Veri Manipülasyonu - Pandas

Pandas Serisi oluşturmak

In [86]:

```
import pandas as pd
```

In [87]:

```
pd.Series([10, 27, 32, 41, 5])
```

Out[87]:

```
0    10
1    27
2    32
3    41
4     5
dtype: int64
```

In [88]:

```
seri = pd.Series([10, 27, 32, 41, 5])
type(seri)
```

Out[88]:

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

In [89]:

```
seri.axes
```

Out[89]:

```
[RangeIndex(start=0, stop=5, step=1)]
```

In [90]:

```
seri.dtype
```

Out[90]:

```
dtype('int64')
```

In [91]:

```
seri.size
```

Out[91]:

```
5
```

In [92]:

```
seri.ndim
```

Out[92]:

```
1
```

In [93]:

```
seri.values
```

Out[93]:

```
array([10, 27, 32, 41,  5], dtype=int64)
```

In [94]:

```
seri.head()
```

Out[94]:

```
0    10
1    27
2    32
3    41
4     5
dtype: int64
```

In [95]:

```
seri.head(2)
```

Out[95]:

```
0    10
1    27
dtype: int64
```

In [96]:

```
seri.tail()
```

Out[96]:

```
0    10
1    27
2    32
3    41
4     5
dtype: int64
```

In [97]:

```
seri.tail(2)
```

Out[97]:

```
3    41
4     5
dtype: int64
```

In [98]:

```
pd.Series([23,67,34,9,3], index=[1, 3, 5, 7, 9])
```

Out[98]:

```
1    23
3    67
5    34
7     9
9     3
dtype: int64
```

In [99]:

```
different_type = pd.Series([23,"Mustafa",3.4,9,3], index=[1, 3, 5, 7, 9])
different_type
```

Out[99]:

```
1         23
3    Mustafa
5        3.4
7          9
9          3
dtype: object
```

In [100]:

```
different_type[1:4]
```

Out[100]:

```
3    Mustafa
5        3.4
7          9
dtype: object
```

In [101]:

```
seri = pd.Series([23,67,34,9,3], index=["Ayşe", "Fatma", "Hayriye", "Mustafa", "9"])
seri
```

Out[101]:

```
Ayşe       23
Fatma      67
Hayriye    34
Mustafa     9
9           3
dtype: int64
```

In [102]:

```
seri["Fatma"]
```

Out[102]:

```
67
```

In [103]:

```
seri["Ayşe":"Hayriye"]
```

Out[103]:

```
Ayşe       23
Fatma      67
Hayriye    34
dtype: int64
```

In [104]:

```
sozluk = {"reg": 10, "loj": 11, "cart": 12}
seri = pd.Series(sozluk)
seri
```

Out[104]:

```
reg     10
loj     11
cart    12
dtype: int64
```

In [105]:

```
pd.concat([seri, seri])
```

Out[105]:

```
reg     10
loj     11
cart    12
reg     10
loj     11
cart    12
dtype: int64
```

Eleman işlemleri

Eleman İşlemleri

In [106]:

```
import numpy as np
a = np.array([1, 2, 33, 444, 75], dtype="int64")
seri = pd.Series(a)
seri
```

Out[106]:

```
0      1
1      2
2     33
3    444
4     75
dtype: int64
```

In [107]:

```
seri[0]
```

Out[107]:

```
1
```

In [108]:

```
seri[0:3]
```

Out[108]:

```
0     1
1     2
2    33
dtype: int64
```

In [109]:

```
seri = pd.Series([121, 200, 150, 99],
                 index=["reg", "loj", "cart", "rf"])
seri
```

Out[109]:

```
reg     121
loj     200
cart    150
rf       99
dtype: int64
```

In [110]:

```
seri.index
```

Out[110]:

```
Index(['reg', 'loj', 'cart', 'rf'], dtype='object')
```

In [111]:

```
seri.keys
```

Out[111]:

```
<bound method Series.keys of reg     121
loj     200
cart    150
rf       99
dtype: int64>
```

In [112]:

```
list(seri.items())
```

Out[112]:

```
[('reg', 121), ('loj', 200), ('cart', 150), ('rf', 99)]
```

In [113]:

```
seri.values
```

Out[113]:

```
array([121, 200, 150,  99], dtype=int64)
```

In [114]:

```
"reg" in seri
```

Out[114]:

```
True
```

In [115]:

```
seri["reg"]
```

Out[115]:

```
121
```

In [116]:

```
seri[["rf", "reg"]]
```

Out[116]:

```
rf      99
reg    121
dtype: int64
```

In [117]:

```
seri["reg"] = 130
seri["reg"]
```

Out[117]:

```
130
```

In [118]:

```
seri
```

Out[118]:

```
reg     130
loj     200
cart    150
rf       99
dtype: int64
```

In [119]:

```
seri["reg":"loj"]
```

Out[119]:

```
reg    130
loj    200
dtype: int64
```

Pandas DataFrame

In [120]:

```
import numpy as np
```

In [121]:

```
liste = [1, 2, 39, 67, 90]
df = pd.DataFrame(liste, columns=["degisken_ismi"])
df
```

Out[121]:

	degisken_ismi
0	**1**
1	**2**
2	**39**
3	**67**
4	**90**

In [122]:

```
m = np.arange(1,10).reshape((3,3))
m_df = pd.DataFrame(m, columns=["degisken1", "degisken2", "degisken3"])
m_df.head()
```

Out[122]:

	degisken1	degisken2	degisken3
0	**1**	**2**	**3**
1	**4**	**5**	**6**
2	**7**	**8**	**9**

In [123]:

```
m_df.columns
```

Out[123]:

```
Index(['degisken1', 'degisken2', 'degisken3'], dtype='object')
```

In [124]:

```
m_df.columns = ["var1", "var2", "var3"]
```

In [125]:

```
type(m_df)
```

Out[125]:

```
pandas.core.frame.DataFrame
```

In [126]:

```
m_df.axes
```

Out[126]:

```
[RangeIndex(start=0, stop=3, step=1),
 Index(['var1', 'var2', 'var3'], dtype='object')]
```

In [127]:

```
m_df.shape
```

Out[127]:

```
(3, 3)
```

In [128]:

```
m_df.ndim
```

Out[128]:

```
2
```

In [129]:

```
m_df.size
```

Out[129]:

```
9
```

In [130]:

```
m_df.values
```

Out[130]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [131]:

```
type(m_df.values)
```

Out[131]:

```
numpy.ndarray
```

In [132]:

```
m_df.head()
```

Out[132]:

	var1	var2	var3
0	1	2	3
1	4	5	6
2	7	8	9

In [133]:

```
m_df.tail(1)
```

Out[133]:

	var1	var2	var3
2	7	8	9

In [134]:

```
a = np.array([1, 2, 3, 4, 5])
```

In [135]:

```
pd.DataFrame(a, columns=["deg1"])
```

Out[135]:

	deg1
0	1

1	deg?
2	3
3	4
4	5

Eleman İşlemleri

In [136]:

```
s1 = np.random.randint(10, size= 5)
s2 = np.random.randint(10, size= 5)
s3 = np.random.randint(10, size= 5)
```

In [137]:

```
dict1 = {"var1": s1, "var2": s2, "var3":s3}
dict1
```

Out[137]:

```
{'var1': array([4, 1, 6, 2, 2]),
 'var2': array([8, 7, 6, 9, 3]),
 'var3': array([9, 5, 5, 5, 8])}
```

In [138]:

```
df = pd.DataFrame(dict1) #index=[0, 1, 2, 3, 4]
df
```

Out[138]:

	var1	var2	var3
0	4	8	9
1	1	7	5
2	6	6	5
3	2	9	5
4	2	3	8

In [139]:

```
df[0:2]
```

Out[139]:

	var1	var2	var3
0	4	8	9
1	1	7	5

In [140]:

```
df.index
```

Out[140]:

```
RangeIndex(start=0, stop=5, step=1)
```

In [141]:

```
df.index = ["a", "b", "c", "d", "e"]
df
```

Out[141]:

	var1	var2	var3
a	4	8	9
b	1	7	5
c	6	6	5
d	2	9	5
e	2	3	8

In [142]:

```
df["c":"e"]
```

Out[142]:

	var1	var2	var3
c	6	6	5
d	2	9	5
e	2	3	8

In [143]:

```
df.drop("a", axis=0)
```

Out[143]:

	var1	var2	var3
b	1	7	5
c	6	6	5
d	2	9	5
e	2	3	8

In [144]:

```
df
```

Out[144]:

	var1	var2	var3
a	4	8	9
b	1	7	5
c	6	6	5
d	2	9	5
e	2	3	8

In [145]:

```
df.drop(["a","b"], axis=0)
```

Out[145]:

	var1	var2	var3
c	6	6	5
d	2	9	5
e	2	3	8

In [146]:

```
df.drop(["a","b"], axis=0, inplace=True)
```

In [147]:

```
df
```

Out[147]:

	var1	var2	var3
c	6	6	5
d	2	9	5
e	2	3	8

In [148]:

```
l = ["c", "e"]
```

In [149]:

```
df.drop(l, axis=0)
```

Out[149]:

	var1	var2	var3
d	2	9	5

In [150]:

```
"var1" in df
```

Out[150]:

```
True
```

In [151]:

```
l = ["var1", "var2", "var4"]
for i in l:
  print(i in df)
```

```
True
True
False
```

In [152]:

```
df["var1"]
```

Out[152]:

```
c    6
d    2
e    2
Name: var1, dtype: int32
```

In [153]:

```
df["var4"] = df["var1"] / df["var2"]
df[["var4"]]
```

Out[153]:

	var4
c	1.000000
d	0.222222

	var4
e	0.666667

In [154]:

```
df
```

Out[154]:

	var1	var2	var3	var4
c	6	6	5	1.000000
d	2	9	5	0.222222
e	2	3	8	0.666667

In [155]:

```
type(df["var1"])
```

Out[155]:

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

In [156]:

```
df.drop("var4", axis=1)
```

Out[156]:

	var1	var2	var3
c	6	6	5
d	2	9	5
e	2	3	8

In [157]:

```
l = ["var1", "var2", "var3"]
df.drop(l, axis=1)
```

Out[157]:

	var4
c	1.000000
d	0.222222
e	0.666667

Gözlem ve Değişken Seçimi: loc & iloc

In [158]:

```
m = np.random.randint(1, 30, size=(10,3))
df = pd.DataFrame(m, columns=["var1", "var2", "var3"])
df
```

Out[158]:

	var1	var2	var3
0	15	2	3
1	1	3	11
2	26	6	8
3	27	15	27

	var1	var2	var3
4	18	15	8
5	8	20	26
6	3	5	9
7	27	13	8
8	4	2	2
9	19	8	13

In [159]:

```
df.loc[0:3]
```

Out[159]:

	var1	var2	var3
0	15	2	3
1	1	3	11
2	26	6	8
3	27	15	27

In [160]:

```
df.iloc[0:3]
```

Out[160]:

	var1	var2	var3
0	15	2	3
1	1	3	11
2	26	6	8

In [161]:

```
df.iloc[0,0]
```

Out[161]:

```
15
```

In [162]:

```
df.iloc[:3, :2]
```

Out[162]:

	var1	var2
0	15	2
1	1	3
2	26	6

In [163]:

```
df.loc[0:3, "var1"]
```

Out[163]:

```
0    15
1     1
2    26
3    27
Name: var1, dtype: int32
```

In [164]:

```
df.iloc[0:3, 1]
```

Out[164]:

```
0    2
1    3
2    6
Name: var2, dtype: int32
```

In [165]:

```
df.iloc[0:3, 1:3]
```

Out[165]:

	var2	var3
0	**2**	**3**
1	**3**	**11**
2	**6**	**8**

In [166]:

```
df.iloc[0:3]["var3"]
```

Out[166]:

```
0     3
1    11
2     8
Name: var3, dtype: int32
```

Koşullu işlemler

In [167]:

```
m = np.random.randint(1, 30, size=(10,3))
df = pd.DataFrame(m, columns=["var1", "var2", "var3"])
df
```

Out[167]:

	var1	var2	var3
0	**6**	**10**	**26**
1	**22**	**14**	**10**
2	**4**	**26**	**8**
3	**15**	**6**	**25**
4	**2**	**25**	**5**
5	**4**	**3**	**28**
6	**7**	**13**	**25**
7	**27**	**8**	**17**
8	**15**	**29**	**2**
9	**15**	**25**	**29**

In [168]:

```
df["var1"]
```

Out[168]:

```
0     6
```

```
1     22
2      4
3     15
4      2
5      4
6      7
7     27
8     15
9     15
Name: var1, dtype: int32
```

In [169]:

```
df["var1"][0:2]
```

Out[169]:

```
0      6
1     22
Name: var1, dtype: int32
```

In [170]:

```
df[0:2]["var1"]
```

Out[170]:

```
0      6
1     22
Name: var1, dtype: int32
```

In [171]:

```
df[0:2][["var1","var2"]]
```

Out[171]:

	var1	var2
0	6	10
1	22	14

In [172]:

```
df
```

Out[172]:

	var1	var2	var3
0	6	10	26
1	22	14	10
2	4	26	8
3	15	6	25
4	2	25	5
5	4	3	28
6	7	13	25
7	27	8	17
8	15	29	2
9	15	25	29

In [173]:

```
df[df["var1"] > 15]["var1"]
```

Out[173]:

```
1    22
7    27
Name: var1, dtype: int32
```

In [174]:

```
df[df.var1 > 15]
```

Out[174]:

	var1	var2	var3
1	22	14	10
7	27	8	17

In [175]:

```
df[df.var1 > 15]["var1"]
```

Out[175]:

```
1    22
7    27
Name: var1, dtype: int32
```

In [176]:

```
df[((df.var1 > 15) & (df.var3 < 10))]
```

Out[176]:

	var1	var2	var3

In [177]:

```
df.loc[((df.var1 > 15), ["var1", "var2"])]
```

Out[177]:

	var1	var2
1	22	14
7	27	8

In [178]:

```
df.loc[(df.var1 > 15)][["var1"]]
```

Out[178]:

	var1
1	22
7	27

In [179]:

```
df.loc[(df.var1 > 15)][["var1", "var2"]]
```

Out[179]:

	var1	var2
1	22	14
7	27	8

Birleştirme(Join) İşlemleri

In [180]:

```
m = np.random.randint(1, 30, size=(10,3))
df1 = pd.DataFrame(m, columns=["var1", "var2", "var3"])
df1
```

Out[180]:

	var1	var2	var3
0	14	14	6
1	15	13	1
2	15	22	24
3	18	5	29
4	9	7	9
5	15	15	15
6	1	27	17
7	14	7	24
8	5	25	1
9	25	29	16

In [181]:

```
df2 = df1 + 10
df2
```

Out[181]:

	var1	var2	var3
0	24	24	16
1	25	23	11
2	25	32	34
3	28	15	39
4	19	17	19
5	25	25	25
6	11	37	27
7	24	17	34
8	15	35	11
9	35	39	26

In [182]:

```
pd.concat([df1, df2])
```

Out[182]:

	var1	var2	var3
0	14	14	6
1	15	13	1
2	15	22	24
3	18	5	29
4	9	7	9

	var1	var2	var3
5	15	18	19
6	1	27	17
7	14	7	24
8	5	25	1
9	25	29	16
0	24	24	16
1	25	23	11
2	25	32	34
3	28	15	39
4	19	17	19
5	25	25	25
6	11	37	27
7	24	17	34
8	15	35	11
9	35	39	26

In [183]:

```
pd.concat([df1, df2], ignore_index=True)
```

Out[183]:

	var1	var2	var3
0	14	14	6
1	15	13	1
2	15	22	24
3	18	5	29
4	9	7	9
5	15	15	15
6	1	27	17
7	14	7	24
8	5	25	1
9	25	29	16
10	24	24	16
11	25	23	11
12	25	32	34
13	28	15	39
14	19	17	19
15	25	25	25
16	11	37	27
17	24	17	34
18	15	35	11
19	35	39	26

In [184]:

```
df1.columns
```

Out[184]:

```
Index(['var1', 'var2', 'var3'], dtype='object')
```

Index(['var1', 'var2', 'var3'], dtype='object')

In [185]:

```
df2.columns = ["var1", "var2", "deg3"]
```

In [186]:

```
df1
```

Out[186]:

	var1	var2	var3
0	14	14	6
1	15	13	1
2	15	22	24
3	18	5	29
4	9	7	9
5	15	15	15
6	1	27	17
7	14	7	24
8	5	25	1
9	25	29	16

In [187]:

```
df2
```

Out[187]:

	var1	var2	deg3
0	24	24	16
1	25	23	11
2	25	32	34
3	28	15	39
4	19	17	19
5	25	25	25
6	11	37	27
7	24	17	34
8	15	35	11
9	35	39	26

In [188]:

```
pd.concat([df2, df1])
```

Out[188]:

	var1	var2	deg3	var3
0	24	24	16.0	NaN
1	25	23	11.0	NaN
2	25	32	34.0	NaN
3	28	15	39.0	NaN
4	19	17	19.0	NaN

	var1	var2	deg3	var3
5	25	25	25.0	NaN
6	11	37	27.0	NaN
7	24	17	34.0	NaN
8	15	35	11.0	NaN
9	35	39	26.0	NaN
0	14	14	NaN	6.0
1	15	13	NaN	1.0
2	15	22	NaN	24.0
3	18	5	NaN	29.0
4	9	7	NaN	9.0
5	15	15	NaN	15.0
6	1	27	NaN	17.0
7	14	7	NaN	24.0
8	5	25	NaN	1.0
9	25	29	NaN	16.0

In [189]:

```
pd.concat([df1,df2], join="inner", ignore_index=True)
```

Out[189]:

	var1	var2
0	14	14
1	15	13
2	15	22
3	18	5
4	9	7
5	15	15
6	1	27
7	14	7
8	5	25
9	25	29
10	24	24
11	25	23
12	25	32
13	28	15
14	19	17
15	25	25
16	11	37
17	24	17
18	15	35
19	35	39

İleri Seviye Birleştirme İşlemleri

In [190]:

```
df1 = pd.DataFrame({'calisanlar': ['Ali', 'Veli', 'Ayse', 'Fatma'],
```

```
                    'Grup': ['Muhasebe', 'Mühendislik', 'Mühendislik', 'İK']})
df1
```

Out[190]:

	calisanlar	Grup
0	Ali	Muhasebe
1	Veli	Mühendislik
2	Ayse	Mühendislik
3	Fatma	İK

In [191]:

```
df2 = pd.DataFrame({'calisanlar': ['Ayse', 'Ali', 'Veli', 'Fatma'],
                    'ilk_giris': [2010, 2009, 2014, 2019]})
df2
```

Out[191]:

	calisanlar	ilk_giris
0	Ayse	2010
1	Ali	2009
2	Veli	2014
3	Fatma	2019

In [192]:

```
pd.merge(df1, df2)
```

Out[192]:

	calisanlar	Grup	ilk_giris
0	Ali	Muhasebe	2009
1	Veli	Mühendislik	2014
2	Ayse	Mühendislik	2010
3	Fatma	İK	2019

In [193]:

```
pd.merge(df1, df2, on="calisanlar")
```

Out[193]:

	calisanlar	Grup	ilk_giris
0	Ali	Muhasebe	2009
1	Veli	Mühendislik	2014
2	Ayse	Mühendislik	2010
3	Fatma	İK	2019

In [194]:

```
# many to one
df3 = pd.merge(df1, df2)
df3
```

Out[194]:

	calisanlar	Grup	ilk_giris

	calisanlar	Grup	ilk_giris
0	Ali	Muhasebe	2009
1	Veli	Mühendislik	2014
2	Ayse	Mühendislik	2010
3	Fatma	İK	2019

In [195]:

```
df4 = pd.DataFrame({'Grup': ["Muhasebe", "Mühendislik", "IK"],
                    'mudur': ["Caner", "Mustafa", "Berkcan"]})
df4
```

Out[195]:

	Grup	mudur
0	Muhasebe	Caner
1	Mühendislik	Mustafa
2	IK	Berkcan

In [196]:

```
pd.merge(df3,df4)
```

Out[196]:

	calisanlar	Grup	ilk_giris	mudur
0	Ali	Muhasebe	2009	Caner
1	Veli	Mühendislik	2014	Mustafa
2	Ayse	Mühendislik	2010	Mustafa

In [197]:

```
# many to many
df5 = pd.DataFrame({'Grup': ['Muhasebe', 'Muhasebe', 'Mühendislik', 'Mühendislik', 'İK',
'İK'],
                    'yetenekler': ['matematik', 'excel', 'kodlama', 'linux', 'excel', 'y
önetim']})
df5
```

Out[197]:

	Grup	yetenekler
0	Muhasebe	matematik
1	Muhasebe	excel
2	Mühendislik	kodlama
3	Mühendislik	linux
4	İK	excel
5	İK	yönetim

In [198]:

```
df1
```

Out[198]:

	calisanlar	Grup
0	Ali	Muhasebe
1	Veli	Mühendislik
2	Ayse	Mühendislik

	calisanlar	Grup
2	Ayse	Mühendislik
3	Fatma	İK

In [199]:

```
pd.merge(df1,df5)
```

Out[199]:

	calisanlar	Grup	yetenekler
0	Ali	Muhasebe	matematik
1	Ali	Muhasebe	excel
2	Veli	Mühendislik	kodlama
3	Veli	Mühendislik	linux
4	Ayse	Mühendislik	kodlama
5	Ayse	Mühendislik	linux
6	Fatma	İK	excel
7	Fatma	İK	yönetim

Aggregation & Grouping

- count()
- first()
- last()
- mean()
- median()
- min()
- max()
- std()
- var()
- sum()

In [200]:

```
import seaborn as sns
```

In [201]:

```
df = sns.load_dataset("planets")
df.head()
```

Out[201]:

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

In [202]:

```
df.shape
```

Out[202]:

```
(1035, 6)
```

In [203]:

```
df.describe().T
```

Out[203]:

	count	mean	std	min	25%	50%	75%	max
number	1035.0	1.785507	1.240976	1.000000	1.00000	1.0000	2.000	7.0
orbital_period	992.0	2002.917596	26014.728304	0.090706	5.44254	39.9795	526.005	730000.0
mass	513.0	2.638161	3.818617	0.003600	0.22900	1.2600	3.040	25.0
distance	808.0	264.069282	733.116493	1.350000	32.56000	55.2500	178.500	8500.0
year	1035.0	2009.070531	3.972567	1989.000000	2007.00000	2010.0000	2012.000	2014.0

In [204]:

```
df.mean()
```

```
c:\Users\Kaan\AppData\Local\Programs\Python\Python37\lib\site-packages\ipykernel_launcher
.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric
_only=None') is deprecated; in a future version this will raise TypeError.  Select only v
alid columns before calling the reduction.
  """Entry point for launching an IPython kernel.
```

Out[204]:

```
number              1.785507
orbital_period    2002.917596
mass                 2.638161
distance           264.069282
year              2009.070531
dtype: float64
```

In [205]:

```
df["mass"].mean()
```

Out[205]:

```
2.6381605847953216
```

In [206]:

```
df["year"].count()
```

Out[206]:

```
1035
```

In [207]:

```
df["year"].min()
```

Out[207]:

```
1989
```

In [208]:

```
df["year"].max()
```

Out[208]:

```
2014
```

In [209]:

```
df["mass"].sum()
```

Out[209]:

```
1353.37638
```

In [210]:

```
df["mass"].std()
```

Out[210]:

```
3.8186166509616046
```

In [211]:

```
df["mass"].var()
```

Out[211]:

```
14.58183312700122
```

In [212]:

```
df.dropna().describe().T
```

Out[212]:

	count	mean	std	min	25%	50%	75%	max
number	498.0	1.734940	1.175720	1.0000	1.00000	1.000	2.0000	6.0
orbital_period	498.0	835.778671	1469.128259	1.3283	38.27225	357.000	999.6000	17337.5
mass	498.0	2.509320	3.636274	0.0036	0.21250	1.245	2.8675	25.0
distance	498.0	52.068213	46.596041	1.3500	24.49750	39.940	59.3325	354.0
year	498.0	2007.377510	4.167284	1989.0000	2005.00000	2009.000	2011.0000	2014.0

Gruplama İşlemleri

In [213]:

```
df = pd.DataFrame({'gruplar': ["A", "B", "C", "A", "B", "C"],
                   'veri': [10, 11, 52, 23, 43, 55]}, columns=['gruplar', 'veri'])
df
```

Out[213]:

	gruplar	veri
0	A	10
1	B	11
2	C	52
3	A	23
4	B	43
5	C	55

In [214]:

```
df.groupby("gruplar")
```

Out[214]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002040A40E408>
```

In [215]:

```
df.groupby("gruplar").mean()
```

Out[215]:

gruplar	veri
A	16.5
B	27.0
C	53.5

In [216]:

```
df.groupby("gruplar").sum()
```

Out[216]:

gruplar	veri
A	33
B	54
C	107

In [217]:

```
df = sns.load_dataset("planets")
df.head()
```

Out[217]:

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

In [218]:

```
df.groupby("method")["orbital_period"].mean()
```

Out[218]:

```
method
Astrometry                         631.180000
Eclipse Timing Variations         4751.644444
Imaging                         118247.737500
Microlensing                      3153.571429
Orbital Brightness Modulation        0.709307
Pulsar Timing                     7343.021201
Pulsation Timing Variations       1170.000000
Radial Velocity                    823.354680
Transit                             21.102073
Transit Timing Variations           79.783500
Name: orbital_period, dtype: float64
```

In [219]:

```
df.groupby("method")["mass"].mean()
```

Out[219]:

```
method
Astrometry                          NaN
Eclipse Timing Variations      5.125000
Imaging                             NaN
Microlensing                        NaN
Orbital Brightness Modulation       NaN
Pulsar Timing                       NaN
Pulsation Timing Variations         NaN
Radial Velocity                2.630699
Transit                        1.470000
Transit Timing Variations           NaN
Name: mass, dtype: float64
```

In [220]:

```
df.groupby("method")["orbital_period"].describe().T
```

Out[220]:

method	Astrometry	Eclipse Timing Variations	Imaging	Microlensing	Orbital Brightness Modulation	Pulsar Timing	Pulsation Timing Variations	Radial Velocity	Tran
count	2.000000	9.000000	12.000000	7.000000	3.000000	5.000000	1.0	553.00000	397.0000
mean	631.180000	4751.644444	118247.737500	3153.571429	0.709307	7343.021201	1170.0	823.35468	21.1020
std	544.217663	2499.130945	213978.177277	1113.166333	0.725493	16313.265573	NaN	1454.92621	46.1858
min	246.360000	1916.250000	4639.150000	1825.000000	0.240104	0.090706	1170.0	0.73654	0.3550
25%	438.770000	2900.000000	8343.900000	2375.000000	0.291496	25.262000	1170.0	38.02100	3.1606
50%	631.180000	4343.500000	27500.000000	3300.000000	0.342887	66.541900	1170.0	360.20000	5.7149
75%	823.590000	5767.000000	94250.000000	3550.000000	0.943908	98.211400	1170.0	982.00000	16.1457
max	1016.000000	10220.000000	730000.000000	5100.000000	1.544929	36525.000000	1170.0	17337.50000	331.6005

İleri Gruplama işlemleri

In [221]:

```
import numpy as np
import pandas as pd
import seaborn as sns
df = pd.DataFrame({'gruplar':['A', 'B', 'C', 'A', 'B', 'C'],
                   'degisken1': [10, 23, 33, 22, 11, 99],
                   'degisken2': [100, 253, 333, 262, 111, 969]},
                  columns=['gruplar', 'degisken1', 'degisken2'])
df
```

Out[221]:

	gruplar	degisken1	degisken2
0	A	10	100
1	B	23	253
2	C	33	333
3	A	22	262
4	B	11	111
5	C	99	969

In [222]:

```
df.groupby("gruplar").mean()
```

Out[222]:

gruplar	degisken1	degisken2
A	16.0	181.0
B	17.0	182.0
C	66.0	651.0

In [223]:

```
df.groupby("gruplar").aggregate([min, np.median, max])
```

Out[223]:

	degisken1			degisken2		
	min	median	max	min	median	max
gruplar						
A	10	16.0	22	100	181.0	262
B	11	17.0	23	111	182.0	253
C	33	66.0	99	333	651.0	969

In [224]:

```
df.groupby("gruplar").aggregate({"degisken1": "min", "degisken2": "std"})
```

Out[224]:

gruplar	degisken1	degisken2
A	10	114.551299
B	11	100.409163
C	33	449.719913

filter

In [225]:

```
df = pd.DataFrame({'gruplar':['A', 'B', 'C', 'A', 'B', 'C'],
                   'degisken1': [10, 23, 33, 22, 11, 99],
                   'degisken2': [100, 253, 333, 262, 111, 969]},
                  columns=['gruplar', 'degisken1', 'degisken2'])
df
```

Out[225]:

	gruplar	degisken1	degisken2
0	A	10	100
1	B	23	253
2	C	33	333
3	A	22	262
4	B	11	111
5	C	99	969

In [226]:

```
def filter_func(x):
  return x["degisken1"].std() > 9
```

In [227]:

```
df.groupby("gruplar").std()
```

Out[227]:

gruplar	degisken1	degisken2
A	8.485281	114.551299
B	8.485281	100.409163
C	46.669048	449.719913

In [228]:

```
df.groupby("gruplar").filter(filter_func)
```

Out[228]:

	gruplar	degisken1	degisken2
2	C	33	333
5	C	99	969

Transform

In [229]:

```
df = pd.DataFrame({'gruplar':['A', 'B', 'C', 'A', 'B', 'C'],
                   'degisken1': [10, 23, 33, 22, 11, 99],
                   'degisken2': [100, 253, 333, 262, 111, 969]},
                  columns=['gruplar', 'degisken1', 'degisken2'])
df
```

Out[229]:

	gruplar	degisken1	degisken2
0	A	10	100
1	B	23	253
2	C	33	333
3	A	22	262
4	B	11	111
5	C	99	969

In [230]:

```
df["degisken1"] * 9
```

Out[230]:

```
0     90
1    207
2    297
3    198
4     99
5    891
Name: degisken1, dtype: int64
```

In [231]:

```
df_a = df.iloc[:, 1:]
df_a
```

Out[231]:

	degisken1	degisken2
0	10	100
1	23	253
2	33	333
3	22	262
4	11	111
5	99	969

In [232]:

```
df_a.transform(lambda x: x - x.mean() / x.std()) # normal distribution - standardizaiton
```

Out[232]:

	degisken1	degisken2
0	9.013055	98.951261
1	22.013055	251.951261
2	32.013055	331.951261
3	21.013055	260.951261
4	10.013055	109.951261
5	98.013055	967.951261

apply

In [233]:

```
df_a
```

Out[233]:

	degisken1	degisken2
0	10	100
1	23	253
2	33	333
3	22	262
4	11	111
5	99	969

In [234]:

```
df_a.apply(np.sum)
```

Out[234]:

```
degisken1     198
degisken2    2028
dtype: int64
```

In [235]:

```
df_a.apply(np.mean)[0]
```

Out[235]:

```
33.0
```

In [236]:

```
df = pd.DataFrame({'gruplar':['A', 'B', 'C', 'A', 'B', 'C'],
                   'degisken1': [10, 23, 33, 22, 11, 99],
                   'degisken2': [100, 253, 333, 262, 111, 969]},
                  columns=['gruplar', 'degisken1', 'degisken2'])
df
```

Out[236]:

	gruplar	degisken1	degisken2
0	A	10	100
1	B	23	253
2	C	33	333
3	A	22	262
4	B	11	111
5	C	99	969

In [237]:

```
df.groupby("gruplar").apply(np.sum)
```

Out[237]:

	gruplar	degisken1	degisken2
gruplar			
A	AA	32	362
B	BB	34	364
C	CC	132	1302

Pivot Tablolar

In [238]:

```
import pandas as pd
import seaborn as sns
titanic = sns.load_dataset("titanic")
titanic.head()
```

Out[238]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alo
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	Fa
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	Tr
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	Tr

In [239]:

```
titanic.groupby("sex")["survived"].mean()
```

Out[239]:

```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

In [240]:

```
titanic.groupby("sex")[["survived"]].mean()
```

Out[240]:

	survived
sex	
female	**0.742038**
male	**0.188908**

In [241]:

```
titanic.groupby(["sex", "class"])[["survived"]].aggregate("mean").unstack()
```

Out[241]:

	survived		
class	**First**	**Second**	**Third**
sex			
female	**0.968085**	**0.921053**	**0.500000**
male	**0.368852**	**0.157407**	**0.135447**

In [242]:

```
titanic.pivot_table("survived", index="sex", columns="class")
```

Out[242]:

class	**First**	**Second**	**Third**
sex			
female	**0.968085**	**0.921053**	**0.500000**
male	**0.368852**	**0.157407**	**0.135447**

In [243]:

```
titanic.age.head()
```

Out[243]:

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
Name: age, dtype: float64
```

In [244]:

```
age = pd.cut(titanic["age"], [0, 18, 90])
age.head(10)
```

Out[244]:

```
0    (18.0, 90.0]
1    (18.0, 90.0]
2    (18.0, 90.0]
3    (18.0, 90.0]
```

```
4    (18.0, 90.0]
5             NaN
6    (18.0, 90.0]
7     (0.0, 18.0]
8    (18.0, 90.0]
9     (0.0, 18.0]
Name: age, dtype: category
Categories (2, interval[int64, right]): [(0, 18] < (18, 90]]
```

In [245]:

```
titanic.pivot_table("survived", ["sex", age], "class")
```

Out[245]:

	class	First	Second	Third
sex	age			
female	(0, 18]	0.909091	1.000000	0.511628
	(18, 90]	0.972973	0.900000	0.423729
male	(0, 18]	0.800000	0.600000	0.215686
	(18, 90]	0.375000	0.071429	0.133663

Dış kaynaklı veri okuma

In [ ]:

```
```