In [2]:
```
import pandas as pd

df = pd.read_csv("HollywoodMovies.csv")
df
```

Out[2]:

	Movie	LeadStudio	RottenTomatoes	AudienceScore	Story	Genre	Thea
0	Spider-Man 3	Sony	61.0	54.0	Metamorphosis	Action	
1	Shrek the Third	Paramount	42.0	57.0	Quest	Animation	
2	Transformers	Paramount	57.0	89.0	Monster Force	Action	
3	Pirates of the Caribbean: At World's End	Disney	45.0	74.0	Rescue	Action	
4	Harry Potter and the Order of the Phoenix	Warner Bros	78.0	82.0	Quest	Adventure	
...	...	...	...	...	...	...	
965	The Canyons	IFC	22.0	NaN	NaN	NaN	
966	The Call	TriStar	43.0	66.0	NaN	NaN	
967	The English Teacher	Cinedigm Entertainment	42.0	NaN	NaN	NaN	
968	John Dies at the End	Magnolia	61.0	53.0	NaN	NaN	
969	Lovelace	Radius-TWC	55.0	37.0	NaN	Biography	

970 rows × 16 columns

In [5]:
```
#1. Find the highest rated movie in the“Quest” story type.
highest_rate = df[df["Story"] == "Quest"]["RottenTomatoes"].max()
highest_rated_movies = df[df["RottenTomatoes"] == highest_rate]
highest_rated_movies
```

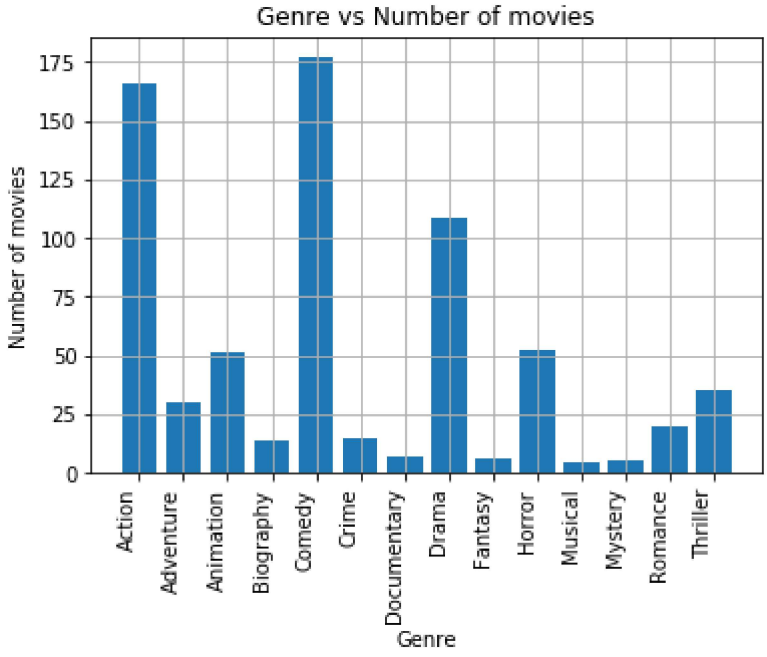
Out[5]:

	Movie	LeadStudio	RottenTomatoes	AudienceScore	Story	Genre	Theaters
10	Ratatouille	Disney	97.0	84.0	Transformation	Animation	
343	The Hurt Locker	Independent	97.0	83.0	Quest	Drama	
611	The Artist	Weinstein	97.0	91.0	NaN	Drama	
629	The Muppets	Disney	97.0	87.0	Quest	Comedy	
830	Gravity	Warner Bros	97.0	85.0	NaN	NaN	

In [12]:

```
#2. Find the genre in which there has been the greatest number of movie releas
es
import matplotlib.pyplot as plt
genre_groups = df.groupby("Genre").groups
genre_groups.keys()

plt.bar(genre_groups.keys(), [len(values) for values in genre_groups.values
()])
plt.xlabel("Genre")
plt.ylabel("Number of movies")
plt.title("Genre vs Number of movies")
plt.grid()

plt.setp(plt.gca().get_xticklabels(), rotation=90, horizontalalignment='right'
)

#Comedy is the genre
```

Out[12]: [None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None]

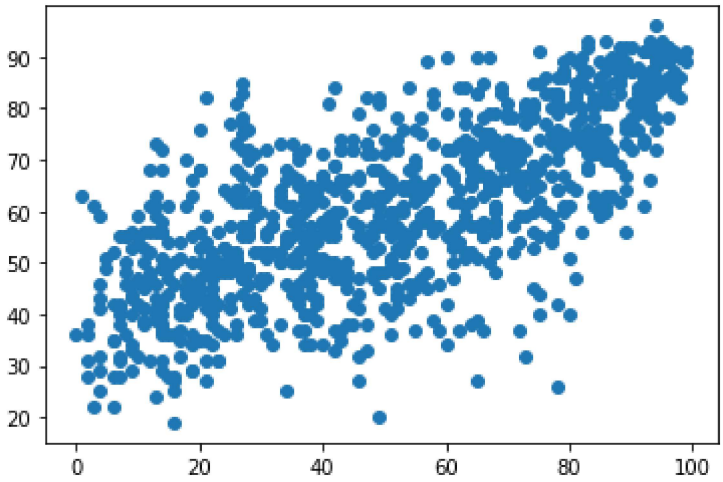


In [29]:
```
# 3. Print the names of the top five movies with the costliest budgets.
sorted_df = df.sort_values("Budget", ascending=False)
sorted_df = sorted_df[sorted_df["Budget"]>0]   #removing all nan values
print(sorted_df.head())
```

```
                                      Movie   LeadStudio  RottenTomatoes  \
3    Pirates of the Caribbean: At World's End       Disney            45.0
468                                  Tangled       Disney            89.0
0                              Spider-Man 3         Sony            61.0
778                    The Dark Knight Rises  Warner Bros            88.0
241    Harry Potter and the Half-Blood Prince  Warner Bros            83.0

     AudienceScore          Story      Genre  TheatersOpenWeek  \
3             74.0         Rescue     Action            4362.0
468           88.0           Love  Animation            3603.0
0             54.0  Metamorphosis     Action            4252.0
778           90.0            NaN        NaN            4404.0
241           75.0          Quest  Adventure            4325.0

     OpeningWeekend  BOAvgOpenWeekend  DomesticGross  ForeignGross  \
3            114.70           26302.0         309.42        654.00
468           48.77           13535.0         200.82        390.97
0            151.10           35540.0         336.53        554.34
778          160.89           36532.0         448.14        636.30
241           77.80           17997.0         302.00        632.00

     WorldGross  Budget  Profitability  OpenProfit  Year
3       963.420   300.0         321.14       38.23  2007
468     591.794   260.0         227.61       18.76  2010
0       890.870   258.0         345.30       58.57  2007
778    1084.440   250.0         433.78       64.36  2012
241     934.000   250.0         373.60       31.12  2009
```

In [30]:
```
# 4. Is there any correspondence between the critics' evaluation of a movie an
d itsacceptance
# by the public? Find out, by plotting the net profitability of a movie agains
t the
# ratings it receives on Rotten Tomatoes.

plt.scatter(df["RottenTomatoes"], df["AudienceScore"])
```

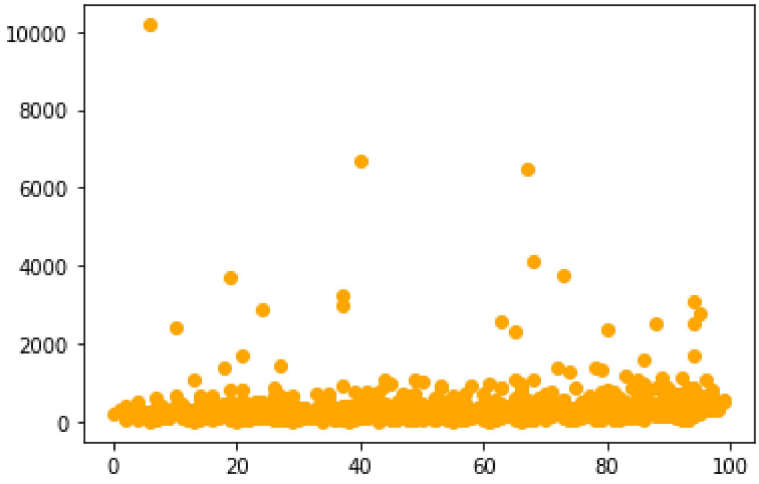
Out[30]: <matplotlib.collections.PathCollection at 0x1f1f7129148>



In [32]:
```
# According to the above graph, we can clearly say that critics response and p
ublic
# evaluation is directly proportional to each other
```

In [39]:
```
# Graph between RottenTomatoes and Profitibility
plt.scatter(df["RottenTomatoes"], df["Profitability"], color='Orange')
```

Out[39]: <matplotlib.collections.PathCollection at 0x1f1f6f20608>



In [40]:
```
# query 5- 5.1- Creating a dataframe from raw data
df = pd.DataFrame({'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
                   'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'
],
                   'age': [42, 52, 36, 24, 73],
                   'preTestScore': [4, 24, 31, ".", "."],
                   'postTestScore': ["25,000", "94,000", 57, 62, 70]})
df
```

Out[40]:

	first_name	last_name	age	preTestScore	postTestScore
0	Jason	Miller	42	4	25,000
1	Molly	Jacobson	52	24	94,000
2	Tina	.	36	31	57
3	Jake	Milner	24	.	62
4	Amy	Cooze	73	.	70

In [41]:
```
#5.2- Save the dataframe into a csv file as example.csv
df.to_csv("example.csv")
print("*"*20)
```

```
********************
```

In [42]:
```
#5.3- Read the example.csv and print the data frame
df1= pd.read_csv("example.csv")
df
```

Out[42]:

	first_name	last_name	age	preTestScore	postTestScore
0	Jason	Miller	42	4	25,000
1	Molly	Jacobson	52	24	94,000
2	Tina	.	36	31	57
3	Jake	Milner	24	.	62
4	Amy	Cooze	73	.	70

In [45]:
```
#5.4- Read the example.csv without column heading
df_without_header = pd.read_csv("example.csv", header=None)
df_without_header
```

Out[45]:

	0	1	2	3	4	5
0	NaN	first_name	last_name	age	preTestScore	postTestScore
1	0.0	Jason	Miller	42	4	25,000
2	1.0	Molly	Jacobson	52	24	94,000
3	2.0	Tina	.	36	31	57
4	3.0	Jake	Milner	24	.	62
5	4.0	Amy	Cooze	73	.	70

In [54]:
```
#5.5- Read the example.csv andmake the index columns as 'First Name' and 'Last Name
df_with_index = pd.read_csv("example.csv", index_col = ["first_name", "last_name"])
print(df_with_index)
print("*"*20)
```

```
                       Unnamed: 0  age preTestScore postTestScore
first_name last_name
Jason      Miller               0   42            4        25,000
Molly      Jacobson             1   52           24        94,000
Tina       .                    2   36           31            57
Jake       Milner               3   24            .            62
Amy        Cooze                4   73            .            70
********************
```

In [3]:
```
#5.6- Print the data frame in a Boolean form as True or False. True for Null/ NaN values and false for non-nullvalues
boolean_df = df.isnull().any()
boolean_df
```

Out[3]:

```
Movie               False
LeadStudio           True
RottenTomatoes       True
AudienceScore        True
Story                True
Genre                True
TheatersOpenWeek     True
OpeningWeekend       True
BOAvgOpenWeekend     True
DomesticGross       False
ForeignGross         True
WorldGross           True
Budget               True
Profitability        True
OpenProfit           True
Year                False
dtype: bool
```

In [4]:
```
#5.7- Read the dataframe by skipping first 3 rows and print the data frame
df_skip_rows = pd.read_csv("example.csv", skiprows=3)
df_skip_rows
```

Out[4]:

	2	Tina	.	36	31	57
0	3	Jake	Milner	24	.	62
1	4	Amy	Cooze	73	.	70

In [9]:
```
# 5.8: Load a csv file while interpreting , in strings around numbers as thous
ands seperators.
# Check the raw data postTestScore column has, as thousands separator.
data = pd.read_csv("example.csv" index_col=4).filter(regex='\d{4}')
data
```

```
  File "<ipython-input-9-f59a1da78ce0>", line 3
    data = pd.read_csv("example.csv" index_col=4).filter(regex='\d{4}')
                                                 ^
SyntaxError: invalid syntax
```

In [10]:
```
# 6.1: From the raw data below create a Pandas Series'Amit', 'Bob', 'Kate',
 'A', 'b', np.nan, 'Car', 'dog', 'cat
import pandas as pd
import numpy as np
import re
```

In [12]:
```
series_1 = pd.Series(['Amit', 'Bob', 'Kate', 'A', 'b', np.nan, 'Car', 'dog',
'cat'])
series_1
```

Out[12]:
```
0    Amit
1     Bob
2    Kate
3       A
4       b
5     NaN
6     Car
7     dog
8     cat
dtype: object
```

In [13]:
```
# a) Print all elements in lower case
lower_series = [str(i).lower() for i in series_1]
lower_series
```

Out[13]:
```
['amit', 'bob', 'kate', 'a', 'b', 'nan', 'car', 'dog', 'cat']
```

In [15]:
```
# b) Print all the elements in upper case
upper_series = [str(i).upper() for i in series_1]
upper_series
```

Out[15]:
```
['AMIT', 'BOB', 'KATE', 'A', 'B', 'NAN', 'CAR', 'DOG', 'CAT']
```

In [17]:
```
# c) Print the length of all the elements
print(len(series_1))
```

```
9
```

In [20]:
```
# 6.2: From the raw data below create a Pandas Series' Atul', 'John ', ' jack
 ', 'Sam
series_2 = pd.Series(['Atul', 'John', 'jack', 'Sam'])
series_2
```

Out[20]:
```
0    Atul
1    John
2    jack
3     Sam
dtype: object
```

In [21]:
```
# a) Print all elements after stripping spaces from the left and right
stripped_series = [str(elem).strip() for elem in series_2]
stripped_series
```

Out[21]:
```
['Atul', 'John', 'jack', 'Sam']
```

In [22]:
```
# b) Print all the elements after removing spaces from the left only
lstripped_series = [str(elem).lstrip() for elem in series_2]
lstripped_series
```

Out[22]:
```
['Atul', 'John', 'jack', 'Sam']
```

In [23]:
```
# c) Print all the elements after removing spaces from the right only
rstripped_series = [str(elem).rstrip() for elem in series_2]
rstripped_series
```

Out[23]:
```
['Atul', 'John', 'jack', 'Sam']
```

In [25]:
```
# 6.3: -Create a series from the raw data below'India_is_big', 'Population_is_
huge', np.nan, 'Has_diverse_culture'
series_3 = pd.Series(['India_is_big', 'Population_is_huge', np.nan, 'Has_diver
se_culture'])
series_3
```

Out[25]:
```
0           India_is_big
1     Population_is_huge
2                    NaN
3    Has_diverse_culture
dtype: object
```

In [29]:
```
# a)split the individual strings wherever '_' comes and create a list out of i
t
splitted_list = [str(ele).split("_") for ele in series_3]
splitted_list
```

Out[29]:
```
[['India', 'is', 'big'],
 ['Population', 'is', 'huge'],
 ['nan'],
 ['Has', 'diverse', 'culture']]
```

In [34]:
```
# b)Access the individual element of a list
lst = []
for sublist in splitted_list:
    for ele in sublist:
        lst.append(ele)
        print(ele)
```

```
India
is
big
Population
is
huge
nan
Has
diverse
culture
```

In [35]:
```
# c)Expand the elements so that all individual elements get splitted by '_' an
d insted of list returns individual elements
print(lst)
```

```
['India', 'is', 'big', 'Population', 'is', 'huge', 'nan', 'Has', 'diverse',
'culture']
```

In [36]:
```
# 6.4: Create a series and replace either X or dog with XX-XX: 'A', 'B', 'C',
 'AabX', 'BacX','', np.nan, 'CABA', 'dog', 'cat
series_4 = pd.Series(['A', 'B', 'C', 'AabX', 'BacX','', np.nan, 'CABA', 'dog',
'cat' ])
series_4
```

Out[36]:
```
0       A
1       B
2       C
3    AabX
4    BacX
5
6     NaN
7    CABA
8     dog
9     cat
dtype: object
```

In [37]:
```
replaced_series = [str(ele).replace("X", "XX-XX").replace("dog", "XX-XX") for
ele in series_4]
replaced_series
```

Out[37]:
```
['A', 'B', 'C', 'AabXX-XX', 'BacXX-XX', '', 'nan', 'CABA', 'XX-XX', 'cat']
```

In [38]:
```
# 6.5: Create a series and remove dollar from the numeric values'12', '-$10',
 '$10,000'
series_5 = pd.Series(['12', '-$10', '$10,000'])
series_5
```

Out[38]:
```
0         12
1       -$10
2    $10,000
dtype: object
```

In [39]:
```
dollar_removed_series = [str(ele).replace("$", "") for ele in series_5]
dollar_removed_series
```

Out[39]:
```
['12', '-10', '10,000']
```

In [41]:
```
# 6.6:-Create a series and reverse all lower case words'india 1998', 'big coun
try', np.nan
series_6 = pd.Series(['india 1998', 'big country', np.nan])
series_6[::-1]     #series reversed
```

Out[41]:
```
2            NaN
1    big country
0     india 1998
dtype: object
```

In [42]:
```
# 6.7: Create pandas series and print true if value is alphanumeric in series
 or false if value is not alpha numeric in series.
series_7 = pd.Series(['1', '2', '1a', '2b', '2003c'])
series_7
```

Out[42]:
```
0        1
1        2
2       1a
3       2b
4    2003c
dtype: object
```

In [43]:
```
alphanum_series = [str(ele).isalnum() for ele in series_7]
alphanum_series
```

Out[43]:
```
[True, True, True, True, True]
```

In [45]:
```
# 6.8: Create pandas series and print true if value is containing ‘A’'1', '2', '1a', '2b', 'America', 'VietnAm','vietnam', '2003c'
series_8 = pd.Series(['A','1', '2', '1a', '2b', 'America', 'VietnAm','vietnam', '2003c'])
series_8
```

Out[45]:
```
0          A
1          1
2          2
3         1a
4         2b
5    America
6    VietnAm
7    vietnam
8      2003c
dtype: object
```

In [46]:
```
A_series = ['A' in ele for ele in series_8]
A_series
```

Out[46]:
```
[True, False, False, False, False, True, True, False, False]
```

In [48]:
```
# 6.9: Create pandas series and print in three columns value 0 or 1 if a or b  or c exists in values'a', 'a|b', np.nan, 'a|c'
series_9 = pd.Series(['a', 'a|b', np.nan, 'a|c'])
series_9
```

Out[48]:
```
0      a
1    a|b
2    NaN
3    a|c
dtype: object
```

In [49]:
```
abc_series = [1 if re.match("[abc]", str(ele)) else 0 for ele in series_9]
abc_series
```

Out[49]:
```
[1, 1, 0, 1]
```

In [51]:
```
# 6.10: Create pandas dataframe having keys and ltable and rtable as below -'k
ey': ['One', 'Two'], 'ltable': [1, 2]'key': ['One', 'Two'], 'rtable': [4, 5].
 Merge both the tables based of key
df1 = pd.DataFrame({'key': ['One', 'Two'], 'ltable': [1, 2]})
df2 = pd.DataFrame({'key': ['One', 'Two'], 'rtable': [4, 5]})
print(df1)
print(df2)

df_merge = pd.merge(df1, df2, sort=True)
df_merge
```

```
   key  ltable
0  One       1
1  Two       2
   key  rtable
0  One       4
1  Two       5
```

Out[51]:

	key	ltable	rtable
0	One	1	4
1	Two	2	5

In [ ]: