Pandas_DataManipulation

June 5, 2025

Pandas Data Manipulations by Isha Borgaonkar

[141]:
```
import pandas as pd
# Load data from a CSV file into a pandas DataFrame
titanic_train = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/
 ↪datasets/master/titanic.csv")
```

[142]:
```
#Display the first 5 rows
titanic_train.head()
```

[142]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

[143]:
```
#Printing type of titanic_train data type
type(titanic_train)
```

[143]:
```
pandas.core.frame.DataFrame
```

[144]:
```
#Printing data types
titanic_train.columns
```

[144]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

[145]:
```
# Check the type of 'titanic_train.dtypes' - this returns a Series object
type(titanic_train.dtypes)
```

[145]:
```
pandas.core.series.Series
```

[146]:
```
# Store the Series of column data types into a variable named 'dy'
dy = titanic_train.dtypes
```

[147]:
```
# Access the data type of the 'Pclass' column from the 'dy' Series
dy['Pclass']
```

[147]:
```
dtype('int64')
```

[148]:
```
#Display the entire Series of column data types
dy
```

[148]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

[149]:
```
# Create a list with integers and strings
l = [0,1,2,3,4,5,"Isha" , "Borgaonkar" ,"UCD"]
```

[150]:
```
#Access the 7th element (index 6) in the list - returns "Isha"
l[6]
```

[150]:
```
'Isha'
```

[151]:
```
#Select every 2nd element (step = 2) from the first 100 entries of the 'dy'␣
 ↪Series (which contains column data types)
dy[0:100:2]
```

[151]:
```
PassengerId     int64
Pclass          int64
```

```
Sex            object
SibSp           int64
Ticket         object
Cabin          object
dtype: object
```

[152]:
```
#Check the type of the 'Age' column - should return <class 'pandas.core.series.
 ↪Series'>
type(titanic_train['Age'])
```

[152]:
```
pandas.core.series.Series
```

[153]:
```
#Get all rows from the 'Age' column except the last one
titanic_train['Age'][:-1]
```

[153]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       …
885    39.0
886    27.0
887    19.0
888     NaN
889    26.0
Name: Age, Length: 890, dtype: float64
```

[154]:
```
#Convert every 2nd value in the 'Age' column (step = 2) into a list
list(titanic_train['Age'][::2])
```

[154]:
```
[22.0,
 26.0,
 35.0,
 54.0,
 27.0,
 4.0,
 20.0,
 14.0,
 2.0,
 31.0,
 35.0,
 15.0,
 8.0,
 nan,
 nan,
 40.0,
```

```
nan,
28.0,
nan,
18.0,
40.0,
nan,
19.0,
nan,
nan,
7.0,
49.0,
65.0,
21.0,
5.0,
22.0,
45.0,
nan,
29.0,
17.0,
32.0,
21.0,
32.0,
nan,
0.83,
22.0,
nan,
17.0,
16.0,
23.0,
29.0,
46.0,
59.0,
71.0,
34.0,
28.0,
21.0,
37.0,
21.0,
38.0,
47.0,
22.0,
17.0,
70.5,
24.0,
21.0,
32.5,
54.0,
```

nan,
nan,
33.0,
47.0,
25.0,
19.0,
16.0,
nan,
24.0,
18.0,
27.0,
36.5,
51.0,
55.5,
nan,
16.0,
nan,
44.0,
26.0,
1.0,
nan,
nan,
61.0,
1.0,
56.0,
nan,
30.0,
nan,
9.0,
4.0,
nan,
40.0,
32.0,
19.0,
44.0,
nan,
nan,
28.0,
34.0,
18.0,
32.0,
16.0,
24.0,
22.0,
nan,
27.0,
32.0,

16.0,
51.0,
38.0,
19.0,
18.0,
35.0,
59.0,
24.0,
44.0,
19.0,
nan,
29.0,
30.0,
25.0,
37.0,
nan,
62.0,
41.0,
nan,
35.0,
nan,
52.0,
nan,
16.0,
58.0,
nan,
41.0,
nan,
45.0,
7.0,
65.0,
16.0,
nan,
30.0,
42.0,
26.0,
36.0,
24.0,
23.5,
nan,
nan,
19.0,
nan,
nan,
30.0,
24.0,
26.0,

43.0,
24.0,
31.0,
22.0,
30.0,
nan,
61.0,
31.0,
nan,
38.0,
nan,
29.0,
45.0,
2.0,
28.0,
36.0,
40.0,
3.0,
23.0,
15.0,
nan,
22.0,
nan,
40.0,
45.0,
nan,
60.0,
nan,
25.0,
19.0,
3.0,
22.0,
20.0,
42.0,
32.0,
nan,
1.0,
nan,
36.0,
28.0,
24.0,
31.0,
23.0,
39.0,
21.0,
20.0,
51.0,

```
21.0,
nan,
33.0,
44.0,
34.0,
30.0,
nan,
29.0,
18.0,
28.0,
nan,
28.0,
42.0,
50.0,
21.0,
64.0,
45.0,
25.0,
nan,
13.0,
5.0,
36.0,
30.0,
nan,
65.0,
50.0,
48.0,
47.0,
nan,
nan,
nan,
nan,
33.0,
22.0,
34.0,
22.0,
9.0,
50.0,
25.0,
35.0,
30.0,
nan,
55.0,
21.0,
54.0,
25.0,
17.0,
```

```
nan,
16.0,
33.0,
28.0,
29.0,
36.0,
24.0,
34.0,
36.0,
30.0,
nan,
nan,
50.0,
39.0,
2.0,
17.0,
30.0,
45.0,
nan,
36.0,
11.0,
50.0,
19.0,
33.0,
17.0,
nan,
22.0,
48.0,
39.0,
nan,
28.0,
nan,
19.0,
nan,
62.0,
36.0,
16.0,
34.0,
nan,
25.0,
54.0,
nan,
47.0,
22.0,
35.0,
47.0,
37.0,
```

```
nan,
nan,
24.0,
nan,
35.0,
30.0,
22.0,
39.0,
nan,
35.0,
34.0,
4.0,
27.0,
20.0,
21.0,
57.0,
26.0,
80.0,
32.0,
9.0,
32.0,
41.0,
20.0,
2.0,
0.75,
19.0,
nan,
nan,
21.0,
18.0,
nan,
23.0,
50.0,
47.0,
20.0,
25.0,
43.0,
40.0,
70.0,
nan,
24.5,
43.0,
nan,
20.0,
60.0,
14.0,
18.0,
```

```
31.0,
nan,
60.0,
44.0,
49.0,
18.0,
18.0,
26.0,
45.0,
22.0,
24.0,
48.0,
52.0,
38.0,
nan,
6.0,
34.0,
27.0,
30.0,
25.0,
29.0,
nan,
23.0,
48.0,
nan,
nan,
21.0,
31.0,
16.0,
19.0,
4.0,
33.0,
48.0,
28.0,
34.0,
nan,
20.0,
16.0,
nan,
nan,
24.0,
57.0,
54.0,
nan,
nan,
13.0,
29.0,
```

25.0,
18.0,
1.0,
nan,
nan,
25.0,
49.0,
30.0,
34.0,
11.0,
27.0,
39.0,
39.0,
26.0,
35.0,
30.5,
23.0,
43.0,
52.0,
38.0,
2.0,
nan,
nan,
15.0,
nan,
18.0,
21.0,
32.0,
20.0,
30.0,
17.0,
nan,
28.0,
4.0,
9.0,
44.0,
45.0,
24.0,
41.0,
48.0,
24.0,
27.0,
nan,
26.0,
33.0,
28.0,
20.0,

```
 nan,
 25.0,
 22.0,
 25.0,
 27.0,
 nan,
 32.0]
```

[155]:
```
# Display the data type of each column in the 'titanic_train' DataFrame
titanic_train.dtypes
```

[155]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

[156]:
```
#Summary stats for numerical columns
titanic_train.describe()
```

[156]:

	PassengerId	Survived	Pclass	Age	SibSp	\
count	891.000000	891.000000	891.000000	714.000000	891.000000	
mean	446.000000	0.383838	2.308642	29.699118	0.523008	
std	257.353842	0.486592	0.836071	14.526497	1.102743	
min	1.000000	0.000000	1.000000	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	20.125000	0.000000	
50%	446.000000	0.000000	3.000000	28.000000	0.000000	
75%	668.500000	1.000000	3.000000	38.000000	1.000000	
max	891.000000	1.000000	3.000000	80.000000	8.000000	

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

[157]:
```
#Create a new DataFrame 'titanic_train_cat' containing only the columns of type␣
↪'object' (typically strings or categorical data)
titanic_train_cat = titanic_train[titanic_train.dtypes[titanic_train.dtypes ==␣
↪'object'].index]
```

[158]:
```
# Check the type of the filtered result - i.e., columns in 'titanic_train' that␣
↪have data type 'object'
type(titanic_train.dtypes[titanic_train.dtypes == 'object'])
```

[158]:
```
pandas.core.series.Series
```

[159]:
```
#Check the data type of the filtered object-type columns from the DataFrame
titanic_train.dtypes[titanic_train.dtypes == 'object']
```

[159]:
```
Name        object
Sex         object
Ticket      object
Cabin       object
Embarked    object
dtype: object
```

[160]:
```
#Summary stats for numerical columns
titanic_train_cat.describe()
```

[160]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

[161]:
```
# Returns the type of the object created by selecting the 'Name' and 'Fare'␣
↪columns from the DataFrame.
# This will output: <class 'pandas.core.frame.DataFrame'> because multiple␣
↪columns are selected, resulting in a new DataFrame.
type(titanic_train[['Name' ,'Fare' ]])
```

[161]:
```
pandas.core.frame.DataFrame
```

[162]:
```
import numpy as np
```

[163]:
```
# Check for missing values
titanic_train_age_null = np.where(titanic_train['Age'].isnull() == True)
```

[164]:
```
# Check for missing values
titanic_train[titanic_train['Age'].isnull() == True]
```

[164]:
```
     PassengerId  Survived  Pclass                                      Name  \
5              6         0       3                          Moran, Mr. James
17            18         1       2                 Williams, Mr. Charles Eugene
19            20         1       3                      Masselmani, Mrs. Fatima
26            27         0       3                      Emir, Mr. Farred Chehab
28            29         1       3                 O'Dwyer, Miss. Ellen "Nellie"
..           …       …     …                                           …
859          860         0       3                             Razi, Mr. Raihed
863          864         0       3            Sage, Miss. Dorothy Edith "Dolly"
868          869         0       3                   van Melkebeke, Mr. Philemon
878          879         0       3                           Laleff, Mr. Kristo
888          889         0       3      Johnston, Miss. Catherine Helen "Carrie"

        Sex  Age  SibSp  Parch      Ticket     Fare Cabin Embarked
5      male  NaN      0      0      330877   8.4583   NaN        Q
17     male  NaN      0      0      244373  13.0000   NaN        S
19   female  NaN      0      0        2649   7.2250   NaN        C
26     male  NaN      0      0        2631   7.2250   NaN        C
28   female  NaN      0      0      330959   7.8792   NaN        Q
..      …  …    …    …          …       …   …      …
859    male  NaN      0      0        2629   7.2292   NaN        C
863  female  NaN      8      2    CA. 2343  69.5500   NaN        S
868    male  NaN      0      0      345777   9.5000   NaN        S
878    male  NaN      0      0      349217   7.8958   NaN        S
888  female  NaN      1      2  W./C. 6607  23.4500   NaN        S

[177 rows x 12 columns]
```

[166]:
```
titanic_train_age_null
len(titanic_train_age_null)
```

[166]:
```
1
```

[167]:
```
titanic_train[['Name' , 'Pclass']]
```

[167]:
```
                                                  Name  Pclass
0                              Braund, Mr. Owen Harris       3
1    Cumings, Mrs. John Bradley (Florence Briggs Th…       1
2                               Heikkinen, Miss. Laina       3
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)       1
4                             Allen, Mr. William Henry       3
..                                                 …     …
886                             Montvila, Rev. Juozas       2
887                      Graham, Miss. Margaret Edith       1
888          Johnston, Miss. Catherine Helen "Carrie"       3
889                             Behr, Mr. Karl Howell       1
890                               Dooley, Mr. Patrick       3
```

```
[891 rows x 2 columns]
```

[168]:
```
# Access rows/columns by integer location
titanic_train.iloc[titanic_train_age_null]
```

[168]:
```
     PassengerId  Survived  Pclass                                      Name  \
5              6         0       3                          Moran, Mr. James
17            18         1       2              Williams, Mr. Charles Eugene
19            20         1       3                   Masselmani, Mrs. Fatima
26            27         0       3                   Emir, Mr. Farred Chehab
28            29         1       3             O'Dwyer, Miss. Ellen "Nellie"
..           ...       ...     ...                                       ...
859          860         0       3                          Razi, Mr. Raihed
863          864         0       3         Sage, Miss. Dorothy Edith "Dolly"
868          869         0       3               van Melkebeke, Mr. Philemon
878          879         0       3                          Laleff, Mr. Kristo
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"

        Sex  Age  SibSp  Parch      Ticket     Fare Cabin Embarked
5      male  NaN      0      0      330877   8.4583   NaN        Q
17     male  NaN      0      0      244373  13.0000   NaN        S
19   female  NaN      0      0        2649   7.2250   NaN        C
26     male  NaN      0      0        2631   7.2250   NaN        C
28   female  NaN      0      0      330959   7.8792   NaN        Q
..      ...  ...    ...    ...         ...      ...   ...      ...
859    male  NaN      0      0        2629   7.2292   NaN        C
863  female  NaN      8      2    CA. 2343  69.5500   NaN        S
868    male  NaN      0      0      345777   9.5000   NaN        S
878    male  NaN      0      0      349217   7.8958   NaN        S
888  female  NaN      1      2  W./C. 6607  23.4500   NaN        S

[177 rows x 12 columns]
```

[169]:
```
# Access rows/columns by integer location
type(titanic_train.iloc[5])
```

[169]:
```
pandas.core.series.Series
```

[170]:
```
# Access rows/columns by integer location
titanic_train.iloc[5]
```

[170]:
```
PassengerId                  6
Survived                     0
Pclass                       3
Name           Moran, Mr. James
Sex                        male
```

```
Age                          NaN
SibSp                          0
Parch                          0
Ticket                    330877
Fare                      8.4583
Cabin                        NaN
Embarked                       Q
Name: 5, dtype: object
```

[171]:
```
# Access rows/columns by integer location
titanic_train.iloc[[1,2,3,6,7]]
```

[171]:
```
   PassengerId  Survived  Pclass  \
1            2         1       1
2            3         1       3
3            4         1       1
6            7         0       1
7            8         0       3

                                                 Name     Sex   Age  SibSp  \
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
6                            McCarthy, Mr. Timothy J    male  54.0      0
7                     Palsson, Master. Gosta Leonard    male   2.0      3

   Parch            Ticket     Fare Cabin Embarked
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
```

[172]:
```
# Access rows/columns by integer location
titanic_train.iloc[np.where(titanic_train['Fare'] ==␣
 ↪max(titanic_train['Fare']))]['Name']
```

[172]:
```
258                      Ward, Miss. Anna
679    Cardeza, Mr. Thomas Drake Martinez
737                Lesurer, Mr. Gustave J
Name: Name, dtype: object
```

[173]:
```
# Returns the name(s) of passenger(s) who paid the highest fare.
titanic_train[titanic_train['Fare']==titanic_train['Fare'].max()]['Name']
```

[173]:
```
258                      Ward, Miss. Anna
679    Cardeza, Mr. Thomas Drake Martinez
```

```
737                Lesurer, Mr. Gustave J
Name: Name, dtype: object
```

[174]:
```
# Returns the name(s) of passenger(s) who are 24 years old.
titanic_train[titanic_train['Age']==24.0]['Name']
```

[174]:
```
89                               Celotti, Mr. Francesco
118                            Baxter, Mr. Quigg Edmond
127                           Madsen, Mr. Fridtjof Arne
139                                  Giglio, Mr. Victor
142    Hakkarainen, Mrs. Pekka Pietari (Elin Matilda …
199               Yrois, Miss. Henriette ("Mrs Harbeck")
210                                      Ali, Mr. Ahmed
234                    Leyson, Mr. Robert William Norman
247                      Hamalainen, Mrs. William (Anna)
293                                Haas, Miss. Aloisia
294                                   Mineff, Mr. Ivan
310                       Hays, Miss. Margaret Bechstein
316                  Kantor, Mrs. Sinai (Miriam Sternin)
341                       Fortune, Miss. Alice Elizabeth
345                        Brown, Miss. Amelia "Mildred"
369                        Aubart, Mme. Leontine Pauline
394    Sandstrom, Mrs. Hjalmar (Agnes Charlotta Bengt…
437                Richards, Mrs. Sidney (Emily Hocking)
499                                  Svensson, Mr. Olof
514                                   Coleff, Mr. Satio
565                                 Davies, Mr. Alfred J
600    Jacobsohn, Mrs. Sidney Samuel (Amy Frances Chr…
615                                  Herman, Miss. Alice
641                                 Sagesser, Mlle. Emma
655                              Hickman, Mr. Leonard Mark
710     Mayne, Mlle. Berthe Antonine ("Mrs de Villiers")
743                                   McNamee, Mr. Neal
770                               Lievens, Mr. Rene Aime
858                Baclini, Mrs. Solomon (Latifa Qurban)
864                                  Gill, Mr. John William
Name: Name, dtype: object
```

[175]:
```
# Returns the name(s) of passenger(s) with the minimum age.
titanic_train[titanic_train['Age'] == titanic_train['Age'].min()]['Name']
```

[175]:
```
803    Thomas, Master. Assad Alexander
Name: Name, dtype: object
```

[176]:
```
# Check for missing values
titanic_train['Age'][titanic_train['Cabin'].isnull()]
```

[176]:
```
0      22.0
2      26.0
4      35.0
5       NaN
7       2.0
       …
884    25.0
885    39.0
886    27.0
888     NaN
890    32.0
Name: Age, Length: 687, dtype: float64
```

[177]:
```
#Check for missing values
titanic_train[['Age','Cabin']][titanic_train.Cabin.isnull()==True]
```

[177]:
```
      Age Cabin
0    22.0   NaN
2    26.0   NaN
4    35.0   NaN
5     NaN   NaN
7     2.0   NaN
..    …    …
884  25.0   NaN
885  39.0   NaN
886  27.0   NaN
888   NaN   NaN
890  32.0   NaN

[687 rows x 2 columns]
```

[178]:
```
titanic_train[titanic_train['Cabin'].isna()]
```

[178]:
```
     PassengerId  Survived  Pclass                                               Name  \
0              1         0       3                            Braund, Mr. Owen Harris
2              3         1       3                             Heikkinen, Miss. Laina
4              5         0       3                       Allen, Mr. William Henry
5              6         0       3                               Moran, Mr. James
7              8         0       3                   Palsson, Master. Gosta Leonard
..           …       …     …                                                  …
884          885         0       3                           Sutehall, Mr. Henry Jr
885          886         0       3             Rice, Mrs. William (Margaret Norton)
886          887         0       2                            Montvila, Rev. Juozas
888          889         0       3          Johnston, Miss. Catherine Helen "Carrie"
890          891         0       3                               Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
```

```
0      male  22.0      1      0         A/5 21171   7.2500   NaN        S
2    female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
4      male  35.0      0      0            373450   8.0500   NaN        S
5      male   NaN      0      0            330877   8.4583   NaN        Q
7      male   2.0      3      1            349909  21.0750   NaN        S
..      …    …    …    …              …       …    …      …
884    male  25.0      0      0   SOTON/OQ 392076   7.0500   NaN        S
885  female  39.0      0      5            382652  29.1250   NaN        Q
886    male  27.0      0      0            211536  13.0000   NaN        S
888  female   NaN      1      2        W./C. 6607  23.4500   NaN        S
890    male  32.0      0      0            370376   7.7500   NaN        Q

[687 rows x 12 columns]
```

[179]:
```
# Returns rows where the 'Cabin' value is missing (NaN).
titanic_train[titanic_train['Cabin'] == 'NaN']['Age']
```

[179]:
```
Series([], Name: Age, dtype: float64)
```

[180]:
```
# Check for missing values
titanic_train[titanic_train["Cabin"].isnull()]["Age"]
```

[180]:
```
0      22.0
2      26.0
4      35.0
5       NaN
7       2.0
       …
884    25.0
885    39.0
886    27.0
888     NaN
890    32.0
Name: Age, Length: 687, dtype: float64
```

[181]:
```
titanic_train['Cabin'] == 'NaN'
```

[181]:
```
0      False
1      False
2      False
3      False
4      False
       …
886    False
887    False
888    False
889    False
```

```
890    False
Name: Cabin, Length: 891, dtype: bool
```

[182]:
```
#Check for missing values
titanic_train.iloc[np.where(titanic_train['Cabin'].isnull() ==
 ↪True)][['Age','Cabin']]
```

[182]:
```
      Age Cabin
0    22.0   NaN
2    26.0   NaN
4    35.0   NaN
5     NaN   NaN
7     2.0   NaN
..    …   …
884  25.0   NaN
885  39.0   NaN
886  27.0   NaN
888   NaN   NaN
890  32.0   NaN

[687 rows x 2 columns]
```

[183]:
```
# Displays the first 20 rows of the titanic_train DataFrame.
titanic_train.head(20)
```

[183]:
```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3
```

```
                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
5                                    Moran, Mr. James    male   NaN      0
6                             McCarthy, Mr. Timothy J    male  54.0      0
7                      Palsson, Master. Gosta Leonard    male   2.0      3
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                 Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
10                    Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                           Bonnell, Miss. Elizabeth  female  58.0      0
12                     Saundercock, Mr. William Henry    male  20.0      0
13                        Andersson, Mr. Anders Johan    male  39.0      1
14                Vestrom, Miss. Hulda Amanda Adolfina  female  14.0      0
15                    Hewlett, Mrs. (Mary D Kingcome)   female  55.0      0
16                                Rice, Master. Eugene    male   2.0      4
17                        Williams, Mr. Charles Eugene    male   NaN      0
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…  female  31.0      1
19                             Masselmani, Mrs. Fatima  female   NaN      0

    Parch            Ticket     Fare Cabin Embarked
0       0         A/5 21171   7.2500   NaN        S
1       0          PC 17599  71.2833   C85        C
2       0  STON/O2. 3101282   7.9250   NaN        S
3       0            113803  53.1000  C123        S
4       0            373450   8.0500   NaN        S
5       0            330877   8.4583   NaN        Q
6       0             17463  51.8625   E46        S
7       1            349909  21.0750   NaN        S
8       2            347742  11.1333   NaN        S
9       0            237736  30.0708   NaN        C
10      1          PP 9549  16.7000    G6        S
11      0            113783  26.5500  C103        S
12      0         A/5. 2151   8.0500   NaN        S
13      5            347082  31.2750   NaN        S
14      0            350406   7.8542   NaN        S
15      0            248706  16.0000   NaN        S
16      1            382652  29.1250   NaN        Q
17      0            244373  13.0000   NaN        S
18      0            345763  18.0000   NaN        S
19      0              2649   7.2250   NaN        C
```

[184]:
```
# Returns the names of male passengers who survived.
titanic_train.Name[(titanic_train.Survived == 1) & (titanic_train.Sex ==␣
 ↪"male")]
```

[184]:
```
17        Williams, Mr. Charles Eugene
21               Beesley, Mr. Lawrence
23        Sloper, Mr. William Thompson
36                     Mamee, Mr. Hanna
55                   Woolner, Mr. Hugh
                    …
838                     Chip, Mr. Chang
839               Marechal, Mr. Pierre
857              Daly, Mr. Peter Denis
869    Johnson, Master. Harold Theodor
889               Behr, Mr. Karl Howell
Name: Name, Length: 109, dtype: object
```

[185]:
```
titanic_train.Name[(titanic_train.Survived == 0) & (titanic_train.Sex ==␣
↪"male")]
```

[185]:
```
0              Braund, Mr. Owen Harris
4            Allen, Mr. William Henry
5                    Moran, Mr. James
6             McCarthy, Mr. Timothy J
7      Palsson, Master. Gosta Leonard
                    …
881                Markun, Mr. Johann
883     Banfield, Mr. Frederick James
884            Sutehall, Mr. Henry Jr
886              Montvila, Rev. Juozas
890                Dooley, Mr. Patrick
Name: Name, Length: 468, dtype: object
```

[186]:
```
# Returns the 'Name' column from the titanic_train DataFrame as a Series.
titanic_train['Name']
```

[186]:
```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th…
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                              …
886                               Montvila, Rev. Juozas
887                        Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                               Behr, Mr. Karl Howell
890                                 Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

[187]:
```
# Same as titanic_train['Name']; returns the 'Name' column as a Series.
titanic_train.Name
```

[187]:
```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th…
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                              …
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

[188]:
```
# Returns the names of male passengers who survived.
titanic_train[(titanic_train['Sex']== 'male') & (titanic_train['Survived']==␣
↪1)]['Name']
```

[188]:
```
17        Williams, Mr. Charles Eugene
21               Beesley, Mr. Lawrence
23        Sloper, Mr. William Thompson
36                   Mamee, Mr. Hanna
55                  Woolner, Mr. Hugh
                    …
838                    Chip, Mr. Chang
839              Marechal, Mr. Pierre
857             Daly, Mr. Peter Denis
869    Johnson, Master. Harold Theodor
889              Behr, Mr. Karl Howell
Name: Name, Length: 109, dtype: object
```

[189]:
```
# Creates a new column 'new_col' in the DataFrame and assigns the value "sudh"␣
↪to all rows.
titanic_train['new_col'] = "sudh"
```

[190]:
```
# Creates a list 'l' containing the elements 1 to 5.
l = [1,2,3,4,5]
```

[191]:
```
# Converts the list 'l' into a pandas Series.
pd.Series(l)
```

[191]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

[194]:
```
# Creates a dictionary 'd' with keys: 'key', 'title', and 'mail_id' and their␣
 ↪corresponding values.
d = {'key' : "Isha" , "title" :"Borgaonkar" , "mail_id" : "isb@gmail.com"}
```

[195]:
```
# Creates a DataFrame from dictionary 'd' with specified row index ['key',␣
 ↪'title', 'mail_id'].
# Each key-value pair becomes a column, and the values are aligned to the given␣
 ↪index.
pd.DataFrame(d,index=['key' , 'title' , 'mail_id'])
```

[195]:

	key	title	mail_id
key	Isha	Borgaonkar	isb@gmail.com
title	Isha	Borgaonkar	isb@gmail.com
mail_id	Isha	Borgaonkar	isb@gmail.com

[196]:
```
# Converts the dictionary 'd' into a pandas Series.
# Dictionary keys become the index, and values become the data in the Series.
pd.Series(d)
```

[196]:
```
key                 Isha
title          Borgaonkar
mail_id    isb@gmail.com
dtype: object
```

[197]:
```
# Creates a 3x4 NumPy array 'arr' filled with random numbers from a standard␣
 ↪normal distribution (mean=0, std=1).
arr  = np.random.randn(3,4)
```

[198]:
```
# Converts the NumPy array 'arr' into a pandas DataFrame.
# Each row and column from the array becomes a row and column in the DataFrame.
pd.DataFrame(arr)
```

[198]:

	0	1	2	3
0	-0.201841	1.045371	0.538162	0.812119
1	0.241106	-0.952510	-0.136267	1.267248
2	0.173634	-1.223255	1.415320	0.457711

[199]:
```
#creating pandas series
ser = pd.Series(np.array([4,5,6,7,8]),index=['a','b','Isha',"Borgaonkar",'c'])
```

[200]:
```
#Printing series
ser
```

[200]:
```
a             4
b             5
Isha          6
Borgaonkar    7
```

```
c             8
dtype: int32
```

[201]:
```
# Accessing value using label 'Isha'
ser['Isha']
```

[201]: 6

[202]:
```
#Access rows/columns by integer location
ser.iloc[0]
```

[202]: 4

[203]:
```
# Access rows/columns by label
ser.loc['Borgaonkar']
```

[203]: 7

[204]:
```
# Creates a pandas Series 'ser1' with values [4, 5, 6, 7, 8] and mixed-type␣
 ↪index [45, 51, 62, "Isha", 83].
ser1 = pd.Series(np.array([4,5,6,7,8]),index=[45,51,62,"Isha",83])
```

[205]:
```
# Displays the Series 'ser1' with its values and mixed-type index.
ser1
```

[205]:
```
45      4
51      5
62      6
Isha    7
83      8
dtype: int32
```

[206]:
```
# Access rows/columns by label
ser1.loc[51]
```

[206]: 5

[207]:
```
# Creates a 4x5 DataFrame 'df' with random integers from 3 to 5 (inclusive of␣
 ↪3, exclusive of 6),
# using custom row labels ['a', 'b', 'c', 'd'] and column labels ['x', 'y',␣
 ↪'z', 'u', 'v'].
df = pd.DataFrame(np.random.
 ↪randint(3,6,(4,5)),index=['a','b','c','d'],columns=['x','y','z','u','v'])
```

[208]:
```
#printing DataFrame
df
```

[208]:
```
   x  y  z  u  v
a  3  5  5  4  4
b  5  3  5  3  4
c  5  3  3  3  4
d  4  3  5  4  4
```

[209]:
```
#  Access rows/columns by label
df.loc[['c','d'],['z','u']]
```

[209]:
```
   z  u
c  3  3
d  5  4
```

[210]:
```
#Access rows/columns by integer location
df.iloc[[2,3],[2,3]]
```

[210]:
```
   z  u
c  3  3
d  5  4
```

[211]:
```
# Access rows/columns by integer location
df.iloc[2:,2:4]
```

[211]:
```
   z  u
c  3  3
d  5  4
```

[212]:
```
# Creates a new column 'new' in the DataFrame by adding values from columns 'x'␣
 ↪and 'y'.
df['new'] = df['x'] + df.y
```

[213]:
```
#Drop specified row/column from DataFrame
df.drop('x',axis=1,inplace=True)
```

[214]:
```
# Drop specified row/column from DataFrame
df = df.drop('a')
```

[215]:
```
# Sets the random seed to 23 for reproducibility of results.
# Creates a 4x5 DataFrame 'df' with random values from the standard normal␣
 ↪distribution,
# using custom row labels ['a', 'b', 'c', 'd'] and column labels ['x', 'y',␣
 ↪'z', 'u', 'v'].

np.random.seed(23)
df = pd.DataFrame(np.random.
 ↪randn(4,5),index=['a','b','c','d'],columns=['x','y','z','u','v'])
```

[216]:
```
# Returns a DataFrame where all values less than 0 are kept,
# and all other values are replaced with NaN (not a number).
df[df<0]
```

[216]:

	x	y	z	u	v
a	NaN	NaN	-0.777619	NaN	NaN
b	-1.051082	-0.367548	-1.137460	-1.322148	NaN
c	-0.347459	NaN	NaN	NaN	-1.043450
d	-1.009942	NaN	NaN	-1.838068	-0.938769

[217]:
```
# Returns a 1D array of all values from the DataFrame that are less than 0.
df.values[df.values<0]
```

[217]:
```
array([-0.77761941, -1.05108156, -0.36754812, -1.13745969, -1.32214752,
       -0.34745899, -1.04345   , -1.00994188, -1.83806777, -0.93876863])
```

[219]:
```
#Access rows/columns by integer location
df.values[df.iloc[0:5]<0]
```

[219]:
```
array([-0.77761941, -1.05108156, -0.36754812, -1.13745969, -1.32214752,
       -0.34745899, -1.04345   , -1.00994188, -1.83806777, -0.93876863])
```

[220]:
```
# Iterates over all DataFrame cells and prints the column and row labels
# where the cell value is less than 0 (i.e., negative).
c = df.values[df.values < 0]

for i in df.columns:
    for j in df.index:
        if df.loc[j, i] in c:
            print(i, j)
```

```
x b
x c
x d
y b
z a
z b
u b
u d
v c
v d
```

[221]:
```
# Recommended: directly check for negative values
for c in df.columns:
    for v in df.index:
        if df.loc[v, c] < 0:
            print(c, v, df.loc[v, c])
```

```
x b -1.0510815639071178
x c -0.34745899102186334
x d -1.0099418765878465
y b -0.3675481161171661
z a -0.7776194131918178
z b -1.1374596907250272
u b -1.3221475225908594
u d -1.8380677677579502
v c -1.0434500017467254
v d -0.9387686311201282
```

[222]:
```
#Access rows/columns by label
if df.loc[j,i]<0:
#Access rows/columns by label
 print(i,j,df.loc[j,i])
```

```
v d -0.9387686311201282
```

[223]:
```
for k in df.values[df.values<0]:
     print(k,"----",(i,j))
```

```
-0.7776194131918178 ---- ('v', 'd')
-1.0510815639071178 ---- ('v', 'd')
-0.3675481161171661 ---- ('v', 'd')
-1.1374596907250272 ---- ('v', 'd')
-1.3221475225908594 ---- ('v', 'd')
-0.34745899102186334 ---- ('v', 'd')
-1.0434500017467254 ---- ('v', 'd')
-1.0099418765878465 ---- ('v', 'd')
-1.8380677677579502 ---- ('v', 'd')
-0.9387686311201282 ---- ('v', 'd')
```

[224]:
```
# Iterates through the DataFrame to find and print positions of negative values
k = df < 0

for i in k.columns:
    for j in k.index:
        if k[i][j] == True:
            print(i, j, df.loc[j, i])
```

```
x b -1.0510815639071178
x c -0.34745899102186334
x d -1.0099418765878465
y b -0.3675481161171661
z a -0.7776194131918178
z b -1.1374596907250272
u b -1.3221475225908594
u d -1.8380677677579502
v c -1.0434500017467254
```

```
v d -0.9387686311201282
```

[225]:
```
# Creates a DataFrame 'df' from dictionary 'd' with three keys: 'key1', 'key2',␣
 ↪and 'key3'.
# Some values are set as np.nan to represent missing data.
d = {'key1':[1,2,3,4,np.nan],"key2":["Isha" ,"B" , np.nan,5,6] ,'key3' :[np.
 ↪nan,np.nan,np.nan,3,8]}
df = pd.DataFrame(d)
```

[226]:
```
# Drop rows with missing values
df.dropna()
```

[226]:
```
   key1 key2  key3
3   4.0    5   3.0
```

[227]:
```
#Drop rows with missing values
df.dropna(axis=1)
```

[227]:
```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
```

[228]:
```
#Drop rows with missing values
df.dropna(thresh=0)
```

[228]:
```
   key1  key2  key3
0   1.0  Isha   NaN
1   2.0     B   NaN
2   3.0   NaN   NaN
3   4.0     5   3.0
4   NaN     6   8.0
```

[229]:
```
# Drop rows with missing values
df.dropna(thresh=1)
```

[229]:
```
   key1  key2  key3
0   1.0  Isha   NaN
1   2.0     B   NaN
2   3.0   NaN   NaN
3   4.0     5   3.0
4   NaN     6   8.0
```

[230]:
```
# Drop rows with missing values
df.dropna(thresh=2)
```

[230]:
```
   key1  key2  key3
0   1.0  Isha   NaN
1   2.0     B   NaN
```

```
3   4.0     5   3.0
4   NaN     6   8.0
```

[231]:
```
# Drop rows with missing values
df.dropna(thresh=3)
```

[231]:
```
   key1 key2  key3
3   4.0    5   3.0
```

[232]:
```
# Drop rows with missing values
df.dropna(thresh=4)
```

[232]:
```
Empty DataFrame
Columns: [key1, key2, key3]
Index: []
```

[233]:
```
# Drop rows with missing values
df.dropna(axis=1 , thresh=3)
```

[233]:
```
   key1  key2
0   1.0  Isha
1   2.0     B
2   3.0   NaN
3   4.0     5
4   NaN     6
```

[234]:
```
#Replace missing values with specified value
df.fillna(567)
```

[234]:
```
    key1  key2   key3
0    1.0  Isha  567.0
1    2.0     B  567.0
2    3.0   567  567.0
3    4.0     5    3.0
4  567.0     6    8.0
```

[235]:
```
#Replace missing values with specified value
df.fillna(value=df['key1'].mean())
```

[235]:
```
   key1  key2  key3
0   1.0  Isha   2.5
1   2.0     B   2.5
2   3.0   2.5   2.5
3   4.0     5   3.0
4   2.5     6   8.0
```

[236]:
```

```

```
d1 = {'name' :['Isha' , 'Shubham' , 'Ritul', 'Avadhut' ,'Manisha'],"mail_id" :
 ↪["isha@gmail.com", "shubham@gmail.com",'ritul@gmail.com','avadhut@gmail.
 ↪com','manisha@gmail.com'],
     'salary' :[100,200,500,200,500]}
```

[237]:
```
df = pd.DataFrame(d1)
```

[ ]:
```

```

[239]:
```
#Group data by specified column(s)
df.groupby('mail_id').sum()
```

[239]:
```
                      name  salary
mail_id
avadhut@gmail.com  Avadhut     200
isha@gmail.com        Isha     100
manisha@gmail.com  Manisha     500
ritul@gmail.com      Ritul     500
shubham@gmail.com  Shubham     200
```

[240]:
```
#Summary stats for numerical columns
type(df.groupby('mail_id').describe())
```

[240]:
```
pandas.core.frame.DataFrame
```

[241]:
```
# Summary stats for numerical columns
df.groupby('mail_id').describe()
```

[241]:
```
                  salary
                   count   mean std    min    25%    50%    75%    max
mail_id
avadhut@gmail.com    1.0  200.0 NaN  200.0  200.0  200.0  200.0  200.0
isha@gmail.com       1.0  100.0 NaN  100.0  100.0  100.0  100.0  100.0
manisha@gmail.com    1.0  500.0 NaN  500.0  500.0  500.0  500.0  500.0
ritul@gmail.com      1.0  500.0 NaN  500.0  500.0  500.0  500.0  500.0
shubham@gmail.com    1.0  200.0 NaN  200.0  200.0  200.0  200.0  200.0
```

[243]:
```
# Summary stats for numerical columns
df.groupby('mail_id').describe().loc['isha@gmail.com']
```

[243]:
```
salary  count      1.0
        mean     100.0
        std        NaN
        min      100.0
        25%      100.0
        50%      100.0
        75%      100.0
        max      100.0
```

```
Name: isha@gmail.com, dtype: float64
```

[245]:
```
# Summary stats for numerical columns
df.groupby('mail_id').describe().loc['isha@gmail.com']['salary']['mean']
```

[245]:
```
100.0
```

[246]:
```
# Summary stats for numerical columns
df.groupby('mail_id').describe().T
```

[246]:
```
mail_id       avadhut@gmail.com  isha@gmail.com  manisha@gmail.com  \
salary count                1.0             1.0                1.0
       mean               200.0           100.0              500.0
       std                  NaN             NaN                NaN
       min                200.0           100.0              500.0
       25%                200.0           100.0              500.0
       50%                200.0           100.0              500.0
       75%                200.0           100.0              500.0
       max                200.0           100.0              500.0

mail_id       ritul@gmail.com  shubham@gmail.com
salary count              1.0                1.0
       mean             500.0              200.0
       std                NaN                NaN
       min              500.0              200.0
       25%              500.0              200.0
       50%              500.0              200.0
       75%              500.0              200.0
       max              500.0              200.0
```

[247]:
```
arr1 = np.random.randn(3,4)
```

[248]:
```
arr2 = np.random.randn(3,4)
```

[249]:
```
arr3 = np.random.randn(3,4)
```

[250]:
```
# Create DataFrame df1 with index labels
df1 = pd.DataFrame(arr1,columns=["A","B","C","D"],index = [0,1,2])
```

[251]:
```
# Create DataFrame df2 with index labels
df2 = pd.DataFrame(arr2,columns=["A","B","C","D"],index = [3,4,5])
```

[252]:
```
df2
```

[252]:
```
          A         B         C         D
3  0.728876  1.968435 -0.547788 -0.679418
4 -2.506230  0.146960  0.606195 -0.022539
5  0.013422  0.935945  0.420623  0.411620
```

[253]:
```
df3 = pd.DataFrame(arr3,columns=["A","B","C","D"],index = [6,7,8])
```

[254]:
```
df1
```

[254]:
```
          A         B         C         D
0 -0.201841  1.045371  0.538162  0.812119
1  0.241106 -0.952510 -0.136267  1.267248
2  0.173634 -1.223255  1.415320  0.457711
```

[255]:
```
df3
```

[255]:
```
          A         B         C         D
6 -0.071324 -0.045438  1.040886 -0.094035
7 -0.420844 -0.551989 -0.121098  0.190141
8  0.512137  0.131538 -0.331617 -1.632386
```

[256]:
```
df4 = pd.concat([df1,df2,df3])
```

[257]:
```
df5 = pd.concat([df1,df2,df3],axis=1)
```

[258]:
```
df4
```

[258]:
```
          A         B         C         D
0 -0.201841  1.045371  0.538162  0.812119
1  0.241106 -0.952510 -0.136267  1.267248
2  0.173634 -1.223255  1.415320  0.457711
3  0.728876  1.968435 -0.547788 -0.679418
4 -2.506230  0.146960  0.606195 -0.022539
5  0.013422  0.935945  0.420623  0.411620
6 -0.071324 -0.045438  1.040886 -0.094035
7 -0.420844 -0.551989 -0.121098  0.190141
8  0.512137  0.131538 -0.331617 -1.632386
```

[259]:
```
# Access rows/columns by integer location
df4.iloc[0]
```

[259]:
```
A   -0.201841
B    1.045371
C    0.538162
D    0.812119
Name: 0, dtype: float64
```

[260]:
```
# Access rows/columns by label
df4.loc[0]
```

[260]:
```
A   -0.201841
B    1.045371
C    0.538162
```

```
D    0.812119
Name: 0, dtype: float64
```

[261]:
```
df5
```

[261]:
```
          A         B         C         D         A         B         C  \
0 -0.201841  1.045371  0.538162  0.812119       NaN       NaN       NaN
1  0.241106 -0.952510 -0.136267  1.267248       NaN       NaN       NaN
2  0.173634 -1.223255  1.415320  0.457711       NaN       NaN       NaN
3       NaN       NaN       NaN       NaN  0.728876  1.968435 -0.547788
4       NaN       NaN       NaN       NaN -2.506230  0.146960  0.606195
5       NaN       NaN       NaN       NaN  0.013422  0.935945  0.420623
6       NaN       NaN       NaN       NaN       NaN       NaN       NaN
7       NaN       NaN       NaN       NaN       NaN       NaN       NaN
8       NaN       NaN       NaN       NaN       NaN       NaN       NaN

          D         A         B         C         D
0       NaN       NaN       NaN       NaN       NaN
1       NaN       NaN       NaN       NaN       NaN
2       NaN       NaN       NaN       NaN       NaN
3 -0.679418       NaN       NaN       NaN       NaN
4 -0.022539       NaN       NaN       NaN       NaN
5  0.411620       NaN       NaN       NaN       NaN
6       NaN -0.071324 -0.045438  1.040886 -0.094035
7       NaN -0.420844 -0.551989 -0.121098  0.190141
8       NaN  0.512137  0.131538 -0.331617 -1.632386
```

[262]:
```
df5['A']
```

[262]:
```
          A         A         A
0 -0.201841       NaN       NaN
1  0.241106       NaN       NaN
2  0.173634       NaN       NaN
3       NaN  0.728876       NaN
4       NaN -2.506230       NaN
5       NaN  0.013422       NaN
6       NaN       NaN -0.071324
7       NaN       NaN -0.420844
8       NaN       NaN  0.512137
```

[263]:
```
# Access rows/columns by integer location
df5.iloc[:,4]
```

[263]:
```
0         NaN
1         NaN
2         NaN
3    0.728876
```

```
4   -2.506230
5    0.013422
6         NaN
7         NaN
8         NaN
Name: A, dtype: float64
```

[264]:
```
#Replace missing values with specified value
pd.concat([df1,df2,df3],axis=1).fillna(0)
```

[264]:

	A	B	C	D	A	B	C	\
0	-0.201841	1.045371	0.538162	0.812119	0.000000	0.000000	0.000000	
1	0.241106	-0.952510	-0.136267	1.267248	0.000000	0.000000	0.000000	
2	0.173634	-1.223255	1.415320	0.457711	0.000000	0.000000	0.000000	
3	0.000000	0.000000	0.000000	0.000000	0.728876	1.968435	-0.547788	
4	0.000000	0.000000	0.000000	0.000000	-2.506230	0.146960	0.606195	
5	0.000000	0.000000	0.000000	0.000000	0.013422	0.935945	0.420623	
6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	

	D	A	B	C	D
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000
3	-0.679418	0.000000	0.000000	0.000000	0.000000
4	-0.022539	0.000000	0.000000	0.000000	0.000000
5	0.411620	0.000000	0.000000	0.000000	0.000000
6	0.000000	-0.071324	-0.045438	1.040886	-0.094035
7	0.000000	-0.420844	-0.551989	-0.121098	0.190141
8	0.000000	0.512137	0.131538	-0.331617	-1.632386

[265]:
```
df11 = pd.DataFrame(arr1,columns=["A","B","C","D"],index = [0,1,2])
df22 = pd.DataFrame(arr2,columns=["A","B","C","D"],index = [3,4,5])
```

[266]:
```
#Printing Df11
df11
```

[266]:

	A	B	C	D
0	-0.201841	1.045371	0.538162	0.812119
1	0.241106	-0.952510	-0.136267	1.267248
2	0.173634	-1.223255	1.415320	0.457711

[267]:
```
#Printing Df22
df22
```

[267]:
```
          A         B         C         D
3  0.728876  1.968435 -0.547788 -0.679418
4 -2.506230  0.146960  0.606195 -0.022539
5  0.013422  0.935945  0.420623  0.411620
```

[268]:
```
# Merge two DataFrames using column keys
pd.merge(df11,df22,on= "A")
```

[268]:
```
Empty DataFrame
Columns: [A, B_x, C_x, D_x, B_y, C_y, D_y]
Index: []
```

[269]:
```
#Create DataFrame df1 with index labels
df1 = pd.DataFrame({'key': ['K0', 'K8', 'K2', 'K3'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})
#Create DataFrame df2 with index labels
df2 = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                        'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']})
```

[270]:
```
# Merge two DataFrames using column keys
pd.merge(df1,df2,on= 'key',how = 'outer')
```

[270]:
```
  key    A    B    C    D
0  K0   A0   B0   C0   D0
1  K1  NaN  NaN   C1   D1
2  K2   A2   B2   C2   D2
3  K3   A3   B3   C3   D3
4  K8   A1   B1  NaN  NaN
```

[275]:
```
import pandas as pd

# Create DataFrame df1
df1 = pd.DataFrame({
    'key1': ['K0', 'K0', 'K1', 'K2'],
    'key2': ['K0', 'K1', 'K0', 'K1'],
    'A': ['A0', 'A1', 'A2', 'A3'],
    'B': ['B0', 'B1', 'B2', 'B3']
})

# Create DataFrame df2
df2 = pd.DataFrame({
    'key1': ['K0', 'K1', 'K1', 'K2'],
    'key2': ['K0', 'K0', 'K0', 'K0'],
    'C': ['C0', 'C1', 'C2', 'C3'],
    'D': ['D0', 'D1', 'D2', 'D3']
})
```

[276]:
```
#Merge two DataFrames using column keys
pd.merge(df1,df2,on = ['key1','key2'])
```

[276]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

[278]:
```
#Create DataFrame df1 with index labels
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                    'B': ['B0', 'B1', 'B2']},
                     index=['K0', 'K1', 'K2'])
#Create DataFrame df2 with index labels
df2 = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                    'D': ['D0', 'D2', 'D3']},
                     index=['K0', 'K2', 'K3'])
```

[279]:
```
#Join two DataFrames based on their index
df1.join(df2,how = 'outer')
```

[279]:

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3