In [1]:
```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.express as px
```

In [3]:
```
import pandas as pd
movies_data_path = ('IMDB Top 250 Movies.csv')

movies_data = pd.read_csv(movies_data_path)
```

Basic Information about Internet Movie Database

In [3]:
```
# Columns of the dataset
movies_data.columns
```

Out[3]:
```
Index(['rank', 'name', 'year', 'rating', 'genre', 'certificate', 'run_time',
       'tagline', 'budget', 'box_office', 'casts', 'directors', 'writers'],
      dtype='object')
```

In [4]:
```
# first 5 records from the dataset
movies_data.head()
```

Out[4]:

	rank	name	year	rating	genre	certificate	run_time	tagline	budget	bo
0	1	The Shawshank Redemption	1994	9.3	Drama	R	2h 22m	Fear can hold you prisoner. Hope can set you f...	25000000	2
1	2	The Godfather	1972	9.2	Crime,Drama	R	2h 55m	An offer you can't refuse.	6000000	25
2	3	The Dark Knight	2008	9.0	Action,Crime,Drama	PG-13	2h 32m	Why So Serious?	185000000	100
3	4	The Godfather Part II	1974	9.0	Crime,Drama	R	3h 22m	All the power on earth can't change destiny.	13000000	4
4	5	12 Angry Men	1957	9.0	Crime,Drama	Approved	1h 36m	Life Is In Their Hands - - Death Is On Their Mi...	350000	

In [5]:
```
# a brief summary of dataset
movies_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   rank         250 non-null    int64
 1   name         250 non-null    object
 2   year         250 non-null    int64
 3   rating       250 non-null    float64
 4   genre        250 non-null    object
 5   certificate  250 non-null    object
 6   run_time     250 non-null    object
 7   tagline      250 non-null    object
 8   budget       250 non-null    object
 9   box_office   250 non-null    object
 10  casts        250 non-null    object
 11  directors    250 non-null    object
 12  writers      250 non-null    object
dtypes: float64(1), int64(2), object(10)
memory usage: 25.5+ KB
```

In [6]:
```
# shape of the data
movies_data.shape
```

Out[6]:
```
(250, 13)
```

In [7]:
```
# statistical information
movies_data.describe()
```

Out[7]:

	rank	year	rating
count	250.000000	250.000000	250.000000
mean	125.500000	1986.360000	8.307200
std	72.312977	25.125356	0.229081
min	1.000000	1921.000000	8.000000
25%	63.250000	1966.250000	8.100000
50%	125.500000	1994.000000	8.200000
75%	187.750000	2006.000000	8.400000
max	250.000000	2022.000000	9.300000

In [8]:
```
# checking for null values
movies_data.isnull().sum()
```

Out[8]:
```
rank           0
name           0
year           0
rating         0
genre          0
certificate    0
run_time       0
tagline        0
budget         0
box_office     0
casts          0
directors      0
writers        0
dtype: int64
```

In [9]:
```
# Loop over columns and count the number of occurrences of "Not Available"
not_avail_cols = []
for col in movies_data.columns:
    if 'Not Available' in movies_data[col].value_counts().index:
        not_avail_count = movies_data[col].value_counts()['Not Available']
        print(f"Column '{col}' has {not_avail_count} 'Not Available' values.")
        not_avail_cols.append(col)

print(f"\nThere are {len(not_avail_cols)} columns with 'Not Available' values: {not_av
```

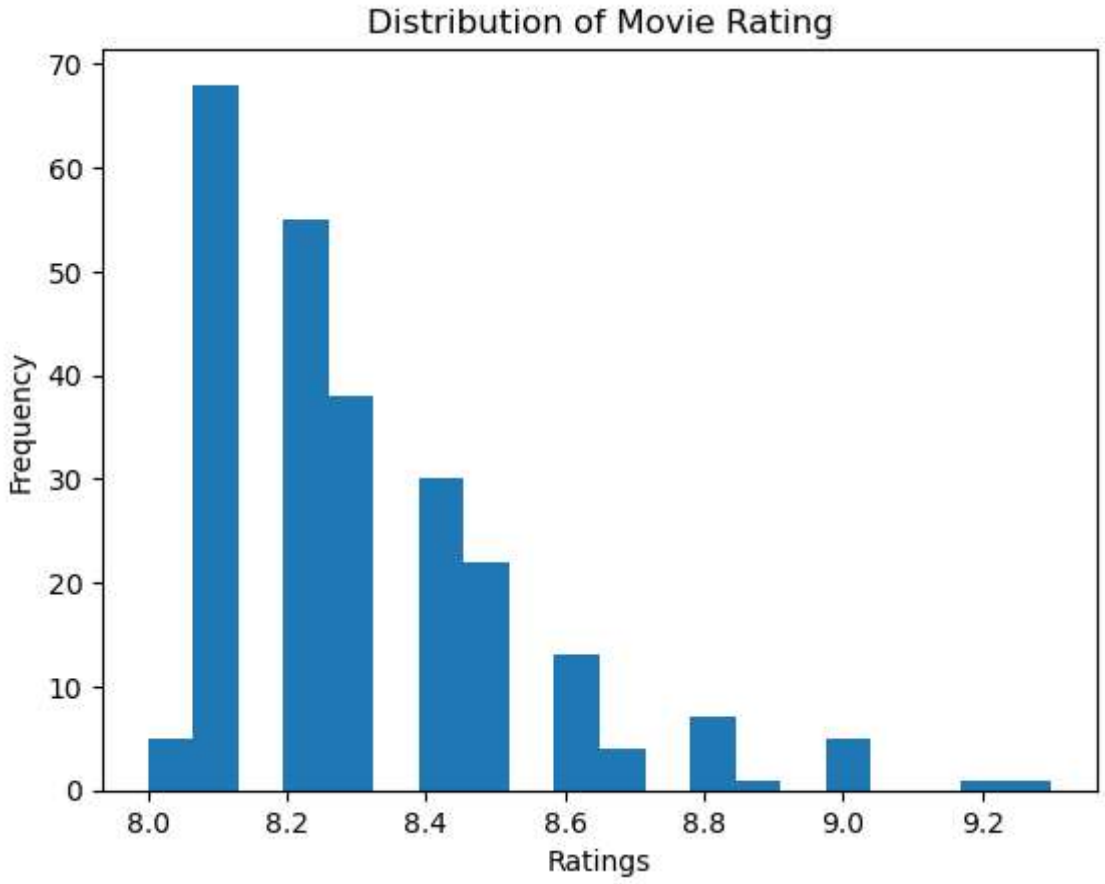
```
Column 'certificate' has 1 'Not Available' values.
Column 'run_time' has 1 'Not Available' values.
Column 'budget' has 39 'Not Available' values.
Column 'box_office' has 30 'Not Available' values.

There are 4 columns with 'Not Available' values: ['certificate', 'run_time', 'budge
t', 'box_office']
```

(EDA) Exploratory Data Analysis

Determine the distribution of movie rating¶

In [10]:
```
plt.hist(movies_data['rating'],bins=20)
plt.xlabel('Ratings')
plt.ylabel('Frequency')
plt.title('Distribution of Movie Rating')
plt.show()
```



we can conclude that the majority of movies in the dataset have a rating between 8.0 and 8.4, with fewer movies having higher ratings above 8.4

Which movie genres are the most common in the dataset?

In [11]:
```
genre_counts = movies_data['genre'].value_counts()

# The Top 5 most common genres
display(genre_counts.head(5))
```

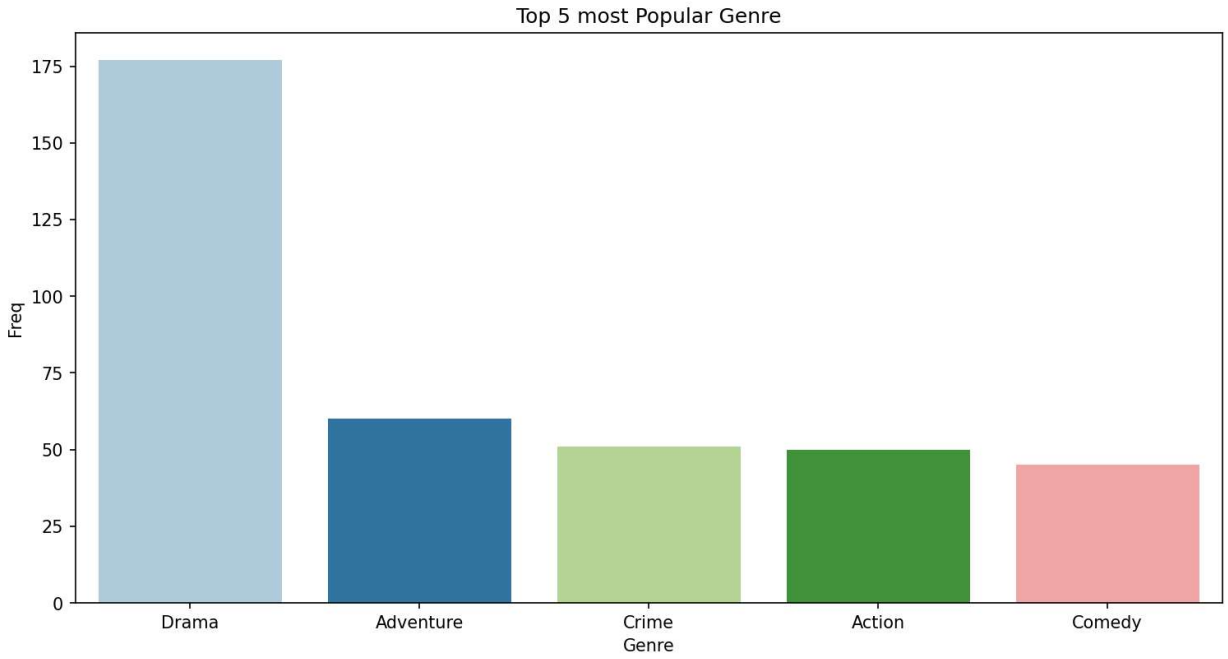
```
Drama                          19
Crime,Drama                    14
Animation,Adventure,Comedy     10
Biography,Drama,History         9
Crime,Drama,Mystery             9
Name: genre, dtype: int64
```

we can conclude that drama is the most frequently occurring genre, followed by crime drama and animation adventure comedy.

In [12]:
```
real_gen = ""
for i in movies_data['genre']:
    real_gen = real_gen+","+i
all_gen = real_gen.split(',')[1:]
unique_genres = list(set(all_gen))

data = []
for i in unique_genres:
    c = 0
    for j in movies_data['genre']:
        if i in j:
            c+=1
    data.append([i, c])

gen_df = pd.DataFrame(data, columns=['Genre', 'Freq'])
gen_df.sort_values(by = 'Freq', ascending = False, inplace = True)
gen_df = gen_df.head(5)
plt.figure(figsize=(12,6),dpi = 150)
sns.barplot(data = gen_df, x = 'Genre', y = 'Freq', palette='Paired')
plt.title('Top 5 most Popular Genre')
plt.show()
```



How many movies were released each year in the dataset?

In [13]:
```
# Count the number of movies released in each year
year_counts = movies_data['year'].value_counts()

print("The number of movies released in each year:\n", year_counts)
```

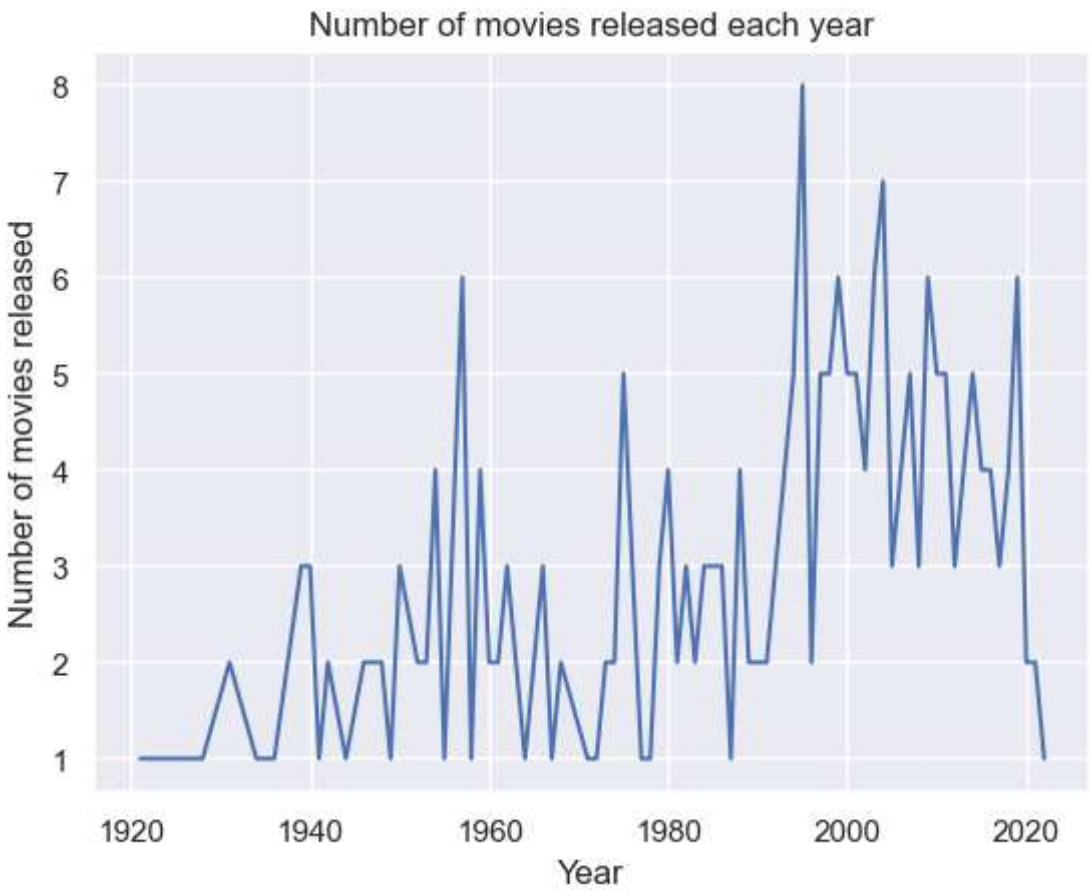
```
The number of movies released in each year:
 1995    8
2004    7
2009    6
1957    6
2003    6
       ..
1941    1
1958    1
2022    1
1987    1
1934    1
Name: year, Length: 86, dtype: int64
```

How does the number of movies released each year vary over time?

In [14]:
```
# Get the count of movies released each year
year_counts = movies_data.groupby('year')['name'].count()

# Create the line plot
sns.set(style="darkgrid")
plt.plot(year_counts.index, year_counts.values)
plt.xlabel('Year')
plt.ylabel('Number of movies released')
```

```
plt.title('Number of movies released each year')
plt.show()
```



What is the Percentage of Movies based on Year

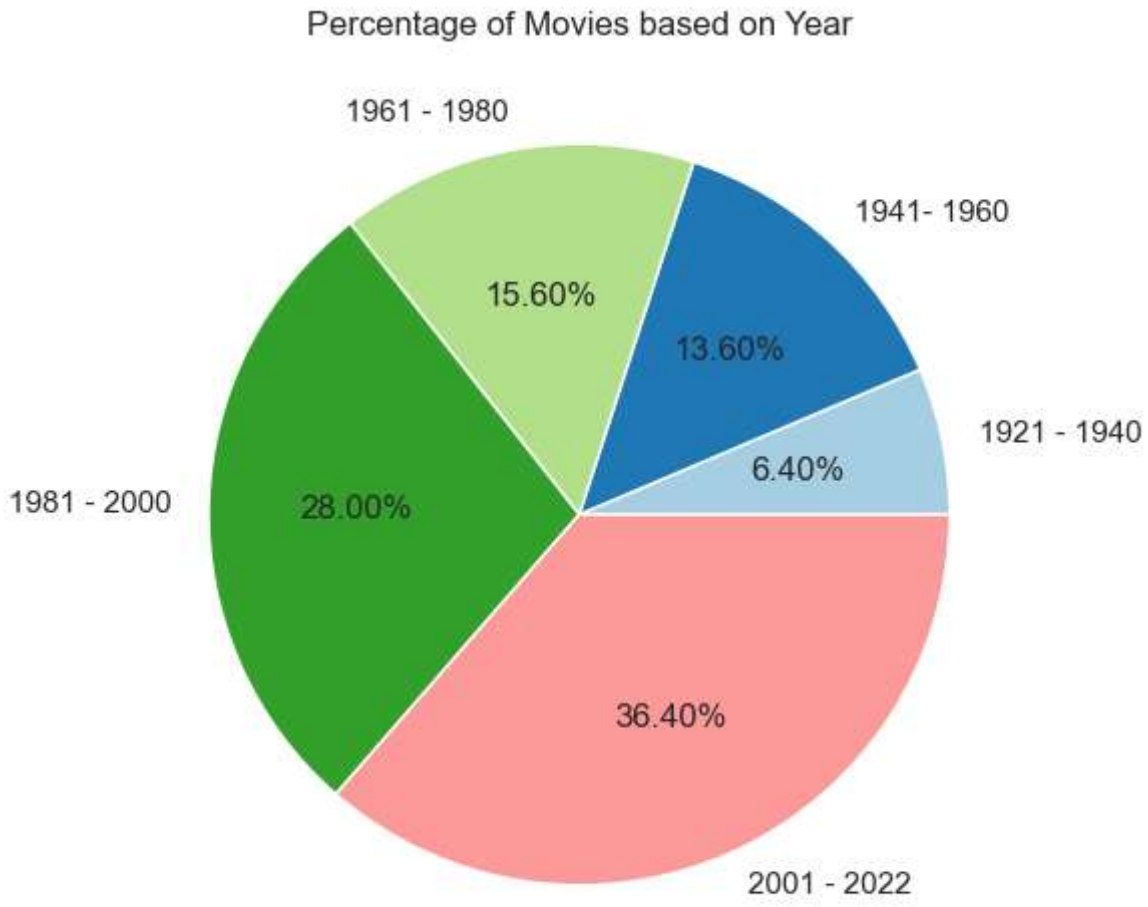
In [15]:
```
movies_data['year'].sort_values()
# year ranges from 1921 to 2022
# lets make group as 1921 - 1940, 1941- 1960, 1961 - 1980, 1981 - 2000, 2001 - 2022

_21to40 = 0
_41to60 = 0
_61to80 = 0
_81to00 = 0
_01to22 = 0

for i in movies_data['year']:
    if i>1920 and i<=1940:
        _21to40+=1
    elif i>1940 and i<=1960:
        _41to60+=1
    elif i>1960 and i<=1980:
        _61to80+=1
    elif i>1980 and i<=2000:
        _81to00+=1
    else:
        _01to22+=1

data = [_21to40, _41to60, _61to80, _81to00, _01to22]
label = ['1921 - 1940', '1941- 1960', '1961 - 1980', '1981 - 2000', '2001 - 2022']
```

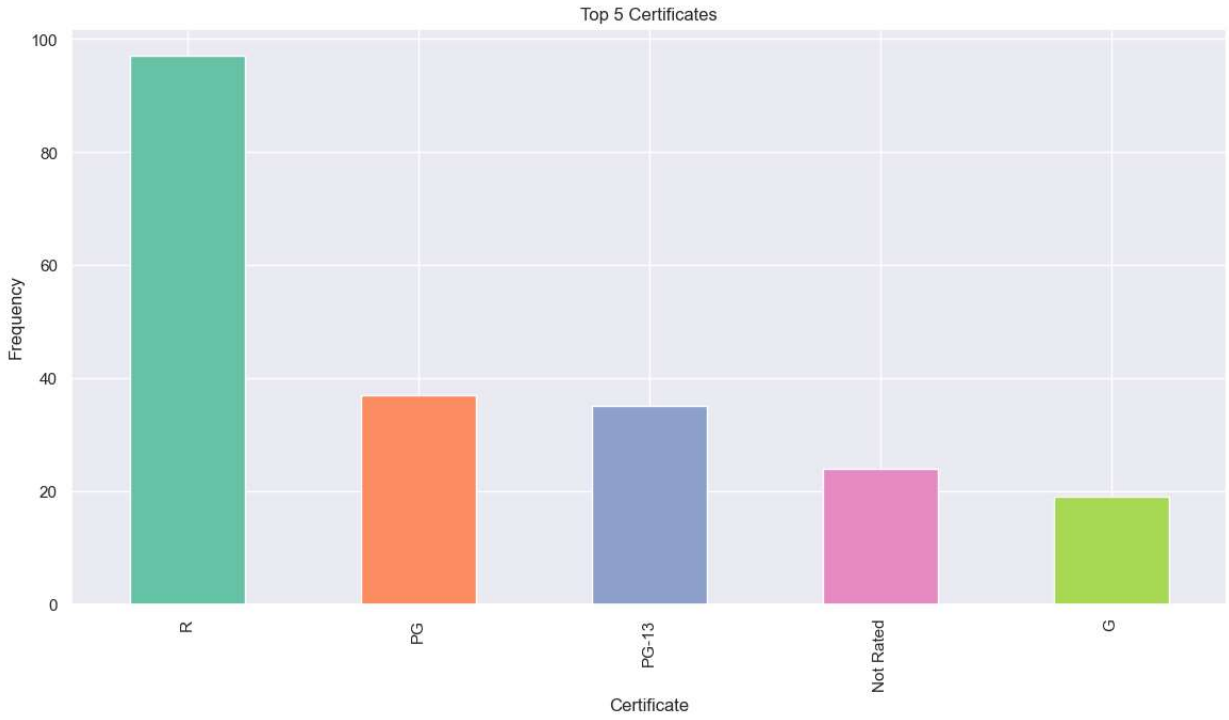
```
plt.figure(figsize=(12,6))
plt.pie(data,labels=label,colors=sns.color_palette('Paired'),autopct='%.2f%%')
plt.title("Percentage of Movies based on Year")
plt.show()
```



As year progresses the trend to movies becomes increasing as a result 36.40 % of the movies lies between the years 2001 - 2022

Distribution of top 5 Certificates

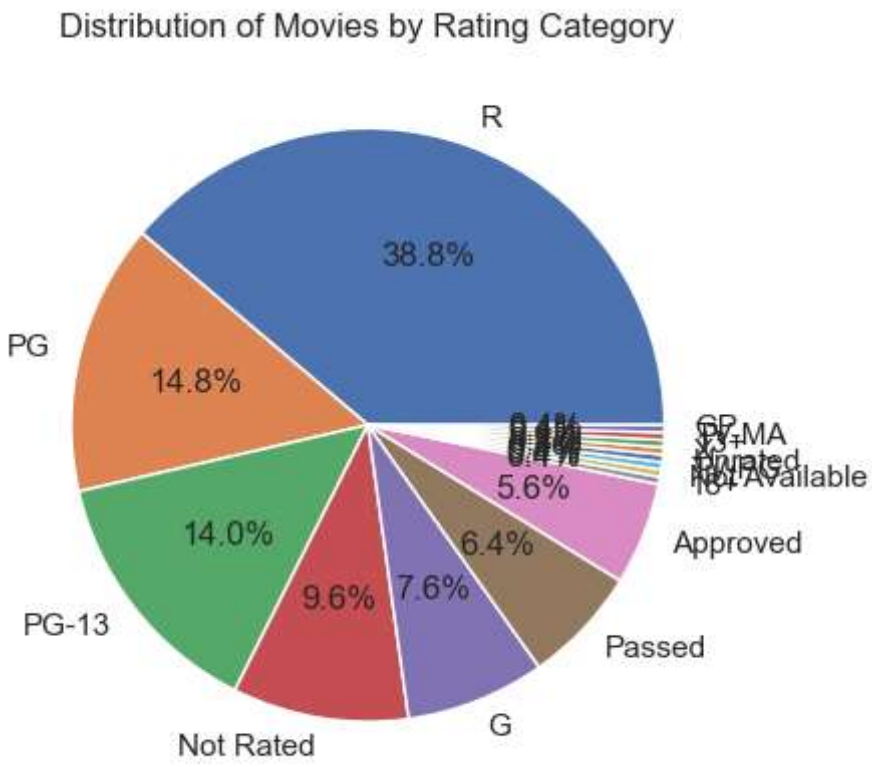
In [16]:
```
plt.figure(figsize=(14,7))
movies_data.groupby('certificate').size().sort_values(ascending=False).head(5).plot(ki
plt.xlabel('Certificate')
plt.ylabel('Frequency')
plt.title("Top 5 Certificates")
plt.show()
```



Distribution of movies by rating category

In [17]:
```
# Count the number of movies in each rating category
rating_counts = movies_data['certificate'].value_counts()

# Create a pie chart of the rating distribution
plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Movies by Rating Category')
plt.show()
```



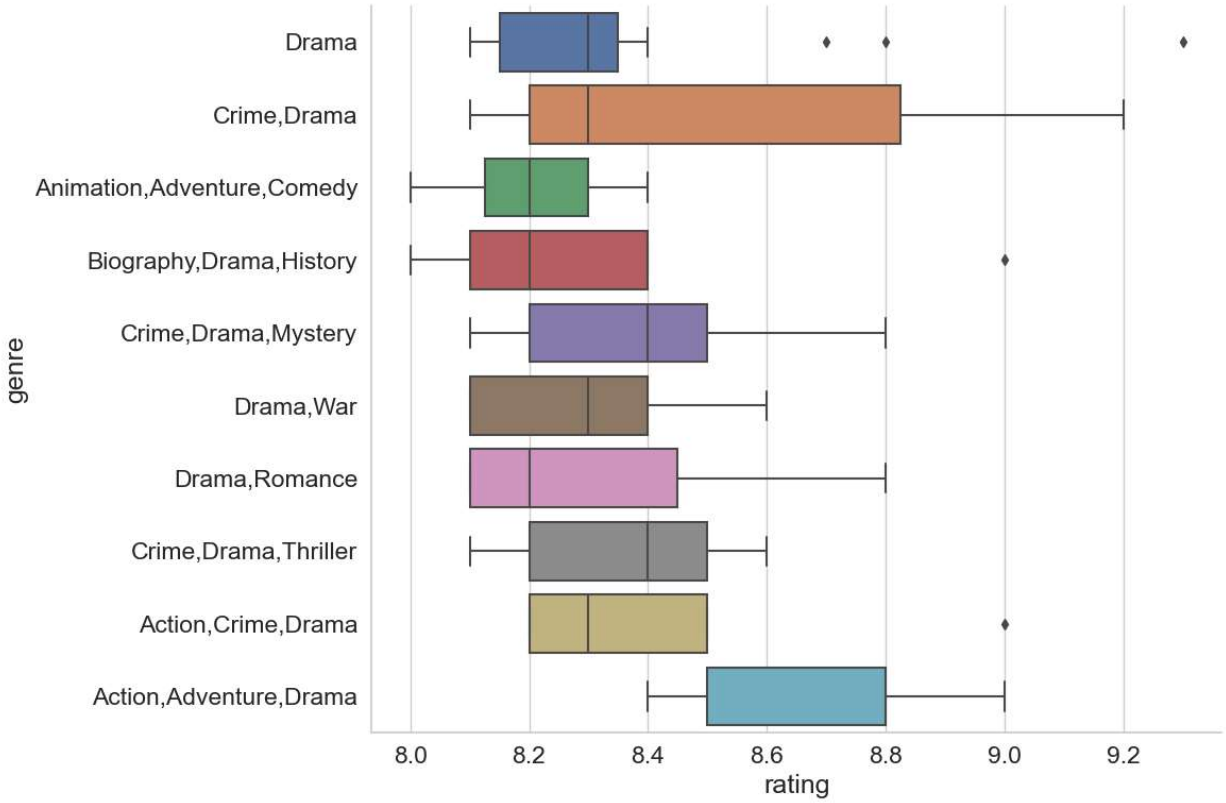
How does the distribution of movie ratings differ by genre?

In [18]:
```
# Get the top 10 genres by movie count
top_genres = movies_data['genre'].str.split('|', expand=True).stack().value_counts().h

# Filter the dataset to only include movies in the top 10 genres
df_top_genres = movies_data[movies_data['genre'].str.contains('|'.join(top_genres))]

# Create a box plot for each genre
sns.set(style='whitegrid', font_scale=1.5)
sns.catplot(x='rating', y='genre', data=df_top_genres, kind='box', height=8, aspect=1.
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x16813e9d0>
```



What are the top 10 highest-grossing movies in the dataset?

In [19]:
```
# Filter out the 'Not Available' box office values and sort by descending order
top_grossing = movies_data[movies_data['box_office'] != 'Not Available'].sort_values(b

# Create a list of 10 different colors
colors = ['#F44336', '#E91E63', '#9C27B0', '#673AB7', '#3F51B5', '#2196F3', '#00BCD4',

# Plot the bar chart
plt.bar(x=top_grossing['name'], height=top_grossing['box_office'], color=colors)
plt.xticks(rotation=90)
plt.xlabel('Movie Title')
plt.ylabel('Box Office ($)')
plt.title('Top 10 Highest-Grossing Movies')
plt.show()
```

Top 10 Highest-Grossing Movies
Box Office ($)
941637960
947944270
9523464
955
962002
96302
968511805
96983009
970263
98690254
Spotlight
The Great Dictator
Rush
The Lion King
Ikiru
The Battle of Algiers
12 Angry Men
ngelove or: How I Learned to Stop Worrying and Love the Bomb
The Lord of the Rings: The Two Towers
Finding Nemo

Dr. Strar

Movie Title

What is the average budget for a movie in the dataset?

In [20]:
```
# Convert the "budget" column to a numeric data type, and replace "Not Available" valu
movies_data['budget'] = pd.to_numeric(movies_data['budget'], errors='coerce')

# Compute the mean budget excluding NaN Values
average_budget = movies_data['budget'].mean(skipna=True)

print(f"The average budget for a movie in the dataset is: {average_budget:.2f}")
```

```
The average budget for a movie in the dataset is: 52912272.62
```

What is the average box office revenue for a movie in the dataset?

In [21]:
```
# Replace "Not Available" value with NaN
movies_data['box_office'] = pd.to_numeric(movies_data['box_office'],errors='coerce')

# Convert the Box-Office column to float datatype
movies_data['box_office'] = movies_data['box_office'].astype(float)

# Calculate the average box office revenue for movies in the dataset
avg_box_office = movies_data['box_office'].mean()

print(f"The average box office revenue for a movie in the dataset is {avg_box_office:.
```

```
The average box office revenue for a movie in the dataset is 238207554.30
```

What is the average runtime of a movie in the dataset?

In [22]:
```
# Define a function to convert runtime string to minutes
def runtime_to_minutes(runtime_str):
    if runtime_str == "Not Available":
        return np.nan
    elif "h " in runtime_str:
        hours, minutes = runtime_str.split("h ")
        return int(hours) * 60 + int(minutes[:-1])
    else:
        return int(runtime_str[:-1])

# Apply the function to the run_time column
movies_data['runtime_minutes'] = movies_data['run_time'].apply(runtime_to_minutes)

# Calculate the average runtime in miunutes
```

```
avg_runtime = movies_data['runtime_minutes'].mean()

# Convert the average runtime back to hours and minutes format
hours = int(avg_runtime/60)
minutes = int(avg_runtime % 60)
print(f"The average runtime of a movie in the dataset is {hours}h {minutes}m.")
```

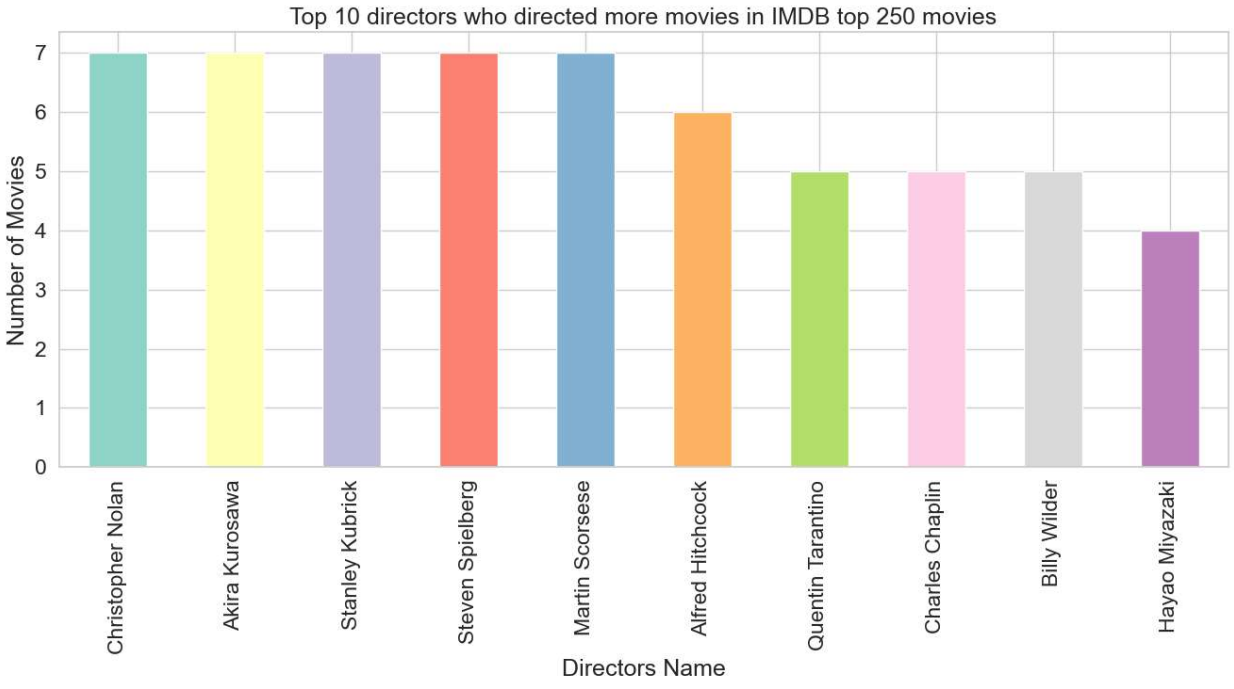
```
The average runtime of a movie in the dataset is 2h 6m.
```

Which directors have the most movies in the dataset?

In [23]:
```
director_count = movies_data['directors'].value_counts()
display(director_count.head(10))
```

```
Martin Scorsese      7
Christopher Nolan    7
Steven Spielberg     7
Stanley Kubrick      7
Akira Kurosawa       7
Alfred Hitchcock     6
Billy Wilder         5
Charles Chaplin      5
Quentin Tarantino    5
Hayao Miyazaki       4
Name: directors, dtype: int64
```

In [24]:
```
plt.figure(figsize=(16,6))
movies_data.groupby('directors').size().sort_values(ascending=False).head(10).plot(kir
plt.xlabel("Directors Name")
plt.ylabel("Number of Movies")
plt.title("Top 10 directors who directed more movies in IMDB top 250 movies")
plt.show()
```



In [ ]: