In [1]:

```
#import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Data Understanding

1. Loading csv file in juypter notebook
2. Dataframe shape
3. Head and Tail
4. dtypes
5. Describe

In [4]:

```
data = pd.read_csv(r"C:\Users\tapas\OneDrive\Desktop\imdb_top_1000.csv")
data
```

Out[4]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min
...	...	...	...	...	...
995	https://m.media-amazon.com/images/M/MV5BNGEwMT...	Breakfast at Tiffany's	1961	A	115 min
996	https://m.media-amazon.com/images/M/MV5BODk3Yj...	Giant	1956	G	201 min
997	https://m.media-amazon.com/images/M/MV5BM2U3Yz...	From Here to Eternity	1953	Passed	118 min
998	https://m.media-amazon.com/images/M/MV5BZTBmMj...	Lifeboat	1944	NaN	97 min

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	
999	https://m.media-amazon.com/images/M/MV5BMTY5OD...	The 39 Steps	1935	NaN	86 min	My T

1000 rows × 16 columns

```
In [5]: data.shape
```

```
Out[5]: (1000, 16)
```

```
In [6]: data.head()
```

Out[6]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genr
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Dram
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime Dram
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action Crime Dram
3	https://m.media-amazon.com/images/M/MV5BMWMwMG...	The Godfather: Part II	1974	A	202 min	Crime Dram
4	https://m.media-amazon.com/images/M/MV5BMWU4N2...	12 Angry Men	1957	U	96 min	Crime Dram

```
In [7]: data.tail()
```

Out[7]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Ge
995	https://m.media-amazon.com/images/M/MV5BNGEwMT...	Breakfast at Tiffany's	1961	A	115 min	Come Dra Roma
996	https://m.media-amazon.com/images/M/MV5BODk3Yj...	Giant	1956	G	201 min	Dra West
997	https://m.media-amazon.com/images/M/MV5BM2U3Yz...	From Here to Eternity	1953	Passed	118 min	Dra Romai
998	https://m.media-amazon.com/images/M/MV5BZTBmMj...	Lifeboat	1944	NaN	97 min	Dra
999	https://m.media-amazon.com/images/M/MV5BMTY5OD...	The 39 Steps	1935	NaN	86 min	Cri Myst Thr

In [8]:

```
data.dtypes
```

Out[8]:

```
Poster_Link       object
Series_Title      object
Released_Year     object
Certificate       object
Runtime           object
Genre             object
IMDB_Rating      float64
Overview          object
Meta_score       float64
Director          object
Star1             object
Star2             object
Star3             object
Star4             object
No_of_Votes        int64
Gross             object
dtype: object
```

In [9]:

```
data.describe()
```

Out[9]:

	IMDB_Rating	Meta_score	No_of_Votes
count	1000.000000	843.000000	1.000000e+03
mean	7.949300	77.971530	2.736929e+05
std	0.275491	12.376099	3.273727e+05
min	7.600000	28.000000	2.508800e+04
25%	7.700000	70.000000	5.552625e+04
50%	7.900000	79.000000	1.385485e+05
75%	8.100000	87.000000	3.741612e+05
max	9.300000	100.000000	2.343110e+06

In [10]:
```python
data.describe(include='object')
```

Out[10]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Ge
count	1000	1000	1000	899	1000	1
unique	1000	999	100	16	140	2
top	https://m.media-amazon.com/images/M/MV5BMDFkYT...	Drishyam	2014	U	100 min	Dra
freq	1	2	32	234	23	

Data Preparation

1. Checking for null values
2. Converting dtypes
3. Dropping irrelevant columns and rows
4. Renaming columns

In [11]:
```python
data.isnull().sum()
```

Out[11]:

```
Poster_Link        0
Series_Title       0
Released_Year      0
Certificate      101
Runtime            0
Genre              0
IMDB_Rating        0
Overview           0
Meta_score       157
Director           0
Star1              0
Star2              0
Star3              0
Star4              0
No_of_Votes        0
Gross            169
dtype: int64
```

In [12]:

```
print(data['Gross'].head(1))
data['Gross'] = data['Gross'].str.replace(',' , '')
print(data['Gross'].head(1))
data['Gross'] = data['Gross'].astype('float64')
data['Gross'] = data['Gross'].replace('Not Rated', 0)
```

```
0    28,341,469
Name: Gross, dtype: object
0    28341469
Name: Gross, dtype: object
```

In [14]:

```
data['Gross'] = data['Gross'].replace(np.nan, 0)
```

In [15]:

```
data['Gross'] = data['Gross'].astype('int64')
```

In [16]:

```
data['Gross'].dtype
```

Out[16]:

```
dtype('int64')
```

In [17]:

```
data.drop(['Poster_Link', 'Overview'], axis = 1)
```

Out[17]:

	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Meta_score	Dire
0	The Shawshank Redemption	1994	A	142 min	Drama	9.3	80.0	F Dara
1	The Godfather	1972	A	175 min	Crime, Drama	9.2	100.0	Fr Cop
2	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	84.0	Christo N
3	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.0	90.0	Fr Cop
4	12 Angry Men	1957	U	96 min	Crime, Drama	9.0	96.0	Si Lu
...	...	...	...	...	...	...	...	
995	Breakfast at Tiffany's	1961	A	115 min	Comedy, Drama, Romance	7.6	76.0	I Edv
996	Giant	1956	G	201 min	Drama, Western	7.6	84.0	Ge Ste
997	From Here to Eternity	1953	Passed	118 min	Drama, Romance, War	7.6	85.0	Zinner
998	Lifeboat	1944	NaN	97 min	Drama, War	7.6	78.0	A Hitch
999	The 39 Steps	1935	NaN	86 min	Crime, Mystery, Thriller	7.6	93.0	A Hitch

1000 rows × 14 columns

In [18]:

```
data = data.rename(columns={"Series_Title": "Movies_Title"})
```

Performing EDA

In [19]:

```
data.corr()
```

```
C:\Users\tapas\AppData\Local\Temp\ipykernel_15340\2627137660.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
sion, it will default to False. Select only valid columns or specify the value of
numeric_only to silence this warning.
  data.corr()
```

Out[19]:

	IMDB_Rating	Meta_score	No_of_Votes	Gross
IMDB_Rating	1.000000	0.268531	0.494979	0.082381
Meta_score	0.268531	1.000000	-0.018507	-0.053659
No_of_Votes	0.494979	-0.018507	1.000000	0.602128
Gross	0.082381	-0.053659	0.602128	1.000000

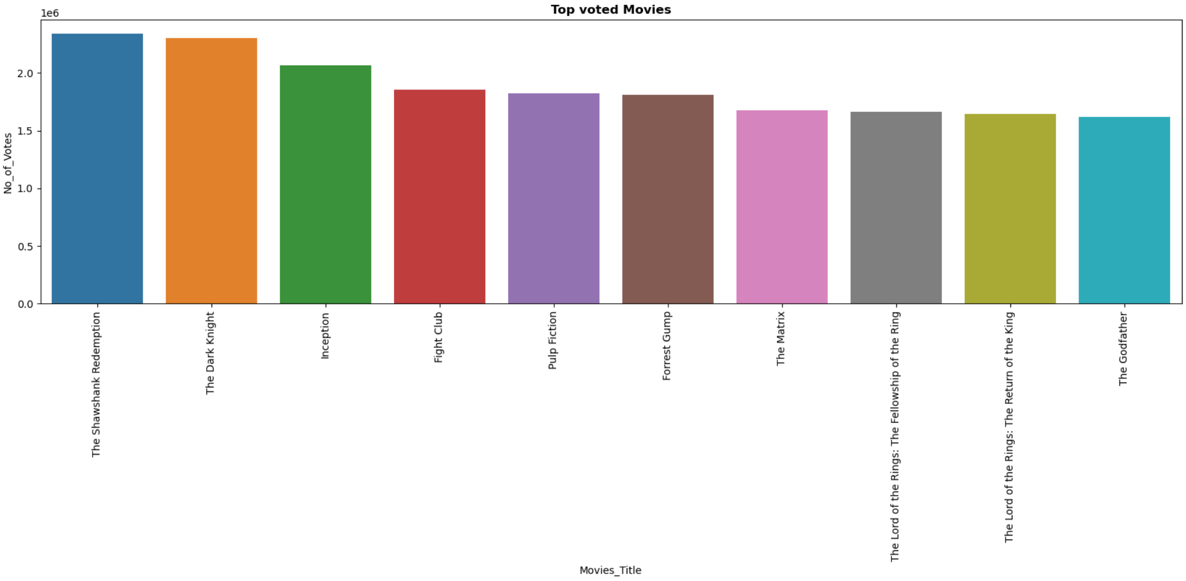
Top voted Movies

In [21]:

```
top_voted = data.sort_values(['No_of_Votes'], ascending = False)
```

In [22]:

```
fig,axs=plt.subplots(figsize=(20,5))
sns.barplot(x = top_voted['Movies_Title'][:10], y = top_voted['No_of_Votes'][:10])
plt.title("Top voted Movies", weight = "bold")
plt.xticks(rotation=90)
plt.show()
```

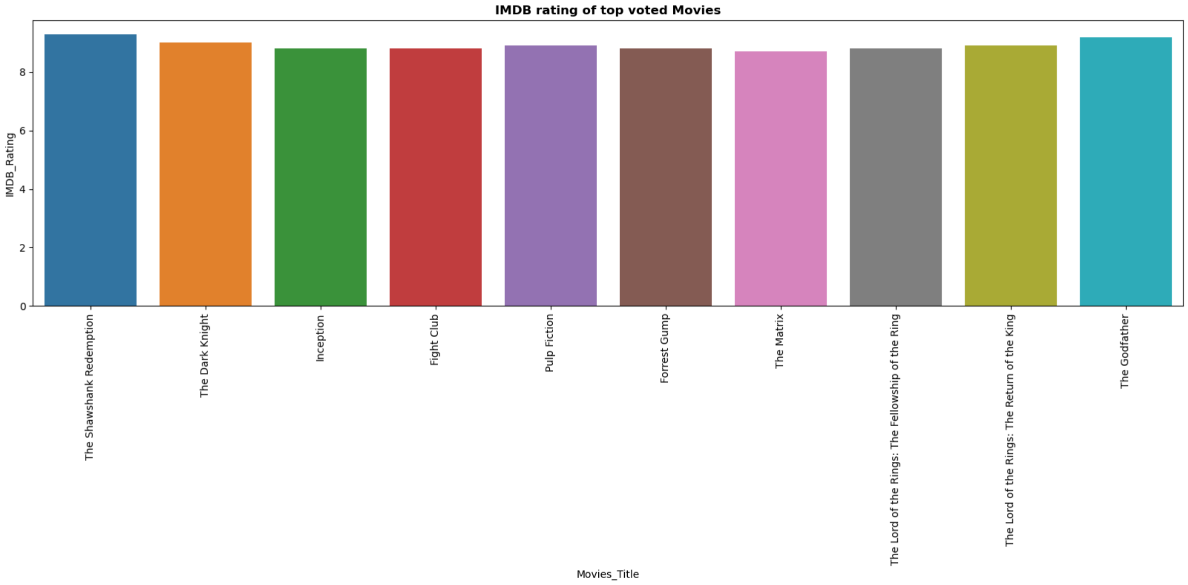


IMDB rating of top voted movies

IMDb's scores are based on users' ratings

In [23]:

```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = top_voted['Movies_Title'][:10], y = top_voted['IMDB_Rating'][:10])
plt.title("IMDB rating of top voted Movies", weight = "bold")
plt.xticks(rotation=90)
plt.show()
```



In [24]:

```
#top 10 most rated movies
data.sort_values(by = 'No_of_Votes', ascending = False).head(10)['Movies_Title']
```

Out[24]:

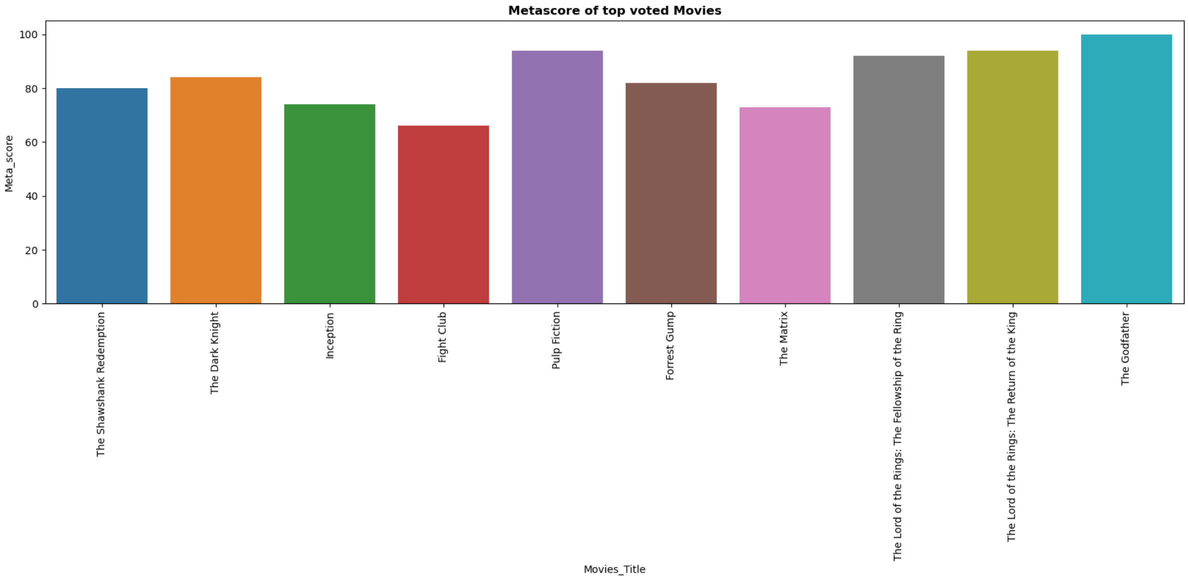
```
0                             The Shawshank Redemption
2                                      The Dark Knight
8                                            Inception
9                                           Fight Club
6                                         Pulp Fiction
11                                        Forrest Gump
14                                          The Matrix
10    The Lord of the Rings: The Fellowship of the Ring
5         The Lord of the Rings: The Return of the King
1                                        The Godfather
Name: Movies_Title, dtype: object
```

Metascore of top rated movies

Metacritic's main scores are based on reviewers' ratings

In [25]:

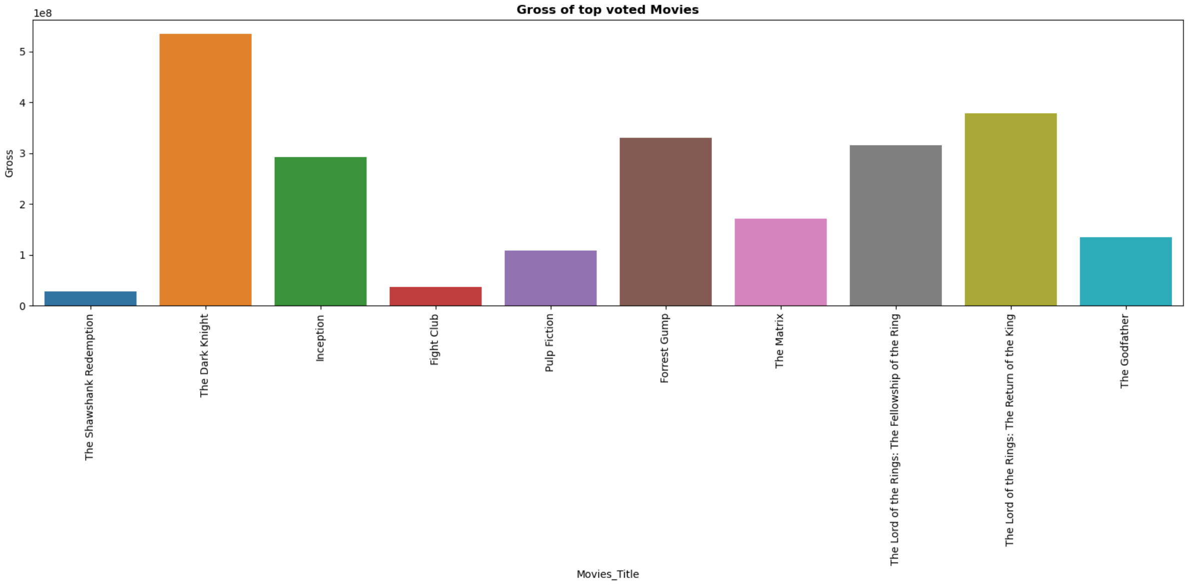
```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = top_voted['Movies_Title'][:10], y = top_voted['Meta_score'][:10])
plt.title("Metascore of top voted Movies", weight = "bold")
plt.xticks(rotation=90)
plt.show()
```



Gross of top voted Movies

In [26]:

```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = top_voted['Movies_Title'][:10], y = top_voted['Gross'][:10])
plt.title("Gross of top voted Movies", weight = "bold")
plt.xticks(rotation=90)
plt.show()
```



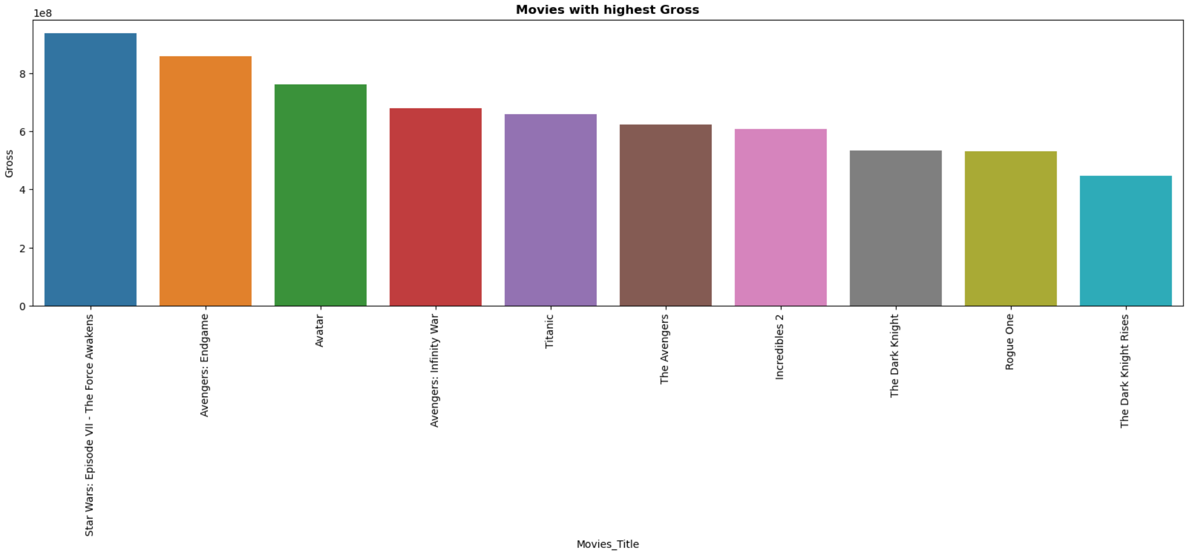
Top movies by Gross

In [27]:

```
highest_earning = data.sort_values(['Gross'], ascending = False)
```

In [28]:

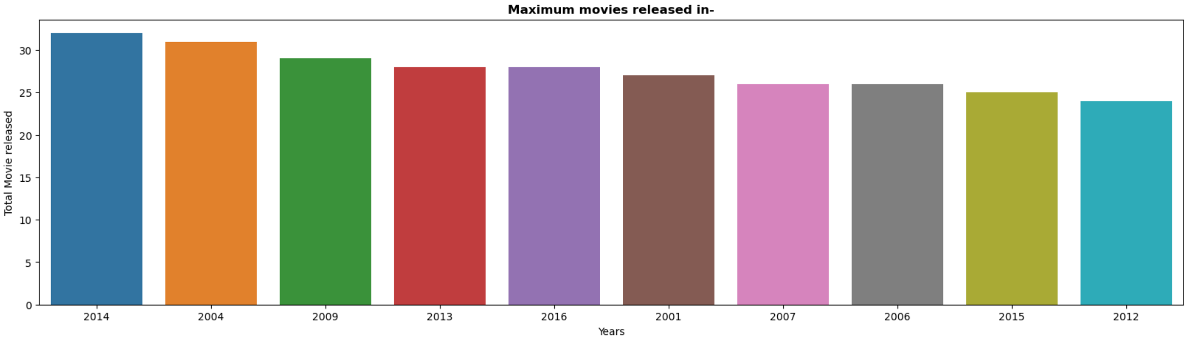
```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = highest_earning['Movies_Title'][:10], y = highest_earning['Gross'][
plt.title("Movies with highest Gross", weight = "bold")
plt.xticks(rotation=90)
plt.show()
```



Maximum movies released in year

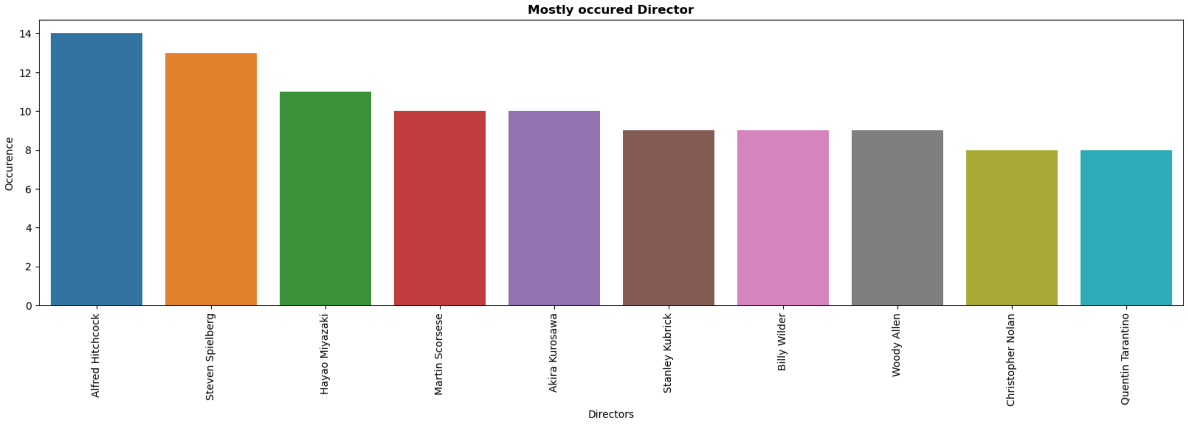
In [29]:

```
fig,axs = plt.subplots(figsize = (20,5))
sns.barplot(x = data['Released_Year'].value_counts()[:10].index, y = data['Releasec
plt.title("Maximum movies released in-", weight = "bold")
plt.xlabel("Years")
plt.ylabel("Total Movie released")
plt.show()
```



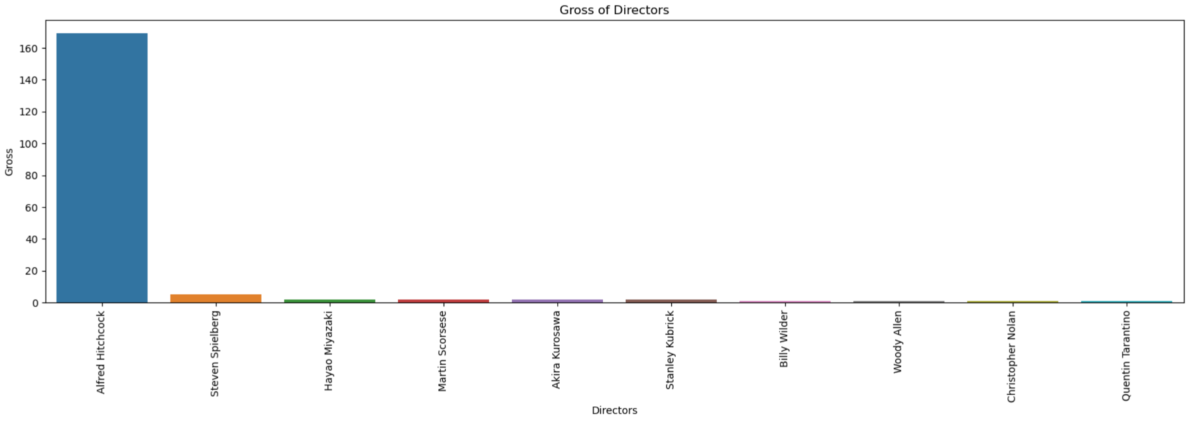
Mostly occured Director

In [30]:
```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = data['Director'].value_counts()[:10].index, y = data['Director'].va
plt.title("Mostly occured Director", weight = "bold")
plt.xlabel("Directors")
plt.ylabel("Occurence")
plt.xticks(rotation = 90)
plt.show()
```



Directors with highest Gross

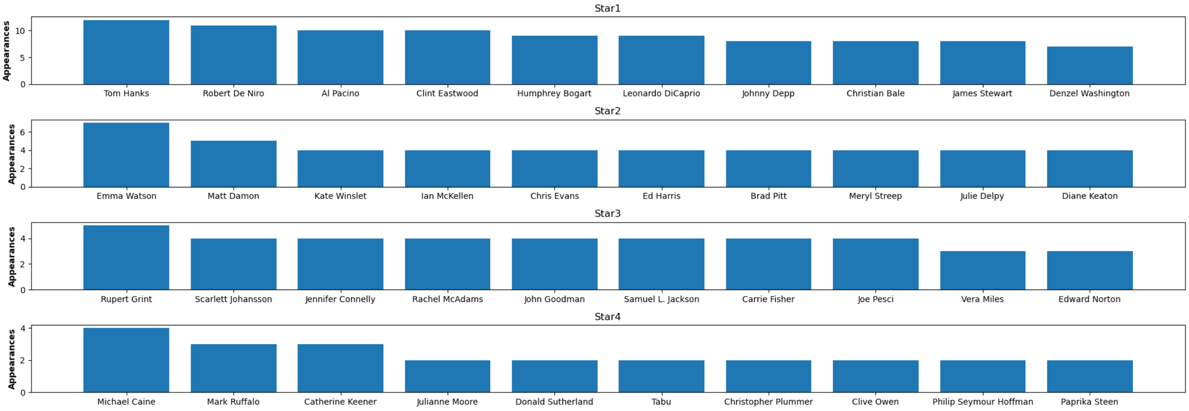
In [31]:
```
fig,axs=plt.subplots(figsize = (20,5))
sns.barplot(x = data['Director'].value_counts()[:10].index, y = data['Gross'].value
plt.title("Gross of Directors")
plt.xlabel("Directors")
plt.xticks(rotation = 90)
plt.show()
```



Stars with most occurences in Movies

In [32]:

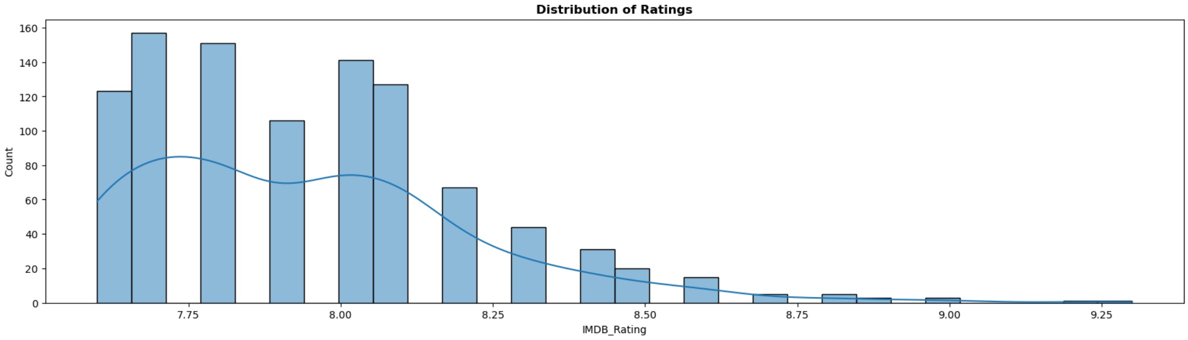
```
stars=['Star1','Star2','Star3','Star4']
fig,axs=plt.subplots(4,1,figsize=(20,7))
ax=0
for x in stars:
    axs[ax].bar(data[x].value_counts()[:10].index,data[x].value_counts()[:10])
    axs[ax].set_title(x)
    axs[ax].set_ylabel("Appearances", weight = "bold")
    ax+=1
    plt.tight_layout()
```



IMDB rating distribution

In [33]:

```
fig,axs=plt.subplots(figsize=(20,5))
sns.histplot(data['IMDB_Rating'],bins=30, kde = True)
plt.title("Distribution of Ratings", weight = "bold")
plt.show()
```



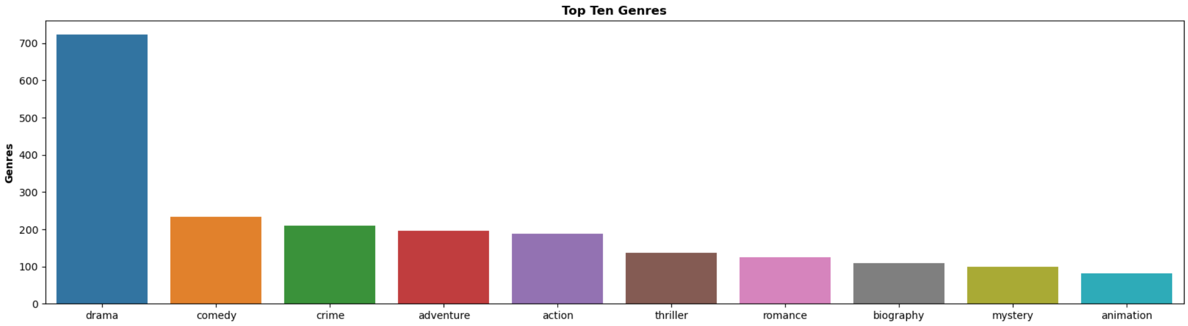
Top 10 Genres

In [34]:

```
from collections import Counter
genre=[]
for x in data['Genre']:
    for y in x.split(','):
        genre.append(y.strip().lower())

count=Counter(genre)
count=count.most_common()[:10]
x,y=map(list,zip(*count))

fig,axs=plt.subplots(figsize=(20,5))
g=sns.barplot(y=y,x=x)
g.set_ylabel("Genres", weight = "bold")
g.set_title("Top Ten Genres", weight = "bold")
plt.show()
```


Top Ten Genres
Genres
drama
comedy
crime
adventure
action
thriller
romance
biography
mystery
animation