```
In [3]: import pandas as pd
```

```
In [4]: movies=pd.read_csv('D:\DATA SCIENCE full stack\My task\movie.csv', sep=',')
```

```
In [5]: movies
```

```
Out[5]:
        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
  ...       ...                               ...                                          ...
27273    131254        Kein Bund für's Leben (2007)                                   Comedy
27274    131256      Feuer, Eis & Dosenbier (2002)                                   Comedy
27275    131258                The Pirates (2014)                                    Adventure
27276    131260               Rentun Ruusu (2001)                           (no genres listed)
27277    131262                 Innocence (2014)                     Adventure|Fantasy|Horror

27278 rows × 3 columns
```

```
In [6]: type(movies)
```

```
Out[6]: pandas.core.frame.DataFrame
```

```
In [7]: movies.shape
```

```
Out[7]: (27278, 3)
```

```
In [8]: movies.head(20)
```

Out[8]:

	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 [9]: tags = pd.read_csv(r"D:\DATA SCIENCE full stack\My task\tag.csv")
```

```
In [10]: tags.head()
```

Out[10]:

	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"D:\DATA SCIENCE full stack\My task\rating.csv" ,  parse
```

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

Out[12]:

	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 [13]:
```
row_0= tags.iloc[0]
type(row_0)
```

Out[13]:
```
pandas.core.series.Series
```

In [14]:
```
row_0
```

Out[14]:
```
userId                        18
movieId                     4141
tag                  Mark Waters
timestamp    2009-04-24 18:19:40
Name: 0, dtype: object
```

In [15]:
```
row_1=tags.iloc[1]
row_1
```

Out[15]:
```
userId                        65
movieId                      208
tag                    dark hero
timestamp    2013-05-10 01:41:18
Name: 1, dtype: object
```

In [16]:
```
row_0.index
```

Out[16]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [17]:
```
row_0['userId']
```

Out[17]:
```
18
```

In [18]:
```
row_1['userId']
```

Out[18]:
```
65
```

In [19]:
```
'rating' in row_0
```

Out[19]:
```
False
```

```
In [20]: 'raing' in row_1
```

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

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

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

```
In [22]: row_0 = row_0.rename('firstRow')
         row_0.name
```

```
Out[22]: 'firstRow'
```

DataFrames

```
In [23]: tags.head()
```

Out[23]:

	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 [24]: tags.index
```

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

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

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

```
In [26]: tags.iloc[[0,11,500]]
```

Out[26]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
11	65	1783	noir thriller	2013-05-10 01:39:43
500	342	55908	entirely dialogue	2012-01-31 18:41:16

In [27]:
```
tags.iloc[[0,6,600]]
```
Out[27]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
6	65	898	screwball comedy	2013-05-10 01:42:40
600	348	608	black comedy	2011-02-09 22:21:05

In [28]:
```
tags.shape
```
Out[28]:
```
(465564, 4)
```

Descriptive Statistics

In [29]:
```
ratings['rating'].describe()
```
Out[29]:
```
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 [30]:
```
ratings.describe()
```
Out[30]:

	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 [31]:
```
ratings['rating'].mean()
```
Out[31]:
```
3.5255285642993797
```

In [32]:
```
ratings.mean()
```

```
C:\Users\NEHA\AppData\Local\Temp\ipykernel_10956\2439446979.py:1: FutureWarni
ng: DataFrame.mean and DataFrame.median with numeric_only=None will include d
atetime64 and datetime64tz columns in a future version.
  ratings.mean()
```

Out[32]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [33]:
```
ratings['rating'].min()
```

Out[33]:
```
0.5
```

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

Out[34]:
```
5.0
```

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

Out[35]:
```
1.0519889192942424
```

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

Out[36]:
```
0    4.0
Name: rating, dtype: float64
```

In [37]:
```
ratings.corr()
```

```
C:\Users\NEHA\AppData\Local\Temp\ipykernel_10956\1007000214.py:1: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  ratings.corr()
```

Out[37]:

	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 [38]:
```
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[38]:
```
False
```

In [39]:
```
filter2=ratings['rating']>0
filter2.all()
```

Out[39]:
```
True
```

Data Cleaning: Handling Missing Data

In [40]:
```
movies.shape
```

Out[40]:
```
(27278, 3)
```

In [41]:
```
movies.isnull().sum()
```

Out[41]:
```
movieId    0
title      0
genres     0
dtype: int64
```

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

Out[42]:
```
movieId    False
title      False
genres     False
dtype: bool
```

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

Out[43]:
```
False
```

In [44]:
```
ratings.shape
```

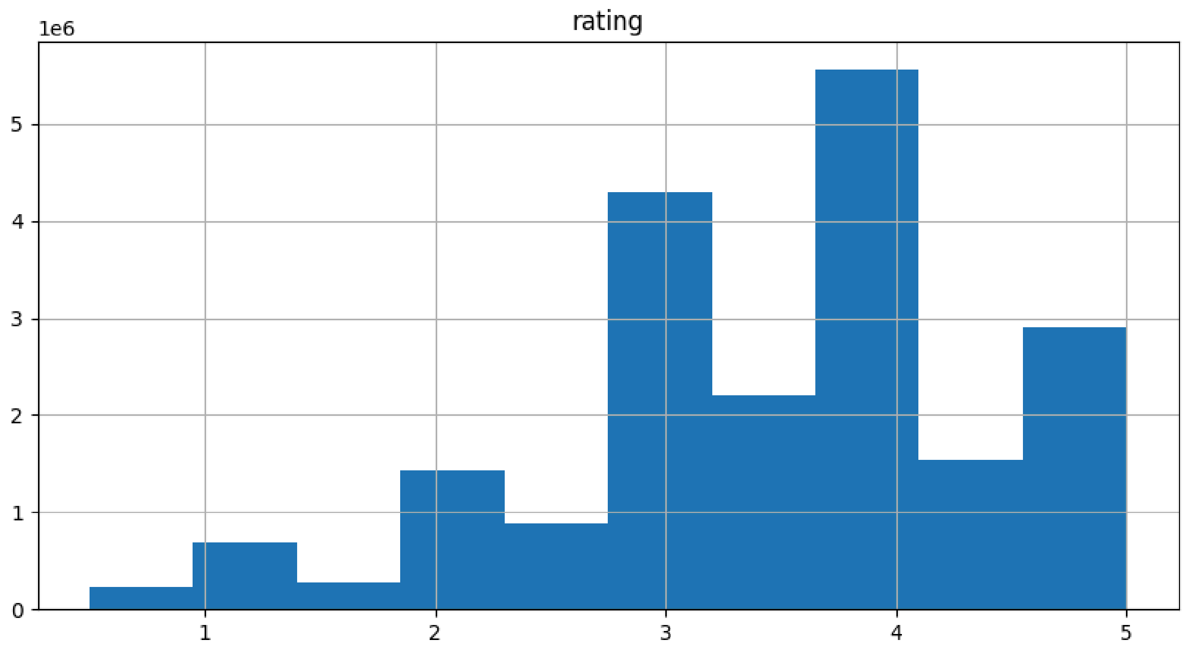
Out[44]:
```
(20000263, 4)
```

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

Out[45]: False

In [46]: tags.shape

Out[46]: (465564, 4)

In [47]: tags.isnull().any().any()

Out[47]: True

In [48]: tags.isnull().sum()

Out[48]: userId        0
         movieId       0
         tag          16
         timestamp     0
         dtype: int64

In [49]: tags=tags.dropna()

In [50]: tags.isnull().any().any()

Out[50]: False

In [51]: tags.isnull().sum()

Out[51]: userId       0
         movieId      0
         tag          0
         timestamp    0
         dtype: int64

In [52]: tags.shape

Out[52]: (465548, 4)
```

Data Visualization

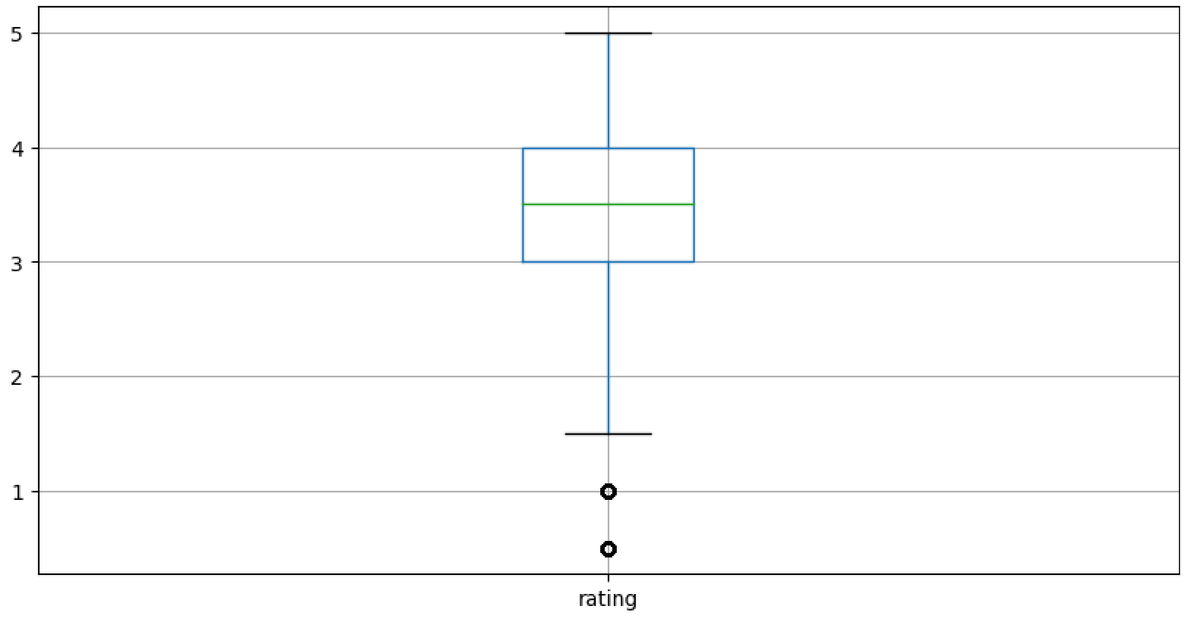
In [53]:
```
%matplotlib inline
import matplotlib.pyplot as plt
ratings.hist(column='rating', figsize=(10,5))
```

Out[53]:
```
array([[<AxesSubplot: title={'center': 'rating'}>]], dtype=object)
```



In [54]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[54]:
```
<AxesSubplot: >
```



slicing

In [55]:
```
tags['tag'].head()
```

Out[55]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [56]:
```
movies[['title','genres']].head()
```

Out[56]:

	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 [57]:
```
ratings[-10:]
```

Out[57]:

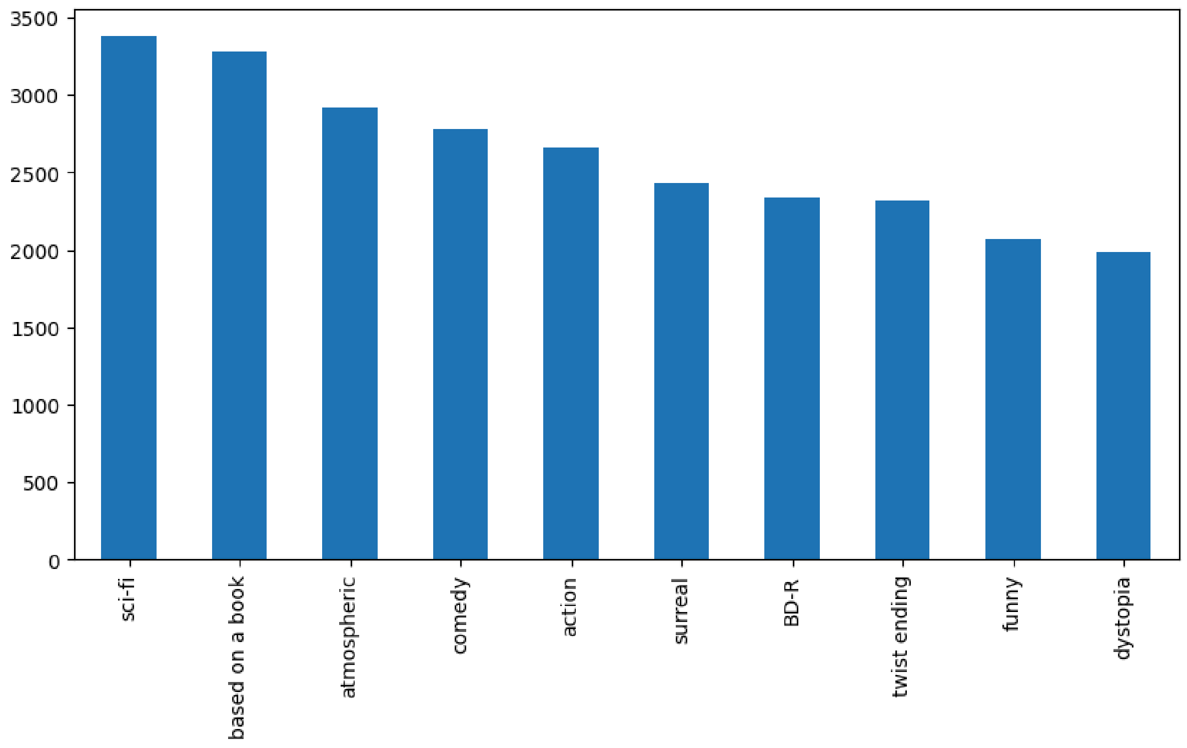
	userId	movieId	rating	timestamp
20000253	138493	60816	4.5	2009-12-03 18:32:43
20000254	138493	61160	4.0	2009-11-16 16:55:37
20000255	138493	65682	4.5	2009-10-17 21:52:53
20000256	138493	66762	4.5	2009-10-17 18:50:08
20000257	138493	68319	4.5	2009-12-07 18:15:20
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

In [58]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[58]:
```
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 [59]: tag_counts[:10].plot(kind='bar', figsize=(10,5))
```

```
Out[59]: <AxesSubplot: >
```



Filters for Selecting Rows

In [60]:
```python
is_highly_rated=ratings['rating'] >=5.0
ratings[is_highly_rated][30:50]
```

Out[60]:

	userId	movieId	rating	timestamp
239	3	50	5.0	1999-12-11 13:13:38
242	3	175	5.0	1999-12-11 13:32:13
244	3	223	5.0	1999-12-11 13:20:44
245	3	260	5.0	1999-12-11 13:09:02
246	3	316	5.0	1999-12-14 12:51:10
247	3	318	5.0	1999-12-11 13:09:26
248	3	329	5.0	1999-12-14 12:53:41
252	3	457	5.0	1999-12-11 13:16:55
253	3	480	5.0	1999-12-14 12:50:20
254	3	490	5.0	1999-12-11 13:30:41
256	3	541	5.0	1999-12-11 13:14:07
258	3	593	5.0	1999-12-11 13:16:55
263	3	858	5.0	1999-12-11 13:01:07
264	3	904	5.0	1999-12-11 13:04:10
267	3	924	5.0	1999-12-11 13:10:12
268	3	953	5.0	1999-12-11 13:11:52
271	3	1060	5.0	1999-12-11 13:18:30
272	3	1073	5.0	1999-12-11 13:20:44
275	3	1084	5.0	1999-12-11 13:15:03
276	3	1089	5.0	1999-12-11 13:05:58

In [61]:
```python
is_action=movies['genres'].str.contains('Action')
movies[is_action][5:15]
```

Out[61]:

	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 [62]:
```
movies[is_action].head(15)
```

Out[62]:

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

Out[64]:

	movieId
rating	
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 [68]:
```
average_rating = ratings[['movieId','rating']].groupby('movieId').mean()
average_rating.head()
```

Out[68]:
```
          rating
movieId
      1   3.921240
      2   3.211977
      3   3.151040
      4   2.861393
      5   3.064592
```

In [69]:
```
movie_count=ratings[['movieId','rating']].groupby('movieId').count()
movie_count.head()
```

Out[69]:
```
        rating
movieId
      1   49695
      2   22243
      3   12735
      4    2756
      5   12161
```

In [70]:
```
movie_count=ratings[['movieId','rating']].groupby('movieId').count()
movie_count.tail()
```

Out[70]:
```
        rating
movieId
 131254       1
 131256       1
 131258       1
 131260       1
 131262       1
```

Merge DataFrames

```
In [71]: tags.head()
```

Out[71]:

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

Out[72]:

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

Out[73]:

	movieId	title	genres	userId	tag	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1644	Watched	2014-12-04 23:44:40
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	computer animation	2007-07-08 13:59:15
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Disney animated feature	2007-07-08 22:21:47
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Pixar animation	2007-07-08 22:46:10
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	TÃ©a Leoni does not star in this movie	2009-06-15 19:19:33

In [76]:
```
avg_ratings= ratings.groupby('movieId', as_index=False).mean()
del avg_ratings['userId']
avg_ratings.head()
```

```
C:\Users\NEHA\AppData\Local\Temp\ipykernel_10956\1123815892.py:1: FutureWarni
ng: The default value of numeric_only in DataFrameGroupBy.mean is deprecated.
In a future version, numeric_only will default to False. Either specify numer
ic_only or select only columns which should be valid for the function.
  avg_ratings= ratings.groupby('movieId', as_index=False).mean()
```

Out[76]:

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

In [77]:
```
box_office = movies.merge(avg_ratings, on='movieId', how='inner')
box_office.tail()
```

Out[77]:

	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 [78]:
```
is_highly_rated = box_office['rating'] >= 4.0
box_office[is_highly_rated][-5:]
```

Out[78]:

	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 [79]:
```
is_Adventure = box_office['genres'].str.contains('Adventure')
box_office[is_Adventure][:5]
```

Out[79]:

	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 [80]:
```
box_office[is_Adventure & is_highly_rated][-5:]
```

Out[80]:

	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 [81]:
```
movies.head()
```

Out[81]:

	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 [82]:
```
movie_genres = movies['genres'].str.split('|', expand=True)
```

In [83]:
```
movie_genres[:10]
```

Out[83]:

	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

In [85]:
```
movie_genres['isComedy']=movies['genres'].str.contains('Comedy')
movie_genres[:10]
```

Out[85]:

	0	1	2	3	4	5	6	7	8	9	isComedy
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None	True
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None	False
2	Comedy	Romance	None	None	None	None	None	None	None	None	True
3	Comedy	Drama	Romance	None	None	None	None	None	None	None	True
4	Comedy	None	None	None	None	None	None	None	None	None	True
5	Action	Crime	Thriller	None	None	None	None	None	None	None	False
6	Comedy	Romance	None	None	None	None	None	None	None	None	True
7	Adventure	Children	None	None	None	None	None	None	None	None	False
8	Action	None	None	None	None	None	None	None	None	None	False
9	Action	Adventure	Thriller	None	None	None	None	None	None	None	False

***Extract year from title eg. 2007

In [86]:
```
movies['year']=movies['title'].str.extract('.*\((.*)\).*', expand=True)
```

In [87]:
```
movies.tail()
```
Out[87]:

	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 Timestamp

In [91]:
```
# Timestamps are common in sensor data or other time series datasets.
# Let us revisit the tags.csv dataset and read the timestamps!

tags = pd.read_csv(r"D:\DATA SCIENCE full stack\My task\tag.csv")
```

In [93]:
```
tags.dtypes
```
Out[93]:
```
userId         int64
movieId        int64
tag           object
timestamp     object
dtype: object
```

In [94]:
```
tags.head(5)
```
Out[94]:

	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 [ ]:
```
# tags['parsed_time'] = pd.to_datetime(tags['timestamp'], unit='s')
```

In [ ]:
```
# tags['parsed_time'].dtype
```

```
In [99]: tags.head(2)
```

Out[99]:

	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

**Selecting rows based on timestamps

In [100]:
```python
greater_than_t = tags['parsed_time'] > '2015-02-01'

selected_rows = tags[greater_than_t]

tags.shape, selected_rows.shape
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\indexes\base.py:3803, in Index.get_loc(self, key, method, tolerance)
   3802 try:
-> 3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\_libs
\index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\_libs
\index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

KeyError: 'parsed_time'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In [100], line 1
----> 1 greater_than_t = tags['parsed_time'] > '2015-02-01'
      3 selected_rows = tags[greater_than_t]
      5 tags.shape, selected_rows.shape

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\frame.py:3804, in DataFrame.__getitem__(self, key)
   3802 if self.columns.nlevels > 1:
   3803     return self._getitem_multilevel(key)
-> 3804 indexer = self.columns.get_loc(key)
   3805 if is_integer(indexer):
   3806     indexer = [indexer]

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\indexes\base.py:3805, in Index.get_loc(self, key, method, tolerance)
   3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:
-> 3805     raise KeyError(key) from err
   3806 except TypeError:
   3807     # If we have a listlike key, _check_indexing_error will raise
   3808     #  InvalidIndexError. Otherwise we fall through and re-raise
   3809     #  the TypeError.
   3810     self._check_indexing_error(key)

KeyError: 'parsed_time'
```

```
In [101]: tags.sort_values(by='parsed_time', ascending=True)[:10]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In [101], line 1
----> 1 tags.sort_values(by='parsed_time', ascending=True)[:10]

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\util
\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<loc
als>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\frame.py:6901, in DataFrame.sort_values(self, by, axis, ascending, inplace,
kind, na_position, ignore_index, key)
   6897 elif len(by):
   6898     # len(by) == 1
   6900     by = by[0]
-> 6901     k = self._get_label_or_level_values(by, axis=axis)
   6903     # need to rewrap column in Series to apply key function
   6904     if key is not None:
   6905         # error: Incompatible types in assignment (expression has typ
e
   6906         # "Series", variable has type "ndarray")

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\generic.py:1850, in NDFrame._get_label_or_level_values(self, key, axis)
   1844     values = (
   1845         self.axes[axis]
   1846         .get_level_values(key)  # type: ignore[assignment]
   1847         ._values
   1848     )
   1849 else:
-> 1850     raise KeyError(key)
   1852 # Check for duplicates
   1853 if values.ndim > 1:

KeyError: 'parsed_time'
```

Average Movie rating over Time

Movie ratings related to the year of launch?

```
In [102]: average_rating = ratings[['movieId','rating']].groupby('movieId', as_index=Fals
          average_rating.tail()
```

Out[102]:

	movieId	rating
26739	131254	4.0
26740	131256	4.0
26741	131258	2.5
26742	131260	3.0
26743	131262	4.0

```
In [103]: joined = movies.merge(average_rating, on='movieId', how='inner')
          joined.head()
          joined.corr()
```

```
C:\Users\NEHA\AppData\Local\Temp\ipykernel_10956\2957516148.py:3: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  joined.corr()
```

Out[103]:

	movieId	rating
movieId	1.000000	-0.090369
rating	-0.090369	1.000000

```
In [ ]:
```