PLOT GRAPHS USING IMDB MOVIE RATING ANALYSIS DATASET THROUGH KAGGLE REPOSITORY

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

In [4]:
```
movies = pd.read_csv(r'C:\Users\sss\Desktop\imdbdataset\movie.csv')
print(type(movies))
movies.head(20)
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Comedy
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller

In [7]:
```
tags = pd.read_csv(r'C:\Users\sss\Desktop\imdbdataset\tag.csv')
tags.head()
```

Out[7]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
1	65	208	dark hero	2013-05-10 01:41:18
2	65	353	dark hero	2013-05-10 01:41:19
3	65	521	noir thriller	2013-05-10 01:39:43
4	65	592	dark hero	2013-05-10 01:41:18

In [11]:
```
ratings = pd.read_csv(r'C:\Users\sss\Desktop\imdbdataset\rating.csv')
ratings.head()
```

Out[11]:

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40

In [12]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [ ]:
```
tags.columns # timestamp column deleted
```

Out[ ]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [ ]:
```
ratings.columns # timestamp column deleted
```

Out[ ]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

In [15]:
```
movies.columns
```

Out[15]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [16]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[16]:
```
pandas.core.series.Series
```

In [17]:
```
print(row_0)  # Display the first row of the tags DataFrame
```

```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [18]:
```
row_0.index  # Display the index of the first row
```

Out[18]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [19]:
```
row_0['userId']  # Access the 'userId' value from the first row
```

Out[19]:
```
np.int64(18)
```

In [20]:
```
'rating'in row_0  # Check if 'rating' is a key in the first row
```

Out[20]:
```
False
```

In [21]:
```
row_0.name
```

Out[21]:
```
0
```

In [22]:
```
row_0=row_0.rename('firstrow')  # Rename the Series to 'firstrow'
row_0.name
```

Out[22]:
```
'firstrow'
```

In [23]:
```
tags.head()  # Display the first few rows of the tags DataFrame
```

Out[23]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero

In [24]:
```
tags.index  # Display the index of the tags DataFrame
```

Out[24]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [25]:
```
tags.columns
```

Out[25]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [26]:
```
tags.iloc[[0,11,500]]  # Display specific rows from the tags DataFrame
```

Out[26]:

	userId	movieId	tag
0	18	4141	Mark Waters
11	65	1783	noir thriller
500	342	55908	entirely dialogue

In [27]:
```
ratings['rating'].describe()  # Get descriptive statistics for the 'rating' co
```

Out[27]:
```
count    2.000026e+07
mean     3.525529e+00
std      1.051989e+00
min      5.000000e-01
25%      3.000000e+00
50%      3.500000e+00
75%      4.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

In [28]:
```
ratings.describe()  # Get descriptive statistics for the entire ratings DataFr
```

Out[28]:

	userId	movieId	rating
count	2.000026e+07	2.000026e+07	2.000026e+07
mean	6.904587e+04	9.041567e+03	3.525529e+00
std	4.003863e+04	1.978948e+04	1.051989e+00
min	1.000000e+00	1.000000e+00	5.000000e-01
25%	3.439500e+04	9.020000e+02	3.000000e+00
50%	6.914100e+04	2.167000e+03	3.500000e+00
75%	1.036370e+05	4.770000e+03	4.000000e+00
max	1.384930e+05	1.312620e+05	5.000000e+00

In [29]:
```
ratings['rating'].mean()  # Calculate the mean of the 'rating' column
```

Out[29]:
```
np.float64(3.5255285642993797)
```

In [30]:
```
ratings.mean()  # Calculate the mean of all numeric columns in ratings DataFra
```

Out[30]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [31]:
```
ratings['rating'].min()  # Get the minimum value of the 'rating' column
```

Out[31]:
```
np.float64(0.5)
```

In [32]:
```
ratings['rating'].max()  # Get the maximum value of the 'rating' column
```

Out[32]:
```
np.float64(5.0)
```

In [33]:
```
ratings['rating'].std()  # Calculate the standard deviation of the 'rating' co
```

Out[33]:
```
np.float64(1.0519889192942418)
```

In [34]:
```
ratings['rating'].mode()  # Calculate the mode of the 'rating' column
```

Out[34]:
```
0    4.0
Name: rating, dtype: float64
```

In [35]:
```
ratings.corr()  # Calculate the correlation between numeric columns in ratings
```

Out[35]:

	userId	movieId	rating
userId	1.000000	-0.000850	0.001175
movieId	-0.000850	1.000000	0.002606
rating	0.001175	0.002606	1.000000

In [36]:
```
filter1 = ratings['rating'] > 10
print(filter1)
filter1.any()
```

```
0           False
1           False
2           False
3           False
4           False
            ...
20000258    False
20000259    False
20000260    False
20000261    False
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

Out[36]:
```
np.False_
```

In [37]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[37]:
```
np.True_
```

In [38]:
```
movies.shape
```

Out[38]:
```
(27278, 3)
```

In [39]:
```
movies.isnull().any().any()  # Check for any null values in the movies DataFra
```

Out[39]: np.False_

In [40]:
```
ratings.shape  # Display the shape of the ratings DataFrame
```

Out[40]: (20000263, 3)

In [41]:
```
ratings.isnull().any().any()
```

Out[41]: np.False_

In [42]:
```
tags.shape  # Display the shape of the tags DataFrame
```

Out[42]: (465564, 3)

In [43]:
```
tags.isnull().any().any()
```

Out[43]: np.True_

In [44]:
```
tags=tags.dropna()  # Drop rows with any null values in the tags DataFrame
```

In [45]:
```
tags.isnull().any().any()
```

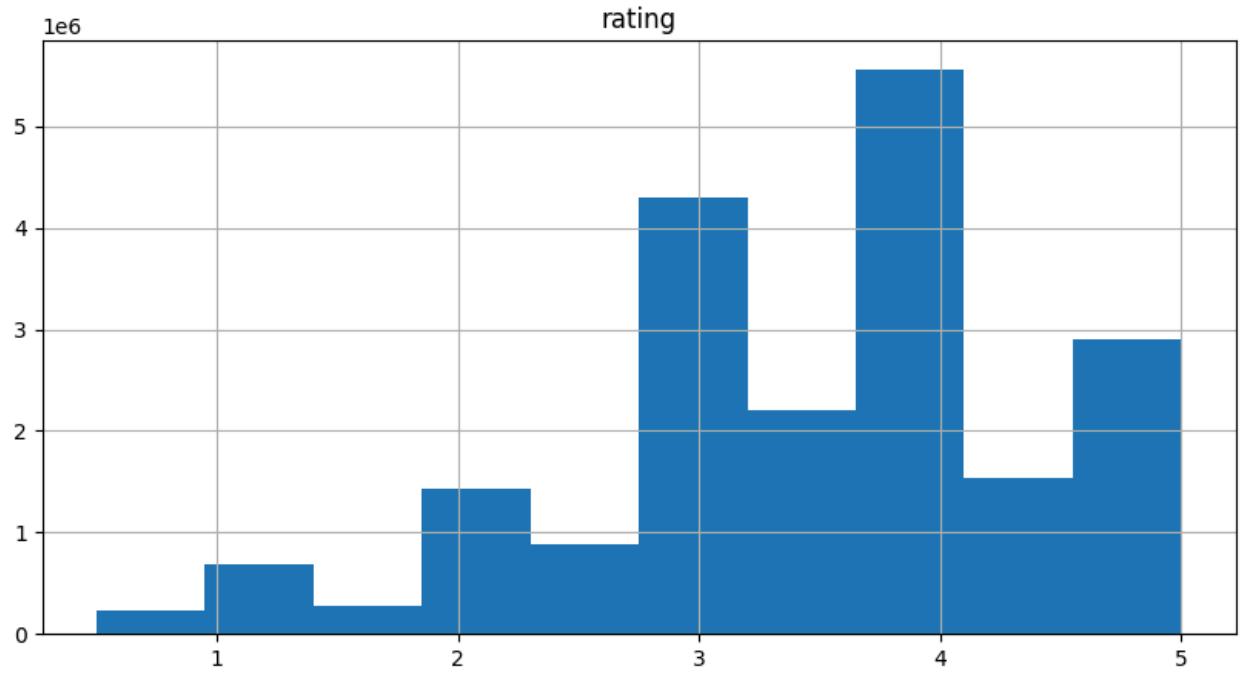
Out[45]: np.False_

In [46]:
```
tags.shape  # Display the shape of the tags DataFrame
```

Out[46]: (465548, 3)

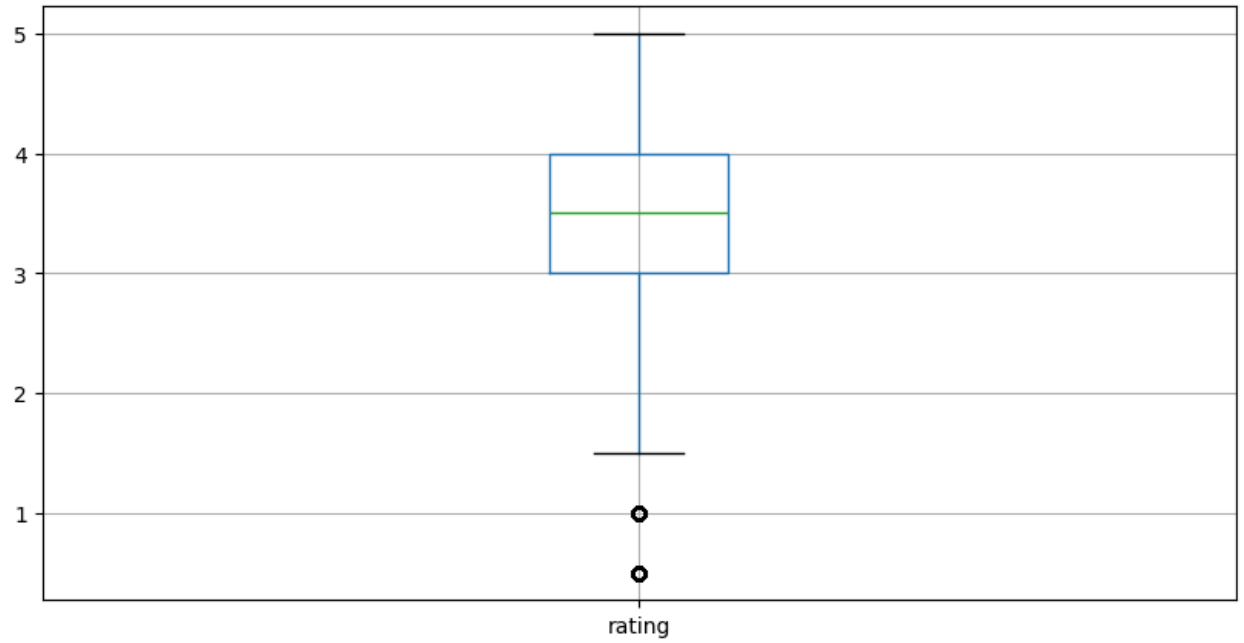
In [47]:
```
%matplotlib inline
import matplotlib.pyplot as plt
ratings.hist(column='rating',figsize=(10,5))
```

Out[47]: array([[<Axes: title={'center': 'rating'}>]], dtype=object)



In [48]:
```
ratings.boxplot(column='rating',figsize=(10,5))  # Create a box plot for the '
```

Out[48]: <Axes: >



In [49]:
```
tags['tag'].head()
```

Out[49]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```