```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data=pd.read_csv('IMDB-Movie-Data.csv')
```

1. Display Top 10 Rows of The Dataset

```python
data.head(10)
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Rank\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 288,\n        \"min\": 1,\n
\"max\": 1000,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          522,\n          738,\n          741\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Title\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
999,\n        \"samples\": [\n          \"Olympus Has Fallen\",\n
\"Man on a Ledge\",\n          \"The Girl with All the Gifts\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Genre\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
207,\n        \"samples\": [\n          \"Mystery,Romance,Sci-Fi\",\n
\"Drama,Mystery,Sci-Fi\",\n          \"Drama,Mystery,Romance\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Description\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n          \"A
lawyer finds himself in over his head when he gets involved in drug
trafficking.\",\n          \"A CIA agent on the ground in Jordan hunts
down a powerful terrorist leader while being caught between the
unclear intentions of his American supervisors and Jordan
Intelligence.\",\n          \"A titan of industry is sent to prison
after she's caught insider trading. When she emerges ready to rebrand
herself as America's latest sweetheart, not everyone she screwed over
is so quick to forgive and forget.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Director\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
644,\n        \"samples\": [\n          \"Patricia Riggen\",\n
\"Gregory Wilson\",\n          \"Chris McCoy\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Actors\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
996,\n        \"samples\": [\n          \"Adrian Titieni, Maria-
Victoria Dragus, Lia Bugnar,Malina Manovici\",\n          \"Madina
Nalwanga, David Oyelowo, Lupita Nyong'o, Martin Kabanza\",\n
\"Ry\\u00fbnosuke Kamiki, Mone Kamishiraishi, Ry\\u00f4 Narita, Aoi
Yuki\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 3,\n        \"min\": 2006,\n        \"max\": 2016,\n
\"num_unique_values\": 11,\n        \"samples\": [\n          2011,\n
2014,\n          2010\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Runtime (Minutes)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 18,\n        \"min\": 66,\n
\"max\": 191,\n        \"num_unique_values\": 94,\n
\"samples\": [\n          106,\n          99,\n          146\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rating\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.9454287892779637,\n        \"min\": 1.9,\n        \"max\":
9.0,\n        \"num_unique_values\": 55,\n        \"samples\": [\n
7.4,\n          6.1,\n          4.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Votes\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 188762,\n
\"min\": 61,\n        \"max\": 1791916,\n
\"num_unique_values\": 997,\n        \"samples\": [\n
214994,\n          4370,\n          23713\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Revenue (Millions)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
103.25354047492473,\n        \"min\": 0.0,\n        \"max\": 936.63,\n
\"num_unique_values\": 814,\n        \"samples\": [\n          89.02,\
n          23.23,\n          202.85\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Metascore\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
17.194757023263836,\n        \"min\": 11.0,\n        \"max\": 100.0,\n
\"num_unique_values\": 84,\n        \"samples\": [\n          27.0,\n
76.0,\n          47.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

2. Check Last 10 Rows of The Dataset

```
data.tail(10)
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"Rank\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 991,\n
\"max\": 1000,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          999,\n          992,\n          996\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Title\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 10,\
n        \"samples\": [\n          \"Search Party\",\n
\"Taare Zameen Par\",\n          \"Secret in Their Eyes\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Genre\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 10,\
n        \"samples\": [\n          \"Adventure,Comedy\",\n
\"Drama,Family,Music\",\n          \"Crime,Drama,Mystery\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          \"A pair of friends embark on a mission to reunite their
pal with the woman he was going to marry.\",\n          \"An eight-
year-old boy is thought to be a lazy trouble-maker, until the new art
teacher has the patience and compassion to discover the real problem
behind his struggles in school.\",\n          \"A tight-knit team of
rising investigators, along with their supervisor, is suddenly torn
apart when they discover that one of their own teenage daughters has
been brutally murdered.\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          \"Scot Armstrong\",\n          \"Aamir Khan\",\n
\"Billy Ray\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Actors\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"Adam
Pally, T.J. Miller, Thomas Middleditch,Shannon Woodward\",\n
\"Darsheel Safary, Aamir Khan, Tanay Chheda, Sachet Engineer\",\n
\"Chiwetel Ejiofor, Nicole Kidman, Julia Roberts, Dean Norris\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Year\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 3,\n
\"min\": 2007,\n        \"max\": 2016,\n        \"num_unique_values\":
9,\n        \"samples\": [\n          2014,\n          2007,\n
2015\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Runtime (Minutes)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 23,\n        \"min\": 87,\n
\"max\": 165,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          93,\n          165,\n          94\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Rating\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.9065195959149355,\n        \"min\": 5.3,\n        \"max\": 8.5,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          5.6,\n
```

```
8.5,\n          6.2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Votes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 55746,\n        \"min\": 4881,\n        \"max\": 164088,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          4881,\n
102697,\n          27585\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Revenue (Millions)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 24.21936297380967,\n
\"min\": 1.2,\n        \"max\": 60.13,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          1.2,\n          17.54,\n
45.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Metascore\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 13.200378782444085,\n        \"min\":
11.0,\n        \"max\": 50.0,\n        \"num_unique_values\": 9,\n
\"samples\": [\n          22.0,\n          42.0,\n          46.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

3. Find Shape of Our Dataset (Number of Rows And Number of Columns)

```
data.shape
```

```
(1000, 12)
```

```
print('Number of Rows',data.shape[0])
print('Number of Columns',data.shape[1])
```

```
Number of Rows 1000
Number of Columns 12
```

4. Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

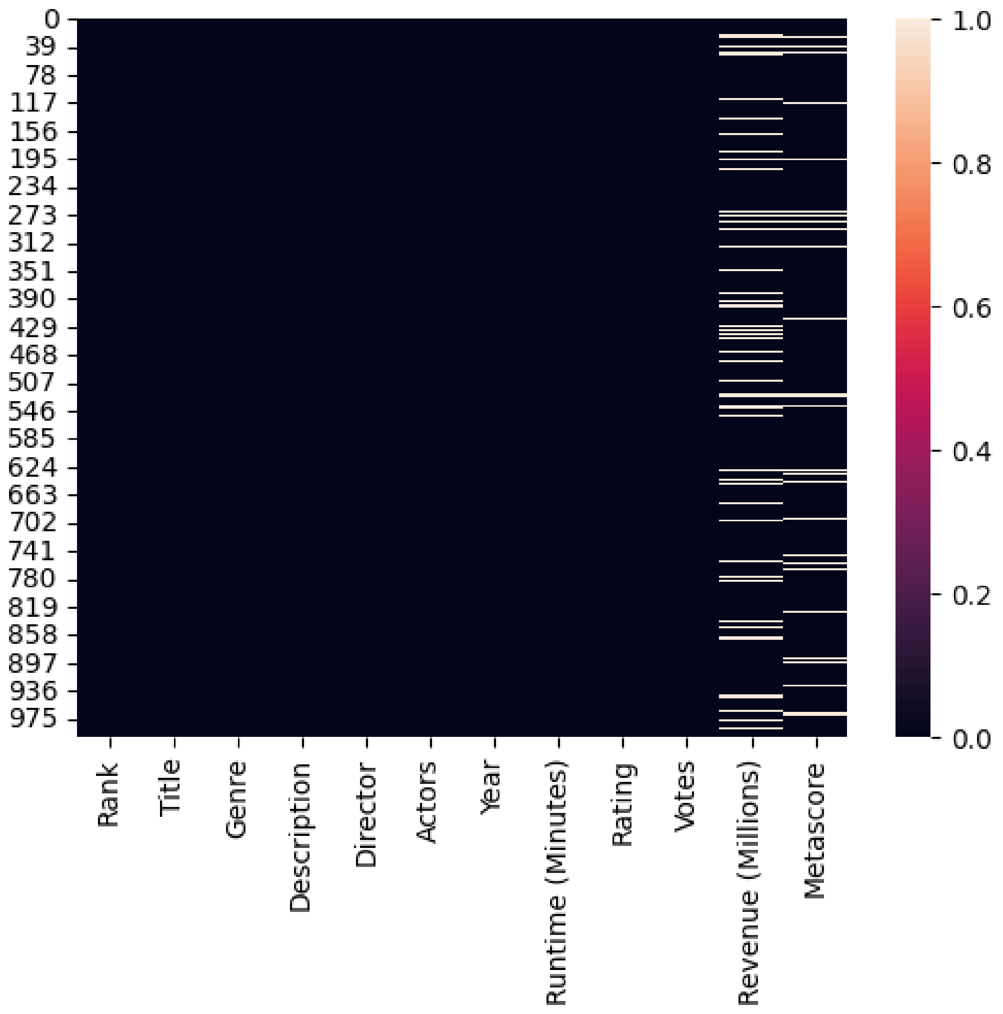
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
```

```
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                1000 non-null   int64  
 1   Title               1000 non-null   object 
 2   Genre               1000 non-null   object 
 3   Description         1000 non-null   object 
 4   Director            1000 non-null   object 
 5   Actors              1000 non-null   object 
 6   Year                1000 non-null   int64  
 7   Runtime (Minutes)   1000 non-null   int64  
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64  
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

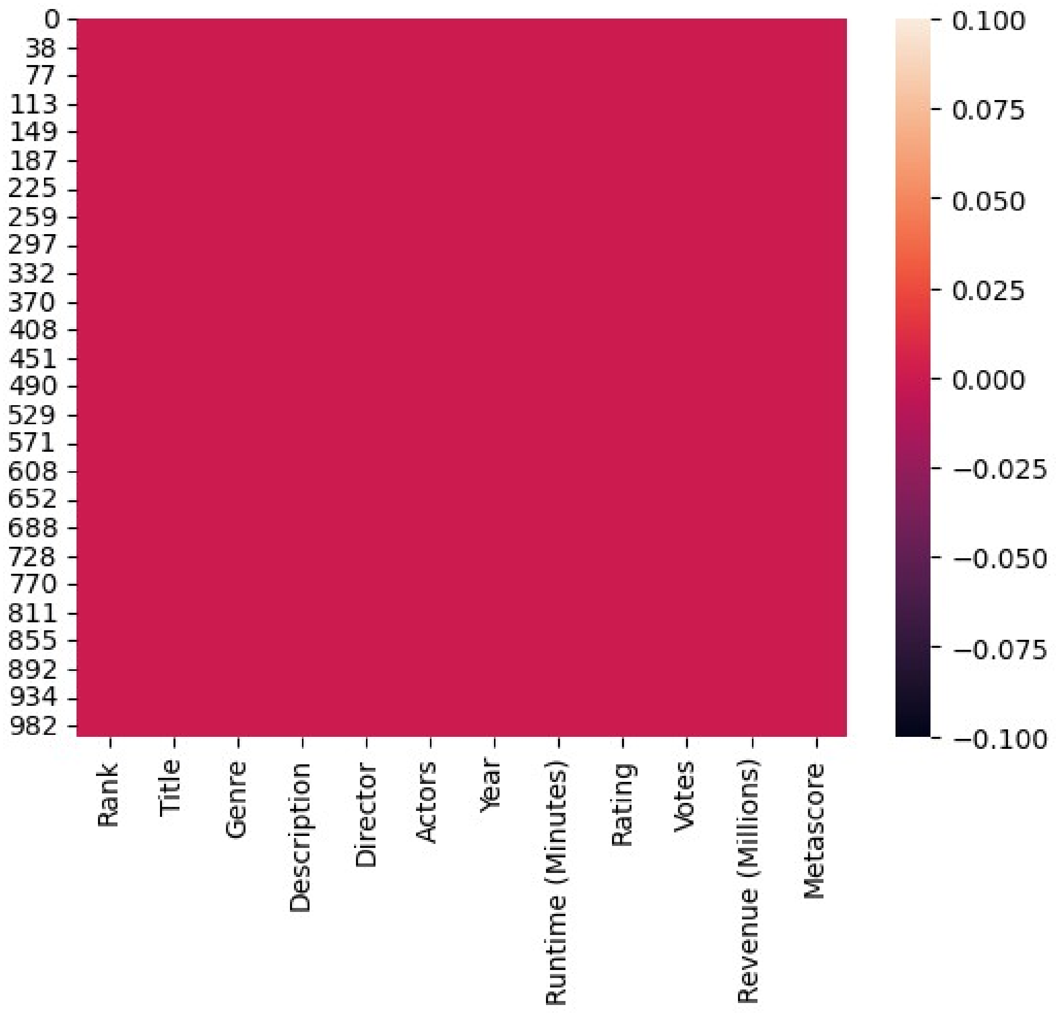
5. Check Null Values In The Dataset

```
data.isnull().sum()

Rank                    0
Title                   0
Genre                   0
Description             0
Director                0
Actors                  0
Year                    0
Runtime (Minutes)       0
Rating                  0
Votes                   0
Revenue (Millions)    128
Metascore              64
dtype: int64

import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(data.isnull())
plt.show()
```



6. Drop All The Missing Values

```
data = data.dropna(axis=0)

sns.heatmap(data.isnull())
plt.show()
```



7. Check For Duplicate Data

```
dup_data=data.duplicated().any()
print("Are there any duplicated values in data?",dup_data)

Are there any duplicated values in data? False
```

8. Get Overall Statistics About The DataFrame

```
data.describe()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"Rank\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 333.92550255275137,\n   \"min\": 1.0,\n   \"max\": 1000.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n

```
485.2470167064439,\n          475.5,\n          838.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Year\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 769.7713287499067,\n
\"min\": 3.172359915266028,\n        \"max\": 2016.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
2012.5071599045345,\n          2013.0,\n          838.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Runtime (Minutes)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
264.1807671963779,\n        \"min\": 18.470922051554556,\n
\"max\": 838.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          114.63842482100239,\n          112.0,\n
838.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Rating\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 294.3064953884777,\n        \"min\": 0.8777538418027501,\n
\"max\": 838.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          6.814319809069212,\n          6.9,\n
838.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Votes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 598181.3436789116,\n        \"min\": 178.0,\n        \"max\":
1791916.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\
n          193230.25178997614,\n          136879.5,\n          838.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Revenue (Millions)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
385.4296330380251,\n        \"min\": 0.0,\n        \"max\": 936.63,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
84.5645584725537,\n          48.150000000000006,\n          838.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Metascore\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
279.24227712556353,\n        \"min\": 11.0,\n        \"max\": 838.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
59.575178997613364,\n          60.0,\n          838.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

9. Display Title of The Movie Having Runtime >= 180 Minutes

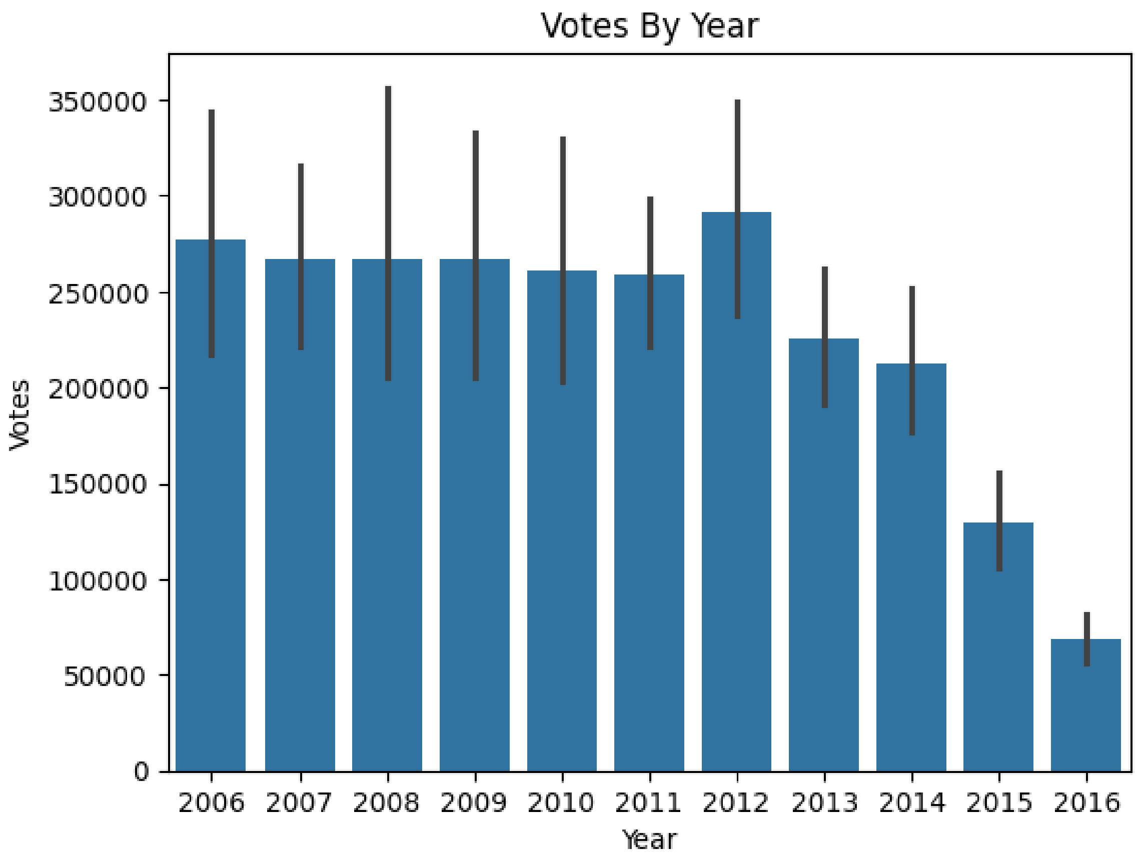
```
data[data['Runtime (Minutes)']>=180]['Title']
```

```
82     The Wolf of Wall Street
88           The Hateful Eight
```

```
311               La vie d'Adèle
Name: Title, dtype: object
```

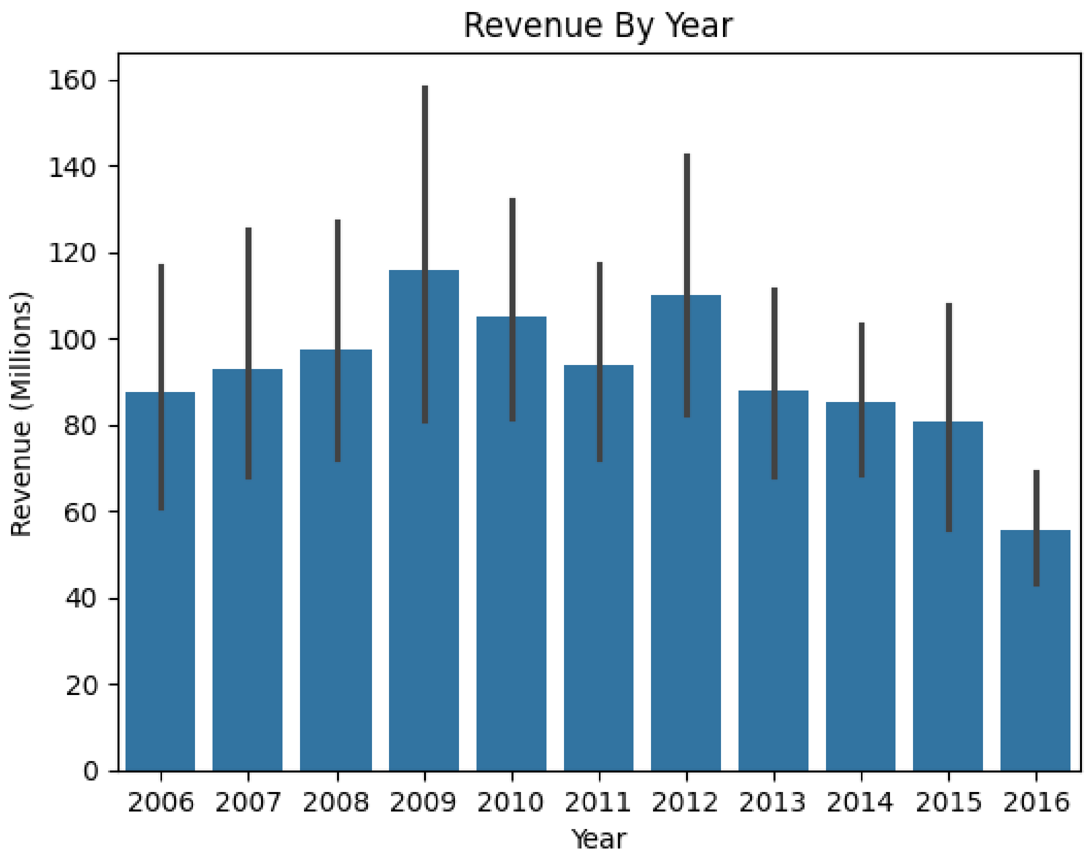
10. In Which Year There Was The Highest Voting?

```
sns.barplot(x='Year',y='Votes',data=data)
plt.title("Votes By Year")
plt.show()
```



11. In Which Year There Was The Highest Revenue?

```
sns.barplot(x='Year',y='Revenue (Millions)',data=data)
plt.title("Revenue By Year")
plt.show()
```



12. Find The Average Rating For Each Director

```
data.groupby('Director')['Rating'].mean().sort_values(ascending=False)

Director
Christopher Nolan                   8.68
Makoto Shinkai                      8.60
Olivier Nakache                     8.60
Aamir Khan                          8.50
Florian Henckel von Donnersmarck    8.50
                                    ...
Sam Taylor-Johnson                  4.10
Joey Curtis                         4.00
George Nolfi                        3.90
James Wong                          2.70
Jason Friedberg                     1.90
Name: Rating, Length: 524, dtype: float64
```

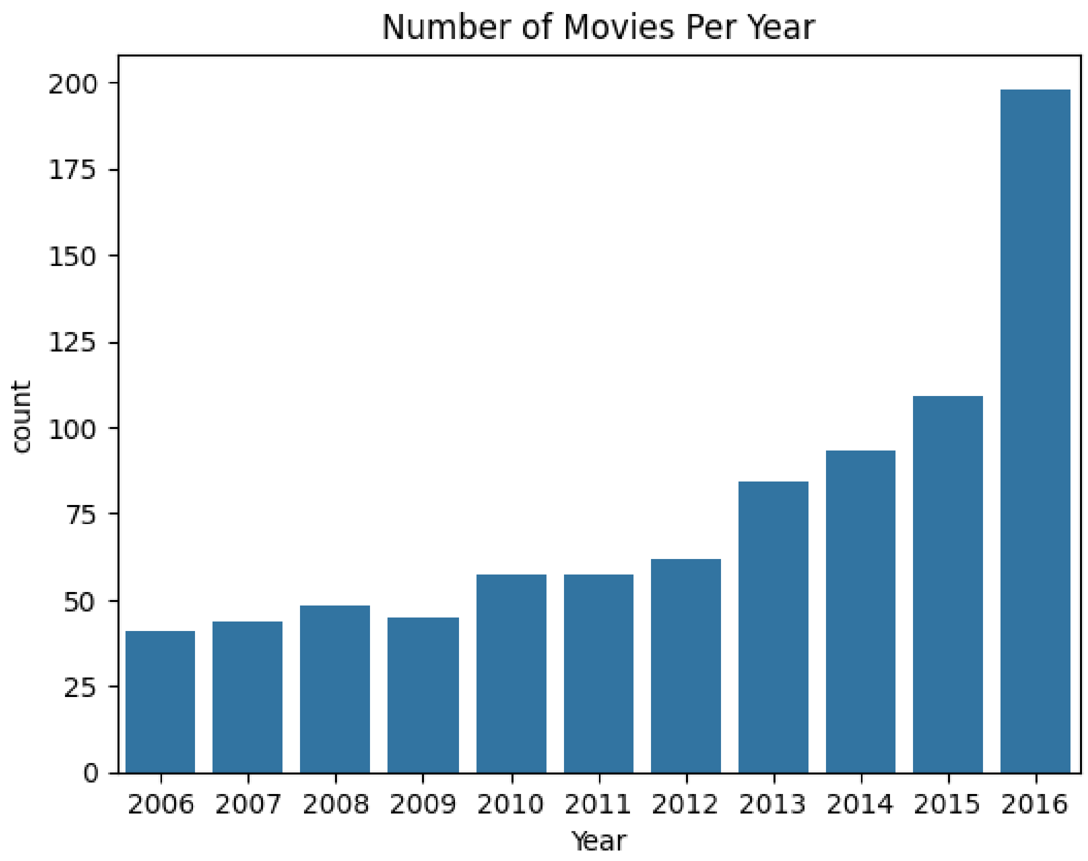
13. Display Top 10 Lengthy Movies Title

```
top_10_lengthy_movies = data.sort_values(
    by='Runtime (Minutes)', ascending=False
).head(10)

top_10_lengthy_movies[['Title', 'Runtime (Minutes)']]
```

```
{"summary":"{\n  \"name\": \"top_10_lengthy_movies[['Title', 'Runtime
(Minutes)']]\",\n  \"rows\": 10,\n  \"fields\": [\n    {\n
\"column\": \"Title\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          \"The Curious Case of Benjamin Button\",\n          \"The
Wolf of Wall Street\",\n          \"Pirates of the Caribbean: At
World's End\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Runtime (Minutes)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7,\n        \"min\": 165,\n
\"max\": 187,\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          187,\n          180,\n          166\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

14. Display Number of Movies Per Year

```
sns.countplot(x='Year',data=data)
plt.title("Number of Movies Per Year")

Text(0.5, 1.0, 'Number of Movies Per Year')
```



15. Find Most Popular Movie Title (Higest Revenue)

```
most_popular_movie = data.loc[
    data['Revenue (Millions)'].idxmax()
]

most_popular_movie[['Title', 'Revenue (Millions)']]

Title                      Star Wars: Episode VII - The Force Awakens
Revenue (Millions)                                             936.63
Name: 50, dtype: object
```

16. Display Top 10 Highest Rated Movie Titles And its Directors

```
top_10=data.nlargest(10,'Rating')
[['Title','Rating','Director']].set_index('Title')
```

```
top_10
```

{"summary":"{\n \"name\": \"top_10\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"Title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 10,\n   \"samples\": [\n   \"Whiplash\",\n   \"Inception\",\n   \"The Prestige\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Rating\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.1663329993316621,\n   \"min\": 8.5,\n   \"max\": 9.0,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   8.8,\n   8.5,\n   9.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Director\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 6,\n   \"samples\": [\n   \"Christopher Nolan\",\n   \"Makoto Shinkai\",\n   \"Florian Henckel von Donnersmarck\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"top_10"}

```
sns.barplot(x=top_10['Rating'], y=top_10.index)
plt.title("Display Top 10 Highest Rated Movie Titles")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Rating'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-2044283178.py in <cell line: 0>()
```

```
----> 1 sns.barplot(x=top_10['Rating'], y=top_10.index)
      2 plt.title("Display Top 10 Highest Rated Movie Titles")

/usr/local/lib/python3.12/dist-packages/pandas/core/frame.py in
__getitem__(self, key)
   4100             if self.columns.nlevels > 1:
   4101                 return self._getitem_multilevel(key)
-> 4102             indexer = self.columns.get_loc(key)
   4103             if is_integer(indexer):
   4104                 indexer = [indexer]

/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error
will raise

KeyError: 'Rating'
```

17. Display Top 10 Highest Revenue Movie Titles

```
data.columns

Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors',
'Year',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore'],
      dtype='object')

data.sort_values(by='Revenue (Millions)',ascending=False).head(10)
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 10,\n \"fields\":
[\n   {\n   \"column\": \"Rank\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 161,\n   \"min\": 13,\n
\"max\": 579,\n   \"num_unique_values\": 10,\n
\"samples\": [\n   125,\n   88,\n   13\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n   \"samples\": [\n   \"The
Dark Knight Rises\",\n   \"Avatar\",\n   \"Rogue One\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Genre\",\n   \"properties\":
{\n   \"dtype\": \"string\",\n   \"num_unique_values\": 7,\n
\"samples\": [\n   \"Action,Adventure,Fantasy\",\n

```
\"Action,Adventure,Sci-Fi\",\n          \"Action,Thriller\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Description\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          \"Eight years after the Joker's reign of anarchy, the
Dark Knight, with the help of the enigmatic Selina, is forced from his
imposed exile to save Gotham City, now on the edge of total
annihilation, from the brutal guerrilla terrorist Bane.\",\n
\"A paraplegic marine dispatched to the moon Pandora on a unique
mission becomes torn between following his orders and protecting the
world he feels is his home.\",\n          \"The Rebel Alliance makes a
risky move to steal the plans for the Death Star, setting up the epic
saga to follow.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Director\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          \"James Cameron\",\n          \"Gareth Edwards\",\n
\"J.J. Abrams\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Actors\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"Christian Bale, Tom Hardy, Anne Hathaway,Gary Oldman\",\n
\"Sam Worthington, Zoe Saldana, Sigourney Weaver, Michelle
Rodriguez\",\n          \"Felicity Jones, Diego Luna, Alan Tudyk,
Donnie Yen\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2,\n        \"min\": 2008,\n        \"max\": 2016,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          2015,\n
2009,\n          2013\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Runtime (Minutes)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 97,\n
\"max\": 164,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          164,\n          162,\n          133\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Rating\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.5827139568909908,\n        \"min\": 7.0,\n        \"max\": 9.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          7.8,\n
7.4,\n          8.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Votes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 490143,\n        \"min\": 157026,\n        \"max\": 1791916,\
n        \"num_unique_values\": 10,\n        \"samples\": [\n
1222645,\n          935408,\n          323118\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Revenue (Millions)\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
161.91619246799664,\n        \"min\": 424.65,\n        \"max\":
936.63,\n        \"num_unique_values\": 10,\n        \"samples\": [\n
448.13,\n          760.51,\n          532.17\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Metascore\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8.275801404630785,\n        \"min\": 59.0,\n        \"max\": 83.0,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          78.0,\n
83.0,\n          65.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
top_10 = data.nlargest(10,'Revenue (Millions)')
[['Title','Director','Revenue (Millions)']].set_index('Title')

sns.barplot(top_10['Revenue (Millions)'],top_10.index)
plt.title("Display Top 10 Highest Revenue Movie Titles")
plt.show()
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
/tmp/ipython-input-2621774543.py in <cell line: 0>()
----> 1 sns.barplot(top_10['Revenue (Millions)'],top_10.index)
      2 plt.title("Display Top 10 Highest Revenue Movie Titles")
      3 plt.show()

TypeError: barplot() takes from 0 to 1 positional arguments but 2 were
given
```

18. Find Average Rating of Movies Year-wise

```
data.columns = data.columns.str.strip()

avg_rating_year = data.groupby('Year')['Rating'].mean()

avg_rating_year
```

```
Year
2006    7.143902
2007    7.140909
2008    6.708333
2009    6.911111
2010    6.894737
2011    6.945614
2012    6.933871
2013    6.832143
2014    6.822581
```

```
2015    6.674312
2016    6.644444
Name: Rating, dtype: float64
```

20. Classify Movies Based on Ratings [Good,Better and Best]

```
data.columns

Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors',
'Year',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore', 'temp'],
      dtype='object')

def rating(rating):
    if rating>=7.0:
        return 'Excellent'
    elif rating>=6.0:
        return 'Good'
    else:
        return 'Average'

data['rating_cat']=data['Rating'].apply(rating)

data.head(1)
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 838,\n  \"fields\":
[\n    {\n      \"column\": \"Rank\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 286,\n        \"min\": 1,\n
\"max\": 1000,\n        \"num_unique_values\": 838,\n
\"samples\": [\n          239,\n          982,\n          90\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 837,\n        \"samples\": [\n          \"The
Disappointments Room\",\n          \"Annie\",\n          \"The
Accountant\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Genre\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 189,\n        \"samples\": [\n
\"Drama,Fantasy,Musical\",\n          \"Drama,Thriller,War\",\n
\"Action,Adventure,Drama\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Description\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 838,\n
\"samples\": [\n          \"The adventures of Gustave H, a legendary
concierge at a famous hotel from the fictional Republic of Zubrowka
between the first and second World Wars, and Zero Moustafa, the lobby
boy who becomes his most trusted friend.\",\n          \"A foster kid,
who lives with her mean foster mom, sees her life change when business
tycoon and New York mayoral candidate Will Stacks makes a thinly-
veiled campaign move and takes her in.\",\n          \"As a math
savant uncooks the books for a new client, the Treasury Department
closes in on his activities and the body count starts to rise.\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Director\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 524,\n        \"samples\": [\n
\"Andrey Kravchuk\",\n          \"John R. Leonetti\",\n
\"Damien Chazelle\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Actors\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 834,\n        \"samples\": [\n
\"Elijah Wood, Brittany Murphy, Hugh Jackman, Robin Williams\",\n
\"Evan Rachel Wood, Jim Sturgess, Joe Anderson, Dana Fuchs\",\n
\"Vin Diesel, Rose Leslie, Elijah Wood, \\u00d3lafur Darri \\
u00d3lafsson\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 3,\n        \"min\": 2006,\n        \"max\": 2016,\n
\"num_unique_values\": 11,\n        \"samples\": [\n          2008,\n
2014,\n          2010\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Runtime (Minutes)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 18,\n        \"min\": 66,\n
\"max\": 187,\n        \"num_unique_values\": 90,\n
\"samples\": [\n          130,\n          144,\n          110\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Rating\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.8777538418027501,\n        \"min\": 1.9,\n        \"max\": 9.0,\n
\"num_unique_values\": 50,\n        \"samples\": [\n          6.6,\n
3.9,\n          6.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Votes\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 193099,\n        \"min\": 178,\n        \"max\": 1791916,\n
\"num_unique_values\": 837,\n        \"samples\": [\n          4895,\n
27312,\n          162122\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Revenue (Millions)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 104.5202265333547,\n
\"min\": 0.0,\n        \"max\": 936.63,\n
\"num_unique_values\": 789,\n        \"samples\": [\n          93.95,\
n          3.44,\n          26.62\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Metascore\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
16.95241649434648,\n        \"min\": 11.0,\n        \"max\": 100.0,\n
\"num_unique_values\": 82,\n        \"samples\": [\n          57.0,\n
76.0,\n          36.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"temp\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating_cat\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Excellent\",\n          \"Good\",\n          \"Average\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

21. Count Number of Action Movies

```python
list1=[]
for value in data['Genre']:
    list1.append(value.split(','))

data['temp']=list1

genre=input("Enter Genre you want to count : ").title()
count=0
for value in data['temp']:
    if genre in value:
        count=count+1
print("Total Count is",count)
```