# IMDb - Exploratory Data Analysis (SQL + Python)

The primary objective of this exploratory data analysis project is to gain insights into the IMDb dataset using SQL and Python. By leveraging the power of structured query language for efficient data manipulation and Python's versatile libraries for in-depth analysis and visualization, we aim to discover valuable information about the movies featured in the dataset. Through this comprehensive analysis, we seek to unravel the intricacies of the movie industry, from genre trends to the impact of directors, actors and many more.

## Reading the dataset

In [231]:

```
# Importing all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlalchemy
from matplotlib import style
```

In [232]:

```
# Connecting with PostgreSQL
engine = sqlalchemy.create_engine('postgresql://postgres:Adi_1997@localhost:54
```

In [233]:

```
# Exploring the datset
df = pd.read_sql('IMDb',engine)
df.head()
```

Out[233]:

| | Title | Year | Certificate | Runtime | Genre | Rating | Meta_score | Director | Star1 | Star2 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | The Shawshank Redemption | 1994 | A | 142 | Drama | 9.3 | 80.0 | Frank Darabont | Tim Robbins | Morgan Freeman |
| 1 | The Godfather | 1972 | A | 175 | Crime, Drama | 9.2 | 100.0 | Francis Ford Coppola | Marlon Brando | Al Pacino |
| 2 | The Dark Knight | 2008 | UA | 152 | Action, Crime, Drama | 9.0 | 84.0 | Christopher Nolan | Christian Bale | Heath Ledger |
| 3 | The Godfather: Part II | 1974 | A | 202 | Crime, Drama | 9.0 | 90.0 | Francis Ford Coppola | Al Pacino | Robert De Niro |
| 4 | 12 Angry Men | 1957 | U | 96 | Crime, Drama | 9.0 | 96.0 | Sidney Lumet | Henry Fonda | Lee J. Cobb |

In [234]:

```
df.columns
```

Out[234]:

```
Index(['Title', 'Year', 'Certificate', 'Runtime', 'Genre', 'Rating',
       'Meta_score', 'Director', 'Star1', 'Star2', 'Star3', 'Star4', 'Votes',
       'Gross'],
      dtype='object')
```

# Movies

## Total number of movies in the dataset

In [235]:
```
query = ''' SELECT COUNT(*) AS Total_no_of_movies
            FROM "IMDb"
            '''
```

In [236]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[236]:

| | total_no_of_movies |
|---|---|
| 0 | 1000 |

## Top 10 highest-rated movies

In [237]:
```
query = ''' SELECT "Title", "Rating"
            FROM "IMDb"
            ORDER BY "Rating" DESC
            LIMIT 10'''
```

In [238]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[238]:

| | Title | Rating |
|---|---|---|
| 0 | The Shawshank Redemption | 9.3 |
| 1 | The Godfather | 9.2 |
| 2 | The Dark Knight | 9.0 |
| 3 | The Godfather: Part II | 9.0 |
| 4 | 12 Angry Men | 9.0 |
| 5 | Pulp Fiction | 8.9 |
| 6 | The Lord of the Rings: The Return of the King | 8.9 |
| 7 | Schindler's List | 8.9 |
| 8 | Fight Club | 8.8 |
| 9 | Inception | 8.8 |

# Actors and Directors

## Top 10 directors with the most movies in the dataset

In [239]:
```
query = ''' SELECT "Director" , COUNT(*) AS Total_movies
            FROM "IMDb"
            GROUP BY "Director"
            ORDER BY Total_movies DESC
            LIMIT 10
            '''
```

In [240]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[240]:

| | Director | total_movies |
|---|---|---|
| **0** | Alfred Hitchcock | 14 |
| **1** | Steven Spielberg | 13 |
| **2** | Hayao Miyazaki | 11 |
| **3** | Martin Scorsese | 10 |
| **4** | Akira Kurosawa | 10 |
| **5** | Billy Wilder | 9 |
| **6** | Woody Allen | 9 |
| **7** | Stanley Kubrick | 9 |
| **8** | David Fincher | 8 |
| **9** | Clint Eastwood | 8 |

## Movies in which 'Al Pacino' has appeared

In [13]:
```
query = ''' SELECT "Title"
            FROM "IMDb"
            WHERE "Star1" = 'Al Pacino' OR "Star2" = 'Al Pacino' OR "Star3" =
            '''
```

In [14]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[14]:

| | Title |
|---|---|
| **0** | The Godfather |
| **1** | The Godfather: Part II |
| **2** | Scarface |
| **3** | Heat |
| **4** | Scent of a Woman |
| **5** | Dog Day Afternoon |
| **6** | The Irishman |
| **7** | Carlito's Way |
| **8** | The Insider |
| **9** | Donnie Brasco |
| **10** | Glengarry Glen Ross |
| **11** | Serpico |
| **12** | The Godfather: Part III |

### Lead actor who has worked with 'Chirstopher Nolan' the most

In [70]:
```
query = ''' SELECT "Star1" AS Actor, COUNT(*) AS Total_movies
            FROM "IMDb"
            WHERE "Director" = 'Christopher Nolan'
            GROUP BY Actor
            ORDER BY Total_movies DESC
            LIMIT 1

            '''
```

In [71]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[71]:

| | actor | total_movies |
|---|---|---|
| **0** | Christian Bale | 4 |

## Genres

### Most popular genre in the dataset

In [241]:
```
query = ''' SELECT "Genre", COUNT (*) AS Total_movies
            FROM "IMDb"
            GROUP BY "Genre"
            ORDER BY Total_movies DESC
            LIMIT 15
             '''
```

In [242]:

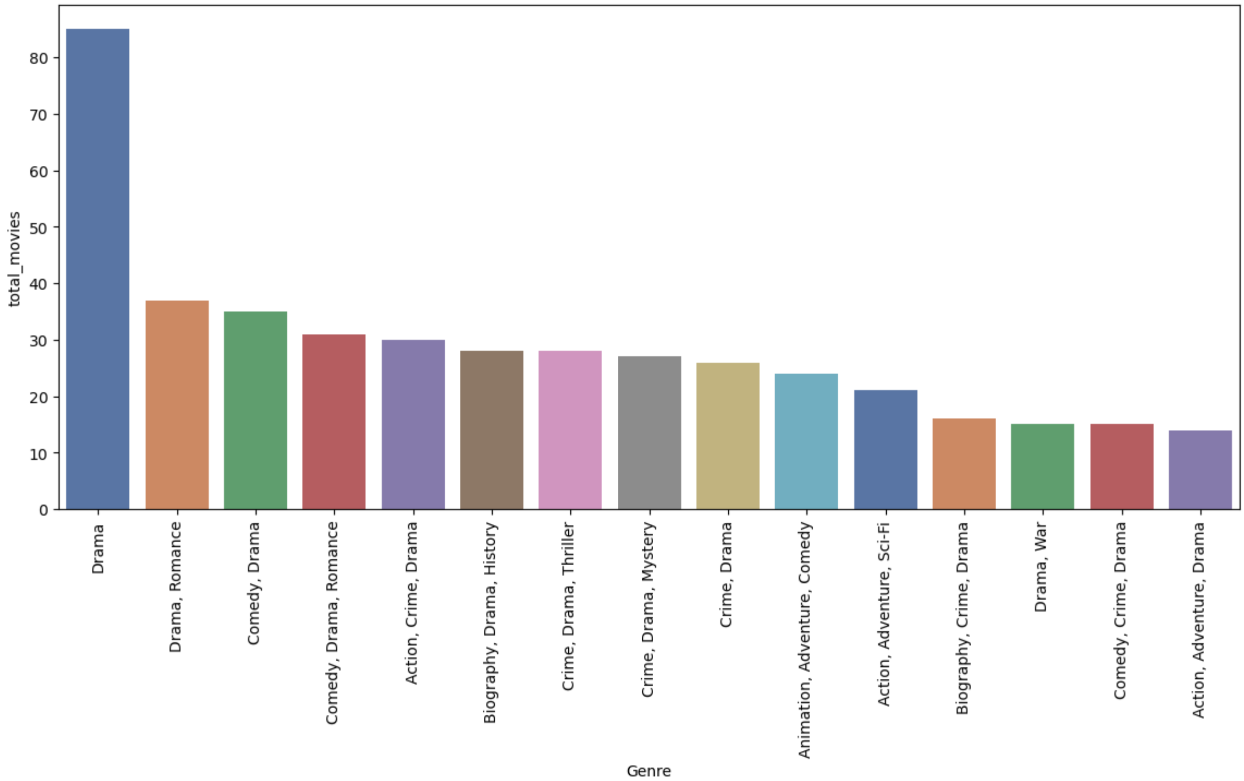
```
df = pd.read_sql_query(query,engine)
df
```

Out[242]:

| | Genre | total_movies |
|---|---|---|
| **0** | Drama | 85 |
| **1** | Drama, Romance | 37 |
| **2** | Comedy, Drama | 35 |
| **3** | Comedy, Drama, Romance | 31 |
| **4** | Action, Crime, Drama | 30 |
| **5** | Biography, Drama, History | 28 |
| **6** | Crime, Drama, Thriller | 28 |
| **7** | Crime, Drama, Mystery | 27 |
| **8** | Crime, Drama | 26 |
| **9** | Animation, Adventure, Comedy | 24 |
| **10** | Action, Adventure, Sci-Fi | 21 |
| **11** | Biography, Crime, Drama | 16 |
| **12** | Drama, War | 15 |
| **13** | Comedy, Crime, Drama | 15 |
| **14** | Action, Adventure, Drama | 14 |

In [243]:
```python
plt.figure(figsize= (14,6))
sns.barplot(x='Genre',y='total_movies', data=df, palette= 'deep')
plt.xticks(rotation = 90)
```

Out[243]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(0, 0, 'Drama'),
  Text(1, 0, 'Drama, Romance'),
  Text(2, 0, 'Comedy, Drama'),
  Text(3, 0, 'Comedy, Drama, Romance'),
  Text(4, 0, 'Action, Crime, Drama'),
  Text(5, 0, 'Biography, Drama, History'),
  Text(6, 0, 'Crime, Drama, Thriller'),
  Text(7, 0, 'Crime, Drama, Mystery'),
  Text(8, 0, 'Crime, Drama'),
  Text(9, 0, 'Animation, Adventure, Comedy'),
  Text(10, 0, 'Action, Adventure, Sci-Fi'),
  Text(11, 0, 'Biography, Crime, Drama'),
  Text(12, 0, 'Drama, War'),
  Text(13, 0, 'Comedy, Crime, Drama'),
  Text(14, 0, 'Action, Adventure, Drama')])
```




As we can see from the above barplot, Drama is the most popular genre in the dataset. Also, Drama is present in the movies which has multiple genres.

## Top 5 genres with the highest average ratings (Min 10 movies)

In [336]:
```sql
query = ''' SELECT DISTINCT "Genre" , CAST(AVG("Rating") AS DECIMAL(10,2)) as
            FROM "IMDb"
            GROUP BY "Genre"
            HAVING COUNT(*) >= 10
            ORDER BY Avg_Rating DESC
            LIMIT 5
            '''
```

In [337]:
```python
df = pd.read_sql_query(query,engine)
df
```

Out[337]:

| | Genre | avg_rating |
|---|---|---|
| **0** | Crime, Drama | 8.16 |
| **1** | Action, Adventure, Drama | 8.15 |
| **2** | Drama, War | 8.07 |
| **3** | Biography, Drama, History | 8.02 |
| **4** | Biography, Drama | 7.98 |

## Most profitable genres

In [333]:
```python
query = ''' SELECT DISTINCT "Genre", CAST(AVG("Gross") AS BIGINT) AS Avg_Gross
            FROM "IMDb"
            WHERE "Gross" <> 0
            GROUP BY "Genre"
            ORDER BY Avg_Gross DESC
            LIMIT 5
            '''
```

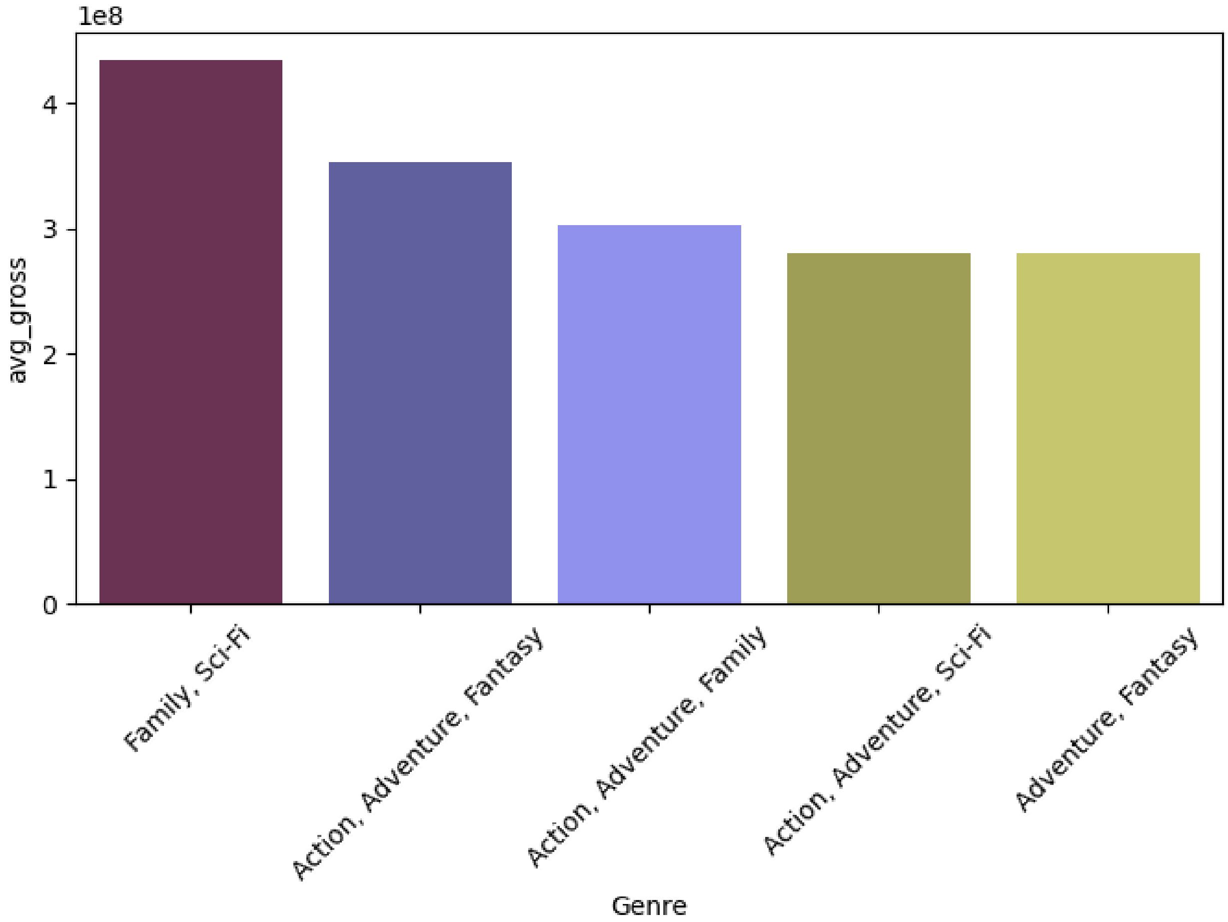
In [334]:
```python
df = pd.read_sql_query(query, engine)
df
```

Out[334]:

| | Genre | avg_gross |
|---|---|---|
| **0** | Family, Sci-Fi | 435110554 |
| **1** | Action, Adventure, Fantasy | 352723505 |
| **2** | Action, Adventure, Family | 301959197 |
| **3** | Action, Adventure, Sci-Fi | 280888546 |
| **4** | Adventure, Fantasy | 280685212 |

In [335]:
```
plt.figure(figsize= (8,4))
sns.barplot(x='Genre',y='avg_gross', data=df, palette= 'gist_stern')
plt.xticks(rotation = 45)
```

Out[335]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Family, Sci-Fi'),
  Text(1, 0, 'Action, Adventure, Fantasy'),
  Text(2, 0, 'Action, Adventure, Family'),
  Text(3, 0, 'Action, Adventure, Sci-Fi'),
  Text(4, 0, 'Adventure, Fantasy')])
```




Action and Adventure are two of the most profitable genres as per above bar plot.

# Box Office Analysis

## Top 10 highest-grossing movies

In [323]:
```
query = ''' SELECT "Title" ,"Director", "Gross", "Genre"
            FROM "IMDb"
            WHERE "Gross" <> 0
            ORDER BY "Gross" DESC
            LIMIT 10
            '''
```

In [324]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[324]:

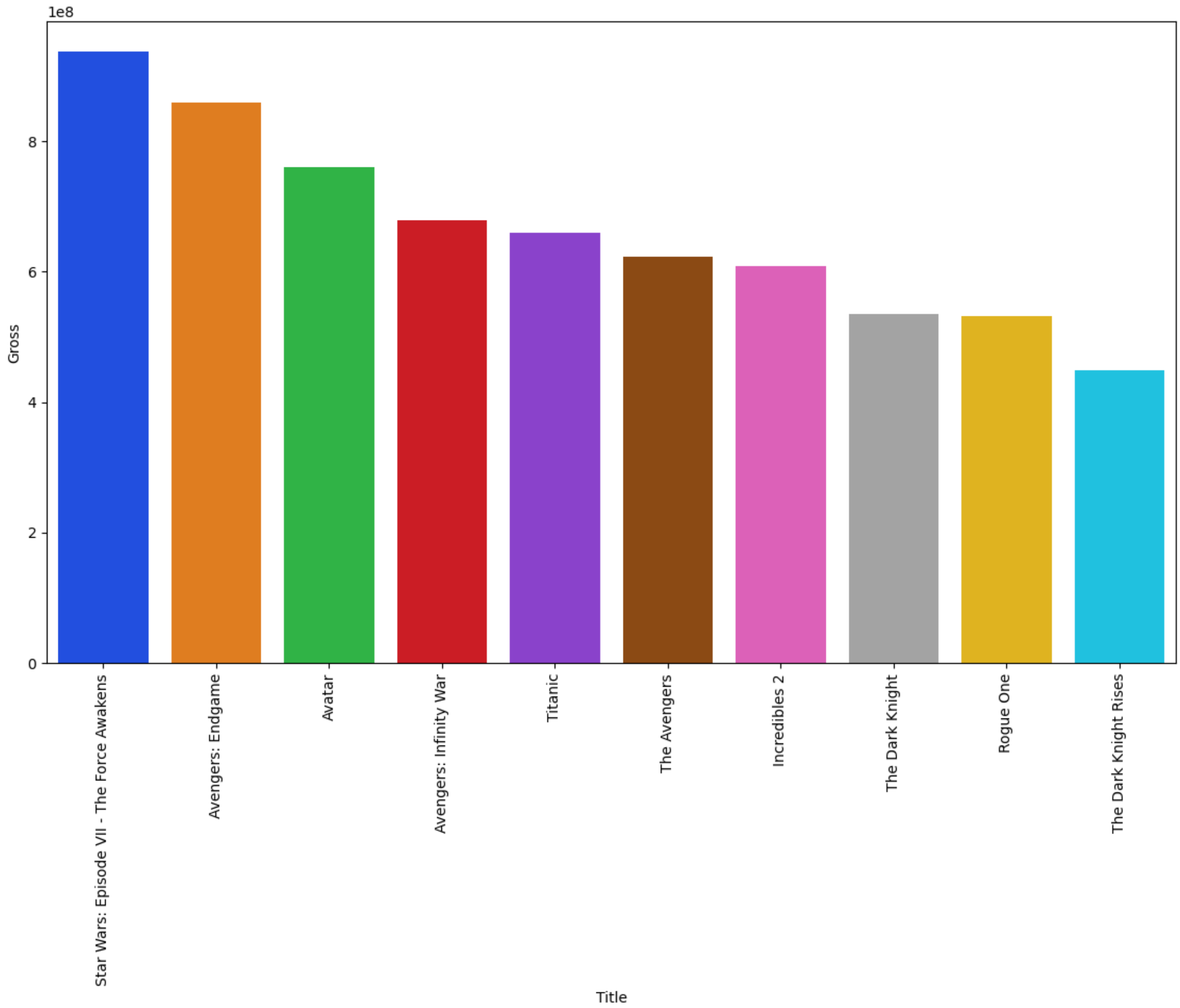
| | Title | Director | Gross | Genre |
|---|---|---|---|---|
| **0** | Star Wars: Episode VII - The Force Awakens | J.J. Abrams | 936662225 | Action, Adventure, Sci-Fi |
| **1** | Avengers: Endgame | Anthony Russo | 858373000 | Action, Adventure, Drama |
| **2** | Avatar | James Cameron | 760507625 | Action, Adventure, Fantasy |
| **3** | Avengers: Infinity War | Anthony Russo | 678815482 | Action, Adventure, Sci-Fi |
| **4** | Titanic | James Cameron | 659325379 | Drama, Romance |
| **5** | The Avengers | Joss Whedon | 623279547 | Action, Adventure, Sci-Fi |
| **6** | Incredibles 2 | Brad Bird | 608581744 | Animation, Action, Adventure |
| **7** | The Dark Knight | Christopher Nolan | 534858444 | Action, Crime, Drama |
| **8** | Rogue One | Gareth Edwards | 532177324 | Action, Adventure, Sci-Fi |
| **9** | The Dark Knight Rises | Christopher Nolan | 448139099 | Action, Adventure |

In [304]:

```python
plt.figure(figsize= (14,8))
sns.barplot(x='Title',y='Gross', data=df, palette= 'bright')
plt.xticks(rotation = 90)
```

Out[304]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Star Wars: Episode VII - The Force Awakens'),
  Text(1, 0, 'Avengers: Endgame'),
  Text(2, 0, 'Avatar'),
  Text(3, 0, 'Avengers: Infinity War'),
  Text(4, 0, 'Titanic'),
  Text(5, 0, 'The Avengers'),
  Text(6, 0, 'Incredibles 2'),
  Text(7, 0, 'The Dark Knight'),
  Text(8, 0, 'Rogue One'),
  Text(9, 0, 'The Dark Knight Rises')])
```




From the above list of top grossers, we can see that there is only one movie which has a 'Drama' genre and rest of them are mostly of 'Action' or 'Adventure'. So, even if Drama is the most popular genre, Action and Adventure are two of the most profitable ones.

## Director with the highest total box office revenue

In [305]:
```
query = ''' SELECT "Director" , CAST(SUM("Gross") AS BIGINT) AS Total_collecti
            FROM "IMDb"
            WHERE "Gross" <> 0
            GROUP BY "Director"
            ORDER BY Total_collection DESC
            LIMIT 5
            '''
```

In [306]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[306]:

| | Director | total_collection |
|---|---|---|
| **0** | Steven Spielberg | 2478133165 |
| **1** | Anthony Russo | 2205039403 |
| **2** | Christopher Nolan | 1937454106 |
| **3** | James Cameron | 1748236602 |
| **4** | Peter Jackson | 1597312443 |

## Box office revenue for each year

In [307]:
```
query = ''' SELECT "Year", CAST(SUM("Gross") AS BIGINT) AS Total_collections
            FROM "IMDb"
            WHERE "Gross" <> 0
            GROUP BY "Year"
            ORDER BY "Year" ASC
            '''
```

In [308]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[308]:

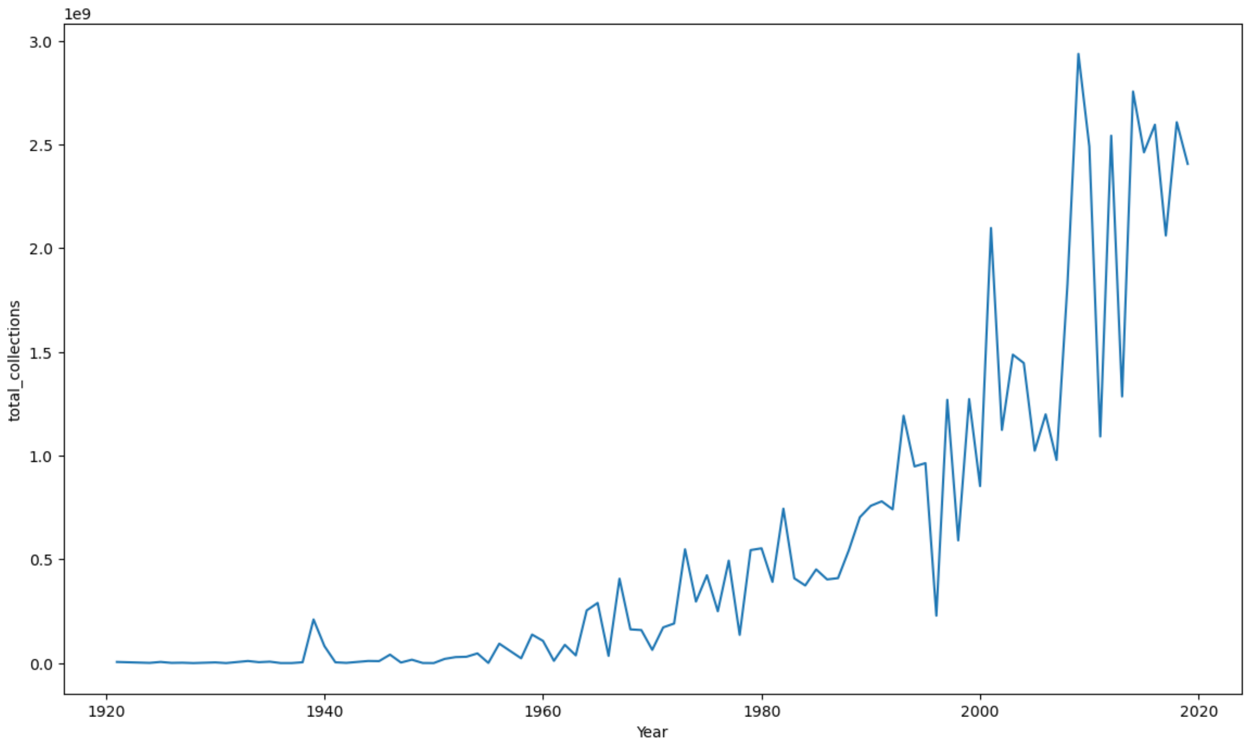
| | Year | total_collections |
|---|---|---|
| **0** | 1921 | 5450000 |
| **1** | 1924 | 977375 |
| **2** | 1925 | 5500970 |
| **3** | 1926 | 1033895 |
| **4** | 1927 | 1775706 |
| **...** | ... | ... |
| **89** | 2015 | 2462336868 |
| **90** | 2016 | 2595557425 |
| **91** | 2017 | 2061312852 |
| **92** | 2018 | 2607757362 |
| **93** | 2019 | 2406742688 |

94 rows × 2 columns

In [309]:
```python
plt.figure(figsize= (14,8))
sns.lineplot(x='Year',y='total_collections', data=df, palette= 'inferno')
plt.xticks(rotation = 0)
```

```
C:\Users\Aditya\AppData\Local\Temp\ipykernel_6280\4263806761.py:2: UserWarning: I
gnoring `palette` because no `hue` variable has been assigned.
  sns.lineplot(x='Year',y='total_collections', data=df, palette= 'inferno')
```

Out[309]:
```
(array([1900., 1920., 1940., 1960., 1980., 2000., 2020., 2040.]),
 [Text(1900.0, 0, '1900'),
  Text(1920.0, 0, '1920'),
  Text(1940.0, 0, '1940'),
  Text(1960.0, 0, '1960'),
  Text(1980.0, 0, '1980'),
  Text(2000.0, 0, '2000'),
  Text(2020.0, 0, '2020'),
  Text(2040.0, 0, '2040')])
```




From the above time series analysis of box office revenue, we can say that last two decades have seen significant grwoth in the box office collection.

# User Reviews

## Top 10 movies with the most user reviews

In [255]:
```python
query = ''' SELECT "Title", "Votes", "Rating"
            FROM "IMDb"
            ORDER BY "Votes" DESC
            LIMIT 10
            '''
```

In [256]:

```
df = pd.read_sql_query(query,engine)
df
```

Out[256]:

| | Title | Votes | Rating |
|---|---|---|---|
| **0** | The Shawshank Redemption | 2343110 | 9.3 |
| **1** | The Dark Knight | 2303232 | 9.0 |
| **2** | Inception | 2067042 | 8.8 |
| **3** | Fight Club | 1854740 | 8.8 |
| **4** | Pulp Fiction | 1826188 | 8.9 |
| **5** | Forrest Gump | 1809221 | 8.8 |
| **6** | The Matrix | 1676426 | 8.7 |
| **7** | The Lord of the Rings: The Fellowship of the Ring | 1661481 | 8.8 |
| **8** | The Lord of the Rings: The Return of the King | 1642758 | 8.9 |
| **9** | The Godfather | 1620367 | 9.2 |

## Movies with the highest and lowest user ratings

In [257]:

```
query = ''' WITH cte AS(
SELECT "Title",
"Rating" AS Highest,
ROW_NUMBER () OVER (ORDER BY "Rating" DESC ) AS Rank_high
FROM "IMDb"
GROUP BY  "Title", "Rating"
),

cte1 AS(
SELECT "Title",
"Rating" AS Lowest,
ROW_NUMBER () OVER (ORDER BY "Rating" ASC ) AS Rank_low
FROM "IMDb"
GROUP BY  "Title", "Rating"
)

SELECT cte."Title", cte.Highest, cte1.Lowest
FROM cte JOIN cte1 ON cte."Title" = cte1."Title"
WHERE cte.Rank_high = 1 OR cte1.Rank_low = 1
ORDER BY "Title" DESC

            '''
```

In [258]:

```
df = pd.read_sql_query(query,engine)
df
```

Out[258]:

| | Title | highest | lowest |
|---|---|---|---|
| **0** | The Shawshank Redemption | 9.3 | 9.3 |
| **1** | The Secret of Kells | 7.6 | 7.6 |

## Correlation between user ratings and box office revenue

In [326]:
```
query = ''' SELECT "Rating", "Gross"
            FROM "IMDb"
            WHERE "Gross" <> 0
            '''
```

In [327]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[327]:

| | Rating | Gross |
|---|---|---|
| **0** | 9.3 | 28341469 |
| **1** | 9.2 | 134966411 |
| **2** | 9.0 | 534858444 |
| **3** | 9.0 | 57300000 |
| **4** | 9.0 | 4360000 |
| **...** | ... | ... |
| **826** | 7.6 | 696690 |
| **827** | 7.6 | 1378435 |
| **828** | 7.6 | 141843612 |
| **829** | 7.6 | 13780024 |
| **830** | 7.6 | 30500000 |

831 rows × 2 columns

In [330]:
```
corr = df["Rating"].corr(df["Gross"])
corr
```

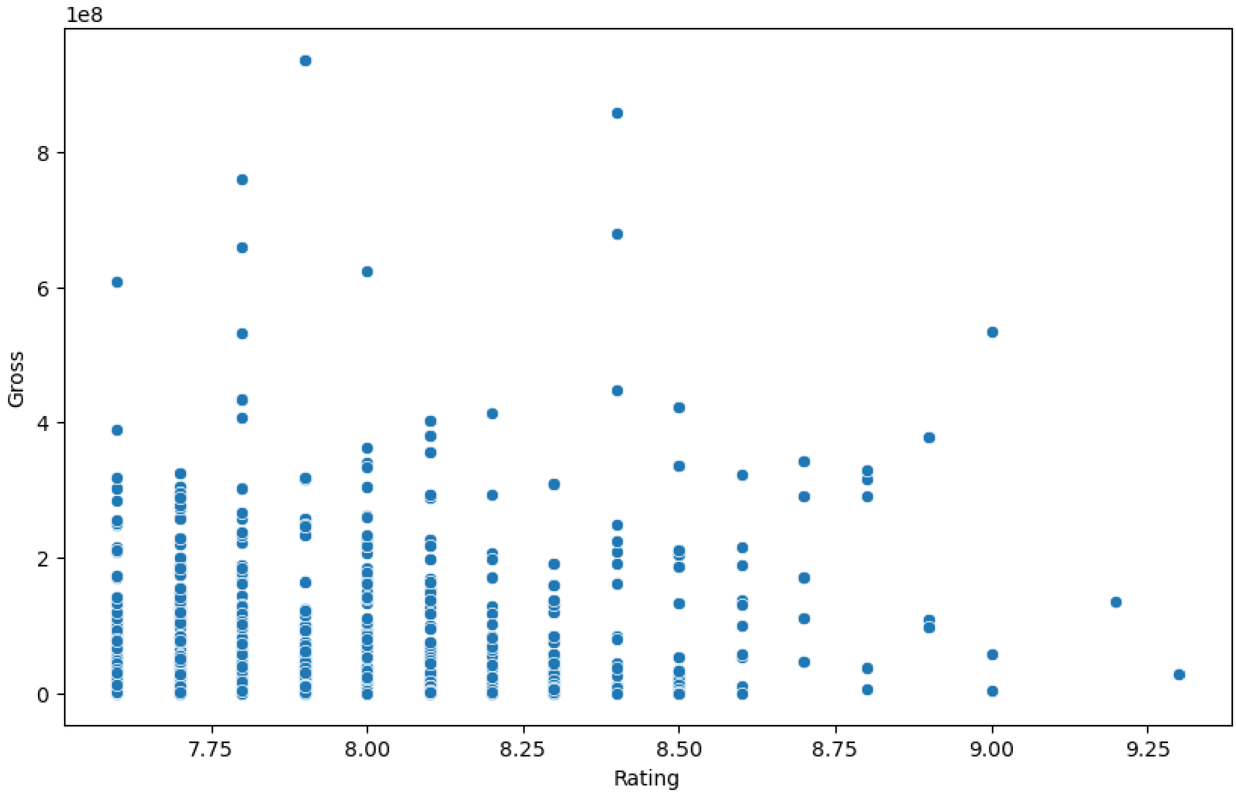
Out[330]: 0.09592277110132366

In [331]:

```python
plt.figure(figsize= (10,6))
sns.scatterplot(x='Rating',y='Gross', data=df)
plt.xticks(rotation = 0)
```

Out[331]:

```
(array([7.5 , 7.75, 8.  , 8.25, 8.5 , 8.75, 9.  , 9.25, 9.5 ]),
 [Text(7.5, 0, '7.50'),
  Text(7.75, 0, '7.75'),
  Text(8.0, 0, '8.00'),
  Text(8.25, 0, '8.25'),
  Text(8.5, 0, '8.50'),
  Text(8.75, 0, '8.75'),
  Text(9.0, 0, '9.00'),
  Text(9.25, 0, '9.25'),
  Text(9.5, 0, '9.50')])
```




Correlation coefficient between user ratings and box office revenue is 0.096, which is close to 0, so we can say that user rating has no direct impact on box office collection and vice versa.

## Director and Cast Analysis

### Average rating for each director's movies (Min 5 movies)

In [338]:

```python
query = ''' SELECT "Director", CAST(AVG("Rating") AS DECIMAL(10,2)) AS Avg_Rat
            FROM "IMDb"
            GROUP BY "Director"
            HAVING COUNT(*) >= 5
            ORDER BY Avg_Rating DESC
            '''
```

In [339]:
```
df = pd.read_sql_query(query,engine)
df
```

Out[339]:

| | Director | avg_rating |
|---|---|---|
| 0 | Christopher Nolan | 8.46 |
| 1 | Peter Jackson | 8.40 |
| 2 | Francis Ford Coppola | 8.40 |
| 3 | Charles Chaplin | 8.33 |
| 4 | Sergio Leone | 8.27 |
| 5 | Stanley Kubrick | 8.23 |
| 6 | Akira Kurosawa | 8.22 |
| 7 | Quentin Tarantino | 8.18 |
| 8 | Martin Scorsese | 8.17 |
| 9 | Billy Wilder | 8.14 |
| 10 | Ingmar Bergman | 8.14 |
| 11 | Andrei Tarkovsky | 8.12 |
| 12 | Robert Zemeckis | 8.12 |
| 13 | Sidney Lumet | 8.10 |
| 14 | James Cameron | 8.08 |
| 15 | Ridley Scott | 8.08 |
| 16 | David Fincher | 8.04 |
| 17 | Steven Spielberg | 8.03 |
| 18 | Hayao Miyazaki | 8.02 |
| 19 | Alfred Hitchcock | 8.01 |
| 20 | Federico Fellini | 8.00 |
| 21 | Roman Polanski | 8.00 |
| 22 | Denis Villeneuve | 7.98 |
| 23 | Ron Howard | 7.92 |
| 24 | Clint Eastwood | 7.91 |
| 25 | Richard Linklater | 7.90 |
| 26 | John Ford | 7.90 |
| 27 | John Huston | 7.90 |
| 28 | Howard Hawks | 7.86 |
| 29 | David Lynch | 7.86 |
| 30 | Rob Reiner | 7.83 |
| 31 | Wes Anderson | 7.83 |
| 32 | Joel Coen | 7.82 |
| 33 | Woody Allen | 7.79 |
| 34 | Alfonso Cuarón | 7.75 |

Christopher Nolan's movies as a director has the highest average rating for directors who have directed minimum 5 movies.

### Lead actors who are associated with higher-grossing movies

In [315]:

```
query = ''' SELECT "Star1" AS Actor, CAST(SUM("Gross") AS BIGINT) AS total_box
            FROM "IMDb"
            WHERE "Gross" <> 0
            GROUP BY "Star1"
            ORDER BY total_boxoffice DESC
            LIMIT 10
            '''
```

In [316]:

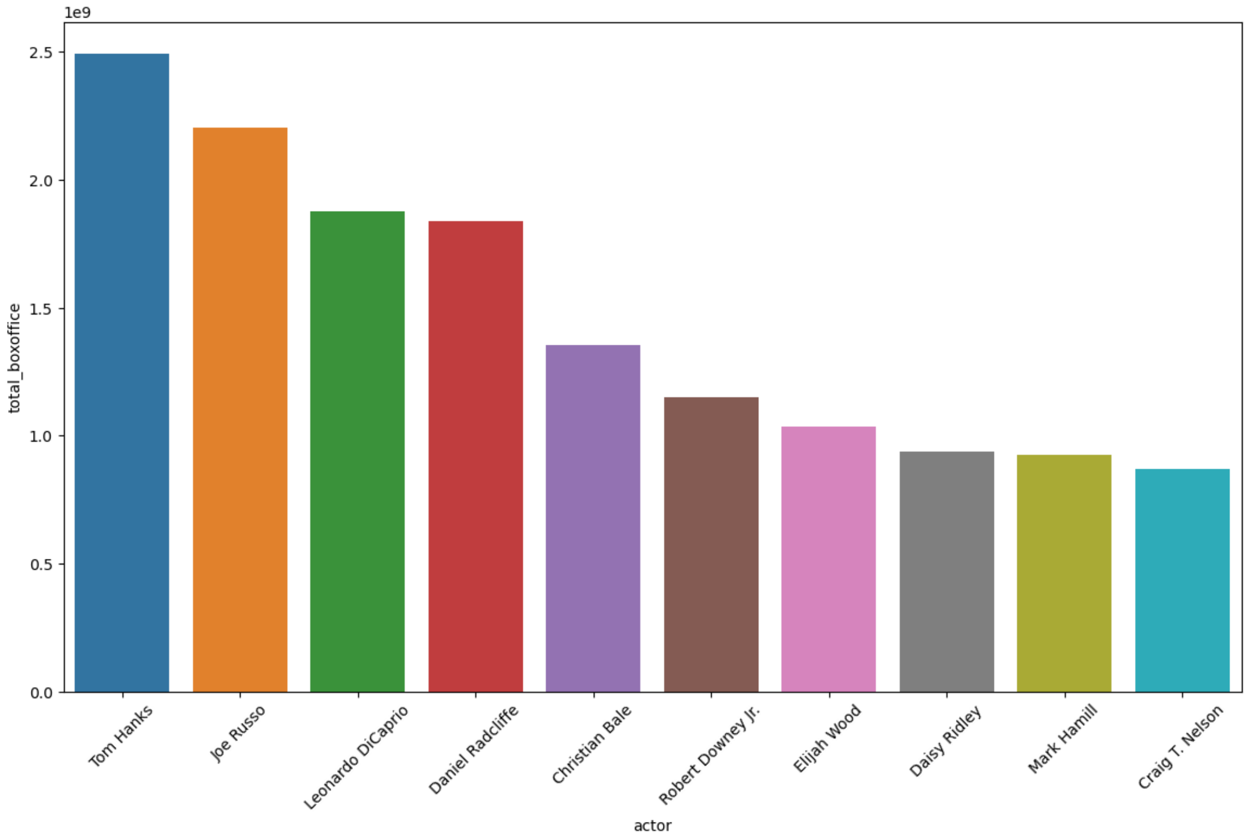
```
df = pd.read_sql_query(query, engine)
df
```

Out[316]:

| | actor | total_boxoffice |
|---|---|---|
| **0** | Tom Hanks | 2493097454 |
| **1** | Joe Russo | 2205039403 |
| **2** | Leonardo DiCaprio | 1877321752 |
| **3** | Daniel Radcliffe | 1835901034 |
| **4** | Christian Bale | 1351591432 |
| **5** | Robert Downey Jr. | 1150720327 |
| **6** | Elijah Wood | 1035942020 |
| **7** | Daisy Ridley | 936662225 |
| **8** | Mark Hamill | 922340616 |
| **9** | Craig T. Nelson | 870022836 |

In [317]:
```
plt.figure(figsize= (14,8))
sns.barplot(x='actor',y='total_boxoffice', data=df, palette= 'tab10')
plt.xticks(rotation = 45)
```

Out[317]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Tom Hanks'),
  Text(1, 0, 'Joe Russo'),
  Text(2, 0, 'Leonardo DiCaprio'),
  Text(3, 0, 'Daniel Radcliffe'),
  Text(4, 0, 'Christian Bale'),
  Text(5, 0, 'Robert Downey Jr.'),
  Text(6, 0, 'Elijah Wood'),
  Text(7, 0, 'Daisy Ridley'),
  Text(8, 0, 'Mark Hamill'),
  Text(9, 0, 'Craig T. Nelson')])
```




Tom Hanks, Joe Russo and Leonardo DiCaprio are few lead actors who have been associated with high grossing movies.

## Runtime Analysis

### Average runtime of movies over the years

In [318]:
```
query = ''' SELECT "Year", CAST(AVG("Runtime") AS DECIMAL(10,2)) AS Avg_Runtim
            FROM "IMDb"
            GROUP BY "Year"
            ORDER BY "Year"
            '''
```

In [319]:

```
df = pd.read_sql_query(query, engine)
df
```

Out[319]:

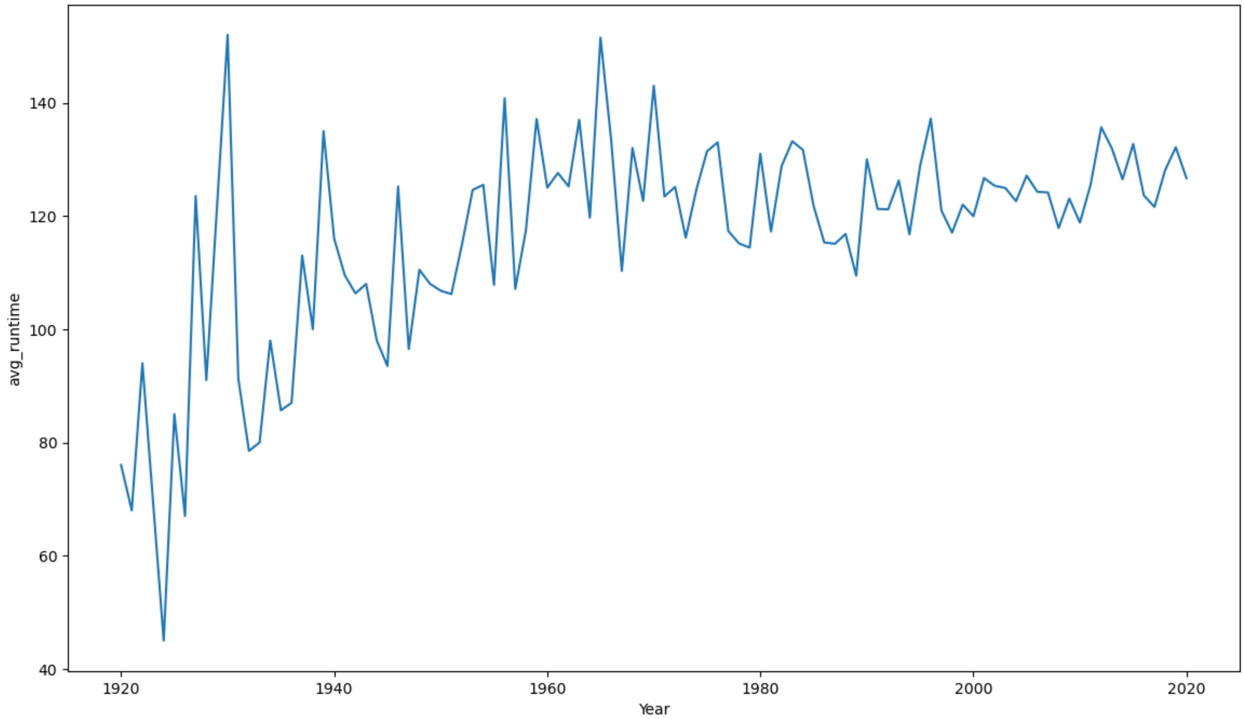
| | Year | avg_runtime |
|---|---|---|
| **0** | 1920 | 76.00 |
| **1** | 1921 | 68.00 |
| **2** | 1922 | 94.00 |
| **3** | 1924 | 45.00 |
| **4** | 1925 | 85.00 |
| **...** | ... | ... |
| **94** | 2016 | 123.64 |
| **95** | 2017 | 121.59 |
| **96** | 2018 | 128.11 |
| **97** | 2019 | 132.13 |
| **98** | 2020 | 126.67 |

99 rows × 2 columns

In [320]:
```
plt.figure(figsize= (14,8))
sns.lineplot(x='Year',y='avg_runtime', data=df, palette= 'inferno')
plt.xticks(rotation = 0)
```

```
C:\Users\Aditya\AppData\Local\Temp\ipykernel_6280\2765243330.py:2: UserWarning: I
gnoring `palette` because no `hue` variable has been assigned.
  sns.lineplot(x='Year',y='avg_runtime', data=df, palette= 'inferno')
```

Out[320]:
```
(array([1900., 1920., 1940., 1960., 1980., 2000., 2020., 2040.]),
 [Text(1900.0, 0, '1900'),
  Text(1920.0, 0, '1920'),
  Text(1940.0, 0, '1940'),
  Text(1960.0, 0, '1960'),
  Text(1980.0, 0, '1980'),
  Text(2000.0, 0, '2000'),
  Text(2020.0, 0, '2020'),
  Text(2040.0, 0, '2040')])
```




From the above time series analysis of average runtime of movies, we can see that over the years there are lot of fluctuations in the average runtime from 45 mins to 150 mins but from the past 2 decades average runtime has consolidated around 120 mins.

## Movies with the Longest and Shortest runtimes

In [270]:

```
query = ''' WITH cte AS(
SELECT "Title",
"Runtime" AS Longest,
ROW_NUMBER () OVER (ORDER BY "Runtime" DESC ) AS Rank_high
FROM "IMDb"
GROUP BY  "Title", "Runtime"
),

cte1 AS(
SELECT "Title",
"Runtime" AS Shortest,
ROW_NUMBER () OVER (ORDER BY "Runtime" ASC ) AS Rank_low
FROM "IMDb"
GROUP BY  "Title", "Runtime"
)

SELECT cte."Title", cte.Longest, cte1.Shortest
FROM cte JOIN cte1 ON cte."Title" = cte1."Title"
WHERE cte.Rank_high = 1 OR cte1.Rank_low = 1
ORDER BY "Title" ASC '''
```

In [271]:

```
df = pd.read_sql_query(query, engine)
df
```

Out[271]:

| | Title | longest | shortest |
|---|---|---|---|
| **0** | Gangs of Wasseypur | 321 | 321 |
| **1** | Sherlock Jr. | 45 | 45 |

## Movies with a runtime longer than the average duration

In [272]:

```
query = ''' SELECT "Title", "Runtime"
            FROM "IMDb"
            WHERE "Runtime" > (SELECT AVG("Runtime") FROM "IMDb")
            ORDER BY "Runtime" DESC
            '''
```

In [273]:
```
df = pd.read_sql_query(query, engine)
df
```

Out[273]:

| | Title | Runtime |
|---|---|---|
| **0** | Gangs of Wasseypur | 321 |
| **1** | Hamlet | 242 |
| **2** | Gone with the Wind | 238 |
| **3** | Once Upon a Time in America | 229 |
| **4** | Lawrence of Arabia | 228 |
| **...** | ... | ... |
| **437** | About Time | 123 |
| **438** | Jodaeiye Nader az Simin | 123 |
| **439** | The Theory of Everything | 123 |
| **440** | Atonement | 123 |
| **441** | The Notebook | 123 |

442 rows × 2 columns

# IMDb rating Vs Meta score

In [274]:
```
query = ''' SELECT "Rating", "Meta_score"
            FROM "IMDb"
            WHERE "Meta_score" <> 0 AND "Rating" <> 0
            '''
```

In [275]:
```
df = pd.read_sql_query(query, engine)
df
```

Out[275]:

| | Rating | Meta_score |
|---|---|---|
| **0** | 9.3 | 80 |
| **1** | 9.2 | 100 |
| **2** | 9.0 | 84 |
| **3** | 9.0 | 90 |
| **4** | 9.0 | 96 |
| **...** | ... | ... |
| **838** | 7.6 | 76 |
| **839** | 7.6 | 84 |
| **840** | 7.6 | 85 |
| **841** | 7.6 | 78 |
| **842** | 7.6 | 93 |

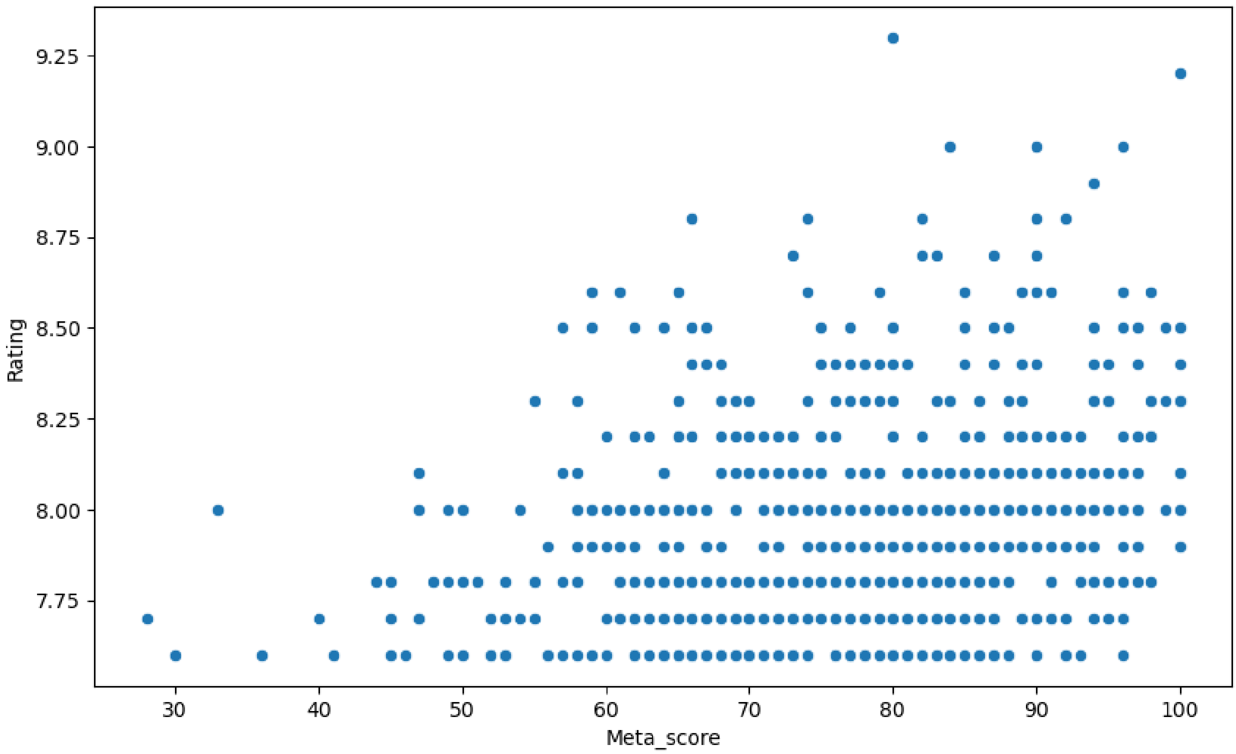
843 rows × 2 columns

In [276]:
```
corr = df["Rating"].corr(df["Meta_score"])
corr
```

Out[276]: 0.26853084455955467

In [277]:
```
plt.figure(figsize= (10,6))
sns.scatterplot(x='Meta_score',y='Rating', data=df)
plt.xticks(rotation = 0)
```

Out[277]:
```
(array([ 20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100., 110.]),
 [Text(20.0, 0, '20'),
  Text(30.0, 0, '30'),
  Text(40.0, 0, '40'),
  Text(50.0, 0, '50'),
  Text(60.0, 0, '60'),
  Text(70.0, 0, '70'),
  Text(80.0, 0, '80'),
  Text(90.0, 0, '90'),
  Text(100.0, 0, '100'),
  Text(110.0, 0, '110')])
```




Correlation coefficient between user ratings and Meta score is 0.268, which is closer to 0 than 1, so we can say that user rating has weak correlation with Meta score and vice versa.

## Conclusion

In conclusion, our exploratory data analysis project on the IMDb movie dataset has unveiled a plethora of intriguing insights into the world of cinema. Notably, we've discovered that while drama stands as the most popular genre among the movies in our dataset, the genres that reign supreme in terms of profitability are action and adventure. This underscores the nuanced relationship between popularity and financial success in the film industry.

Our analysis of box office revenue over time has revealed a remarkable growth trend in the past two decades, reflecting the dynamic nature of the movie business. However, the correlation coefficient between user ratings and box office revenue suggests a weak connection, indicating that audience preferences do not directly dictate a movie's financial performance.

Further, we found that Christopher Nolan's directorial ventures consistently earn the highest average ratings, emphasizing his influence in the industry. Additionally, actors like Tom Hanks, Joe Russo, and Leonardo DiCaprio have been associated with high-grossing movies, highlighting the impact of star power on a film's earnings.

The analysis of average movie runtime showed fluctuations over the years, with a consolidation around 120 minutes in the past two decades. Finally, the correlation between user ratings and Meta scores is weak, implying that audience opinions and critical acclaim often diverge.

Our EDA project has not only provided valuable insights into the movie industry's dynamics but also challenged preconceived notions, showcasing the complex interplay of factors that contribute to a movie's success. As the cinematic landscape continues to evolve, these findings serve as a testament to the ever changing nature of the art of filmmaking.

In [ ]:

```
 
```