# Pandas With Data Science.AI

## Import Libraries

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

In [2]:
```
pd.__version__
```

Out[2]: '2.2.2'

In [3]:
```
ratings  = pd.read_csv(r"D:\fullstackNaresh\imdbproject\archive\rating.csv")
tags  = pd.read_csv(r"D:\fullstackNaresh\imdbproject\archive\tag.csv")
movies = pd.read_csv(r"D:\fullstackNaresh\imdbproject\archive\movie.csv")
```

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

Out[4]: (20000263, 4)

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

Out[5]: (465564, 4)

In [6]:
```
movies.shape
```

Out[6]: (27278, 3)

In [7]:
```
ratings.head()
```

Out[7]:

| | 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 [8]:
```
tags.head()
```

Out[8]:

| | 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 [9]:
```
movies.head()
```

Out[9]:

| | 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 |

In [10]:
```
del ratings['timestamp']
```

In [11]:
```
del tags['timestamp']
```

In [12]:
```
ratings.head()
```

Out[12]:

| | userId | movieId | rating |
|---|---|---|---|
| **0** | 1 | 2 | 3.5 |
| **1** | 1 | 29 | 3.5 |
| **2** | 1 | 32 | 3.5 |
| **3** | 1 | 47 | 3.5 |
| **4** | 1 | 50 | 3.5 |

In [13]:
```
tags.head()
```

Out[13]:

| | 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 [14]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[14]:
```
pandas.core.series.Series
```

In [15]:
```
tags.iloc[0]
```

Out[15]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [16]:
```
tags.iloc[0,2]
```

Out[16]:
```
'Mark Waters'
```

In [17]:
```
print(row_0)
```

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

In [18]:
```
row_0.index
```

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

In [19]:
```
row_0['userId']
```

Out[19]:
```
18
```

In [20]:
```
row_0['movieId']
```

Out[20]:
```
4141
```

In [21]:
```
row_0['tag']
```

Out[21]:
```
'Mark Waters'
```

In [22]:
```
'rating' in row_0
```

Out[22]:
```
False
```

In [23]:
```
row_0.name
```

Out[23]:
```
0
```

In [24]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[24]:
```
'firstRow'
```

In [25]:
```
row_0.index
```

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

In [26]:
```
tags.head()
```

Out[26]:

| | 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 [27]:
```
tags.index
```

Out[27]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [28]:
```
tags.columns
```

Out[28]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [29]:
```
tags.iloc[ [0,11,500] ]
```

Out[29]:

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

In [30]:
```
ratings['rating'].describe()
```

Out[30]:
```
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 [31]:
```
ratings.describe()
```

Out[31]:

| | 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 [32]:
```
ratings['rating'].mean()
```

Out[32]:
```
3.5255285642993797
```

In [33]:
```
ratings.mean()
```

Out[33]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [34]:
```
ratings['rating'].min()
```

Out[34]:
```
0.5
```

In [35]:
```
ratings['rating'].max()
```

Out[35]:
```
5.0
```

In [36]:
```
ratings['rating'].std()
```

Out[36]:
```
1.051988919275684
```

In [37]:
```
ratings['rating'].mode()
```

Out[37]:
```
0    4.0
Name: rating, dtype: float64
```

In [38]:
```
ratings.corr()
```

Out[38]:

| | 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 [39]:
```
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[39]:
```
False
```

In [40]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[40]:
```
True
```

# Data Cleaning: Handling Missing Data

In [41]:
```
movies.shape
```

Out[41]:
```
(27278, 3)
```

In [42]:
```
movies.isnull().any().any()
```

Out[42]:
```
False
```

In [43]:
```
ratings.shape
```

Out[43]:
```
(20000263, 3)
```

In [44]:
```
ratings.isnull().any().any()
```

Out[44]: False

In [45]:
```
tags.shape
```

Out[45]: (465564, 3)

In [46]:
```
tags.isnull().any().any()
```

Out[46]: True

In [47]:
```
tags=tags.dropna()
```

In [48]:
```
tags.isnull().any().any()
```

Out[48]: False

In [49]:
```
tags.shape
```

Out[49]: (465548, 3)

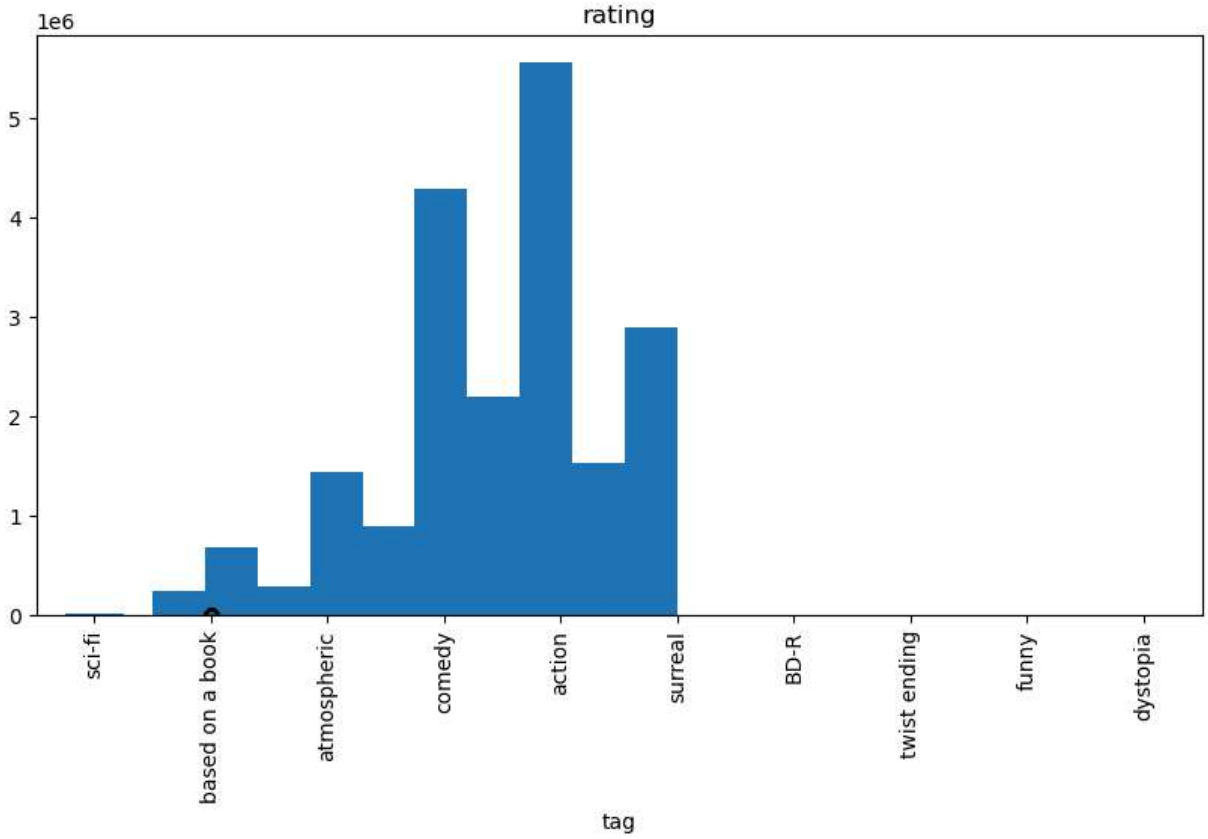
# Data Visualization

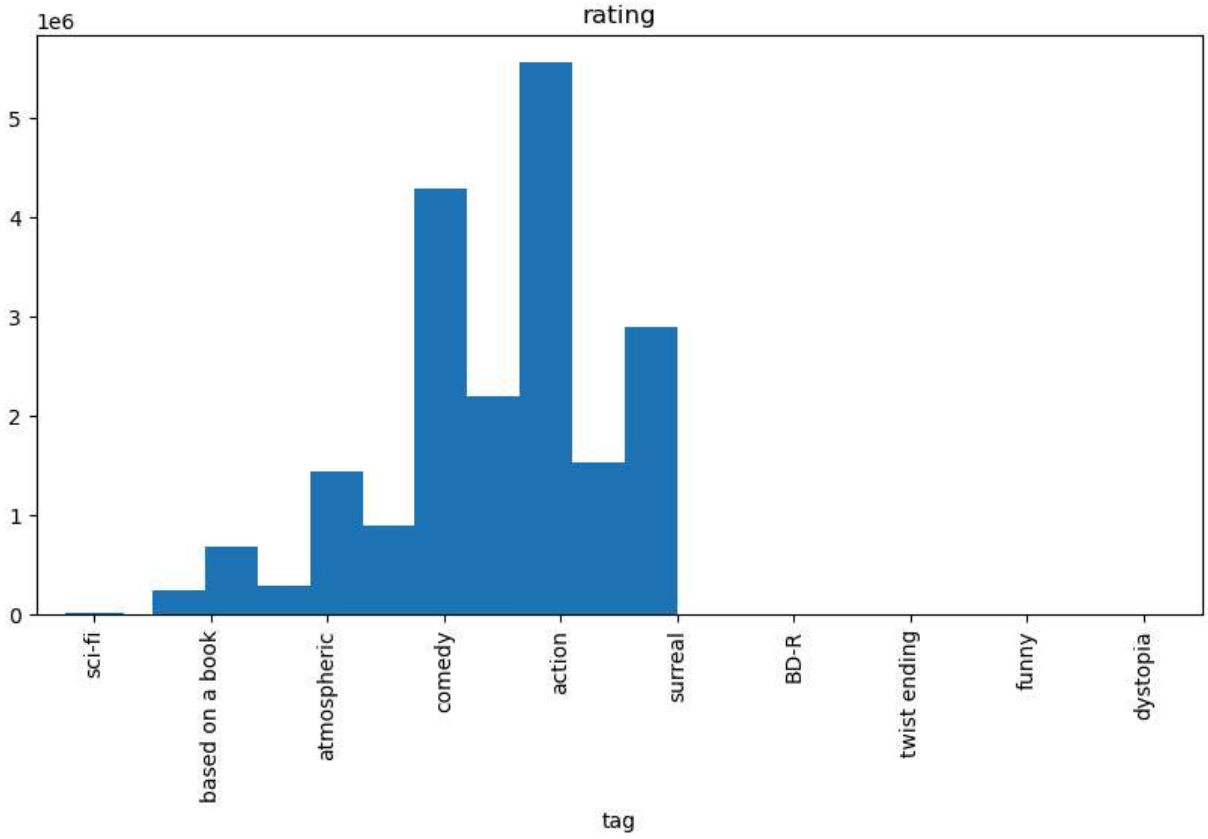
In [82]:
```
%matplotlib inline
```

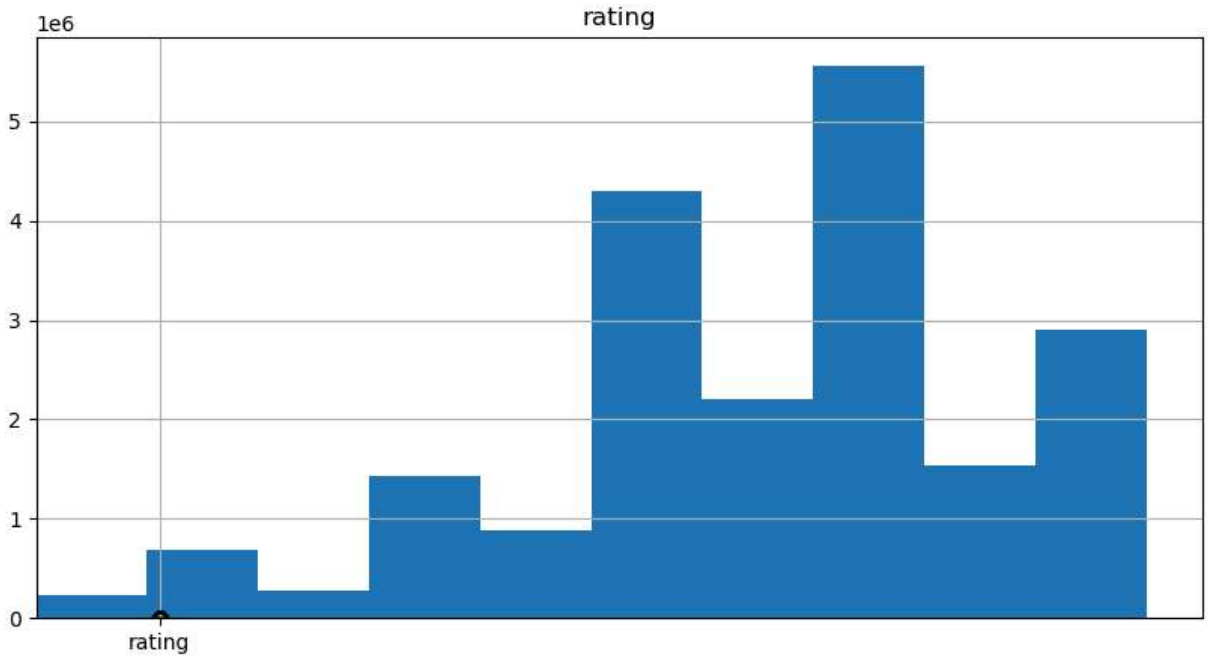
In [83]:
```
ratings.hist(column='rating', figsize=(10,5))
```

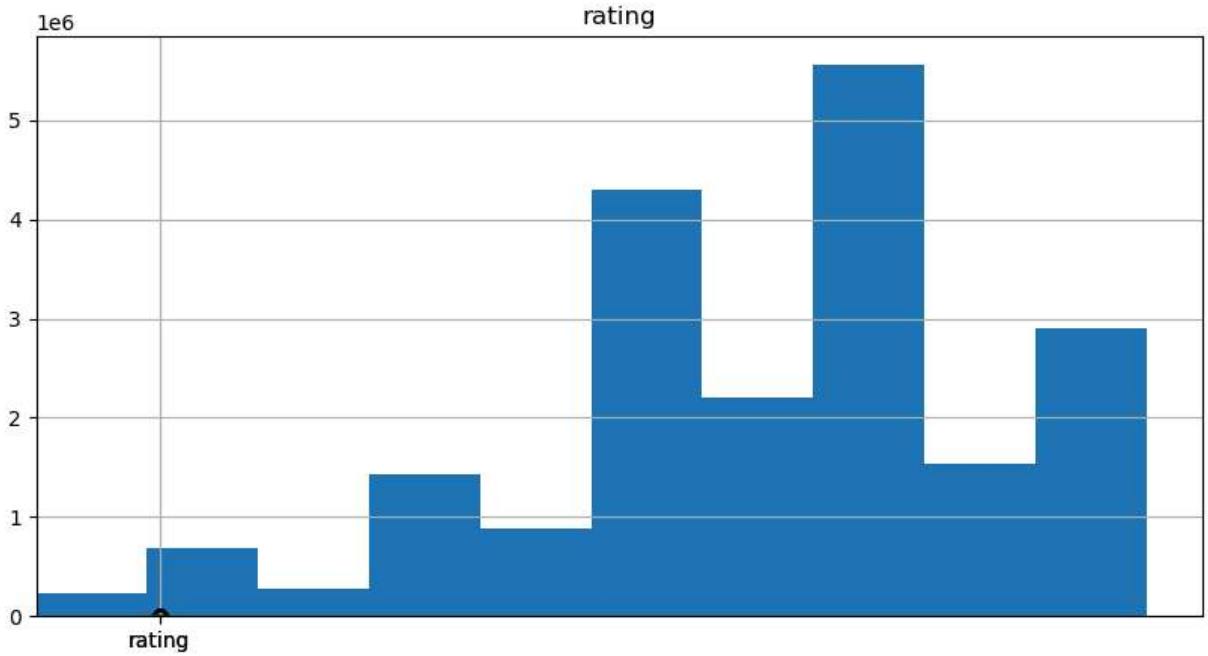
Out[83]: array([[<Axes: title={'center': 'rating'}>]], dtype=object)

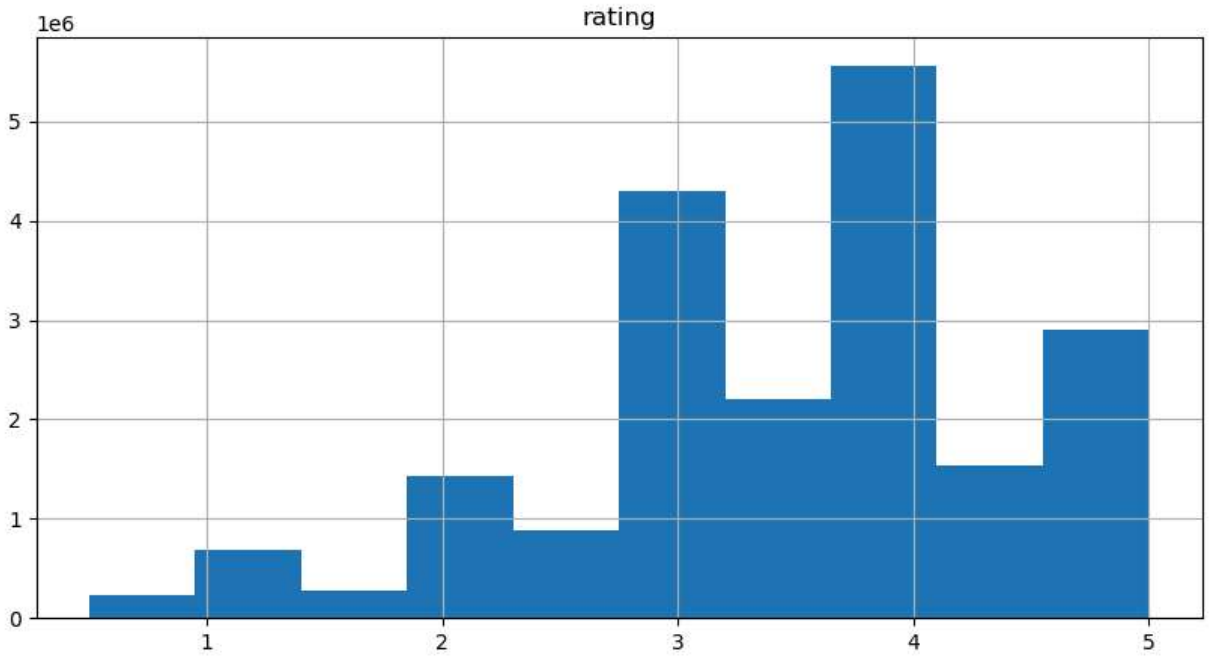
In [86]:
```
%matplotlib inline
import matplotlib.pyplot as plt
ratings.hist(column='rating',figsize=(10,5))
plt.show()
```


rating
1e6
0
1
2
3
4
5
sci-fi
based on a book
atmospheric
comedy
action
surreal
BD-R
twist ending
funny
dystopia
tag


rating
1e6
0
1
2
3
4
5
sci-fi
based on a book
atmospheric
comedy
action
surreal
BD-R
twist ending
funny
dystopia
tag


rating
1e6
5
4
3
2
1
0
rating


rating
1e6
5
4
3
2
1
0
rating




In [85]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[85]:
```
<Axes: title={'center': 'rating'}>
```

In [52]:
```
tags['tag'].head()
```

Out[52]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [53]:
```
movies[['title','genres']].head()
```

Out[53]:

| | title | genres |
|---|---|---|
| **0** | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| **1** | Jumanji (1995) | Adventure\|Children\|Fantasy |
| **2** | Grumpier Old Men (1995) | Comedy\|Romance |
| **3** | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| **4** | Father of the Bride Part II (1995) | Comedy |

In [54]:
```
ratings[-10:]
```

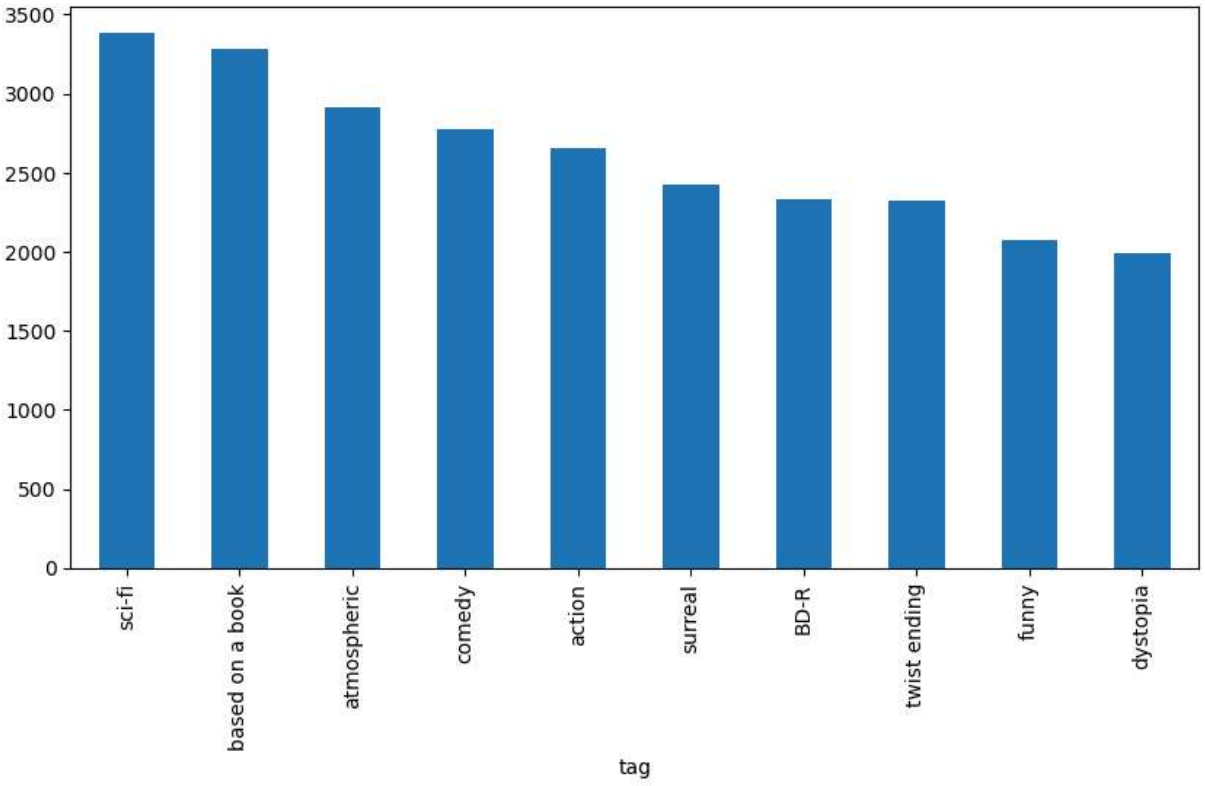
Out[54]:

| | userId | movieId | rating |
|---|---|---|---|
| **20000253** | 138493 | 60816 | 4.5 |
| **20000254** | 138493 | 61160 | 4.0 |
| **20000255** | 138493 | 65682 | 4.5 |
| **20000256** | 138493 | 66762 | 4.5 |
| **20000257** | 138493 | 68319 | 4.5 |
| **20000258** | 138493 | 68954 | 4.5 |
| **20000259** | 138493 | 69526 | 4.5 |
| **20000260** | 138493 | 69644 | 3.0 |
| **20000261** | 138493 | 70286 | 5.0 |
| **20000262** | 138493 | 71619 | 2.5 |

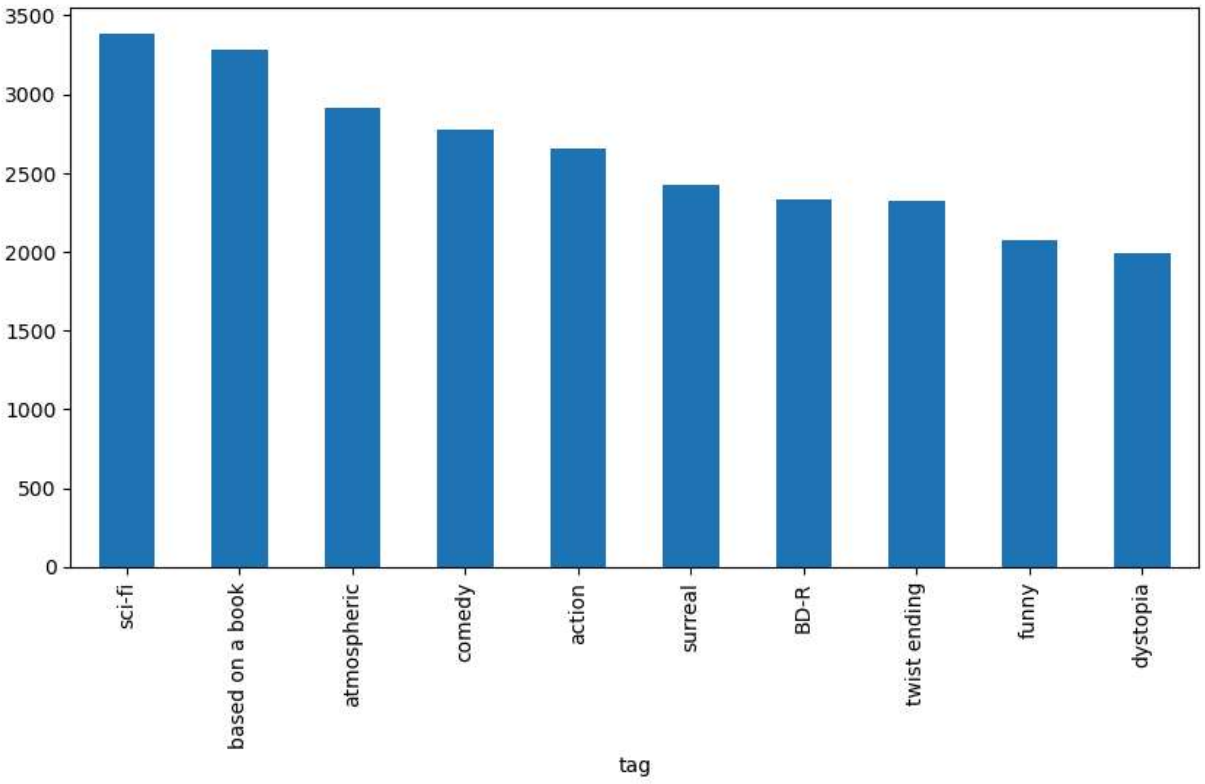
In [55]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[55]:
```
tag
missing child                  1
Ron Moore                      1
Citizen Kane                   1
mullet                         1
biker gang                     1
Paul Adelstein                 1
the wig                        1
killer fish                    1
genetically modified monsters  1
topless scene                  1
Name: count, dtype: int64
```

In [87]:
```
%matplotlib inline
import matplotlib.pyplot as plt
tag_counts[:10].plot(kind='bar', figsize=(10,5))
plt.show()
```




In [88]:
```
%matplotlib inline
import matplotlib.pyplot as plt
tag_counts[:10].plot(kind='bar', figsize=(10,5))
plt.show()
```




# Filters for Selecting Rows

In [58]:
```
is_highly_rated = ratings['rating'] >= 5.0
ratings[is_highly_rated][30:50]
```

Out[58]:

| | userId | movieId | rating |
|---|---|---|---|
| **239** | 3 | 50 | 5.0 |
| **242** | 3 | 175 | 5.0 |
| **244** | 3 | 223 | 5.0 |
| **245** | 3 | 260 | 5.0 |
| **246** | 3 | 316 | 5.0 |
| **247** | 3 | 318 | 5.0 |
| **248** | 3 | 329 | 5.0 |
| **252** | 3 | 457 | 5.0 |
| **253** | 3 | 480 | 5.0 |
| **254** | 3 | 490 | 5.0 |
| **256** | 3 | 541 | 5.0 |
| **258** | 3 | 593 | 5.0 |
| **263** | 3 | 858 | 5.0 |
| **264** | 3 | 904 | 5.0 |
| **267** | 3 | 924 | 5.0 |
| **268** | 3 | 953 | 5.0 |
| **271** | 3 | 1060 | 5.0 |
| **272** | 3 | 1073 | 5.0 |
| **275** | 3 | 1084 | 5.0 |
| **276** | 3 | 1089 | 5.0 |

In [59]:
```
is_action= movies['genres'].str.contains('Action')
movies[is_action][5:15]
```

Out[59]:

| | movieId | title | genres |
|---|---|---|---|
| **22** | 23 | Assassins (1995) | Action\|Crime\|Thriller |
| **41** | 42 | Dead Presidents (1995) | Action\|Crime\|Drama |
| **43** | 44 | Mortal Kombat (1995) | Action\|Adventure\|Fantasy |
| **50** | 51 | Guardian Angel (1994) | Action\|Drama\|Thriller |
| **65** | 66 | Lawnmower Man 2: Beyond Cyberspace (1996) | Action\|Sci-Fi\|Thriller |
| **69** | 70 | From Dusk Till Dawn (1996) | Action\|Comedy\|Horror\|Thriller |
| **70** | 71 | Fair Game (1995) | Action |
| **75** | 76 | Screamers (1995) | Action\|Sci-Fi\|Thriller |
| **77** | 78 | Crossing Guard, The (1995) | Action\|Crime\|Drama\|Thriller |
| **85** | 86 | White Squall (1996) | Action\|Adventure\|Drama |

In [60]:
```
movies[is_action].head(15)
```

Out[60]:

| | movieId | title | genres |
|---|---|---|---|
| **5** | 6 | Heat (1995) | Action\|Crime\|Thriller |
| **8** | 9 | Sudden Death (1995) | Action |
| **9** | 10 | GoldenEye (1995) | Action\|Adventure\|Thriller |
| **14** | 15 | Cutthroat Island (1995) | Action\|Adventure\|Romance |
| **19** | 20 | Money Train (1995) | Action\|Comedy\|Crime\|Drama\|Thriller |
| **22** | 23 | Assassins (1995) | Action\|Crime\|Thriller |
| **41** | 42 | Dead Presidents (1995) | Action\|Crime\|Drama |
| **43** | 44 | Mortal Kombat (1995) | Action\|Adventure\|Fantasy |
| **50** | 51 | Guardian Angel (1994) | Action\|Drama\|Thriller |
| **65** | 66 | Lawnmower Man 2: Beyond Cyberspace (1996) | Action\|Sci-Fi\|Thriller |
| **69** | 70 | From Dusk Till Dawn (1996) | Action\|Comedy\|Horror\|Thriller |
| **70** | 71 | Fair Game (1995) | Action |
| **75** | 76 | Screamers (1995) | Action\|Sci-Fi\|Thriller |
| **77** | 78 | Crossing Guard, The (1995) | Action\|Crime\|Drama\|Thriller |
| **85** | 86 | White Squall (1996) | Action\|Adventure\|Drama |

# Group By and Aggregate

In [61]:
```
ratings_count = ratings[['movieId','rating']].groupby('rating').count()
ratings_count
```

Out[61]:

| rating | movieId |
|---|---|
| **0.5** | 239125 |
| **1.0** | 680732 |
| **1.5** | 279252 |
| **2.0** | 1430997 |
| **2.5** | 883398 |
| **3.0** | 4291193 |
| **3.5** | 2200156 |
| **4.0** | 5561926 |
| **4.5** | 1534824 |
| **5.0** | 2898660 |

In [62]:
```
average_rating = ratings[['movieId','rating']].groupby('movieId').mean()
average_rating.head()
```

Out[62]:

| movieId | rating |
|---|---|
| **1** | 3.921240 |
| **2** | 3.211977 |
| **3** | 3.151040 |
| **4** | 2.861393 |
| **5** | 3.064592 |

In [63]:
```
movie_count = ratings[['movieId','rating']].groupby('movieId').count()
movie_count.head()
```

Out[63]:

| movieId | rating |
|---|---|
| 1 | 49695 |
| 2 | 22243 |
| 3 | 12735 |
| 4 | 2756 |
| 5 | 12161 |

In [64]:
```
movie_count = ratings[['movieId','rating']].groupby('movieId').count()
movie_count.tail()
```

Out[64]:

| movieId | rating |
|---|---|
| 131254 | 1 |
| 131256 | 1 |
| 131258 | 1 |
| 131260 | 1 |
| 131262 | 1 |

# Merge Dataframes

In [65]:
```
tags.head()
```

Out[65]:

| | 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 [66]:
```
movies.head()
```

Out[66]:

| | 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 |

In [67]:
```
t = movies.merge(tags, on='movieId', how='inner')
t.head()
```

Out[67]:

| | movieId | title | genres | userId | tag |
|---|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1644 | Watched |
| **1** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1741 | computer animation |
| **2** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1741 | Disney animated feature |
| **3** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1741 | Pixar animation |
| **4** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1741 | TÃ©a Leoni does not star in this movie |

In [68]:
```
avg_ratings= ratings.groupby('movieId', as_index=False).mean()
del avg_ratings['userId']
avg_ratings.head()
```

Out[68]:

| | movieId | rating |
|---|---|---|
| **0** | 1 | 3.921240 |
| **1** | 2 | 3.211977 |
| **2** | 3 | 3.151040 |
| **3** | 4 | 2.861393 |
| **4** | 5 | 3.064592 |

In [69]:
```
box_office = movies.merge(avg_ratings, on='movieId', how='inner')
box_office.tail()
```

Out[69]:

| | movieId | title | genres | rating |
|---|---|---|---|---|
| **26739** | 131254 | Kein Bund für's Leben (2007) | Comedy | 4.0 |
| **26740** | 131256 | Feuer, Eis & Dosenbier (2002) | Comedy | 4.0 |
| **26741** | 131258 | The Pirates (2014) | Adventure | 2.5 |
| **26742** | 131260 | Rentun Ruusu (2001) | (no genres listed) | 3.0 |
| **26743** | 131262 | Innocence (2014) | Adventure\|Fantasy\|Horror | 4.0 |

In [70]:
```
is_highly_rated = box_office['rating'] >= 4.0
box_office[is_highly_rated][-5:]
```

Out[70]:

| | movieId | title | genres | rating |
|---|---|---|---|---|
| **26737** | 131250 | No More School (2000) | Comedy | 4.0 |
| **26738** | 131252 | Forklift Driver Klaus: The First Day on the Jo... | Comedy\|Horror | 4.0 |
| **26739** | 131254 | Kein Bund für's Leben (2007) | Comedy | 4.0 |
| **26740** | 131256 | Feuer, Eis & Dosenbier (2002) | Comedy | 4.0 |
| **26743** | 131262 | Innocence (2014) | Adventure\|Fantasy\|Horror | 4.0 |

In [71]:
```
is_Adventure = box_office['genres'].str.contains('Adventure')
box_office[is_Adventure][:5]
```

Out[71]:

| | movieId | title | genres | rating |
|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 3.921240 |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy | 3.211977 |
| **7** | 8 | Tom and Huck (1995) | Adventure\|Children | 3.142049 |
| **9** | 10 | GoldenEye (1995) | Action\|Adventure\|Thriller | 3.430029 |
| **12** | 13 | Balto (1995) | Adventure\|Animation\|Children | 3.272416 |

In [72]:
```
box_office[is_Adventure & is_highly_rated][-5:]
```

Out[72]:

| | movieId | title | genres | rating |
|---|---|---|---|---|
| **26611** | 130586 | Itinerary of a Spoiled Child (1988) | Adventure\|Drama | 4.5 |
| **26655** | 130996 | The Beautiful Story (1992) | Adventure\|Drama\|Fantasy | 5.0 |
| **26667** | 131050 | Stargate SG-1 Children of the Gods - Final Cut... | Adventure\|Sci-Fi\|Thriller | 5.0 |
| **26736** | 131248 | Brother Bear 2 (2006) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 4.0 |
| **26743** | 131262 | Innocence (2014) | Adventure\|Fantasy\|Horror | 4.0 |

# Vectorized String Operations

In [73]:
```
movies.head()
```

Out[73]:

| | 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 |

# Split 'genres' into multiple columns

In [74]:
```
movie_genres = movies['genres'].str.split('|', expand=True)
```

In [75]:
```
movie_genres[:10]
```

Out[75]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Adventure | Animation | Children | Comedy | Fantasy | None | None | None | None | None |
| **1** | Adventure | Children | Fantasy | None | None | None | None | None | None | None |
| **2** | Comedy | Romance | None | None | None | None | None | None | None | None |
| **3** | Comedy | Drama | Romance | None | None | None | None | None | None | None |
| **4** | Comedy | None | None | None | None | None | None | None | None | None |
| **5** | Action | Crime | Thriller | None | None | None | None | None | None | None |
| **6** | Comedy | Romance | None | None | None | None | None | None | None | None |
| **7** | Adventure | Children | None | None | None | None | None | None | None | None |
| **8** | Action | None | None | None | None | None | None | None | None | None |
| **9** | Action | Adventure | Thriller | None | None | None | None | None | None | None |

# Extract year from title e.g. (2007)

In [76]:
```
movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
movies.tail()
```

```
<>:1: SyntaxWarning: invalid escape sequence '\('
<>:1: SyntaxWarning: invalid escape sequence '\('
C:\Users\ekris\AppData\Local\Temp\ipykernel_44088\3464295542.py:1: SyntaxWarning: in
valid escape sequence '\('
  movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
```

Out[76]:

| | movieId | title | genres | year |
|---|---|---|---|---|
| **27273** | 131254 | Kein Bund für's Leben (2007) | Comedy | 2007 |
| **27274** | 131256 | Feuer, Eis & Dosenbier (2002) | Comedy | 2002 |
| **27275** | 131258 | The Pirates (2014) | Adventure | 2014 |
| **27276** | 131260 | Rentun Ruusu (2001) | (no genres listed) | 2001 |
| **27277** | 131262 | Innocence (2014) | Adventure\|Fantasy\|Horror | 2014 |

In [77]:
```
%matplotlib inline

ratings.hist(column='rating', figsize=(10,5))
```

Out[77]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```

In [ ]: