Import the dataset

In [58]:
```
import pandas as pd
```

Read the dataset

In [59]:
```
movies = pd.read_csv(r'D:\data science pandas\movie.csv')
```

In [60]:
```
movies = pd.read_csv(r'D:\data science pandas\movie.csv')
print(type(movies))
movies.head(20)
```

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

Out[60]:

	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 [61]:
```
ratings = pd.read_csv(r'D:\data science pandas\rating.csv')
```

In [62]:
```
tags = pd.read_csv(r'D:\data science pandas\tag.csv')
```

In [63]:
```
print(movies.shape)
print(ratings.shape)
print(tags.shape)
```

```
(27278, 3)
(20000263, 4)
(465564, 4)
```

In [64]:
```
print(movies.columns)
print(ratings.columns)
print(tags.columns)
```

```
Index(['movieId', 'title', 'genres'], dtype='object')
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [65]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [66]:
```
print(movies.columns)
print(ratings.columns)
print(tags.columns)
```

```
Index(['movieId', 'title', 'genres'], dtype='object')
Index(['userId', 'movieId', 'rating'], dtype='object')
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [67]:
```
tags.head(2)
```

Out[67]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero

In [68]:
```
row_0 = tags.iloc[1]
```

In [69]:
```
print(row_0)
```

```
userId            65
movieId          208
tag        dark hero
Name: 1, dtype: object
```

In [70]:
```
row_0.index
```

Out[70]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [71]:
```
row_0['userId']
```

Out[71]:
```
65
```

In [72]:
```
'raying' in row_0
```

Out[72]:
```
False
```

In [73]:
```
row_0.name
```

Out[73]:
```
1
```

In [74]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[74]:
```
'firstRow'
```

```
Data frames
```

In [75]:
```
tags.head()
```

Out[75]:

	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 [76]:
```
tags.index
```

Out[76]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [77]:
```
tags.columns
```

Out[77]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [78]:
```
tags.iloc[[0,11,500]]
```

Out[78]:

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

```
Descriptive statistics

Lets look how the ratings are distributed
```

In [79]:
```
ratings['rating'].describe()
```

Out[79]:
```
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 [80]:
```
ratings.describe()
```

Out[80]:

	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 [81]:
```
ratings['rating'].mean()
```

Out[81]:
```
3.5255285642993797
```

In [82]:
```
ratings.mean()
```

Out[82]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [83]:
```
ratings['rating'].min()
```

Out[83]:
```
0.5
```

In [85]:
```
ratings['rating'].max()
```

Out[85]:
```
5.0
```

In [86]:
```
ratings['rating'].std()
```

Out[86]:
```
1.051988919275684
```

In [87]:
```
ratings['rating'].mode()
```

Out[87]:
```
0    4.0
Name: rating, dtype: float64
```

In [88]:
```
ratings.corr()
```

Out[88]:

	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 [89]:
```
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[89]:
```
False
```

In [90]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[90]:
```
True
```

```
Data cleaning handling missing data
```

In [91]:
```
movies.shape
```

Out[91]:
```
(27278, 3)
```

In [92]:
```
movies.isnull().any().any()
```

Out[92]:
```
False
```

In [93]:
```
ratings.shape
```

Out[93]:
```
(20000263, 3)
```

In [94]:
```
ratings.isnull().any().any()
```

Out[94]:
```
False
```

In [95]:
```
tags.shape
```

Out[95]:
```
(465564, 3)
```

In [96]:
```
tags.isnull().any().any()
```

Out[96]:
```
True
```

```
We have some tags which are NULL.
```

In [97]:
```
tags=tags.dropna()
```

In [98]:
```
tags.isnull().any().any()
```

Out[98]:
```
False
```

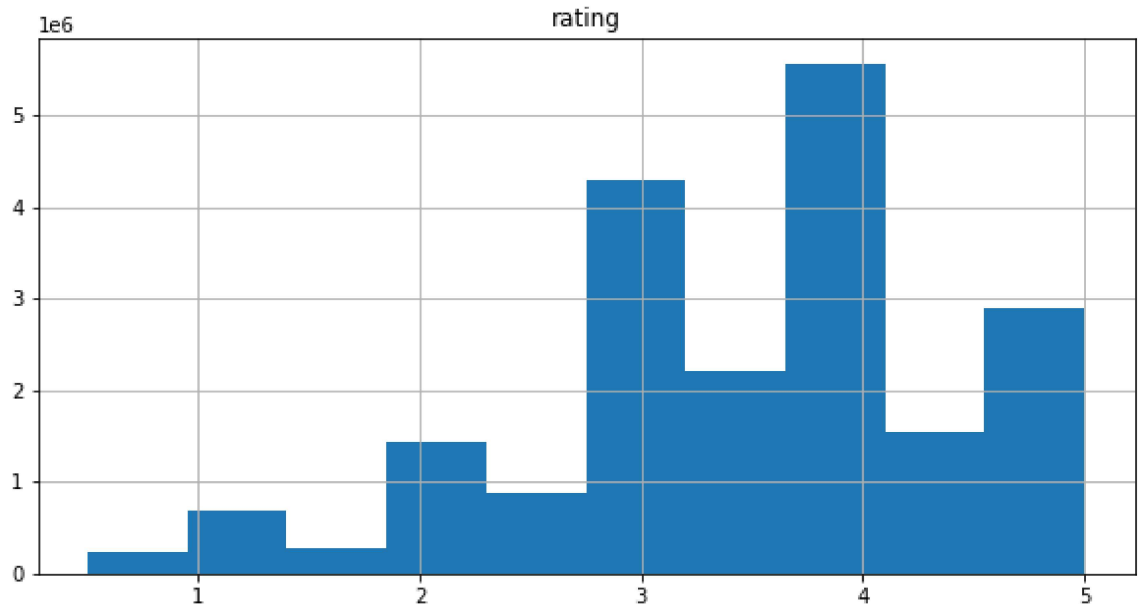
In [99]:
```
tags.shape
```

Out[99]:
```
(465548, 3)
```

```
Data visulation
```

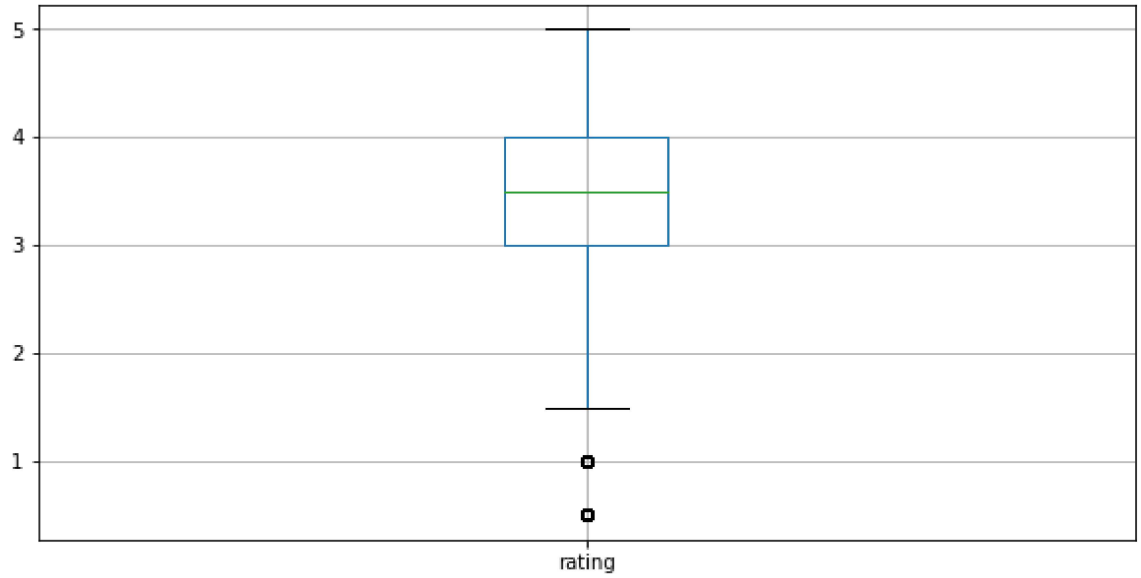
In [100]:
```
%matplotlib inline

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

Out[100]:
```
array([[<AxesSubplot:title={'center':'rating'}>]], dtype=object)
```



In [101]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[101]:
```
<AxesSubplot:>
```




```
Slicing out columns
```

In [102]:
```
tags['tag'].head()
```

Out[102]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [103]:
```
movies[['title','genres']].head()
```

Out[103]:

	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 [104]:
```
ratings[-10:]
```

Out[104]:

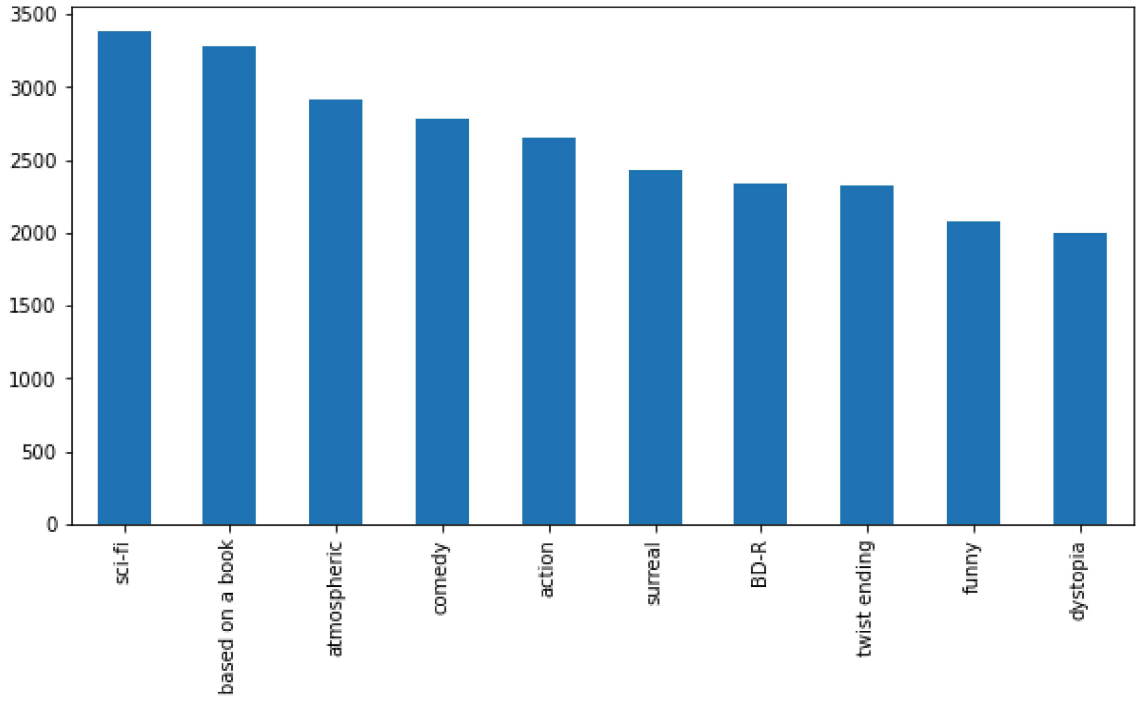
	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [105]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[105]:
```
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: tag, dtype: int64
```

In [106]:
```
tag_counts[:10].plot(kind='bar',figsize=(10,5))
```

Out[106]:
```
<AxesSubplot:>
```




```
Filter for selecting rows
```

In [107]:
```
is_highly_rated = ratings['rating'] >= 5.0
ratings[is_highly_rated][30:50]
```

Out[107]:

	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 [108]:
```
is_action=  movies['genres'] .str.contains('Action')
movies[is_action][5:15]
```

Out[108]:

	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 [109]:
```
movies[is_action].head(15)
```

Out[109]:

	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 [110]:
```
ratings_count = ratings[['movieId','rating']].groupby('rating').count()
ratings_count
```

Out[110]:

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 [111]:
```
ratings
```

Out[111]:

	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
...	...	...	...
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

20000263 rows × 3 columns

In [112]:
```
average_rating = ratings[['movieId','rating']].groupby('movieId').mean()
average_rating.head()
```

Out[112]:

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

In [113]:
```
movie_count = ratings[['movieId','rating']].groupby('movieId').count()
movie_count.head()
```

Out[113]:

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

In [114]:
```
movie_count = ratings[['movieId','rating']].groupby('movieId').count()
movie_count.tail()
```

Out[114]:

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

```
Merge Dataframes
```

In [115]:
```
tags.head()
```

Out[115]:

	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 [116]:
```
movies.head()
```

Out[116]:

	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 [117]:
```
t = movies.merge(tags, on='movieId', how='inner')
t.head()
```

Out[117]:

	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

```
Combine aggreagation, merging, and filters to get useful analytics
```

In [119]:
```
avg_ratings= ratings.groupby('movieId', as_index=False).mean()
del avg_ratings['userId']
avg_ratings.head()
```

Out[119]:

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

In [121]:
```
box_office = movies.merge(avg_ratings, on='movieId', how='inner')
box_office.tail()
```

Out[121]:

	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 [122]:
```
is_highly_rated = box_office['rating'] >= 4.0
box_office[is_highly_rated][-5:]
```

Out[122]:

	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 [123]:
```
is_Adventure = box_office['genres'].str.contains('Adventure')
box_office[is_Adventure][:5]
```

Out[123]:

	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 [124]:
```
box_office[is_Adventure & is_highly_rated][-5:]
```

Out[124]:

	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 [125]:
```
movies.head()
```

Out[125]:

	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 [136]:
```
movie_genres = movies['genres'].str.split('i', expand=True)
```

In [137]:
```
movie_genres[:10]
```

Out[137]:

0	1	2	3	4	5	6	7	8
Adventure|An	mat	on|Ch	ldren|Comedy|Fantasy	None	None	None	None	None
Adventure|Ch	ldren|Fantasy	None	None	None	None	None	None	None
ɔmedy|Romance	None	None	None	None	None	None	None	None
Drama|Romance	None	None	None	None	None	None	None	None
Comedy	None	None	None	None	None	None	None	None
Act	on|Cr	me|Thr	ller	None	None	None	None	None
ɔmedy|Romance	None	None	None	None	None	None	None	None
Adventure|Ch	ldren	None	None	None	None	None	None	None
Act	on	None	None	None	None	None	None	None
Act	on|Adventure|Thr	ller	None	None	None	None	None	None

Add a new column for comedy genre flag

In [138]:
```
movie_genres['isComedy'] = movies['genres'].str.contains('Comedy')
```

In [140]:
```
movie_genres[:10]
```

Out[140]:

	0	1	2	3	4	5	6
0	Adventure|An	mat	on|Ch	ldren|Comedy|Fantasy	None	None	None
1	Adventure|Ch	ldren|Fantasy	None	None	None	None	None
2	Comedy|Romance	None	None	None	None	None	None
3	Comedy|Drama|Romance	None	None	None	None	None	None
4	Comedy	None	None	None	None	None	None
5	Act	on|Cr	me|Thr	ller	None	None	None
6	Comedy|Romance	None	None	None	None	None	None
7	Adventure|Ch	ldren	None	None	None	None	None
8	Act	on	None	None	None	None	None
9	Act	on|Adventure|Thr	ller	None	None	None	None

Extract year from title e.g.(2007)

In [141]:
```
movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
```

In [142]:
```
movies.tail()
```

Out[142]:

	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

Parsing Timestamps

. Timestamps are common in sensor data or other time series datasets. Let us revisit the tags.csv dataset and read the timestamps!

In [150]:
```
tags = pd.read_csv(r'D:\data science pandas\tag.csv')
```

In [151]:
```
tags.head(5)
```

Out[151]:

	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 [ ]:
```
```