Pandas modülü

```
import numpy as np
import pandas as pd

print("numpy version:", np.__version__)
print("pandas version:", pd.__version__)
```

```
numpy version: 1.24.3
pandas version: 1.4.4
```

pandas serileri

Tanımlama 1

```
labels = ["Hagi", "Popescu", "Tafarel", "Okan Buruk", "Bülent
Korkmaz"]
dx = [10, 20, 30, 40, 50]

pd.Series(data = dx)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
p1 = pd.Series(data = dx, index = labels)
p1
```

```
Hagi              10
Popescu           20
Tafarel           30
Okan Buruk        40
Bülent Korkmaz    50
dtype: int64
```

```
type(p1)
```

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

```
pd.Series(dx, labels)
```

```
Hagi              10
Popescu           20
```

```
Tafarel           30
Okan Buruk        40
Bülent Korkmaz    50
dtype: int64
```

Tanımlama 2

```
arr = np.array([100,200,300,400,500])

pd.Series(arr)
```

```
0    100
1    200
2    300
3    400
4    500
dtype: int32
```

```
pd.Series(data = arr)
```

```
0    100
1    200
2    300
3    400
4    500
dtype: int32
```

```
pd.Series(data = arr, index = labels)
```

```
Hagi              100
Popescu           200
Tafarel           300
Okan Buruk        400
Bülent Korkmaz    500
dtype: int32
```

```
pd.Series(arr, index = ["A","B","C","D","E"])
```

```
A    100
B    200
C    300
D    400
E    500
dtype: int32
```

Tanımlama 3

```
dataDict = {"Hagi":10,"Tafarel":80,"Aykut":11}
pd.Series(dataDict)
```

```
Hagi       10
Tafarel    80
Aykut      11
dtype: int64
```

pandas serileri üzerine işlemler

```
s2019 = pd.Series([5, 10, 14, 20], ["Buğday", "Mısır", "Kiraz",
"Erik"])
s2020 = pd.Series([2, 12, 12, 6], ["Buğday", "Mısır", "Çilek",
"Erik"])

s2019, s2020
```

```
(Buğday     5
 Mısır     10
 Kiraz     14
 Erik      20
 dtype: int64,
 Buğday     2
 Mısır     12
 Çilek     12
 Erik       6
 dtype: int64)
```

```
s2019.info()
```

```
<class 'pandas.core.series.Series'>
Index: 4 entries, Buğday to Erik
Series name: None
Non-Null Count  Dtype
--------------  -----
4 non-null      int64
dtypes: int64(1)
memory usage: 64.0+ bytes
```

```
s2019.describe()
```

```
count     4.000000
mean     12.250000
std       6.344289
min       5.000000
25%       8.750000
50%      12.000000
```

```
75%      15.500000
max      20.000000
dtype: float64
```

```
s2020.info()
```

```
<class 'pandas.core.series.Series'>
Index: 4 entries, Buğday to Erik
Series name: None
Non-Null Count  Dtype
--------------  -----
4 non-null      int64
dtypes: int64(1)
memory usage: 64.0+ bytes
```

```
s2020.describe()
```

```
count     4.000000
mean      8.000000
std       4.898979
min       2.000000
25%       5.000000
50%       9.000000
75%      12.000000
max      12.000000
dtype: float64
```

```
toplam = s2019 + s2020
toplam
```

```
Buğday     7.0
Erik      26.0
Kiraz      NaN
Mısır     22.0
Çilek      NaN
dtype: float64
```

```
toplam.info()
```

```
<class 'pandas.core.series.Series'>
Index: 5 entries, Buğday to Çilek
Series name: None
Non-Null Count  Dtype
--------------  -----
3 non-null      float64
dtypes: float64(1)
memory usage: 252.0+ bytes
```

```
toplam.describe()
```

```
count     3.000000
mean     18.333333
std      10.016653
min       7.000000
25%      14.500000
50%      22.000000
75%      24.000000
max      26.000000
dtype: float64
```

```
toplam.isna()
```

```
Buğday    False
Erik      False
Kiraz      True
Mısır     False
Çilek      True
dtype: bool
```

```
toplam.isna().sum()
```

```
2
```

pandas DataFrameleri

DataFrame tanımlama 1

```
df = pd.DataFrame()
df
```

```
Empty DataFrame
Columns: []
Index: []
```

```
df["x"] = np.array([i for i in range(1, 11, 2)])
df
```

```
   x
0  1
1  3
2  5
3  7
4  9
```

```
df["y"] = df.x.values ** 2
df
```

```
   x   y
0  1   1
1  3   9
2  5  25
3  7  49
4  9  81
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x       5 non-null      int32
 1   y       5 non-null      int32
dtypes: int32(2)
memory usage: 168.0 bytes
```

```
df.describe()
```

```
              x          y
count  5.000000   5.000000
mean   5.000000  33.000000
std    3.162278  32.496154
min    1.000000   1.000000
25%    3.000000   9.000000
50%    5.000000  25.000000
75%    7.000000  49.000000
max    9.000000  81.000000
```

```
df.x.values
```

```
array([1, 3, 5, 7, 9])
```

```
df.x
```

```
0    1
1    3
2    5
3    7
4    9
Name: x, dtype: int32
```

```
len(df.x)
```

```
5
```

DataFrame tanımlama 2

```
from numpy.random import randn

df = pd.DataFrame(randn(5, 3),
                  index = ["A", "B", "C", "D", "E"],
                  columns = ["s1", "s2", "s3"])
df
```

```
          s1        s2        s3
A -0.280314  1.252665 -0.645268
B  0.710631 -1.071232 -0.524408
C  0.353716  1.587944  0.892008
D  0.838767  0.413116  0.499758
E -0.556107  0.461905  0.500079
```

```
df["s1"]
```

```
A   -0.280314
B    0.710631
C    0.353716
D    0.838767
E   -0.556107
Name: s1, dtype: float64
```

```
df.s1
```

```
A   -0.280314
B    0.710631
C    0.353716
D    0.838767
E   -0.556107
Name: s1, dtype: float64
```

```
df.s1.values
```

```
array([-0.28031449,  0.71063058,  0.35371562,  0.83876668, -
0.5561071 ])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, A to E
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   s1      5 non-null      float64
 1   s2      5 non-null      float64
 2   s3      5 non-null      float64
dtypes: float64(3)
memory usage: 332.0+ bytes
```

```
df.describe()
```

```
             s1        s2        s3
count  5.000000  5.000000  5.000000
mean   0.213338  0.528879  0.144434
std    0.611127  1.027585  0.686036
min   -0.556107 -1.071232 -0.645268
25%   -0.280314  0.413116 -0.524408
50%    0.353716  0.461905  0.499758
75%    0.710631  1.252665  0.500079
max    0.838767  1.587944  0.892008
```

DataFrame Tanımlama 3

```
data1 = {"Hagi":10,"Tafarel":80,"Aykut":11}
df1 = pd.DataFrame([data1])
df1
```

```
   Hagi  Tafarel  Aykut
0    10       80     11
```

```
data = {"Sin": [np.sin(i*np.pi/180) for i in range(0, 195, 15)],
        "Cos": [np.cos(i*np.pi/180) for i in range(0, 195, 15)]}

print("data veri tipi :", type(data))
```

```
data veri tipi : <class 'dict'>
```

```
df1 = pd.DataFrame(data)
df1
```

```
             Sin           Cos
0   0.000000e+00  1.000000e+00
1   2.588190e-01  9.659258e-01
2   5.000000e-01  8.660254e-01
3   7.071068e-01  7.071068e-01
4   8.660254e-01  5.000000e-01
5   9.659258e-01  2.588190e-01
6   1.000000e+00  6.123234e-17
7   9.659258e-01 -2.588190e-01
8   8.660254e-01 -5.000000e-01
9   7.071068e-01 -7.071068e-01
10  5.000000e-01 -8.660254e-01
11  2.588190e-01 -9.659258e-01
12  1.224647e-16 -1.000000e+00
```

```
df1["n"] = [i for i in range(0, 195, 15)]
df1
```

```
              Sin           Cos    n
0   0.000000e+00  1.000000e+00    0
1   2.588190e-01  9.659258e-01   15
2   5.000000e-01  8.660254e-01   30
3   7.071068e-01  7.071068e-01   45
4   8.660254e-01  5.000000e-01   60
5   9.659258e-01  2.588190e-01   75
6   1.000000e+00  6.123234e-17   90
7   9.659258e-01 -2.588190e-01  105
8   8.660254e-01 -5.000000e-01  120
9   7.071068e-01 -7.071068e-01  135
10  5.000000e-01 -8.660254e-01  150
11  2.588190e-01 -9.659258e-01  165
12  1.224647e-16 -1.000000e+00  180
```

```
df1.set_index("n", inplace = True)
df1
```

```
               Sin           Cos
n
0    0.000000e+00  1.000000e+00
15   2.588190e-01  9.659258e-01
30   5.000000e-01  8.660254e-01
45   7.071068e-01  7.071068e-01
60   8.660254e-01  5.000000e-01
75   9.659258e-01  2.588190e-01
90   1.000000e+00  6.123234e-17
105  9.659258e-01 -2.588190e-01
120  8.660254e-01 -5.000000e-01
135  7.071068e-01 -7.071068e-01
150  5.000000e-01 -8.660254e-01
165  2.588190e-01 -9.659258e-01
180  1.224647e-16 -1.000000e+00
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13 entries, 0 to 180
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Sin     13 non-null     float64
 1   Cos     13 non-null     float64
dtypes: float64(2)
memory usage: 312.0 bytes
```

```
df1
```

```
               Sin           Cos
n
0    0.000000e+00  1.000000e+00
```

```
15   2.588190e-01  9.659258e-01
30   5.000000e-01  8.660254e-01
45   7.071068e-01  7.071068e-01
60   8.660254e-01  5.000000e-01
75   9.659258e-01  2.588190e-01
90   1.000000e+00  6.123234e-17
105  9.659258e-01 -2.588190e-01
120  8.660254e-01 -5.000000e-01
135  7.071068e-01 -7.071068e-01
150  5.000000e-01 -8.660254e-01
165  2.588190e-01 -9.659258e-01
180  1.224647e-16 -1.000000e+00
```

```
df1.Sin
```

```
n
0      0.000000e+00
15     2.588190e-01
30     5.000000e-01
45     7.071068e-01
60     8.660254e-01
75     9.659258e-01
90     1.000000e+00
105    9.659258e-01
120    8.660254e-01
135    7.071068e-01
150    5.000000e-01
165    2.588190e-01
180    1.224647e-16
Name: Sin, dtype: float64
```

```
df1["Sin_kare"] = df1.Sin.values ** 2
df1["Cos_kare"] = df1.Cos.values ** 2
df1
```

```
               Sin           Cos      Sin_kare      Cos_kare
n
0    0.000000e+00  1.000000e+00  0.000000e+00  1.000000e+00
15   2.588190e-01  9.659258e-01  6.698730e-02  9.330127e-01
30   5.000000e-01  8.660254e-01  2.500000e-01  7.500000e-01
45   7.071068e-01  7.071068e-01  5.000000e-01  5.000000e-01
60   8.660254e-01  5.000000e-01  7.500000e-01  2.500000e-01
75   9.659258e-01  2.588190e-01  9.330127e-01  6.698730e-02
90   1.000000e+00  6.123234e-17  1.000000e+00  3.749399e-33
105  9.659258e-01 -2.588190e-01  9.330127e-01  6.698730e-02
120  8.660254e-01 -5.000000e-01  7.500000e-01  2.500000e-01
135  7.071068e-01 -7.071068e-01  5.000000e-01  5.000000e-01
150  5.000000e-01 -8.660254e-01  2.500000e-01  7.500000e-01
165  2.588190e-01 -9.659258e-01  6.698730e-02  9.330127e-01
180  1.224647e-16 -1.000000e+00  1.499760e-32  1.000000e+00
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13 entries, 0 to 180
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Sin       13 non-null     float64
 1   Cos       13 non-null     float64
 2   Sin_kare  13 non-null     float64
 3   Cos_kare  13 non-null     float64
dtypes: float64(4)
memory usage: 520.0 bytes
```

```
df1.describe()
```

```
              Sin           Cos   Sin_kare      Cos_kare
count  13.000000  1.300000e+01  13.000000  1.300000e+01
mean    0.584289  3.416071e-17   0.461538  5.384615e-01
std     0.360773  7.637626e-01   0.379777  3.797773e-01
min     0.000000 -1.000000e+00   0.000000  3.749399e-33
25%     0.258819 -7.071068e-01   0.066987  2.500000e-01
50%     0.707107  6.123234e-17   0.500000  5.000000e-01
75%     0.866025  7.071068e-01   0.750000  9.330127e-01
max     1.000000  1.000000e+00   1.000000  1.000000e+00
```

```
df1.iloc[0]
```

```
Sin         0.0
Cos         1.0
Sin_kare    0.0
Cos_kare    1.0
Name: 0, dtype: float64
```

```
df1.loc[[0, 45, 90, 135, 180]] # index in ismine göre
```

```
              Sin           Cos      Sin_kare      Cos_kare
n                                                          
0    0.000000e+00  1.000000e+00  0.000000e+00  1.000000e+00
45   7.071068e-01  7.071068e-01  5.000000e-01  5.000000e-01
90   1.000000e+00  6.123234e-17  1.000000e+00  3.749399e-33
135  7.071068e-01 -7.071068e-01  5.000000e-01  5.000000e-01
180  1.224647e-16 -1.000000e+00  1.499760e-32  1.000000e+00
```

```
df1.iloc[[0, 2, 4]] #index in indis numarasına göre
```

```
         Sin       Cos  Sin_kare  Cos_kare
n                                         
0   0.000000  1.000000      0.00      1.00
```

```
30  0.500000  0.866025      0.25      0.75
60  0.866025  0.500000      0.75      0.25
```

```
df1.loc[[0, 45, 90, 135, 180],["Sin"]]
```

```
              Sin
n
0    0.000000e+00
45   7.071068e-01
90   1.000000e+00
135  7.071068e-01
180  1.224647e-16
```

```
df1.loc[[0, 45, 90, 135, 180],["Sin", "Sin_kare"]]
```

```
              Sin      Sin_kare
n
0    0.000000e+00  0.000000e+00
45   7.071068e-01  5.000000e-01
90   1.000000e+00  1.000000e+00
135  7.071068e-01  5.000000e-01
180  1.224647e-16  1.499760e-32
```

DataFramelerde filtreleme

```
df1 > 0
```

```
       Sin    Cos  Sin_kare  Cos_kare
n
0    False   True     False      True
15    True   True      True      True
30    True   True      True      True
45    True   True      True      True
60    True   True      True      True
75    True   True      True      True
90    True   True      True      True
105   True  False      True      True
120   True  False      True      True
135   True  False      True      True
150   True  False      True      True
165   True  False      True      True
180   True  False      True      True
```

```
df1 > 1
```

```
       Sin    Cos  Sin_kare  Cos_kare
n
0    False  False     False     False
15   False  False     False     False
```

```
30   False  False     False     False
45   False  False     False     False
60   False  False     False     False
75   False  False     False     False
90   False  False     False     False
105  False  False     False     False
120  False  False     False     False
135  False  False     False     False
150  False  False     False     False
165  False  False     False     False
180  False  False     False     False
```

```
(df1 > 0) & (df1 > 1)
```

```
       Sin    Cos  Sin_kare  Cos_kare
n
0    False  False     False     False
15   False  False     False     False
30   False  False     False     False
45   False  False     False     False
60   False  False     False     False
75   False  False     False     False
90   False  False     False     False
105  False  False     False     False
120  False  False     False     False
135  False  False     False     False
150  False  False     False     False
165  False  False     False     False
180  False  False     False     False
```

```
(df1 > 0) | (df1 > 1) #or operatörü altctrl + eksi
```

```
       Sin    Cos  Sin_kare  Cos_kare
n
0    False   True     False      True
15    True   True      True      True
30    True   True      True      True
45    True   True      True      True
60    True   True      True      True
75    True   True      True      True
90    True   True      True      True
105   True  False      True      True
120   True  False      True      True
135   True  False      True      True
150   True  False      True      True
165   True  False      True      True
180   True  False      True      True
```

```
df1.isna()
```

```
       Sin    Cos  Sin_kare  Cos_kare
n
0    False  False     False     False
15   False  False     False     False
30   False  False     False     False
45   False  False     False     False
60   False  False     False     False
75   False  False     False     False
90   False  False     False     False
105  False  False     False     False
120  False  False     False     False
135  False  False     False     False
150  False  False     False     False
165  False  False     False     False
180  False  False     False     False
```

```
df1.isna().sum()
```

```
Sin         0
Cos         0
Sin_kare    0
Cos_kare    0
dtype: int64
```

DataFrame'den sütun silme

```
df1.drop("Sin_kare", axis = 1) # kalıcı olmayan silme
```

```
              Sin           Cos      Cos_kare
n
0    0.000000e+00  1.000000e+00  1.000000e+00
15   2.588190e-01  9.659258e-01  9.330127e-01
30   5.000000e-01  8.660254e-01  7.500000e-01
45   7.071068e-01  7.071068e-01  5.000000e-01
60   8.660254e-01  5.000000e-01  2.500000e-01
75   9.659258e-01  2.588190e-01  6.698730e-02
90   1.000000e+00  6.123234e-17  3.749399e-33
105  9.659258e-01 -2.588190e-01  6.698730e-02
120  8.660254e-01 -5.000000e-01  2.500000e-01
135  7.071068e-01 -7.071068e-01  5.000000e-01
150  5.000000e-01 -8.660254e-01  7.500000e-01
165  2.588190e-01 -9.659258e-01  9.330127e-01
180  1.224647e-16 -1.000000e+00  1.000000e+00
```

```
df1
```

```
              Sin           Cos      Sin_kare      Cos_kare
n
0    0.000000e+00  1.000000e+00  0.000000e+00  1.000000e+00
```

```
15   2.588190e-01  9.659258e-01  6.698730e-02  9.330127e-01
30   5.000000e-01  8.660254e-01  2.500000e-01  7.500000e-01
45   7.071068e-01  7.071068e-01  5.000000e-01  5.000000e-01
60   8.660254e-01  5.000000e-01  7.500000e-01  2.500000e-01
75   9.659258e-01  2.588190e-01  9.330127e-01  6.698730e-02
90   1.000000e+00  6.123234e-17  1.000000e+00  3.749399e-33
105  9.659258e-01 -2.588190e-01  9.330127e-01  6.698730e-02
120  8.660254e-01 -5.000000e-01  7.500000e-01  2.500000e-01
135  7.071068e-01 -7.071068e-01  5.000000e-01  5.000000e-01
150  5.000000e-01 -8.660254e-01  2.500000e-01  7.500000e-01
165  2.588190e-01 -9.659258e-01  6.698730e-02  9.330127e-01
180  1.224647e-16 -1.000000e+00  1.499760e-32  1.000000e+00
```

```
df1.drop("Sin_kare", axis = 1, inplace = True) # kalıcı silme
df1
```

```
               Sin           Cos      Cos_kare
n
0    0.000000e+00  1.000000e+00  1.000000e+00
15   2.588190e-01  9.659258e-01  9.330127e-01
30   5.000000e-01  8.660254e-01  7.500000e-01
45   7.071068e-01  7.071068e-01  5.000000e-01
60   8.660254e-01  5.000000e-01  2.500000e-01
75   9.659258e-01  2.588190e-01  6.698730e-02
90   1.000000e+00  6.123234e-17  3.749399e-33
105  9.659258e-01 -2.588190e-01  6.698730e-02
120  8.660254e-01 -5.000000e-01  2.500000e-01
135  7.071068e-01 -7.071068e-01  5.000000e-01
150  5.000000e-01 -8.660254e-01  7.500000e-01
165  2.588190e-01 -9.659258e-01  9.330127e-01
180  1.224647e-16 -1.000000e+00  1.000000e+00
```

```
df1
```

```
               Sin           Cos      Cos_kare
n
0    0.000000e+00  1.000000e+00  1.000000e+00
15   2.588190e-01  9.659258e-01  9.330127e-01
30   5.000000e-01  8.660254e-01  7.500000e-01
45   7.071068e-01  7.071068e-01  5.000000e-01
60   8.660254e-01  5.000000e-01  2.500000e-01
75   9.659258e-01  2.588190e-01  6.698730e-02
90   1.000000e+00  6.123234e-17  3.749399e-33
105  9.659258e-01 -2.588190e-01  6.698730e-02
120  8.660254e-01 -5.000000e-01  2.500000e-01
135  7.071068e-01 -7.071068e-01  5.000000e-01
150  5.000000e-01 -8.660254e-01  7.500000e-01
165  2.588190e-01 -9.659258e-01  9.330127e-01
180  1.224647e-16 -1.000000e+00  1.000000e+00
```

Concatenating objects

```
df1 = pd.DataFrame(
    {
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    },
    index=[0, 1, 2, 3],
)


df2 = pd.DataFrame(
    {
        "A": ["A4", "A5", "A6", "A7"],
        "B": ["B4", "B5", "B6", "B7"],
        "C": ["C4", "C5", "C6", "C7"],
        "D": ["D4", "D5", "D6", "D7"],
    },
    index=[4, 5, 6, 7],
)


df3 = pd.DataFrame(
    {
        "A": ["A8", "A9", "A10", "A11"],
        "B": ["B8", "B9", "B10", "B11"],
        "C": ["C8", "C9", "C10", "C11"],
        "D": ["D8", "D9", "D10", "D11"],
    },
    index=[8, 9, 10, 11],
)

frames = [df1, df2, df3]

result = pd.concat(frames)
result
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9

```
10  A10  B10  C10  D10
11  A11  B11  C11  D11
```

```
pd.concat(
    frames,
    axis = 0,
    join = "outer",
    ignore_index = False,
    keys = None,
    levels = None,
    names = None,
    verify_integrity = False,
    copy = True,
)
```

```
      A    B    C    D
0    A0   B0   C0   D0
1    A1   B1   C1   D1
2    A2   B2   C2   D2
3    A3   B3   C3   D3
4    A4   B4   C4   D4
5    A5   B5   C5   D5
6    A6   B6   C6   D6
7    A7   B7   C7   D7
8    A8   B8   C8   D8
9    A9   B9   C9   D9
10  A10  B10  C10  D10
11  A11  B11  C11  D11
```

```
pd.concat(
    frames,
    axis = 0,
    join = "inner",
    ignore_index = False,
    keys = None,
    levels = None,
    names = None,
    verify_integrity = False,
    copy = True,
)
```

```
      A    B    C    D
0    A0   B0   C0   D0
1    A1   B1   C1   D1
2    A2   B2   C2   D2
3    A3   B3   C3   D3
4    A4   B4   C4   D4
5    A5   B5   C5   D5
6    A6   B6   C6   D6
7    A7   B7   C7   D7
```

```
8    A8   B8   C8   D8
9    A9   B9   C9   D9
10  A10  B10  C10  D10
11  A11  B11  C11  D11
```

```
result = pd.concat(frames, keys=["x", "y", "z"])
result
```

```
        A    B    C    D
x 0    A0   B0   C0   D0
  1    A1   B1   C1   D1
  2    A2   B2   C2   D2
  3    A3   B3   C3   D3
y 4    A4   B4   C4   D4
  5    A5   B5   C5   D5
  6    A6   B6   C6   D6
  7    A7   B7   C7   D7
z 8    A8   B8   C8   D8
  9    A9   B9   C9   D9
  10  A10  B10  C10  D10
  11  A11  B11  C11  D11
```

```
result.loc["x"]
```

```
    A   B   C   D
0  A0  B0  C0  D0
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
```

```
result.loc["x"] == df1
```

```
      A     B     C     D
0  True  True  True  True
1  True  True  True  True
2  True  True  True  True
3  True  True  True  True
```

```
df4 = pd.DataFrame(
    {
        "B": ["B2", "B3", "B6", "B7"],
        "D": ["D2", "D3", "D6", "D7"],
        "F": ["F2", "F3", "F6", "F7"],
    },
    index=[2, 3, 6, 7],
)
```

```
df1
```

```
    A   B   C   D
0  A0  B0  C0  D0
```

```
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
```

```
df4
```

```
    B   D   F
2  B2  D2  F2
3  B3  D3  F3
6  B6  D6  F6
7  B7  D7  F7
```

```
result = pd.concat([df1, df4], axis = 1) # default olarak outer.
tümünü alır
result
```

```
     A    B    C    D    B    D    F
0   A0   B0   C0   D0  NaN  NaN  NaN
1   A1   B1   C1   D1  NaN  NaN  NaN
2   A2   B2   C2   D2   B2   D2   F2
3   A3   B3   C3   D3   B3   D3   F3
6  NaN  NaN  NaN  NaN   B6   D6   F6
7  NaN  NaN  NaN  NaN   B7   D7   F7
```

```
result = pd.concat([df1, df4], axis=1, join = "inner") # inner,
kesişimleri alır
result
```

```
    A   B   C   D   B   D   F
2  A2  B2  C2  D2  B2  D2  F2
3  A3  B3  C3  D3  B3  D3  F3
```

```
result = pd.concat([df1, df4], axis=1).reindex(df1.index)
result
```

```
    A   B   C   D    B    D    F
0  A0  B0  C0  D0  NaN  NaN  NaN
1  A1  B1  C1  D1  NaN  NaN  NaN
2  A2  B2  C2  D2   B2   D2   F2
3  A3  B3  C3  D3   B3   D3   F3
```

Concatenating with mixed ndims

```
s1 = pd.Series(["X0", "X1", "X2", "X3"], name="X")

result = pd.concat([df1, s1], axis = 1)
result
```

```
    A   B   C   D   X
0  A0  B0  C0  D0  X0
```

```
1  A1  B1  C1  D1  X1
2  A2  B2  C2  D2  X2
3  A3  B3  C3  D3  X3
```

```
s2 = pd.Series(["_0", "_1", "_2", "_3"])

result = pd.concat([df1, s2, s2, s2], axis=1)
result
```

```
    A   B   C   D   0   1   2
0  A0  B0  C0  D0  _0  _0  _0
1  A1  B1  C1  D1  _1  _1  _1
2  A2  B2  C2  D2  _2  _2  _2
3  A3  B3  C3  D3  _3  _3  _3
```

```
result = pd.concat([df1, s1], axis=1, ignore_index=True)
result
```

```
    0   1   2   3   4
0  A0  B0  C0  D0  X0
1  A1  B1  C1  D1  X1
2  A2  B2  C2  D2  X2
3  A3  B3  C3  D3  X3
```

```
s3 = pd.Series([0, 1, 2, 3], name = "foo")

s4 = pd.Series([0, 1, 2, 3])

s5 = pd.Series([0, 1, 4, 5])

pd.concat([s3, s4, s5], axis = 1)
```

```
   foo  0  1
0    0  0  0
1    1  1  1
2    2  2  4
3    3  3  5
```

```
s3 = pd.Series([0, 1, 2, 3], name = "foo")

s4 = pd.Series([0, 1, 2, 3], name = "goo")

s5 = pd.Series([0, 1, 4, 5])

pd.concat([s3, s4, s5], axis = 1)
```

```
   foo  goo  0
0    0    0  0
1    1    1  1
2    2    2  4
3    3    3  5
```

```
pd.concat([s3, s4, s5], axis=1, keys = ["red", "blue", "yellow"])
```

```
   red  blue  yellow
0    0     0       0
1    1     1       1
2    2     2       4
3    3     3       5
```

```
frames
```

```
[    A   B   C   D
 0  A0  B0  C0  D0
 1  A1  B1  C1  D1
 2  A2  B2  C2  D2
 3  A3  B3  C3  D3,
     A   B   C   D
 4  A4  B4  C4  D4
 5  A5  B5  C5  D5
 6  A6  B6  C6  D6
 7  A7  B7  C7  D7,
       A    B    C    D
 8    A8   B8   C8   D8
 9    A9   B9   C9   D9
 10  A10  B10  C10  D10
 11  A11  B11  C11  D11]
```

```
result = pd.concat(frames, keys = ["x", "y", "z"])
result
```

```
        A    B    C    D
x 0    A0   B0   C0   D0
  1    A1   B1   C1   D1
  2    A2   B2   C2   D2
  3    A3   B3   C3   D3
y 4    A4   B4   C4   D4
  5    A5   B5   C5   D5
  6    A6   B6   C6   D6
  7    A7   B7   C7   D7
z 8    A8   B8   C8   D8
  9    A9   B9   C9   D9
  10  A10  B10  C10  D10
  11  A11  B11  C11  D11
```

```
pieces = {"x": df1, "y": df2, "z": df3}

result = pd.concat(pieces)
result
```

```
        A    B    C    D
x 0    A0   B0   C0   D0
  1    A1   B1   C1   D1
  2    A2   B2   C2   D2
  3    A3   B3   C3   D3
y 4    A4   B4   C4   D4
  5    A5   B5   C5   D5
  6    A6   B6   C6   D6
  7    A7   B7   C7   D7
z 8    A8   B8   C8   D8
  9    A9   B9   C9   D9
  10  A10  B10  C10  D10
  11  A11  B11  C11  D11
```

```
pieces
```

```
{'x':     A   B   C   D
 0  A0  B0  C0  D0
 1  A1  B1  C1  D1
 2  A2  B2  C2  D2
 3  A3  B3  C3  D3,
 'y':     A   B   C   D
 4  A4  B4  C4  D4
 5  A5  B5  C5  D5
 6  A6  B6  C6  D6
 7  A7  B7  C7  D7,
 'z':       A    B    C    D
 8    A8   B8   C8   D8
 9    A9   B9   C9   D9
 10  A10  B10  C10  D10
 11  A11  B11  C11  D11}
```

```
result = pd.concat(pieces, keys=["z", "y"])
result
```

```
        A    B    C    D
z 8    A8   B8   C8   D8
  9    A9   B9   C9   D9
  10  A10  B10  C10  D10
  11  A11  B11  C11  D11
y 4    A4   B4   C4   D4
  5    A5   B5   C5   D5
  6    A6   B6   C6   D6
  7    A7   B7   C7   D7
```

```
result.index.levels
```

```
FrozenList([['z', 'y'], [4, 5, 6, 7, 8, 9, 10, 11]])
```

```
result = pd.concat(
    pieces, keys=["x", "y", "z"], levels=[["z", "y", "x", "w"]],
```

```
names=["group_key"])
result
```

```
                A    B    C    D
group_key
x         0    A0   B0   C0   D0
          1    A1   B1   C1   D1
          2    A2   B2   C2   D2
          3    A3   B3   C3   D3
y         4    A4   B4   C4   D4
          5    A5   B5   C5   D5
          6    A6   B6   C6   D6
          7    A7   B7   C7   D7
z         8    A8   B8   C8   D8
          9    A9   B9   C9   D9
          10  A10  B10  C10  D10
          11  A11  B11  C11  D11
```

```
result.index.levels
```

```
FrozenList([['z', 'y', 'x', 'w'], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11]])
```

```
result = pd.concat(
    pieces, keys=["x", "y", "z"], levels=[["x", "y", "z", "w"]],
names=["group_key"])
result
```

```
                A    B    C    D
group_key
x         0    A0   B0   C0   D0
          1    A1   B1   C1   D1
          2    A2   B2   C2   D2
          3    A3   B3   C3   D3
y         4    A4   B4   C4   D4
          5    A5   B5   C5   D5
          6    A6   B6   C6   D6
          7    A7   B7   C7   D7
z         8    A8   B8   C8   D8
          9    A9   B9   C9   D9
          10  A10  B10  C10  D10
          11  A11  B11  C11  D11
```

```
result.index.levels
```

```
FrozenList([['x', 'y', 'z', 'w'], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11]])
```

```
s2 = pd.Series(["X0", "X1", "X2", "X3"], index=["A", "B", "C", "D"])
s2
```

```
A    X0
B    X1
C    X2
D    X3
dtype: object
```

```
result = pd.concat([df1, s2.to_frame().T], ignore_index = True)
result
```

```
    A   B   C   D
0  A0  B0  C0  D0
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
4  X0  X1  X2  X3
```

Database Style

Merge

```
left = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K3"],
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
    }
)


right = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)

result = pd.merge(left, right, on = "key")
result
```

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

```
left = pd.DataFrame(
    {
        "key1": ["K0", "K0", "K1", "K2"],
        "key2": ["K0", "K1", "K0", "K1"],
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
    }
)


right = pd.DataFrame(
    {
        "key1": ["K0", "K1", "K1", "K2"],
        "key2": ["K0", "K0", "K0", "K0"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)

left
```

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

```
right
```

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

```
result = pd.merge(left, right, on=["key1", "key2"]) # kesişimleri alır

result
```

	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

```
result = pd.merge(left, right, how = "left", on = ["key1", "key2"])
result
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1

```
3   K1   K0  A2  B2   C2   D2
4   K2   K1  A3  B3  NaN  NaN
```

```
result = pd.merge(left, right, how = "inner", on = ["key1", "key2"])
result
```

```
  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
```

```
result = pd.merge(left, right, how="cross")
result
```

```
   key1_x key2_x   A   B key1_y key2_y   C   D
0      K0     K0  A0  B0     K0     K0  C0  D0
1      K0     K0  A0  B0     K1     K0  C1  D1
2      K0     K0  A0  B0     K1     K0  C2  D2
3      K0     K0  A0  B0     K2     K0  C3  D3
4      K0     K1  A1  B1     K0     K0  C0  D0
5      K0     K1  A1  B1     K1     K0  C1  D1
6      K0     K1  A1  B1     K1     K0  C2  D2
7      K0     K1  A1  B1     K2     K0  C3  D3
8      K1     K0  A2  B2     K0     K0  C0  D0
9      K1     K0  A2  B2     K1     K0  C1  D1
10     K1     K0  A2  B2     K1     K0  C2  D2
11     K1     K0  A2  B2     K2     K0  C3  D3
12     K2     K1  A3  B3     K0     K0  C0  D0
13     K2     K1  A3  B3     K1     K0  C1  D1
14     K2     K1  A3  B3     K1     K0  C2  D2
15     K2     K1  A3  B3     K2     K0  C3  D3
```

```
df = pd.DataFrame({"Let": ["A", "B", "C"], "Num": [1, 2, 3]})
df
```

```
  Let  Num
0   A    1
1   B    2
2   C    3
```

```
ser = pd.Series(
    ["a", "b", "c", "d", "e", "f"],
    index = pd.MultiIndex.from_arrays(
        [["A", "B", "C"] * 2, [1, 2, 3, 4, 5, 6]],
        names=["Let", "Num"]),
)

ser
```

```
Let  Num
A    1      a
B    2      b
C    3      c
A    4      d
B    5      e
C    6      f
dtype: object
```

```
pd.merge(df, ser.reset_index(), on=["Let", "Num"])
```

```
  Let  Num  0
0   A    1  a
1   B    2  b
2   C    3  c
```

```
left = pd.DataFrame({"A": [1, 2], "B": [2, 2]})
right = pd.DataFrame({"A": [4, 5, 6], "B": [2, 2, 2]})
```

```
left
```

```
   A  B
0  1  2
1  2  2
```

```
right
```

```
   A  B
0  4  2
1  5  2
2  6  2
```

```
result = pd.merge(left, right, on = "B", how = "outer")
result
```

```
   A_x  B  A_y
0    1  2    4
1    1  2    5
2    1  2    6
3    2  2    4
4    2  2    5
5    2  2    6
```

Joining on index

```
left = pd.DataFrame({"A": ["A0", "A1", "A2"],
                     "B": ["B0", "B1", "B2"]},
                    index=["K0", "K1", "K2"])
```

```
right = pd.DataFrame({"C": ["C0", "C2", "C3"],
                      "D": ["D0", "D2", "D3"]},
                     index=["K0", "K2", "K3"])

left
```

```
     A   B
K0  A0  B0
K1  A1  B1
K2  A2  B2
```

```
right
```

```
     C   D
K0  C0  D0
K2  C2  D2
K3  C3  D3
```

```
result = left.join(right)
result
```

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

```
result = left.join(right, how="outer")
result
```

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

```
result = left.join(right, how="inner")
result
```

```
     A   B   C   D
K0  A0  B0  C0  D0
K2  A2  B2  C2  D2
```

```
result = pd.merge(left, right, left_index = True, right_index = True,
how = "outer")
result
```

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

```
result = pd.merge(left, right, left_index = True, right_index = True,
how = "inner")
result
```

```
     A   B   C   D
K0  A0  B0  C0  D0
K2  A2  B2  C2  D2
```

Joining key columns on an index

```
left = pd.DataFrame({"A": ["A0", "A1", "A2", "A3"],
                     "B": ["B0", "B1", "B2", "B3"],
                     "key": ["K0", "K1", "K0", "K1"]})


right = pd.DataFrame({"C": ["C0", "C1"],
                      "D": ["D0", "D1"]},
                     index=["K0", "K1"])

left
```

```
    A   B key
0  A0  B0  K0
1  A1  B1  K1
2  A2  B2  K0
3  A3  B3  K1
```

```
right
```

```
     C   D
K0  C0  D0
K1  C1  D1
```

```
result = left.join(right, on = "key")
result
```

```
    A   B key   C   D
0  A0  B0  K0  C0  D0
1  A1  B1  K1  C1  D1
2  A2  B2  K0  C0  D0
3  A3  B3  K1  C1  D1
```

```
left = pd.DataFrame({"A": ["A0", "A1", "A2", "A3"],
                     "B": ["B0", "B1", "B2", "B3"],
                     "key1": ["K0", "K0", "K1", "K2"],
                     "key2": ["K0", "K1", "K0", "K1"]})

index = pd.MultiIndex.from_tuples([("K0", "K0"),
                                   ("K1", "K0"),
```

```
                                    ("K2", "K0"),
                                    ("K2", "K1")])

left
```

```
    A   B key1 key2
0  A0  B0   K0   K0
1  A1  B1   K0   K1
2  A2  B2   K1   K0
3  A3  B3   K2   K1
```

```
index
```

```
MultiIndex([('K0', 'K0'),
            ('K1', 'K0'),
            ('K2', 'K0'),
            ('K2', 'K1')],
           )
```

```
right = pd.DataFrame({"C": ["C0", "C1", "C2", "C3"],
                      "D": ["D0", "D1", "D2", "D3"]},
                     index = index)

result = left.join(right, on=["key1", "key2"])
result
```

```
    A   B key1 key2    C    D
0  A0  B0   K0   K0   C0   D0
1  A1  B1   K0   K1  NaN  NaN
2  A2  B2   K1   K0   C1   D1
3  A3  B3   K2   K1   C3   D3
```

```
result = left.join(right, on=["key1", "key2"], how="inner")
result
```

```
    A   B key1 key2   C   D
0  A0  B0   K0   K0  C0  D0
2  A2  B2   K1   K0  C1  D1
3  A3  B3   K2   K1  C3  D3
```

Joining a single Index to a MultiIndex

```
left = pd.DataFrame({"A": ["A0", "A1", "A2"],
                     "B": ["B0", "B1", "B2"]},
                    index=pd.Index(["K0", "K1", "K2"], name="key"))


index = pd.MultiIndex.from_tuples(
    [("K0", "Y0"), ("K1", "Y1"), ("K2", "Y2"), ("K2", "Y3")],
    names=["key", "Y"],
```

```
)


right = pd.DataFrame({"C": ["C0", "C1", "C2", "C3"],
                      "D": ["D0", "D1", "D2", "D3"]},
                     index=index)

left
```

```
      A   B
key
K0   A0  B0
K1   A1  B1
K2   A2  B2
```

```
index
```

```
MultiIndex([('K0', 'Y0'),
            ('K1', 'Y1'),
            ('K2', 'Y2'),
            ('K2', 'Y3')],
           names=['key', 'Y'])
```

```
right
```

```
         C   D
key Y
K0  Y0  C0  D0
K1  Y1  C1  D1
K2  Y2  C2  D2
    Y3  C3  D3
```

```
result = left.join(right, how = "inner")
result
```

```
         A   B   C   D
key Y
K0  Y0  A0  B0  C0  D0
K1  Y1  A1  B1  C1  D1
K2  Y2  A2  B2  C2  D2
    Y3  A2  B2  C3  D3
```

Joining with two MultiIndexes

```
leftindex = pd.MultiIndex.from_product(
    [list("abc"), list("xy"), [1, 2]], names=["abc", "xy", "num"]
)


left = pd.DataFrame({"v1": range(12)}, index=leftindex)
```

```
left
```

```
             v1
abc xy num
a   x  1     0
       2     1
    y  1     2
       2     3
b   x  1     4
       2     5
    y  1     6
       2     7
c   x  1     8
       2     9
    y  1    10
       2    11
```

```
rightindex = pd.MultiIndex.from_product(
    [list("abc"), list("xy")], names=["abc", "xy"]
)


right = pd.DataFrame({"v2": [100 * i for i in range(1, 7)]},
index=rightindex)

right
```

```
          v2
abc xy
a   x    100
    y    200
b   x    300
    y    400
c   x    500
    y    600
```

```
left.join(right, on=["abc", "xy"], how="inner")
```

```
             v1   v2
abc xy num
a   x  1     0  100
       2     1  100
    y  1     2  200
       2     3  200
b   x  1     4  300
       2     5  300
    y  1     6  400
       2     7  400
c   x  1     8  500
       2     9  500
```

```
    y  1    10  600
       2    11  600
```

```
leftindex = pd.MultiIndex.from_tuples(
    [("K0", "X0"), ("K0", "X1"), ("K1", "X2")], names=["key", "X"]
)


left = pd.DataFrame(
    {"A": ["A0", "A1", "A2"], "B": ["B0", "B1", "B2"]},
index=leftindex
)


rightindex = pd.MultiIndex.from_tuples(
    [("K0", "Y0"), ("K1", "Y1"), ("K2", "Y2"), ("K2", "Y3")],
names=["key", "Y"]
)


right = pd.DataFrame(
    {"C": ["C0", "C1", "C2", "C3"], "D": ["D0", "D1", "D2", "D3"]},
index=rightindex
)


result = pd.merge(
    left.reset_index(), right.reset_index(), on=["key"], how="inner"
).set_index(["key", "X", "Y"])
result
```

```
            A   B   C   D
key X  Y
K0  X0 Y0  A0  B0  C0  D0
    X1 Y0  A1  B1  C0  D0
K1  X2 Y1  A2  B2  C1  D1
```

Timeseries friendly merging

Merging ordered data

```
left = pd.DataFrame({"k": ["K0", "K1", "K1", "K2"],
                     "lv": [1, 2, 3, 4],
                     "s": ["a", "b", "c", "d"]}
)
```

```
right = pd.DataFrame({"k": ["K1", "K2", "K4"], "rv": [1, 2, 3]})

pd.merge_ordered(left, right, fill_method="ffill", left_by="s")
```

```
     k   lv  s   rv
0   K0  1.0  a  NaN
1   K1  1.0  a  1.0
2   K2  1.0  a  2.0
3   K4  1.0  a  3.0
4   K1  2.0  b  1.0
5   K2  2.0  b  2.0
6   K4  2.0  b  3.0
7   K1  3.0  c  1.0
8   K2  3.0  c  2.0
9   K4  3.0  c  3.0
10  K1  NaN  d  1.0
11  K2  4.0  d  2.0
12  K4  4.0  d  3.0
```

```
trades = pd.DataFrame(
    {
        "time": pd.to_datetime(
            [
                "20160525 13:30:00.023",
                "20160525 13:30:00.038",
                "20160525 13:30:00.048",
                "20160525 13:30:00.048",
                "20160525 13:30:00.048",
            ]
        ),
        "ticker": ["MSFT", "MSFT", "GOOG", "GOOG", "AAPL"],
        "price": [51.95, 51.95, 720.77, 720.92, 98.00],
        "quantity": [75, 155, 100, 100, 100],
    },
    columns=["time", "ticker", "price", "quantity"],
)

trades
```

```
                     time ticker   price  quantity
0 2016-05-25 13:30:00.023   MSFT   51.95        75
1 2016-05-25 13:30:00.038   MSFT   51.95       155
2 2016-05-25 13:30:00.048   GOOG  720.77       100
3 2016-05-25 13:30:00.048   GOOG  720.92       100
4 2016-05-25 13:30:00.048   AAPL   98.00       100
```

```
quotes = pd.DataFrame(
    {
        "time": pd.to_datetime(
            [
```

```
                "20160525 13:30:00.023",
                "20160525 13:30:00.023",
                "20160525 13:30:00.030",
                "20160525 13:30:00.041",
                "20160525 13:30:00.048",
                "20160525 13:30:00.049",
                "20160525 13:30:00.072",
                "20160525 13:30:00.075",
            ]
        ),
        "ticker": ["GOOG", "MSFT", "MSFT", "MSFT", "GOOG", "AAPL",
"GOOG", "MSFT"],
        "bid": [720.50, 51.95, 51.97, 51.99, 720.50, 97.99, 720.50,
52.01],
        "ask": [720.93, 51.96, 51.98, 52.00, 720.93, 98.01, 720.88,
52.03],
    },
    columns=["time", "ticker", "bid", "ask"],
)

quotes
```

	time	ticker	bid	ask
0	2016-05-25 13:30:00.023	GOOG	720.50	720.93
1	2016-05-25 13:30:00.023	MSFT	51.95	51.96
2	2016-05-25 13:30:00.030	MSFT	51.97	51.98
3	2016-05-25 13:30:00.041	MSFT	51.99	52.00
4	2016-05-25 13:30:00.048	GOOG	720.50	720.93
5	2016-05-25 13:30:00.049	AAPL	97.99	98.01
6	2016-05-25 13:30:00.072	GOOG	720.50	720.88
7	2016-05-25 13:30:00.075	MSFT	52.01	52.03

```
pd.merge_asof(trades, quotes, on="time", by="ticker")
```

	time	ticker	price	quantity	bid	ask
0	2016-05-25 13:30:00.023	MSFT	51.95	75	51.95	51.96
1	2016-05-25 13:30:00.038	MSFT	51.95	155	51.97	51.98
2	2016-05-25 13:30:00.048	GOOG	720.77	100	720.50	720.93
3	2016-05-25 13:30:00.048	GOOG	720.92	100	720.50	720.93
4	2016-05-25 13:30:00.048	AAPL	98.00	100	NaN	NaN

```
pd.merge_asof(trades, quotes, on="time", by="ticker",
tolerance=pd.Timedelta("2ms"))
```

	time	ticker	price	quantity	bid	ask
0	2016-05-25 13:30:00.023	MSFT	51.95	75	51.95	51.96
1	2016-05-25 13:30:00.038	MSFT	51.95	155	NaN	NaN
2	2016-05-25 13:30:00.048	GOOG	720.77	100	720.50	720.93
3	2016-05-25 13:30:00.048	GOOG	720.92	100	720.50	720.93
4	2016-05-25 13:30:00.048	AAPL	98.00	100	NaN	NaN

```
pd.merge_asof(
    trades,
    quotes,
    on="time",
    by="ticker",
    tolerance=pd.Timedelta("10ms"),
    allow_exact_matches=False,
)
```

	time	ticker	price	quantity	bid	ask
0	2016-05-25 13:30:00.023	MSFT	51.95	75	NaN	NaN
1	2016-05-25 13:30:00.038	MSFT	51.95	155	51.97	51.98
2	2016-05-25 13:30:00.048	GOOG	720.77	100	NaN	NaN
3	2016-05-25 13:30:00.048	GOOG	720.92	100	NaN	NaN
4	2016-05-25 13:30:00.048	AAPL	98.00	100	NaN	NaN

```
# DataFrame' i dışarıya excel dosyası olarak göndermek

df1.to_excel("solardata.xlsx", sheet_name='Sheet_name_1')

web den data çekmek

url = "https://nssdc.gsfc.nasa.gov/planetary/factsheet/index.html"
data = pd.read_html(url)
data



 # str olan sütun değerlerini float yapmak için

d1["mass"] = pd.to_numeric(d1["mass"])
type(d1["mass"][1])

# dataframe i sıralamak için
df.sort_values(by="mass", inplace=True)
df
```