     29
Romance       18
Name: Genre, dtype: int64
```

# EDA :

- **Number of movies release in each year:**

In [13]:

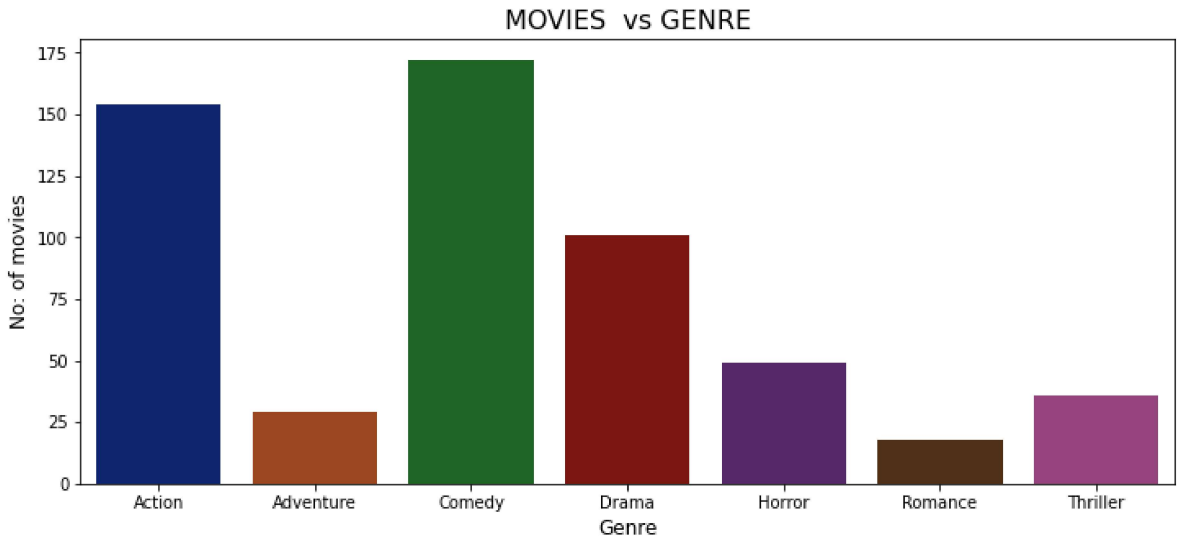
```
## importing some imp library witch is usefull to graph plotting:
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [14]:

```
plt.figure(figsize=(12,5))
sns.countplot(x=df['Genre'],order=df['Genre'].value_counts().index.sort_values(),palette='d
plt.xlabel('Genre',fontsize=12)
plt.ylabel('No: of movies',fontsize=12)
plt.title('MOVIES  vs GENRE',fontsize=16)
```

Out[14]:

```
Text(0.5, 1.0, 'MOVIES  vs GENRE')
```




Observations:

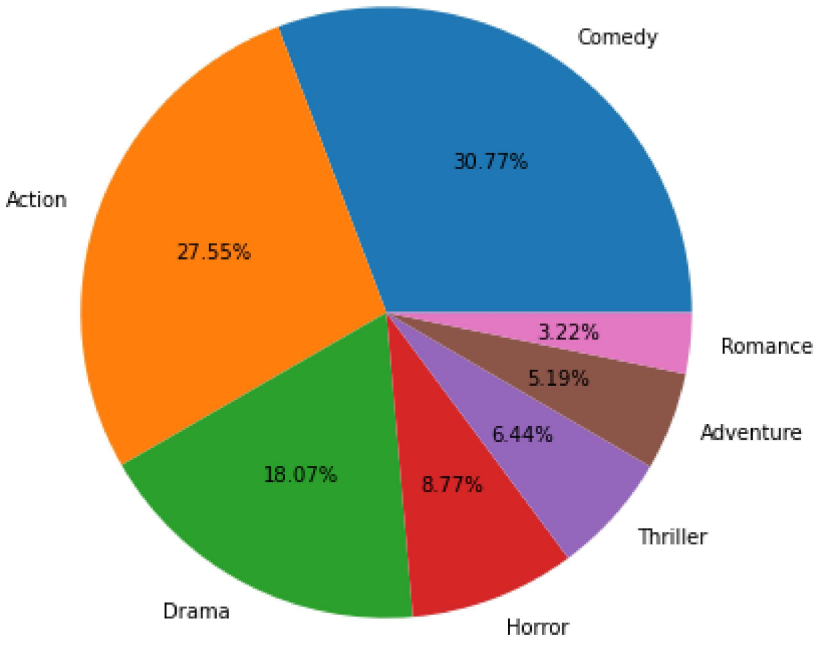
According to the Audience Rantings most of the people like comedy movies...

In [20]:

```
plt.figure(figsize =(14,7))
plt.pie(df['Genre'].value_counts(),labels=df['Genre'].value_counts().index,autopct='%1.2f%%
```

Out[20]:

```
([<matplotlib.patches.Wedge at 0x29c26fa0a30>,
  <matplotlib.patches.Wedge at 0x29c26fb4190>,
  <matplotlib.patches.Wedge at 0x29c26fb48b0>,
  <matplotlib.patches.Wedge at 0x29c26fb4fd0>,
  <matplotlib.patches.Wedge at 0x29c26fc0730>,
  <matplotlib.patches.Wedge at 0x29c26fc0e50>,
  <matplotlib.patches.Wedge at 0x29c26fcf5b0>],
 [Text(0.6248711888047935, 0.9052822749848162, 'Comedy'),
  Text(-1.0359910525893912, 0.36976011001015974, 'Action'),
  Text(-0.5084564369146221, -0.9754342887965787, 'Drama'),
  Text(0.3900654498212439, -1.0285178388612182, 'Horror'),
  Text(0.8192494545611191, -0.7340506325869551, 'Thriller'),
  Text(1.027417752332683, -0.39295389321351365, 'Adventure'),
  Text(1.0943764227924597, -0.1110866564263214, 'Romance')],
 [Text(0.3408388302571601, 0.4937903318098996, '30.77%'),
  Text(-0.5650860286851224, 0.2016873327328144, '27.55%'),
  Text(-0.27733987468070287, -0.5320550666163155, '18.07%'),
  Text(0.21276297262976937, -0.5610097302879371, '8.77%'),
  Text(0.44686333885151946, -0.4003912541383391, '6.44%'),
  Text(0.5604096830905544, -0.21433848720737106, '5.19%'),
  Text(0.5969325942504325, -0.06059272168708439, '3.22%')])
```



Observation:

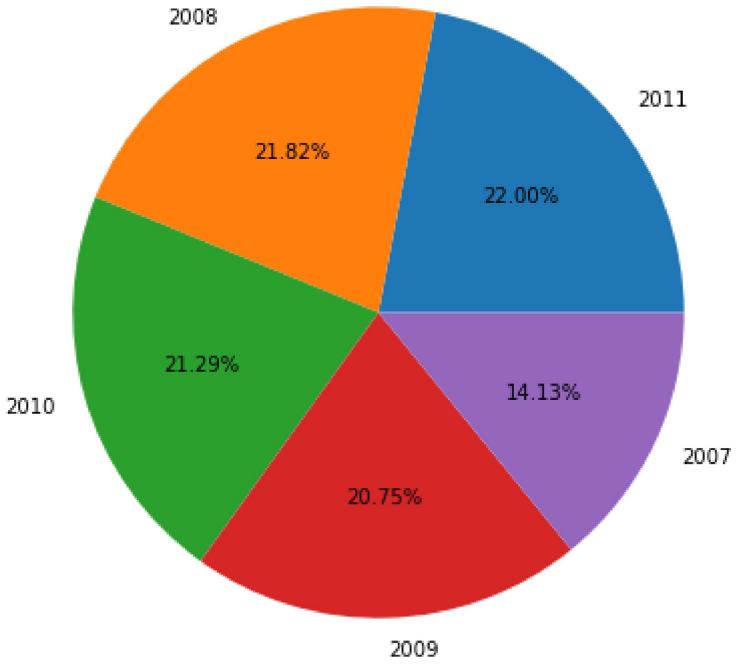
```
30.77% of people like comedy movies
27.55% of people like Action movies
3.22% of people like Romantic Movies
```

In [23]:

```
plt.figure(figsize =(14,7))
plt.pie(df['Year of release'].value_counts(),labels=df['Year of release'].value_counts().in
```

Out[23]:

```
([<matplotlib.patches.Wedge at 0x29c2701baf0>,
  <matplotlib.patches.Wedge at 0x29c2702b2b0>,
  <matplotlib.patches.Wedge at 0x29c2702b9d0>,
  <matplotlib.patches.Wedge at 0x29c270380d0>,
  <matplotlib.patches.Wedge at 0x29c270387f0>],
 [Text(0.8474857539798061, 0.7012616464639142, '2011'),
  Text(-0.5248293299828135, 0.9667234218688359, '2008'),
  Text(-1.0568561278975968, -0.3050493811259071, '2010'),
  Text(0.033995723258231546, -1.0994745521384977, '2009'),
  Text(0.9933533479262744, -0.4724924614887122, '2007')],
 [Text(0.4622649567162578, 0.3825063526166805, '22.00%'),
  Text(-0.2862705436269891, 0.5273036846557286, '21.82%'),
  Text(-0.5764669788532345, -0.16639057152322204, '21.29%'),
  Text(0.018543121777217203, -0.5997133920755441, '20.75%'),
  Text(0.5418290988688769, -0.2577231608120248, '14.13%')])
```

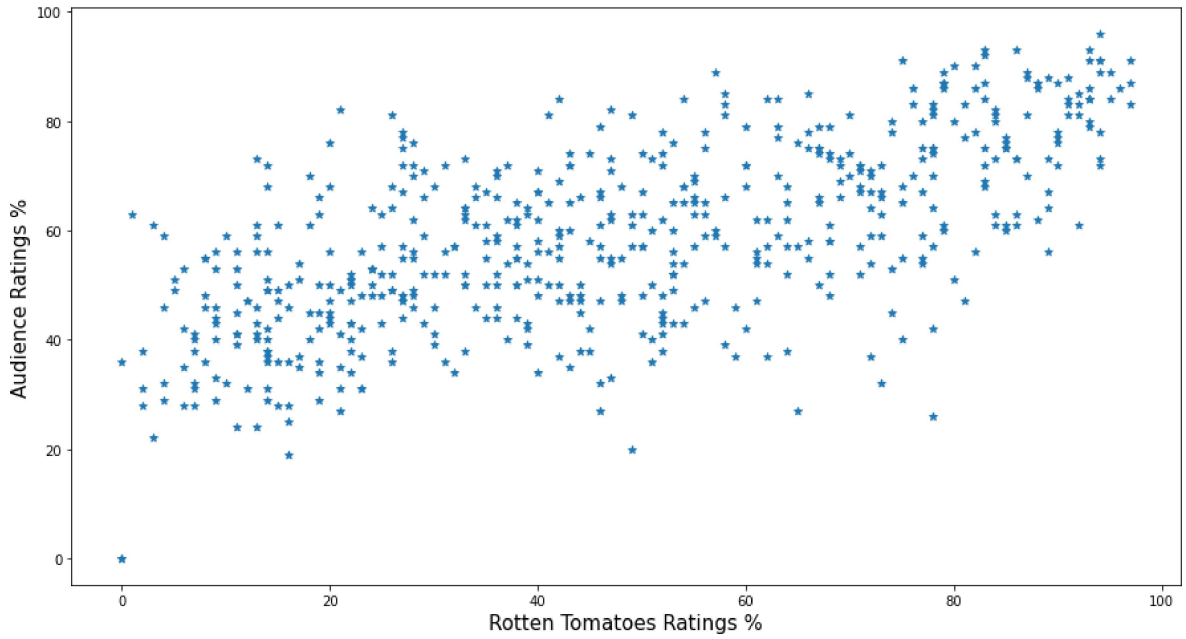



Observations:

we can see in this graph avg percentage of movies released.

In [24]:

```
plt.figure(figsize=(15,8))
X=df['Rotten Tomatoes Ratings %']
Y=df['Audience Ratings %']
plt.xlabel('Rotten Tomatoes Ratings %',fontsize=15)
plt.ylabel('Audience Ratings %',fontsize=15)
plt.scatter(X,Y,marker='*')
plt.show()
```




Observations:

positive relation between rotten tomatoes rating and audience rating....

Type *Markdown* and LaTeX: $\alpha^2$

In [17]:

```
df.head()
```

Out[17]:

| | Film | Genre | Rotten Tomatoes Ratings % | Audience Ratings % | Budget (million $) | Year of release |
|---|---|---|---|---|---|---|
| 0 | (500) Days of Summer | Comedy | 87 | 81 | 8 | 2009 |
| 1 | 10,000 B.C. | Adventure | 9 | 44 | 105 | 2008 |
| 2 | 12 Rounds | Action | 30 | 52 | 20 | 2009 |
| 3 | 127 Hours | Adventure | 93 | 84 | 18 | 2010 |
| 4 | 17 Again | Comedy | 55 | 70 | 20 | 2009 |

In [57]:

```
df.Genre.value_counts()
```

Out[57]:

```
Comedy       172
Action       154
Drama        101
Horror        49
Thriller      36
Adventure     29
Romance       18
Name: Genre, dtype: int64
```

In [23]:

```
(df[df['Year of release']==2009]['Genre'].value_counts())
```

Out[23]:

```
Comedy       41
Action       35
Drama        22
Horror        9
Adventure     8
Romance       1
Name: Genre, dtype: int64
```

In [28]:

```
(df[df['Year of release']==2008]['Genre'].value_counts())
```

Out[28]:

```
Comedy       41
Action       32
Drama        19
Adventure    11
Horror       10
Thriller      9
Name: Genre, dtype: int64
```

In [31]:

```
(df[df['Year of release']==2007]['Genre'].value_counts())
```

Out[31]:

```
Comedy       22
Action       18
Drama        12
Thriller     11
Horror        9
Romance       5
Adventure     2
Name: Genre, dtype: int64
```

In [43]:

```
(df[df['Year of release']==2010]['Genre'].value_counts())
```

Out[43]:

```
Comedy       39
Action       38
Drama        25
Horror        8
Adventure     6
Thriller      2
Romance       1
Name: Genre, dtype: int64
```

In [45]:

```
(df[df['Year of release']==2011]['Genre'].value_counts())
```

Out[45]:

```
Comedy       39
Action       38
Drama        25
Horror        8
Adventure     6
Thriller      2
Romance       1
Name: Genre, dtype: int64
```

In [71]:

```
df.Genre[df['Budget (million $)']==df['Budget (million $)'].max()]
```

Out[71]:

```
304    Action
Name: Genre, dtype: object
```

# Thankyou

In [ ]: