- Importing the libries

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

- Loading the dataset

In [3]:
```
df= pd.read_csv("movie_metadata.csv")
```

In [4]:
```
df.shape
```

Out[4]:
```
(5043, 28)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   color                      5024 non-null   object 
 1   director_name              4939 non-null   object 
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object 
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object 
 10  actor_1_name               5036 non-null   object 
 11  movie_title                5043 non-null   object 
 12  num_voted_users            5043 non-null   int64  
 13  cast_total_facebook_likes  5043 non-null   int64  
 14  actor_3_name               5020 non-null   object 
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object 
 17  movie_imdb_link            5043 non-null   object 
 18  num_user_for_reviews       5022 non-null   float64
 19  language                   5031 non-null   object 
 20  country                    5038 non-null   object 
 21  content_rating             4740 non-null   object 
 22  budget                     4551 non-null   float64
 23  title_year                 4935 non-null   float64
 24  actor_2_facebook_likes     5030 non-null   float64
 25  imdb_score                 5043 non-null   float64
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64  
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

In [6]:
```
df.describe
```

Out[6]:
```
<bound method NDFrame.describe of       color      director_name  num_critic_for_r
iews  duration  \
0     Color      James Cameron                    723.0     178.0
1     Color     Gore Verbinski                    302.0     169.0
2     Color         Sam Mendes                    602.0     148.0
3     Color  Christopher Nolan                    813.0     164.0
4       NaN        Doug Walker                      NaN       NaN
...     ...                ...                    ...       ...
5038  Color        Scott Smith                      1.0      87.0
```

```
5039  Color               NaN                    43.0      43.0
5040  Color   Benjamin Roberds                   13.0      76.0
5041  Color        Daniel Hsia                   14.0     100.0
5042  Color           Jon Gunn                   43.0      90.0

      director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                         0.0                   855.0  Joel David Moore
1                       563.0                  1000.0     Orlando Bloom
2                         0.0                   161.0      Rory Kinnear
3                     22000.0                 23000.0    Christian Bale
4                       131.0                     NaN        Rob Walker
...                       ...                     ...               ...
5038                      2.0                   318.0     Daphne Zuniga
5039                      NaN                   319.0     Valorie Curry
5040                      0.0                     0.0     Maxwell Moody
5041                      0.0                   489.0     Daniel Henney
5042                     16.0                    16.0  Brian Herzlinger

      actor_1_facebook_likes        gross                           genres  \
0                     1000.0  760505847.0  Action|Adventure|Fantasy|Sci-Fi
1                    40000.0  309404152.0         Action|Adventure|Fantasy
2                    11000.0  200074175.0        Action|Adventure|Thriller
3                    27000.0  448130642.0                  Action|Thriller
4                      131.0          NaN                      Documentary
...                      ...          ...                              ...
5038                   637.0          NaN                     Comedy|Drama
5039                   841.0          NaN     Crime|Drama|Mystery|Thriller
5040                     0.0          NaN            Drama|Horror|Thriller
5041                   946.0      10443.0             Comedy|Drama|Romance
5042                    86.0      85222.0                      Documentary

      ... num_user_for_reviews language  country  content_rating       budget  \
0     ...               3054.0  English      USA           PG-13  237000000.0
1     ...               1238.0  English      USA           PG-13  300000000.0
2     ...                994.0  English       UK           PG-13  245000000.0
3     ...               2701.0  English      USA           PG-13  250000000.0
4     ...                  NaN      NaN      NaN             NaN          NaN
...   ...                  ...      ...      ...             ...          ...
5038  ...                  6.0  English   Canada             NaN          NaN
5039  ...                359.0  English      USA           TV-14          NaN
5040  ...                  3.0  English      USA             NaN       1400.0
5041  ...                  9.0  English      USA           PG-13          NaN
5042  ...                 84.0  English      USA              PG       1100.0

      title_year actor_2_facebook_likes imdb_score  aspect_ratio  \
0         2009.0                  936.0        7.9          1.78
1         2007.0                 5000.0        7.1          2.35
2         2015.0                  393.0        6.8          2.35
3         2012.0                23000.0        8.5          2.35
4            NaN                   12.0        7.1           NaN
...          ...                    ...        ...           ...
5038      2013.0                  470.0        7.7           NaN
5039         NaN                  593.0        7.5         16.00
5040      2013.0                    0.0        6.3           NaN
5041      2012.0                  719.0        6.3          2.35
5042      2004.0                   23.0        6.6          1.85

     movie_facebook_likes
0                   33000
1                       0
2                   85000
3                  164000
4                       0
...                   ...
5038                   84
5039                32000
5040                   16
5041                  660
5042                  456
```

```
[5043 rows x 28 columns]>
```

In [7]:
```
df.describe()
```

Out[7]:

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_fa
count	4993.000000	5028.000000	4939.000000	5020.000000	
mean	140.194272	107.201074	686.509212	645.009761	
std	121.601675	25.197441	2813.328607	1665.041728	
min	1.000000	7.000000	0.000000	0.000000	
25%	50.000000	93.000000	7.000000	133.000000	
50%	110.000000	103.000000	49.000000	371.500000	
75%	195.000000	118.000000	194.500000	636.000000	
max	813.000000	511.000000	23000.000000	23000.000000	6

In [8]:
```
df
```

Out[8]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebo
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	
...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	
5039	Color	NaN	43.0	43.0	NaN	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	
5041	Color	Daniel Hsia	14.0	100.0	0.0	
5042	Color	Jon Gunn	43.0	90.0	16.0	

5043 rows × 28 columns

Lets print the max. no of columns in the dataframe

In [9]:
```
pd.set_option("display.max.columns", None)
```

In [10]:
```
df
```

Out[10]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebo
0	Color	James Cameron	723.0	178.0	0.0	
1	Color	Gore Verbinski	302.0	169.0	563.0	
2	Color	Sam Mendes	602.0	148.0	0.0	
3	Color	Christopher Nolan	813.0	164.0	22000.0	
4	NaN	Doug Walker	NaN	NaN	131.0	
...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	
5039	Color	NaN	43.0	43.0	NaN	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	
5041	Color	Daniel Hsia	14.0	100.0	0.0	
5042	Color	Jon Gunn	43.0	90.0	16.0	

5043 rows × 28 columns

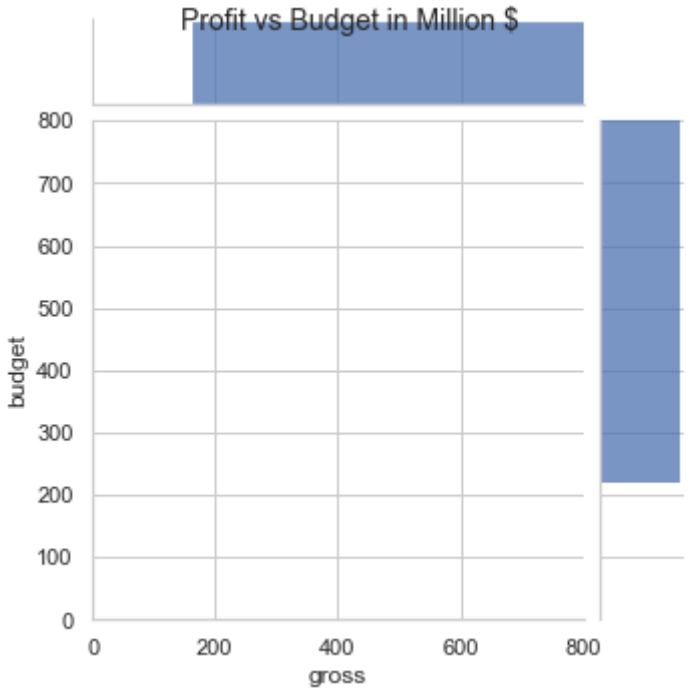
In [11]:
```
import seaborn as sns
```

In [12]:
```
import matplotlib.pyplot as plt
```

Plot profit vs budget

In [13]:
```
sns.set(style="whitegrid")
plot = sns.jointplot(df['gross'].astype(float), df['budget'].astype(float), height=5
plot.ax_marg_x.set_xlim(0, 800)
plot.ax_marg_y.set_ylim(0, 800)
plt.suptitle('Profit vs Budget in Million $', fontsize=14)

plt.show()
```

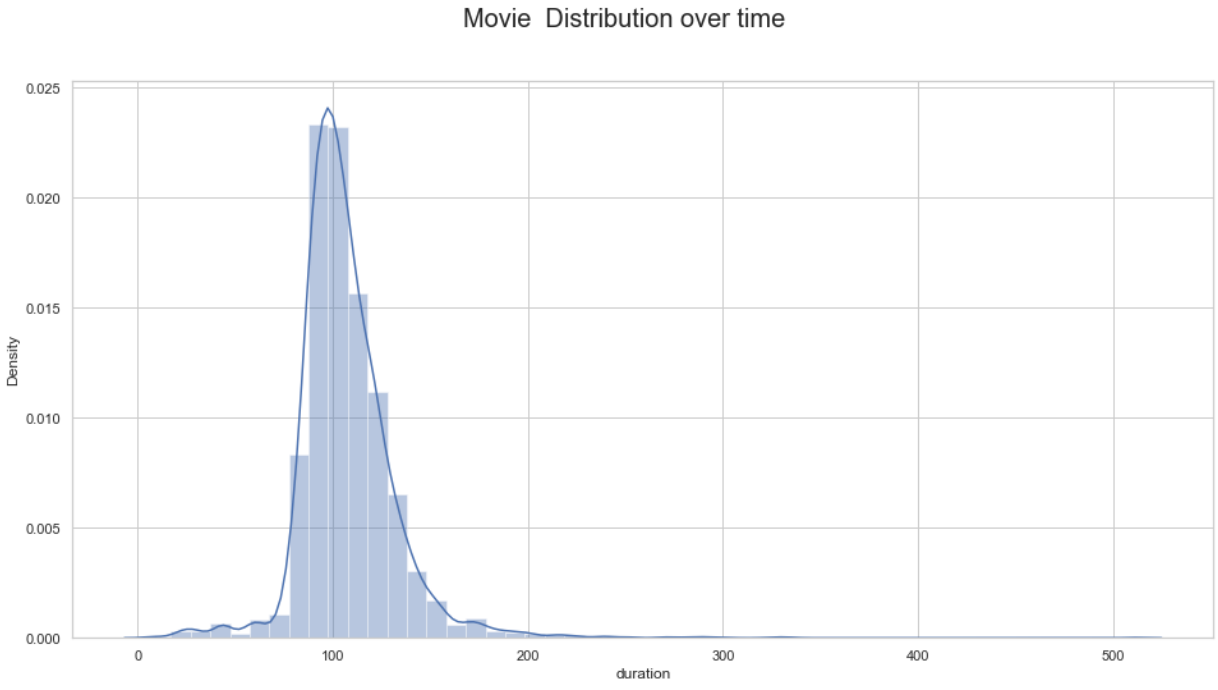
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
Pass the following variables as keyword args: x, y. From version 0.12, the only va
d positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

In [15]:
```
plt.figure(figsize=[16,8])
run = sns.distplot(df['duration'])
plt.suptitle("Movie  Distribution over time", fontsize=20)
plt.show()
```

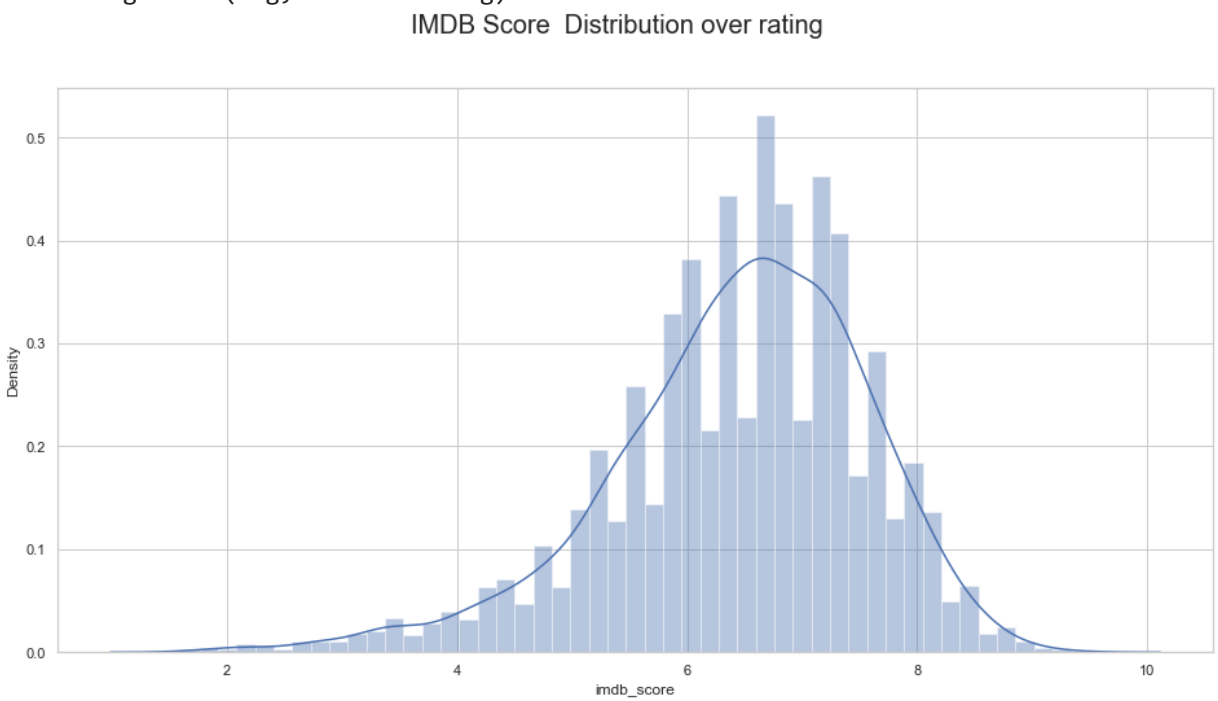
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWa
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [16]:
```
plt.figure(figsize=[16,8])
run = sns.distplot(df['imdb_score'])
plt.suptitle("IMDB Score  Distribution over rating", fontsize=20)
plt.show()
```

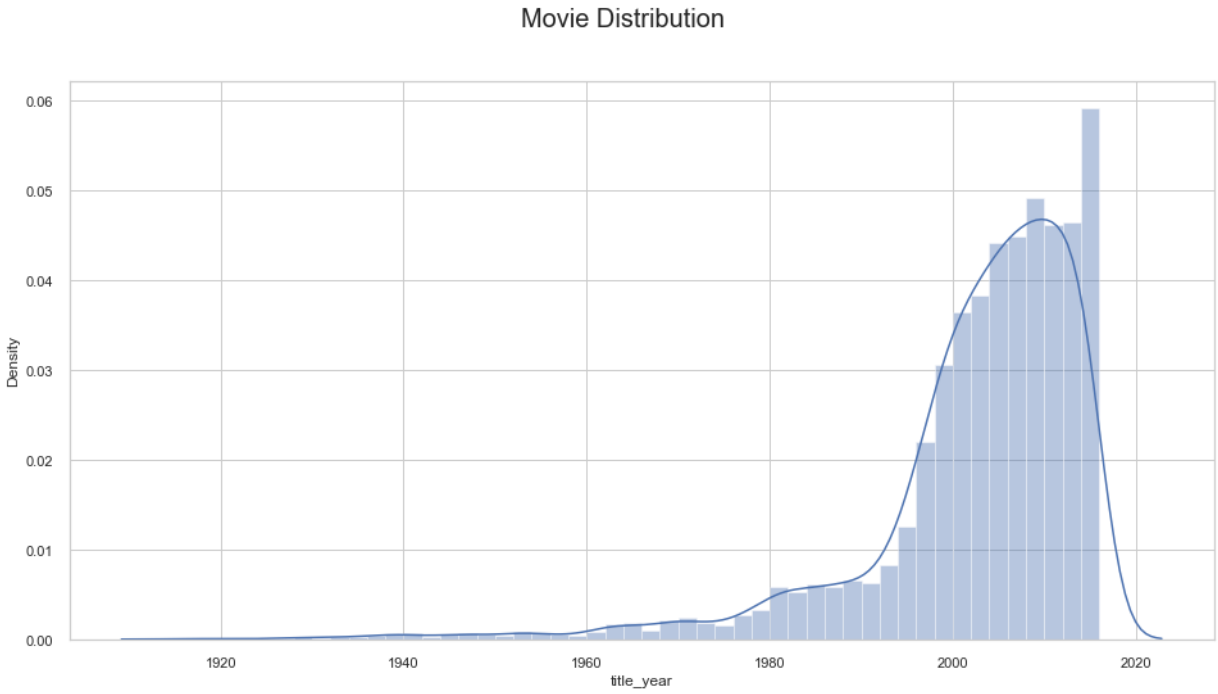
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWa
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
```

```
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [43]:
```
plt.figure(figsize=[16,8])
run = sns.distplot(df['title_year'])
plt.suptitle("Movie Distribution", fontsize=20)
plt.show()
```

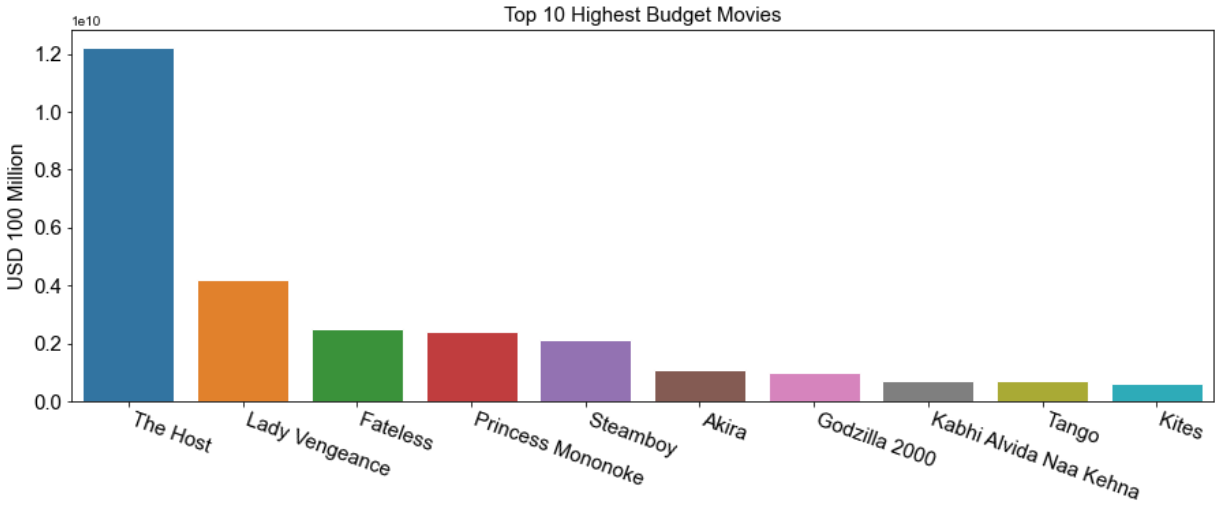
```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWa
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Stats

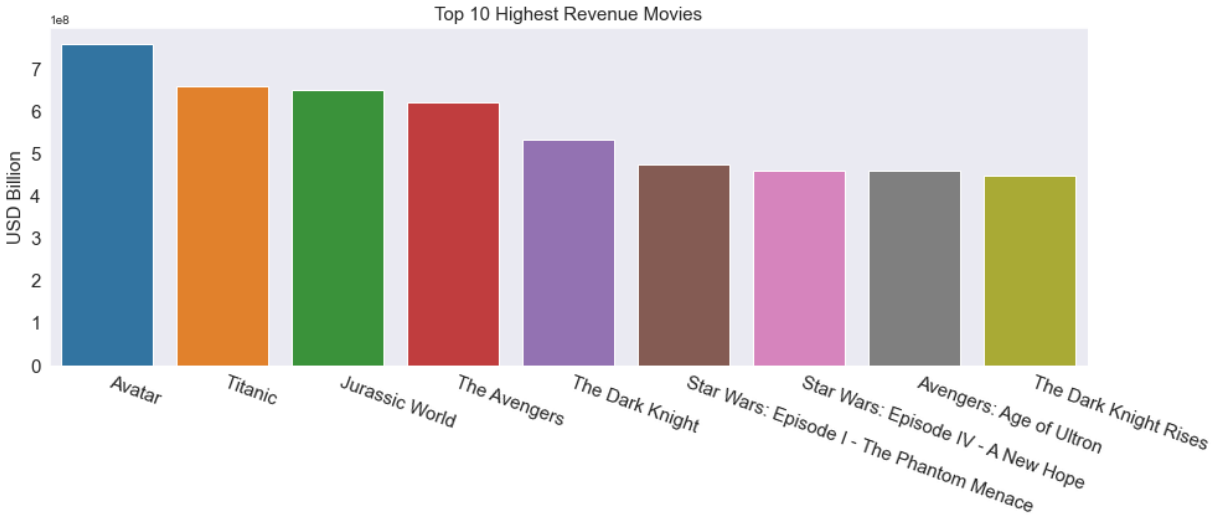
In [16]:
```
cols = ['movie_title', 'budget']
budget_df = df.sort_values('budget', ascending=False)[cols].set_index('movie_title')
top_10_budget = budget_df.head(10)
```

```
fig, ax = plt.subplots(figsize=(15,5))
sns.set_style('dark')
sns.barplot(data=top_10_budget, x=top_10_budget.index, y='budget');
plt.xticks(ha='left', rotation=-20, fontsize=15); plt.yticks(fontsize=15)
plt.xlabel(''); plt.ylabel('USD 100 Million', fontsize=15);
plt.title('Top 10 Highest Budget Movies', fontsize=15);
```



In [19]:

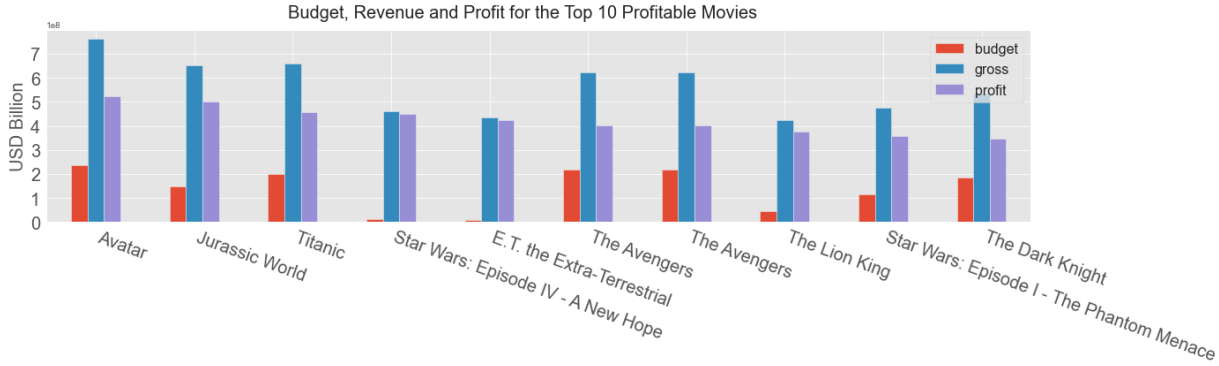
```
cols = ['movie_title', 'gross']
revenue_df = df.sort_values('gross', ascending=False)[cols].set_index('movie_title')
top_10_revenue = revenue_df.head(10)

fig, ax = plt.subplots(figsize=(15,5))
sns.set_style('dark')
sns.barplot(data=top_10_revenue, x=top_10_revenue.index, y='gross');
plt.xticks(ha='left', rotation=-20, fontsize=15); plt.yticks(fontsize=15)
plt.xlabel(''); plt.ylabel('USD Billion', fontsize=15);
plt.title('Top 10 Highest Revenue Movies', fontsize=15);
```



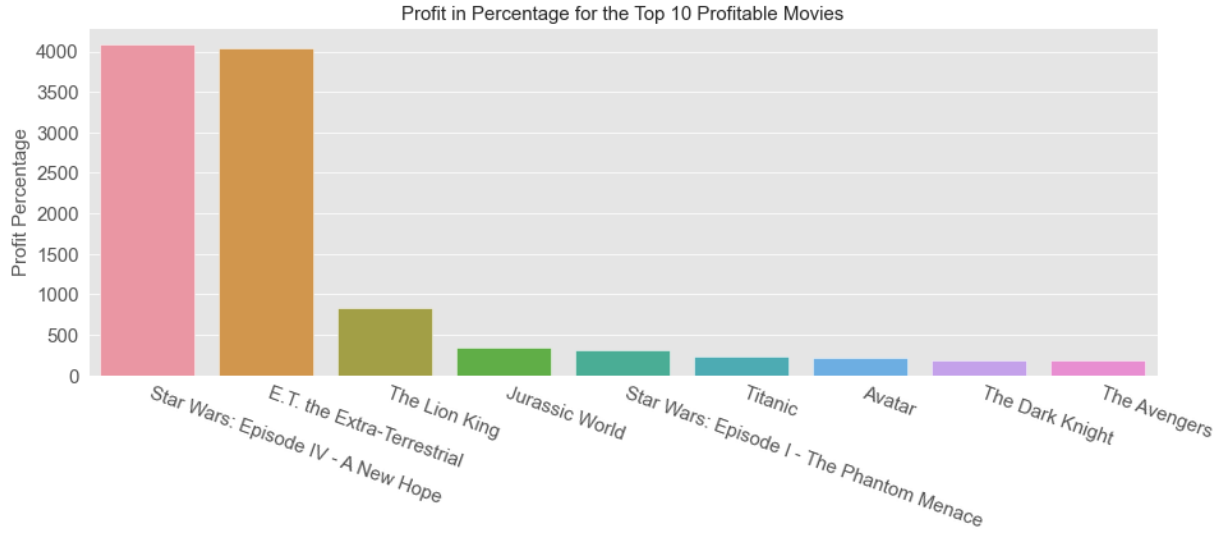
In [22]:

```
profits_ser = df['gross'] - df['budget']
profits_ser.name = 'profit'
profits_df = df.join(profits_ser)[['movie_title', 'budget', 'gross', 'profit']].sort
top_10_profits = profits_df.head(10).set_index('movie_title')

plt.style.use('ggplot')
top_10_profits.plot(kind='bar', figsize=(20,4), fontsize=20)
plt.ylabel('USD Billion', fontsize=20); plt.xlabel('')
plt.xticks(rotation=-20, ha='left')
plt.suptitle('Budget, Revenue and Profit for the Top 10 Profitable Movies', fontsize
plt.axes().legend(fontsize=16);
```

```
<ipython-input-22-5cfa4cb39f3e>:11: MatplotlibDeprecationWarning: Adding an axes usi
ng the same arguments as a previous axes currently reuses the earlier instance.  In
a future version, a new instance will always be created and returned.  Meanwhile, th
is warning can be suppressed, and the future behavior ensured, by passing a unique l
abel to each axes instance.
  plt.axes().legend(fontsize=16);
```

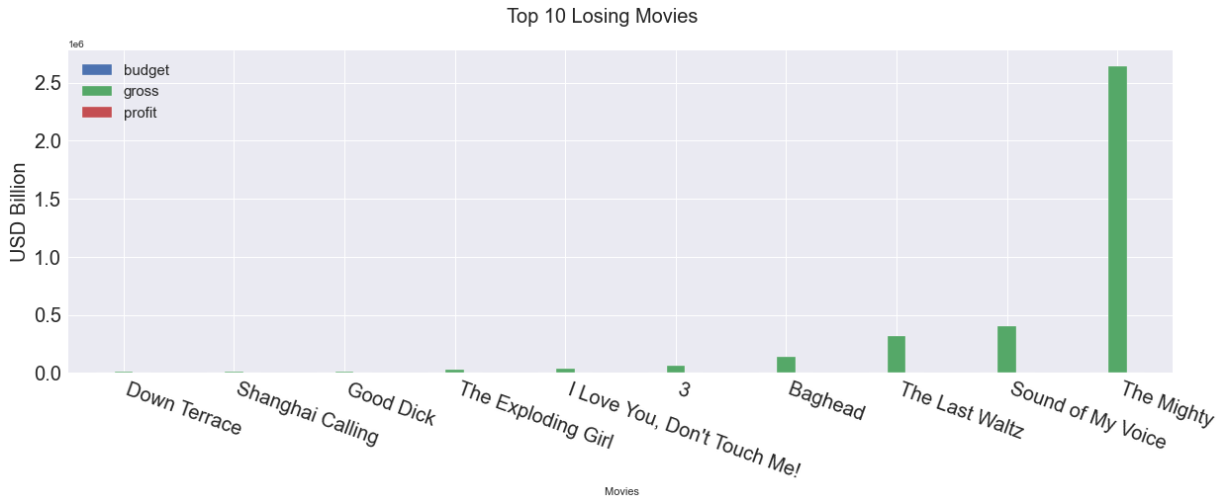


In [23]:
```
profits_ser_perc = (top_10_profits['profit'] / top_10_profits['budget'] * 100)
profits_ser_perc = profits_ser_perc.sort_values(ascending=False).to_frame().rename(c

fig, ax = plt.subplots(figsize=(15,5))
sns.set_style('dark')
sns.barplot(data=profits_ser_perc, x=profits_ser_perc.index, y='Profit Percentage')
plt.xticks(ha='left', rotation=-20, fontsize=15); plt.yticks(fontsize=15)
plt.xlabel(''); plt.ylabel('Profit Percentage', fontsize=15);
plt.title('Profit in Percentage for the Top 10 Profitable Movies', fontsize=15);
```



In [51]:
```
top_10_loss = profits_df[profits_df['gross'] > 0].tail(10).sort_values(['profit',  g

plt.style.use('seaborn')
top_10_loss.plot(kind='bar', figsize=(20,6), fontsize=20)
plt.ylabel('USD Billion', fontsize=20); plt.xlabel(' Movies')
plt.xticks(rotation=-20, ha='left')
plt.suptitle('Top 10 Losing Movies', fontsize=20)
plt.axes().legend(fontsize=15);
```

```
<ipython-input-51-36890f5ac418>:8: MatplotlibDeprecationWarning: Adding an axes us
g the same arguments as a previous axes currently reuses the earlier instance.  In a
future version, a new instance will always be created and returned.  Meanwhile, this
warning can be suppressed, and the future behavior ensured, by passing a unique labe
l to each axes instance.
  plt.axes().legend(fontsize=15);
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Shortcomings:

- The dataset was highly skewed towards large values due to large number of small or zero values. The lack of information for other language movies was evident when we we tried to answer the second question. There were many missing values in the object data type columns - genre, title, etc. These were filtered on the go during the analysis. Assuming these shortcomings were not present we can be confident with our results above and gain more insights by comparing other categorical features that were not explored.

In [ ]: