```
import pandas as pd

rating = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie ratings\
rating.csv")
rating.head()
```

```
   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
```

```
movies = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie ratings\
movie.csv")
movies.head()
```

```
   movieId                                title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                        Comedy|Drama|Romance
4                                      Comedy
```

```
tags = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie ratings\
tag.csv")
tags.head()
```

```
   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
```

```
movies = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie ratings\
movie.csv")
movies.shape
```

```
(27278, 3)
```

```
ratings = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie
ratings\rating.csv")
ratings.shape
```

```
(20000263, 4)
```

```
movies = pd.read_csv(r"C:\Users\Admin\Downloads\archive movie ratings\
movie.csv")
print(type(movies))
movies.head(20)
```

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

	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)	
19	20	Money Train (1995)	

```
18                                          Comedy
19           Action|Comedy|Crime|Drama|Thriller

##For current analysis,we will remove timestamp

del ratings['timestamp']
del tags['timestamp']

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call
last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805,
in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'timestamp'

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

KeyError                                  Traceback (most recent call
last)
Cell In[34], line 1
----> 1 del ratings['timestamp']
      2 del tags['timestamp']

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4506, in
NDFrame.__delitem__(self, key)
   4501             deleted = True
   4502 if not deleted:
   4503     # If the above loop ran and didn't delete anything because
   4504     # there was no match, this call should raise the
appropriate
   4505     # exception:
-> 4506     loc = self.axes[-1].get_loc(key)
   4507     self._mgr = self._mgr.idelete(loc)
   4509 # delete from the caches

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812,
```

```
in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will
raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-
raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'timestamp'
```

```
#Data Structures
import pandas as pd
Series=pd.Series

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

```
pandas.core.series.Series
```

```
print(row_0)
```

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

```
row_0['userId']
```

```
18
```

```
'rating' in row_0
```

```
False
```

```
row_0.name
```

```
0
```

```
row_0=row_0.rename('firstRow')
row_0.name
```

```
'firstRow'
```

DataFrames

```
tags.head()
```

```
   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
```

```
tags.index
```

```
RangeIndex(start=0, stop=465564, step=1)
```

```
tags.columns
```

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

```
tags.iloc
```

```
<pandas.core.indexing._iLocIndexer at 0x18b8052a530>
```

```
tags.iloc[[0,11,500]]
```

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

Descriptive Statistics

```
ratings['rating'].describe()
```

```
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
```

```
ratings.describe()
```

```
             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
```

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

```
3.5255285642993797
```

```
ratings.mean()
```

```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

```
ratings['rating'].min()
```

```
0.5
```

```
ratings['rating'].max()
```

```
5.0
```

```
ratings['rating'].std()
```

```
1.051988919275684
```

```
ratings['rating'].mode()
```

```
0    4.0
Name: rating, dtype: float64
```

```
ratings.corr()
```

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

```
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
```

```
False
```

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

```
True
```

Data Cleaning: Handling Missing Data

```
movies.shape
```

```
(27278, 3)
```

```
movies.isnull().any().any()
```

```
False
```

```
ratings.shape
```

```
(20000263, 3)
```

```
ratings.isnull().any().any()
```

```
False
```

```
tags.shape
```

```
(465564, 3)
```

```
tags.isnull().any().any()
```

```
True
```

```
tags=tags.dropna()
```

```
tags.isnull().any().any()
```

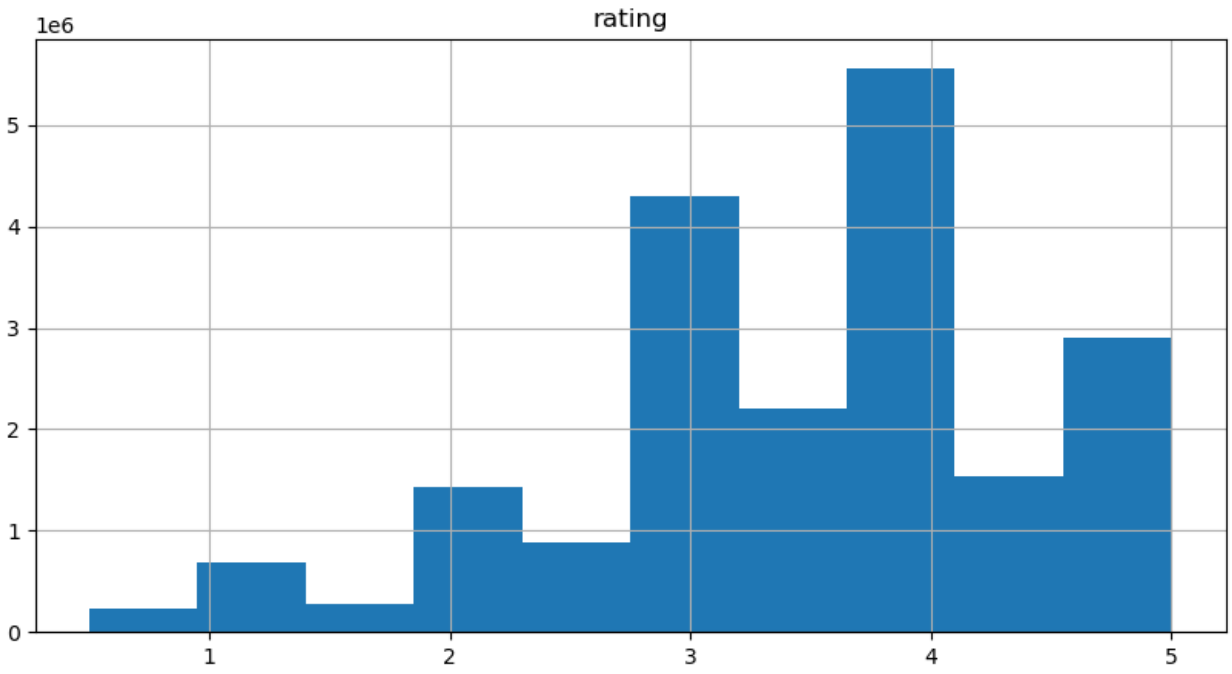
```
False
```

```
tags.shape
```

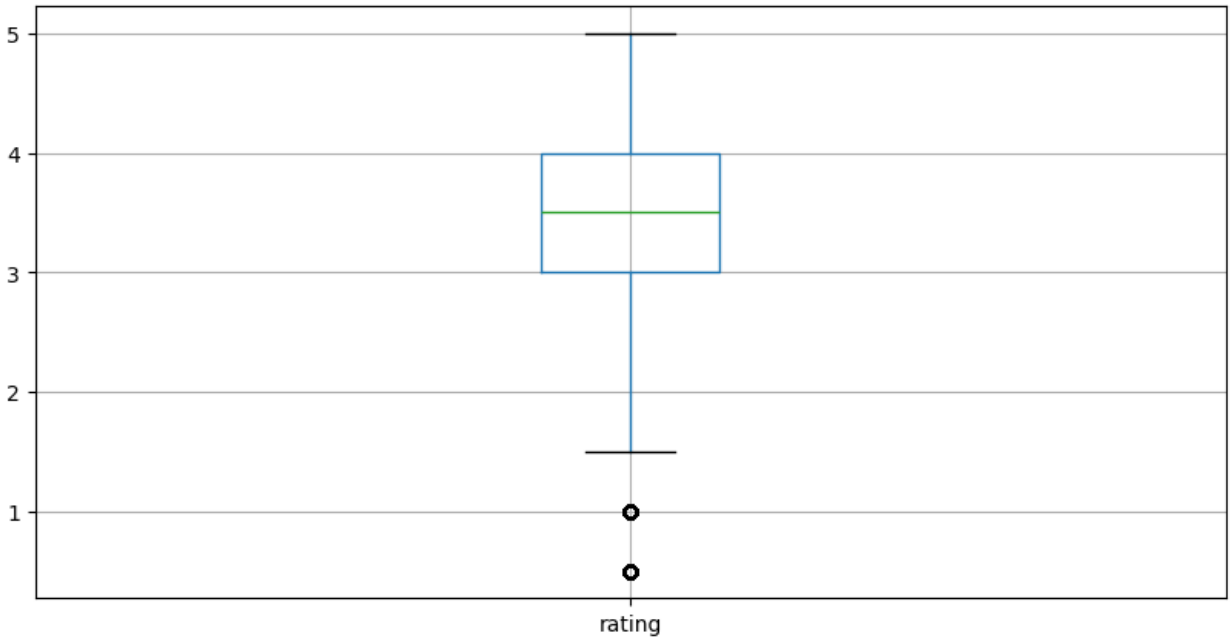
```
(465548, 3)
```

Data Visualization

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




```
plt.show(ratings.boxplot(column='rating',figsize=(10,5)))
```



Slicing out columns

```
tags['tag'].head()
```

```
0      Mark Waters
1        dark hero
```

```
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

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

```
                                title  \
0                    Toy Story (1995)
1                      Jumanji (1995)
2             Grumpier Old Men (1995)
3            Waiting to Exhale (1995)
4  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                              Comedy|Romance
3                        Comedy|Drama|Romance
4                                      Comedy
```

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

```
          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
```

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

```
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
```

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