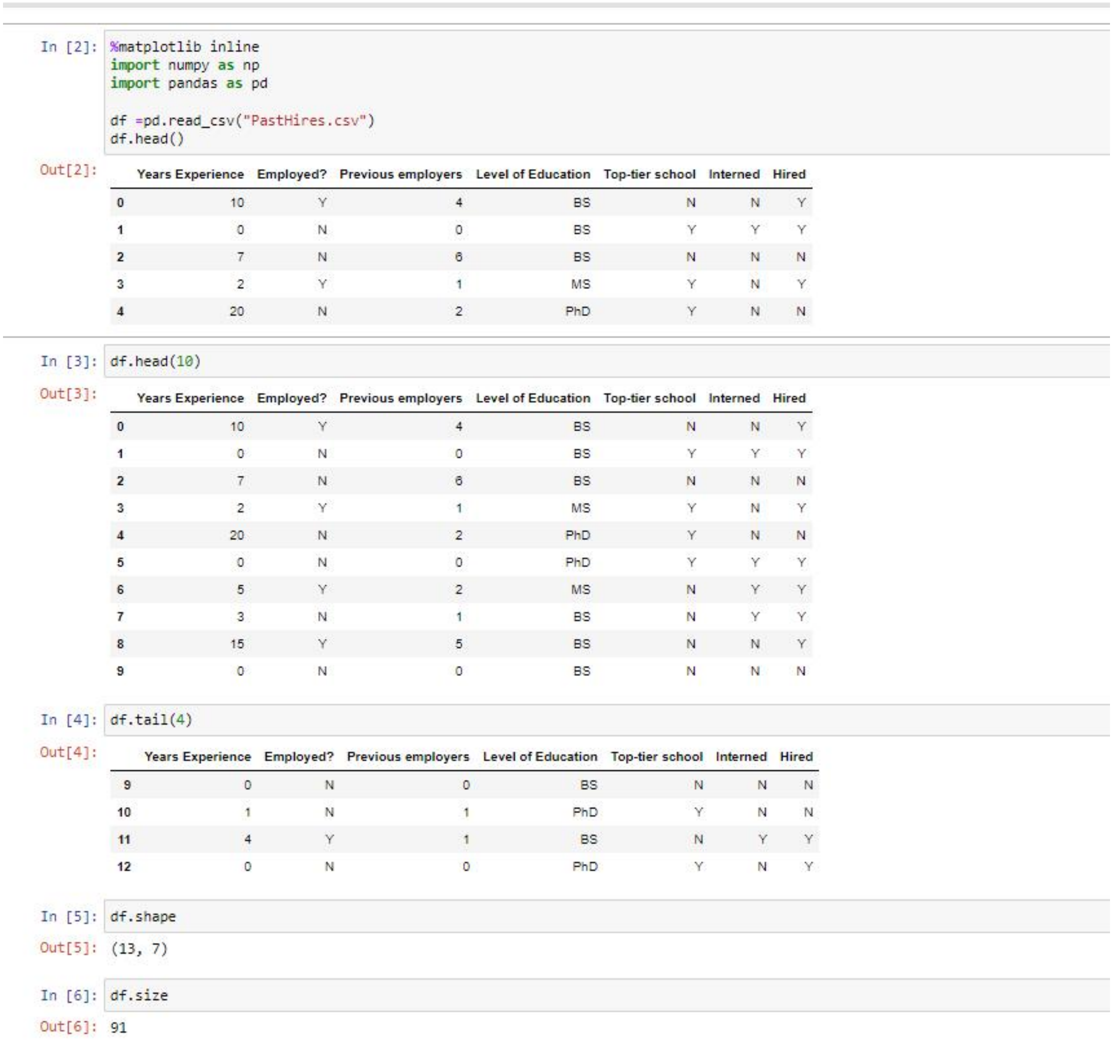
Link Github: paulbboone/DataMining_ThucHanh (github.com)

HOMEWORK LAB 02



In [2]:
```
%matplotlib inline
import numpy as np
import pandas as pd

df =pd.read_csv("PastHires.csv")
df.head()
```

Out[2]:

	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
0	10	Y	4	BS	N	N	Y
1	0	N	0	BS	Y	Y	Y
2	7	N	6	BS	N	N	N
3	2	Y	1	MS	Y	N	Y
4	20	N	2	PhD	Y	N	N

In [3]:
```
df.head(10)
```

Out[3]:

	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
0	10	Y	4	BS	N	N	Y
1	0	N	0	BS	Y	Y	Y
2	7	N	6	BS	N	N	N
3	2	Y	1	MS	Y	N	Y
4	20	N	2	PhD	Y	N	N
5	0	N	0	PhD	Y	Y	Y
6	5	Y	2	MS	N	Y	Y
7	3	N	1	BS	N	Y	Y
8	15	Y	5	BS	N	N	Y
9	0	N	0	BS	N	N	N

In [4]:
```
df.tail(4)
```

Out[4]:

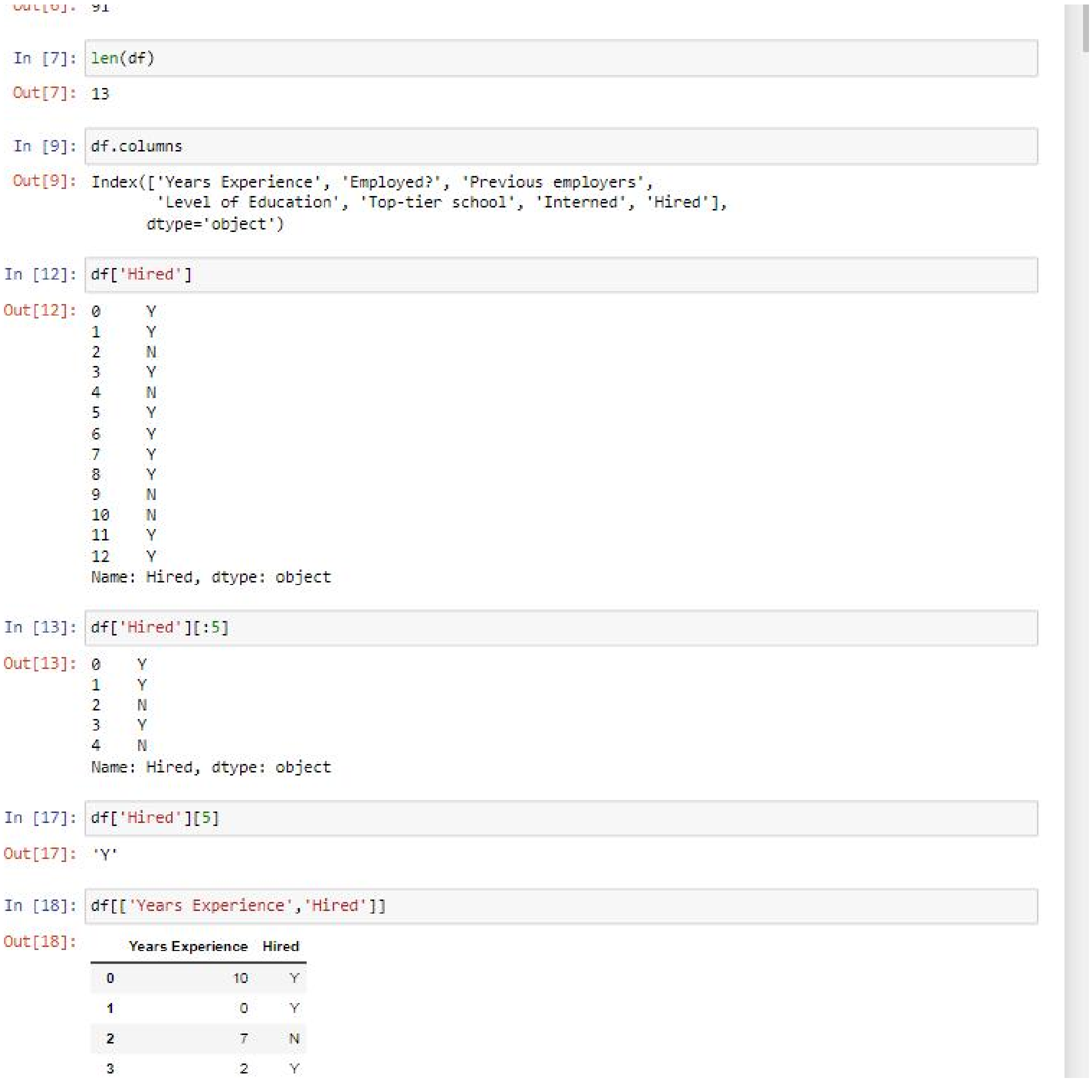
	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
9	0	N	0	BS	N	N	N
10	1	N	1	PhD	Y	N	N
11	4	Y	1	BS	N	Y	Y
12	0	N	0	PhD	Y	N	Y

In [5]:
```
df.shape
```

Out[5]:
```
(13, 7)
```

In [6]:
```
df.size
```

Out[6]:
```
91
```



```
Out[6]: 91
```

```
In [7]: len(df)
```

```
Out[7]: 13
```

```
In [9]: df.columns
```

```
Out[9]: Index(['Years Experience', 'Employed?', 'Previous employers',
               'Level of Education', 'Top-tier school', 'Interned', 'Hired'],
              dtype='object')
```

```
In [12]: df['Hired']
```

```
Out[12]: 0     Y
         1     Y
         2     N
         3     Y
         4     N
         5     Y
         6     Y
         7     Y
         8     Y
         9     N
         10    N
         11    Y
         12    Y
         Name: Hired, dtype: object
```

```
In [13]: df['Hired'][:5]
```

```
Out[13]: 0    Y
         1    Y
         2    N
         3    Y
         4    N
         Name: Hired, dtype: object
```

```
In [17]: df['Hired'][5]
```

```
Out[17]: 'Y'
```

```
In [18]: df[['Years Experience','Hired']]
```

Out[18]:

	Years Experience	Hired
0	10	Y
1	0	Y
2	7	N
3	2	Y

In [19]:
```
df[['Years Experience','Hired']][:5]
```

Out[19]:

	Years Experience	Hired
0	10	Y
1	0	Y
2	7	N
3	2	Y
4	20	N

In [20]:
```
df.sort_values(['Years Experience'])
```

Out[20]:

	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
1	0	N	0	BS	Y	Y	Y
5	0	N	0	PhD	Y	Y	Y
9	0	N	0	BS	N	N	N
12	0	N	0	PhD	Y	N	Y
10	1	N	1	PhD	Y	N	N
3	2	Y	1	MS	Y	N	Y
7	3	N	1	BS	N	Y	Y
11	4	Y	1	BS	N	Y	Y
6	5	Y	2	MS	N	Y	Y
2	7	N	6	BS	N	N	N
0	10	Y	4	BS	N	N	Y
8	15	Y	5	BS	N	N	Y
4	20	N	2	PhD	Y	N	N

In [22]:
```
degree_counts=df['Level of Education'].value_counts()
degree_counts
```

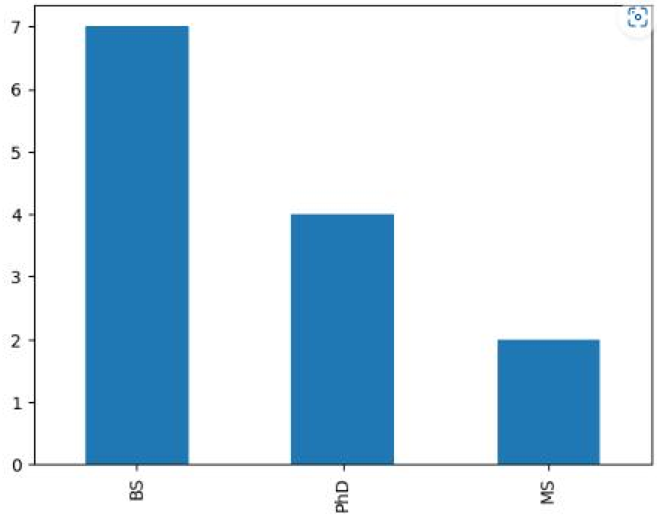
Out[22]:
```
BS     7
PhD    4
MS     2
Name: Level of Education, dtype: int64
```

In [23]:
```
degree_counts.plot(kind='bar')
```

Out[23]:
```
<AxesSubplot:>
```

```
In [23]: degree_counts.plot(kind='bar')
```

```
Out[23]: <AxesSubplot:>
```



```
In [25]: # Series:
```

```
In [26]: labels=['a','b','c']
         my_list=[10,20,30]
         arr=np.array([10,20,30])
         d={'a':10,'b':20,'c':30}
```

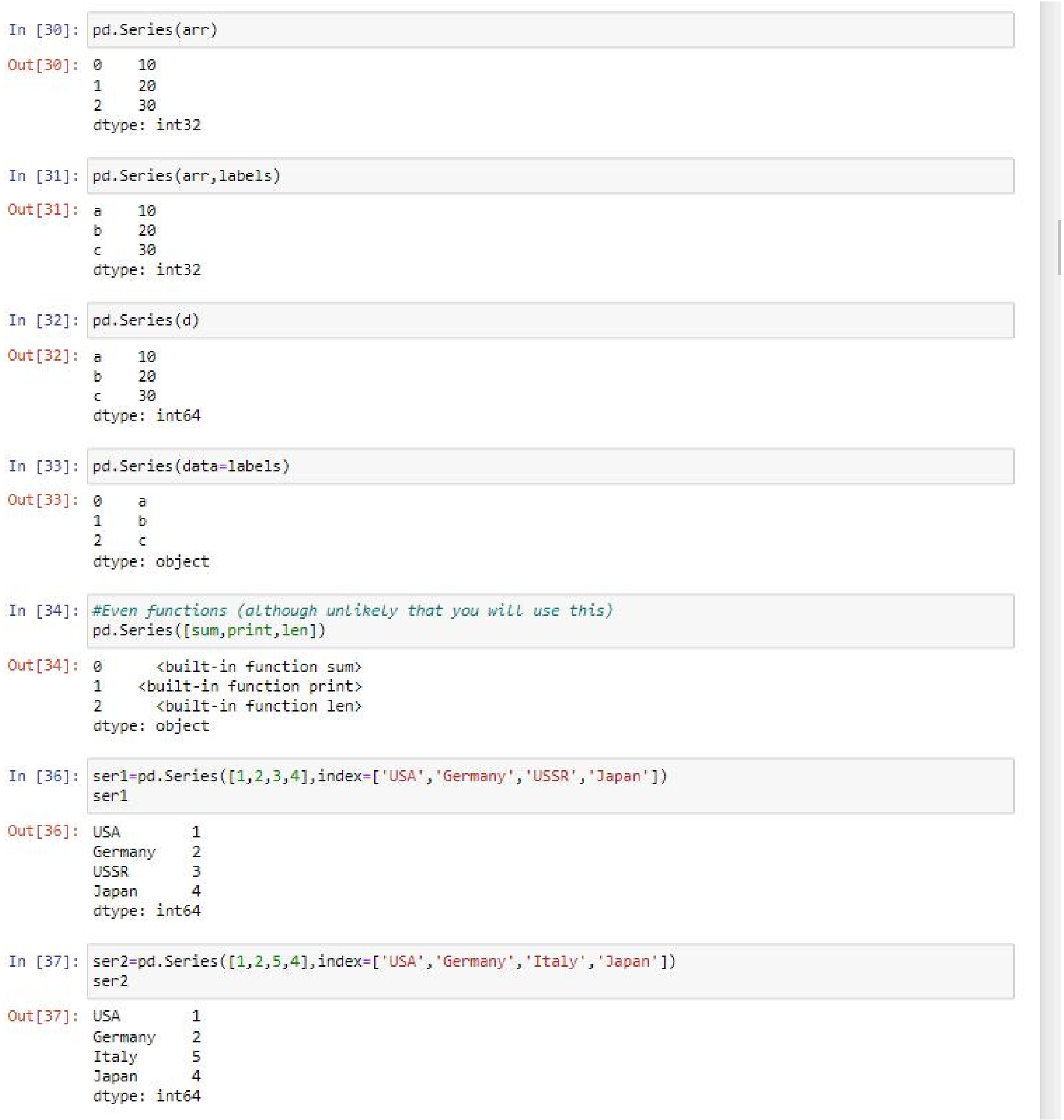
```
In [27]: pd.Series(data=my_list)
```

```
Out[27]: 0    10
         1    20
         2    30
         dtype: int64
```

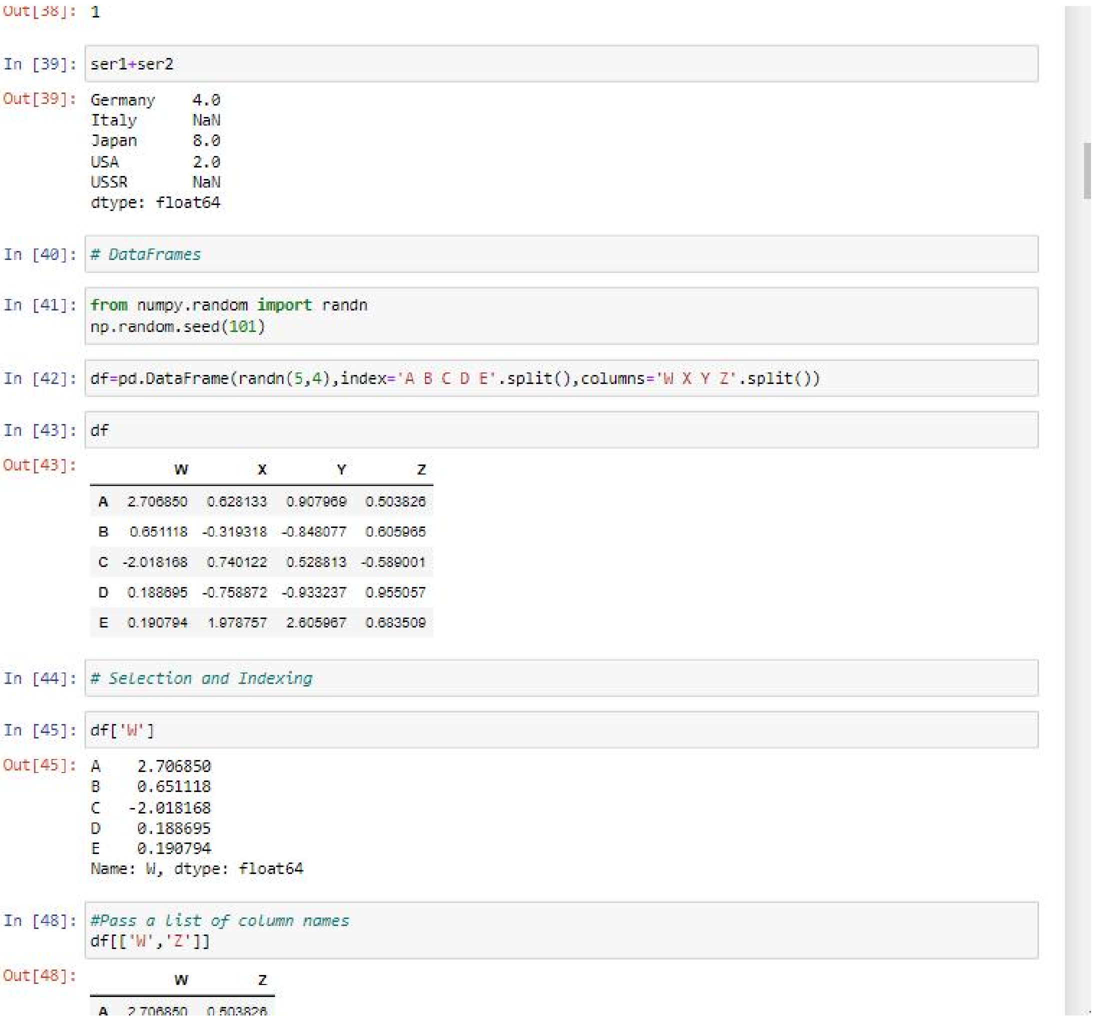
```
In [28]: pd.Series(data=my_list,index=labels)
```

```
Out[28]: a    10
         b    20
         c    30
         dtype: int64
```

```
In [29]: pd.Series(my_list, labels)
```



```
In [30]: pd.Series(arr)

Out[30]: 0    10
         1    20
         2    30
         dtype: int32

In [31]: pd.Series(arr,labels)

Out[31]: a    10
         b    20
         c    30
         dtype: int32

In [32]: pd.Series(d)

Out[32]: a    10
         b    20
         c    30
         dtype: int64

In [33]: pd.Series(data=labels)

Out[33]: 0    a
         1    b
         2    c
         dtype: object

In [34]: #Even functions (although unlikely that you will use this)
         pd.Series([sum,print,len])

Out[34]: 0      <built-in function sum>
         1    <built-in function print>
         2      <built-in function len>
         dtype: object

In [36]: ser1=pd.Series([1,2,3,4],index=['USA','Germany','USSR','Japan'])
         ser1

Out[36]: USA        1
         Germany    2
         USSR       3
         Japan      4
         dtype: int64

In [37]: ser2=pd.Series([1,2,5,4],index=['USA','Germany','Italy','Japan'])
         ser2

Out[37]: USA        1
         Germany    2
         Italy      5
         Japan      4
         dtype: int64
```



```
Out[38]: 1
```

```
In [39]: ser1+ser2
```

```
Out[39]: Germany    4.0
         Italy      NaN
         Japan      8.0
         USA        2.0
         USSR       NaN
         dtype: float64
```

```
In [40]: # DataFrames
```

```
In [41]: from numpy.random import randn
         np.random.seed(101)
```

```
In [42]: df=pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

```
In [43]: df
```

Out[43]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
In [44]: # Selection and Indexing
```

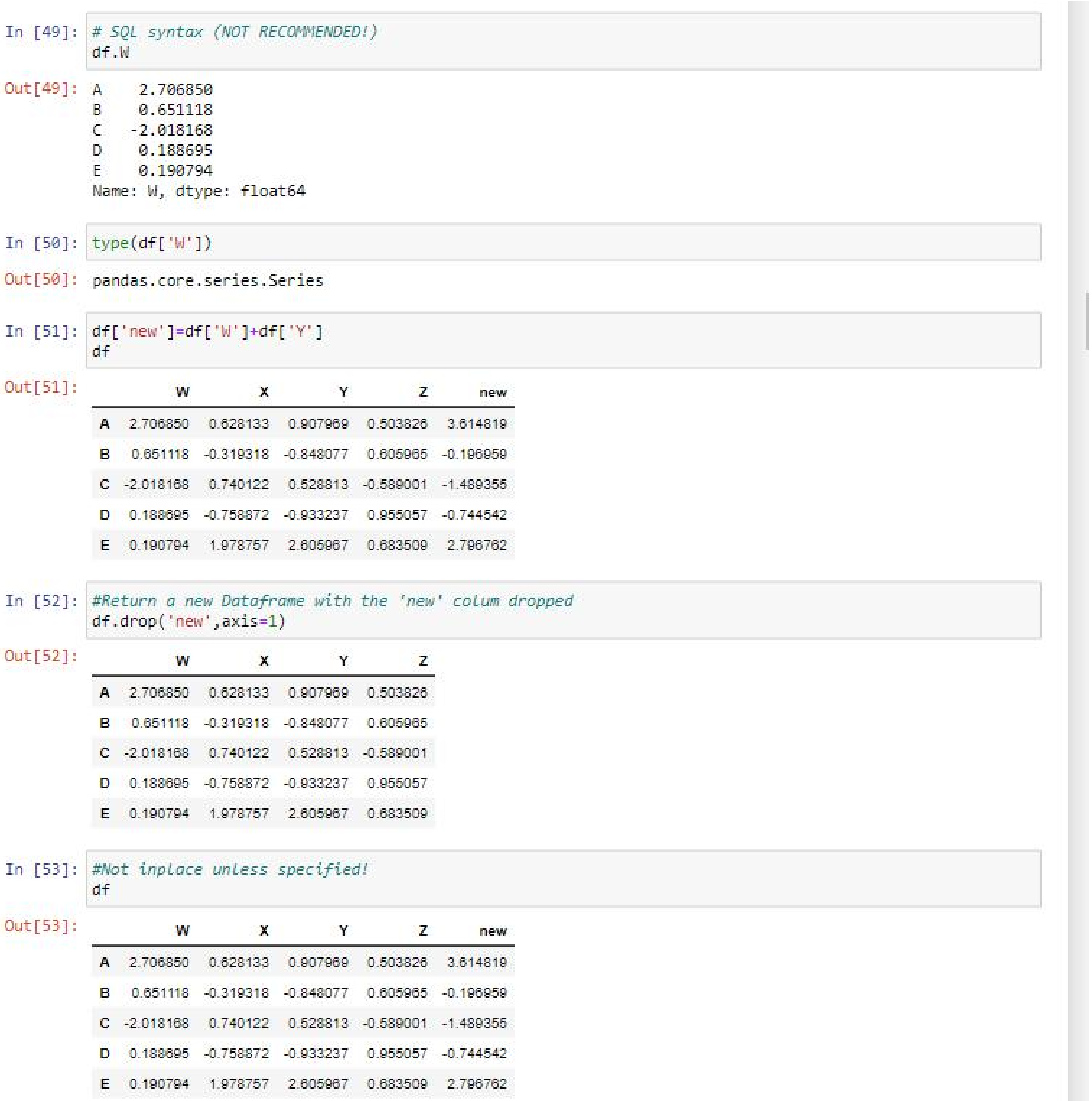
```
In [45]: df['W']
```

```
Out[45]: A    2.706850
         B    0.651118
         C   -2.018168
         D    0.188695
         E    0.190794
         Name: W, dtype: float64
```

```
In [48]: #Pass a list of column names
         df[['W','Z']]
```

Out[48]:

	W	Z
A	2.706850	0.503826



```
In [49]: # SQL syntax (NOT RECOMMENDED!)
         df.W

Out[49]: A    2.706850
         B    0.651118
         C   -2.018168
         D    0.188695
         E    0.190794
         Name: W, dtype: float64
```

```
In [50]: type(df['W'])

Out[50]: pandas.core.series.Series
```

```
In [51]: df['new']=df['W']+df['Y']
         df
```

Out[51]:

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

```
In [52]: #Return a new Dataframe with the 'new' colum dropped
         df.drop('new',axis=1)
```

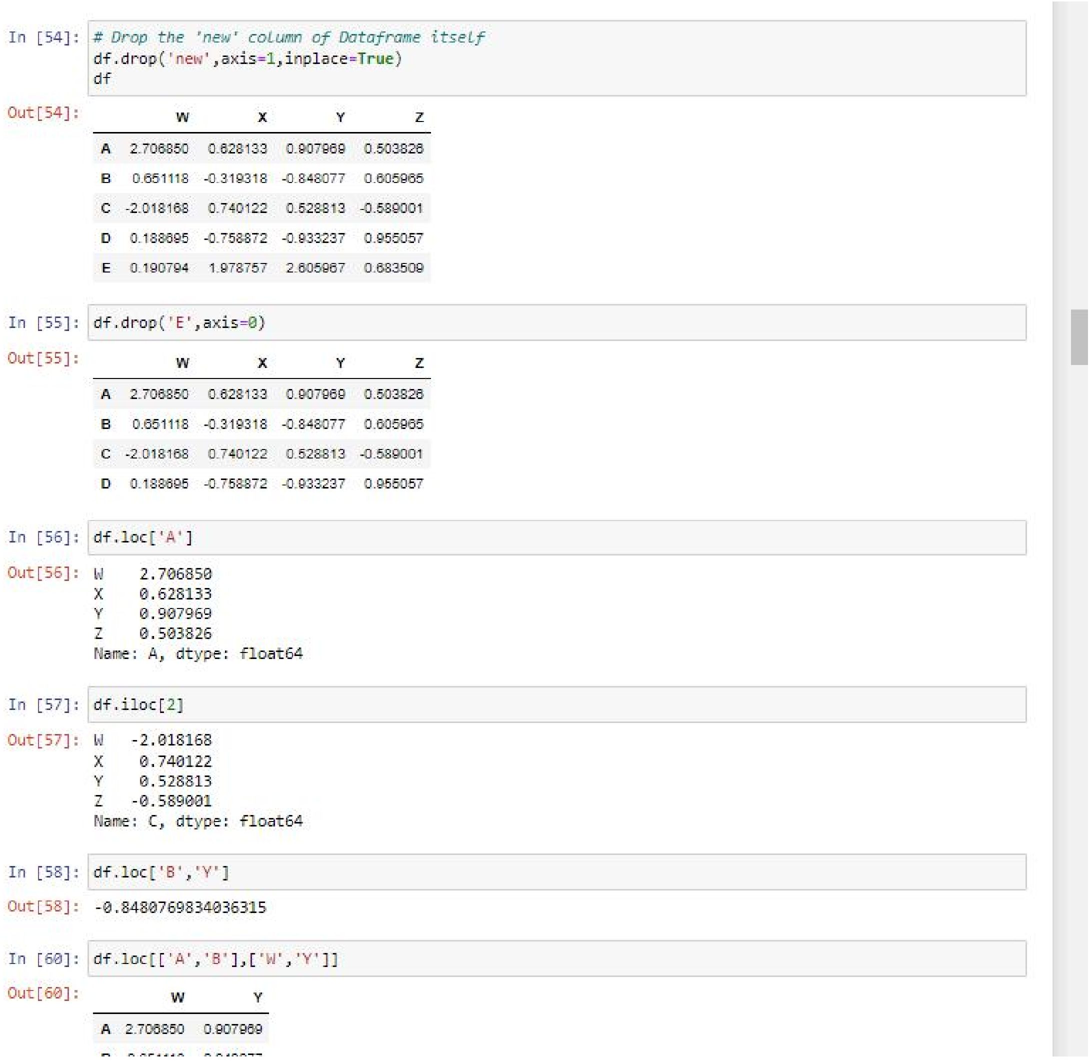
Out[52]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
In [53]: #Not inplace unless specified!
         df
```

Out[53]:

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762



In [54]:
```
# Drop the 'new' column of Dataframe itself
df.drop('new',axis=1,inplace=True)
df
```

Out[54]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

In [55]:
```
df.drop('E',axis=0)
```

Out[55]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057

In [56]:
```
df.loc['A']
```

Out[56]:
```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

In [57]:
```
df.iloc[2]
```

Out[57]:
```
W   -2.018168
X    0.740122
Y    0.528813
Z   -0.589001
Name: C, dtype: float64
```

In [58]:
```
df.loc['B','Y']
```

Out[58]:
```
-0.8480769834036315
```

In [60]:
```
df.loc[['A','B'],['W','Y']]
```

Out[60]:

	W	Y
A	2.706850	0.907969

```
In [60]: df.loc[['A','B'],['W','Y']]
```

Out[60]:

	W	Y
A	2.706850	0.907969
B	0.651118	-0.848077

```
In [61]: # Conditional Selection
         df
```

Out[61]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
In [62]: df>0
```

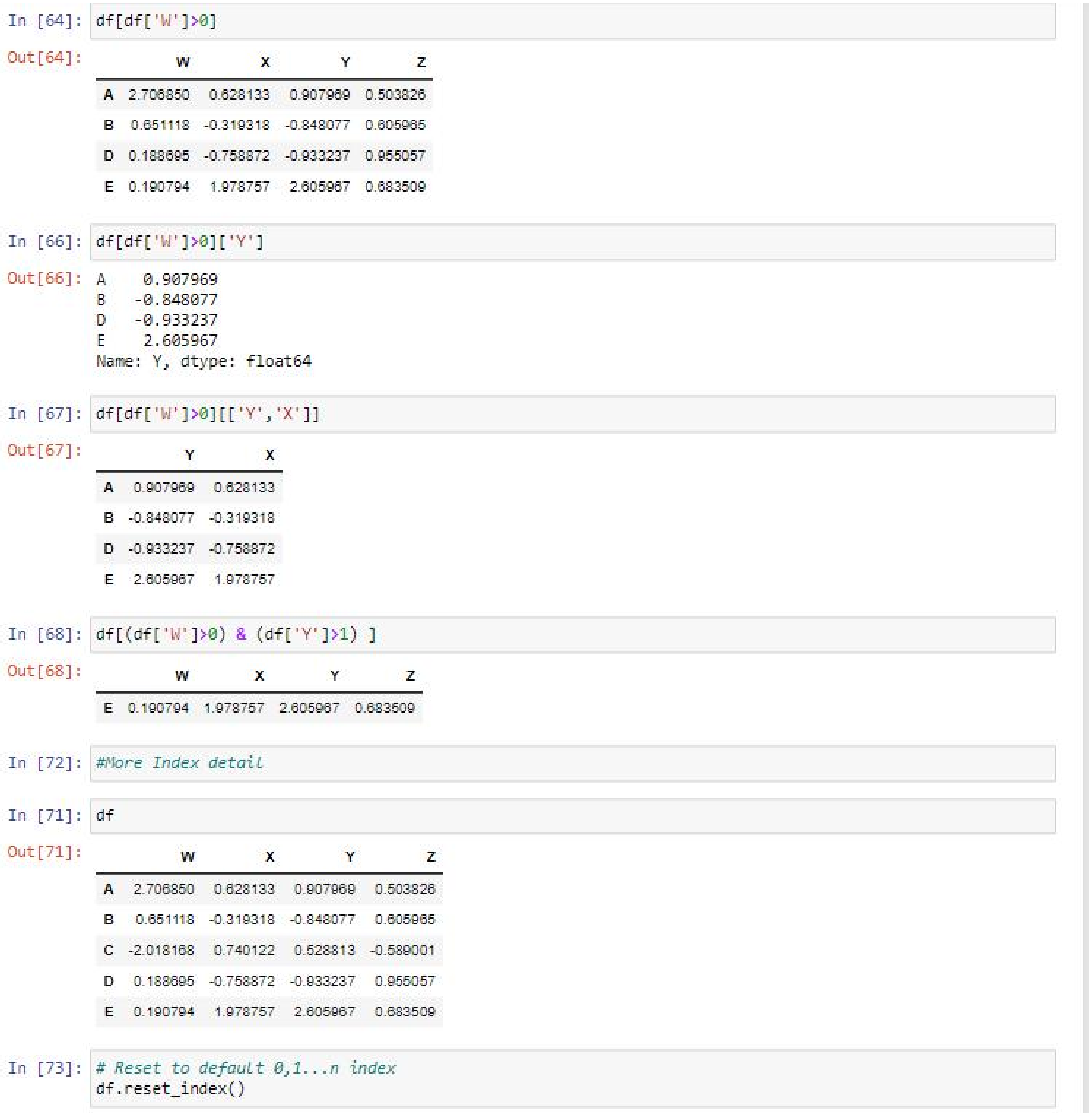
Out[62]:

	W	X	Y	Z
A	True	True	True	True
B	True	False	False	True
C	False	True	True	False
D	True	False	False	True
E	True	True	True	True

```
In [63]: df[df>0]
```

Out[63]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	NaN	NaN	0.605965
C	NaN	0.740122	0.528813	NaN
D	0.188695	NaN	NaN	0.955057
E	0.190794	1.978757	2.605967	0.683509



```
In [64]: df[df['W']>0]
```

Out[64]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
In [66]: df[df['W']>0]['Y']
```

```
Out[66]: A    0.907969
         B   -0.848077
         D   -0.933237
         E    2.605967
         Name: Y, dtype: float64
```

```
In [67]: df[df['W']>0][['Y','X']]
```

Out[67]:

	Y	X
A	0.907969	0.628133
B	-0.848077	-0.319318
D	-0.933237	-0.758872
E	2.605967	1.978757

```
In [68]: df[(df['W']>0) & (df['Y']>1) ]
```

Out[68]:

	W	X	Y	Z
E	0.190794	1.978757	2.605967	0.683509

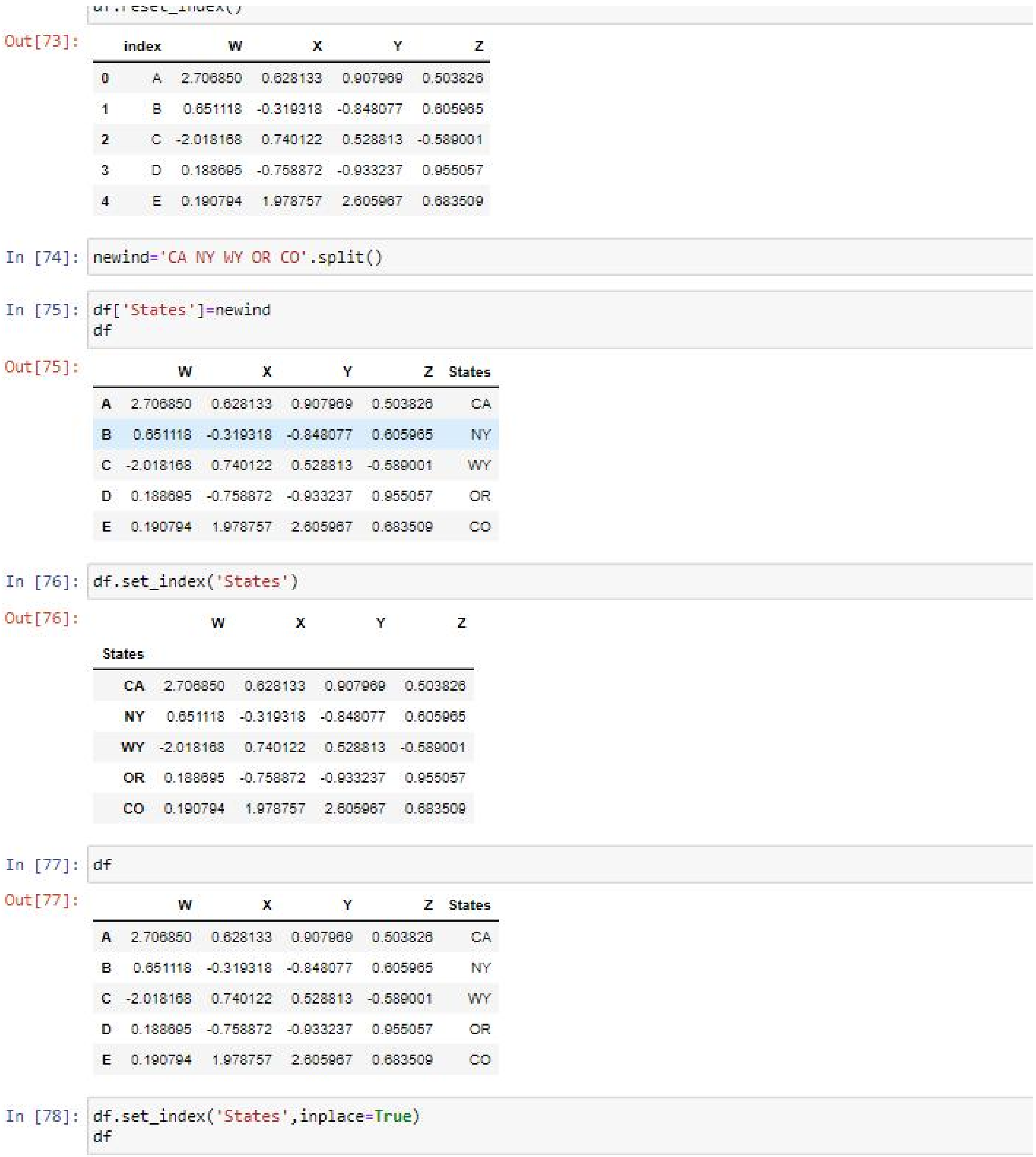
```
In [72]: #More Index detail
```

```
In [71]: df
```

Out[71]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
In [73]: # Reset to default 0,1...n index
         df.reset_index()
```



```
df.reset_index()
```

Out[73]:

	index	W	X	Y	Z
0	A	2.706850	0.628133	0.907969	0.503826
1	B	0.651118	-0.319318	-0.848077	0.605965
2	C	-2.018168	0.740122	0.528813	-0.589001
3	D	0.188695	-0.758872	-0.933237	0.955057
4	E	0.190794	1.978757	2.605967	0.683509

In [74]:

```
newind='CA NY WY OR CO'.split()
```

In [75]:

```
df['States']=newind
df
```

Out[75]:

	W	X	Y	Z	States
A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

In [76]:

```
df.set_index('States')
```

Out[76]:

	W	X	Y	Z
States				
CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

In [77]:

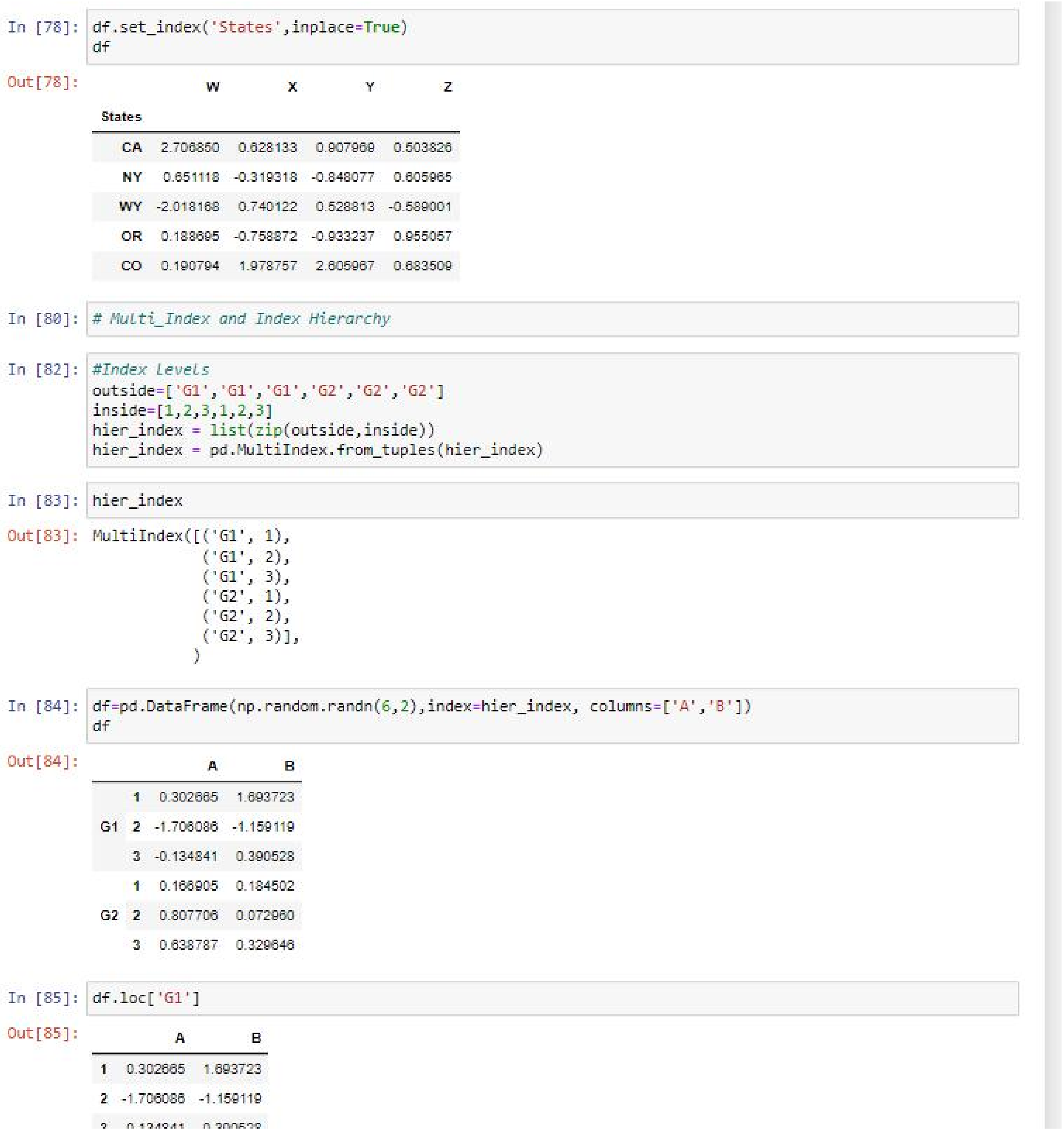
```
df
```

Out[77]:

	W	X	Y	Z	States
A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

In [78]:

```
df.set_index('States',inplace=True)
df
```



In [78]:
```
df.set_index('States',inplace=True)
df
```

Out[78]:

States	W	X	Y	Z
CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

In [80]:
```
# Multi_Index and Index Hierarchy
```

In [82]:
```
#Index Levels
outside=['G1','G1','G1','G2','G2','G2']
inside=[1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

In [83]:
```
hier_index
```

Out[83]:
```
MultiIndex([('G1', 1),
            ('G1', 2),
            ('G1', 3),
            ('G2', 1),
            ('G2', 2),
            ('G2', 3)],
           )
```

In [84]:
```
df=pd.DataFrame(np.random.randn(6,2),index=hier_index, columns=['A','B'])
df
```

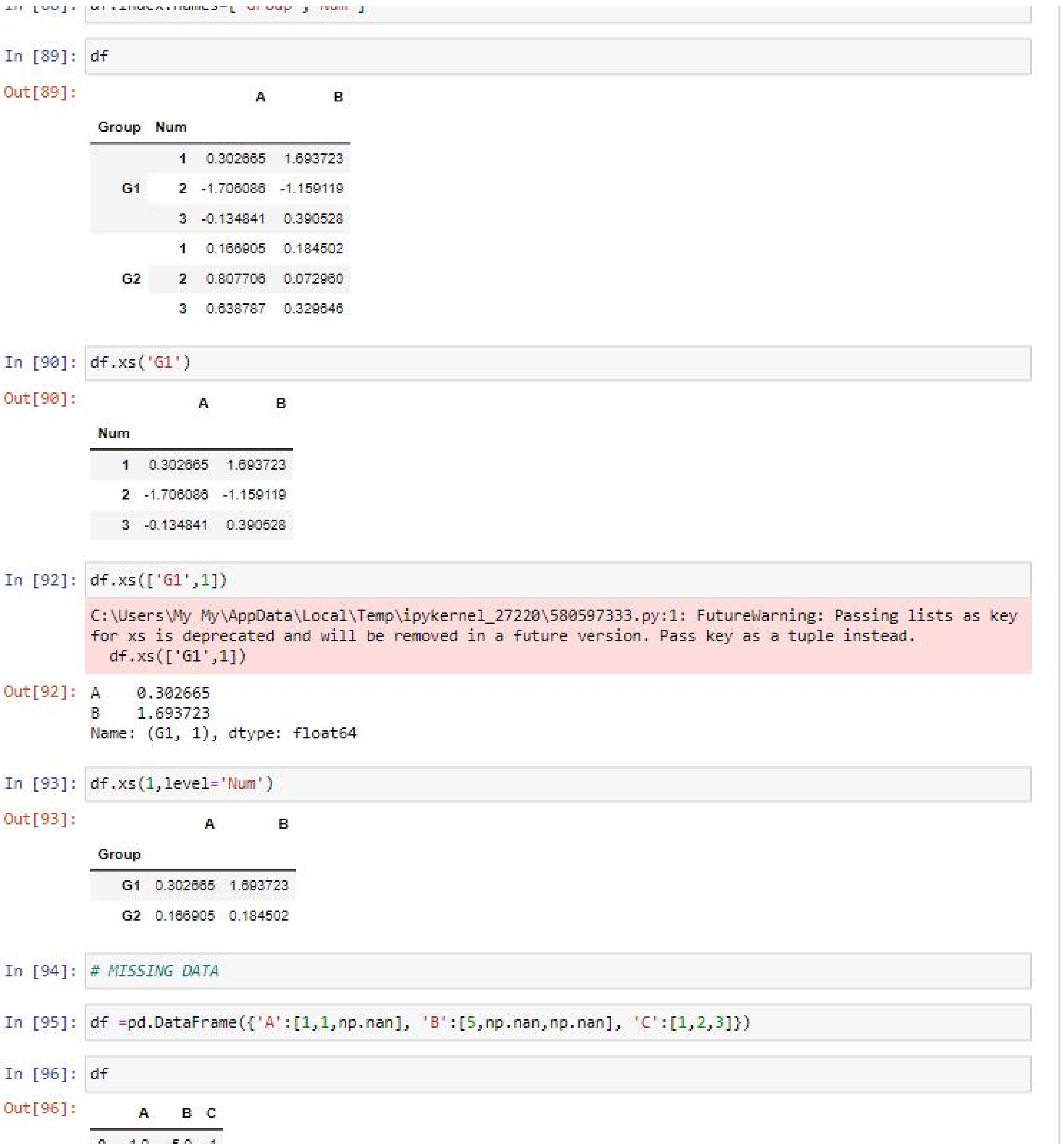
Out[84]:

		A	B
	1	0.302665	1.693723
G1	2	-1.706086	-1.159119
	3	-0.134841	0.390528
	1	0.166905	0.184502
G2	2	0.807706	0.072960
	3	0.638787	0.329646

In [85]:
```
df.loc['G1']
```

Out[85]:

	A	B
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528



```
In [88]: df.index.names=['Group','Num']
```

```
In [89]: df
```

Out[89]:

Group	Num	A	B
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

```
In [90]: df.xs('G1')
```

Out[90]:

Num	A	B
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528

```
In [92]: df.xs(['G1',1])
```

```
C:\Users\My My\AppData\Local\Temp\ipykernel_27220\580597333.py:1: FutureWarning: Passing lists as key
for xs is deprecated and will be removed in a future version. Pass key as a tuple instead.
  df.xs(['G1',1])
```

```
Out[92]: A    0.302665
         B    1.693723
         Name: (G1, 1), dtype: float64
```

```
In [93]: df.xs(1,level='Num')
```

Out[93]:

Group	A	B
G1	0.302665	1.693723
G2	0.166905	0.184502

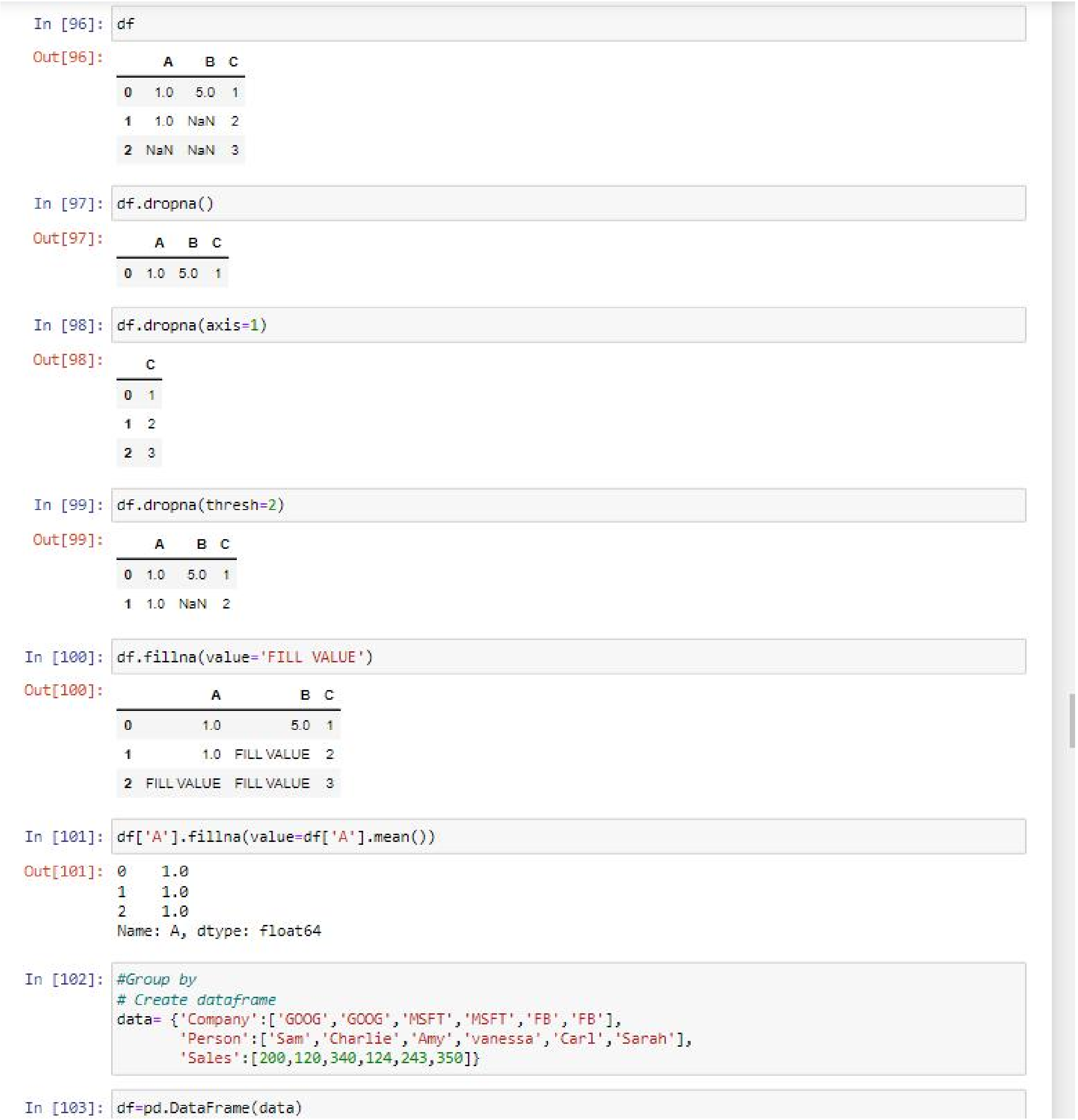
```
In [94]: # MISSING DATA
```

```
In [95]: df =pd.DataFrame({'A':[1,1,np.nan], 'B':[5,np.nan,np.nan], 'C':[1,2,3]})
```

```
In [96]: df
```

Out[96]:

	A	B	C
0	1.0	5.0	1



In [96]:
```
df
```

Out[96]:

	A	B	C
0	1.0	5.0	1
1	1.0	NaN	2
2	NaN	NaN	3

In [97]:
```
df.dropna()
```

Out[97]:

	A	B	C
0	1.0	5.0	1

In [98]:
```
df.dropna(axis=1)
```

Out[98]:

	C
0	1
1	2
2	3

In [99]:
```
df.dropna(thresh=2)
```

Out[99]:

	A	B	C
0	1.0	5.0	1
1	1.0	NaN	2

In [100]:
```
df.fillna(value='FILL VALUE')
```

Out[100]:

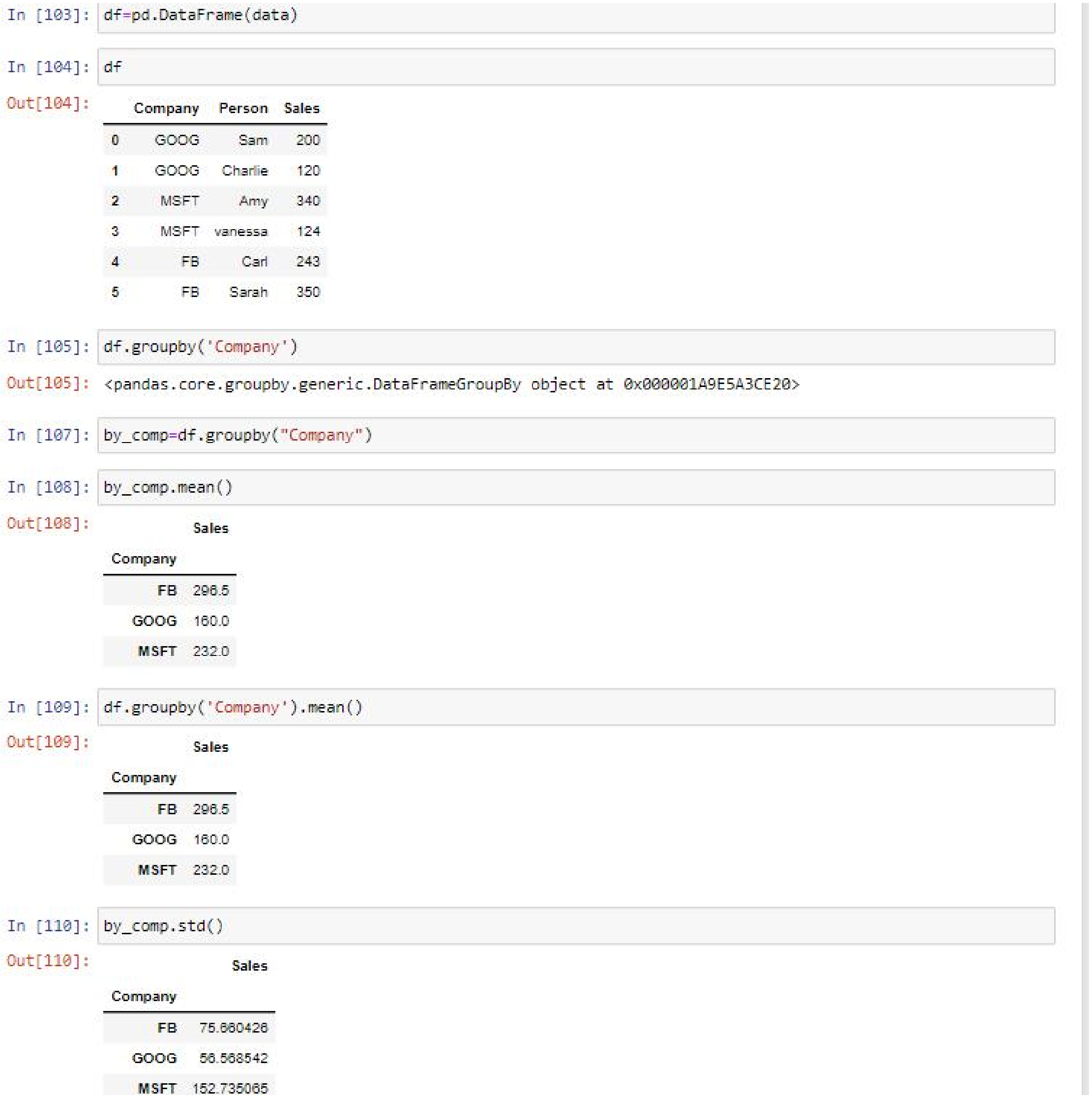
	A	B	C
0	1.0	5.0	1
1	1.0	FILL VALUE	2
2	FILL VALUE	FILL VALUE	3

In [101]:
```
df['A'].fillna(value=df['A'].mean())
```

Out[101]:
```
0    1.0
1    1.0
2    1.0
Name: A, dtype: float64
```

In [102]:
```
#Group by
# Create dataframe
data= {'Company':['GOOG','GOOG','MSFT','MSFT','FB','FB'],
       'Person':['Sam','Charlie','Amy','vanessa','Carl','Sarah'],
       'Sales':[200,120,340,124,243,350]}
```

In [103]:
```
df=pd.DataFrame(data)
```



In [103]:
```
df=pd.DataFrame(data)
```

In [104]:
```
df
```

Out[104]:

	Company	Person	Sales
0	GOOG	Sam	200
1	GOOG	Charlie	120
2	MSFT	Amy	340
3	MSFT	vanessa	124
4	FB	Carl	243
5	FB	Sarah	350

In [105]:
```
df.groupby('Company')
```

Out[105]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A9E5A3CE20>
```

In [107]:
```
by_comp=df.groupby("Company")
```

In [108]:
```
by_comp.mean()
```

Out[108]:

	Sales
Company	
FB	296.5
GOOG	160.0
MSFT	232.0

In [109]:
```
df.groupby('Company').mean()
```

Out[109]:

	Sales
Company	
FB	296.5
GOOG	160.0
MSFT	232.0

In [110]:
```
by_comp.std()
```

Out[110]:

	Sales
Company	
FB	75.660426
GOOG	56.568542
MSFT	152.735065

```
In [111]: by_comp.min()
```

```
Out[111]:
                 Person  Sales
         Company
              FB   Carl    243
            GOOG Charlie   120
            MSFT    Amy    124
```

```
In [112]: by_comp.max()
```

```
Out[112]:
                 Person  Sales
         Company
              FB   Sarah   350
            GOOG     Sam   200
            MSFT vanessa   340
```

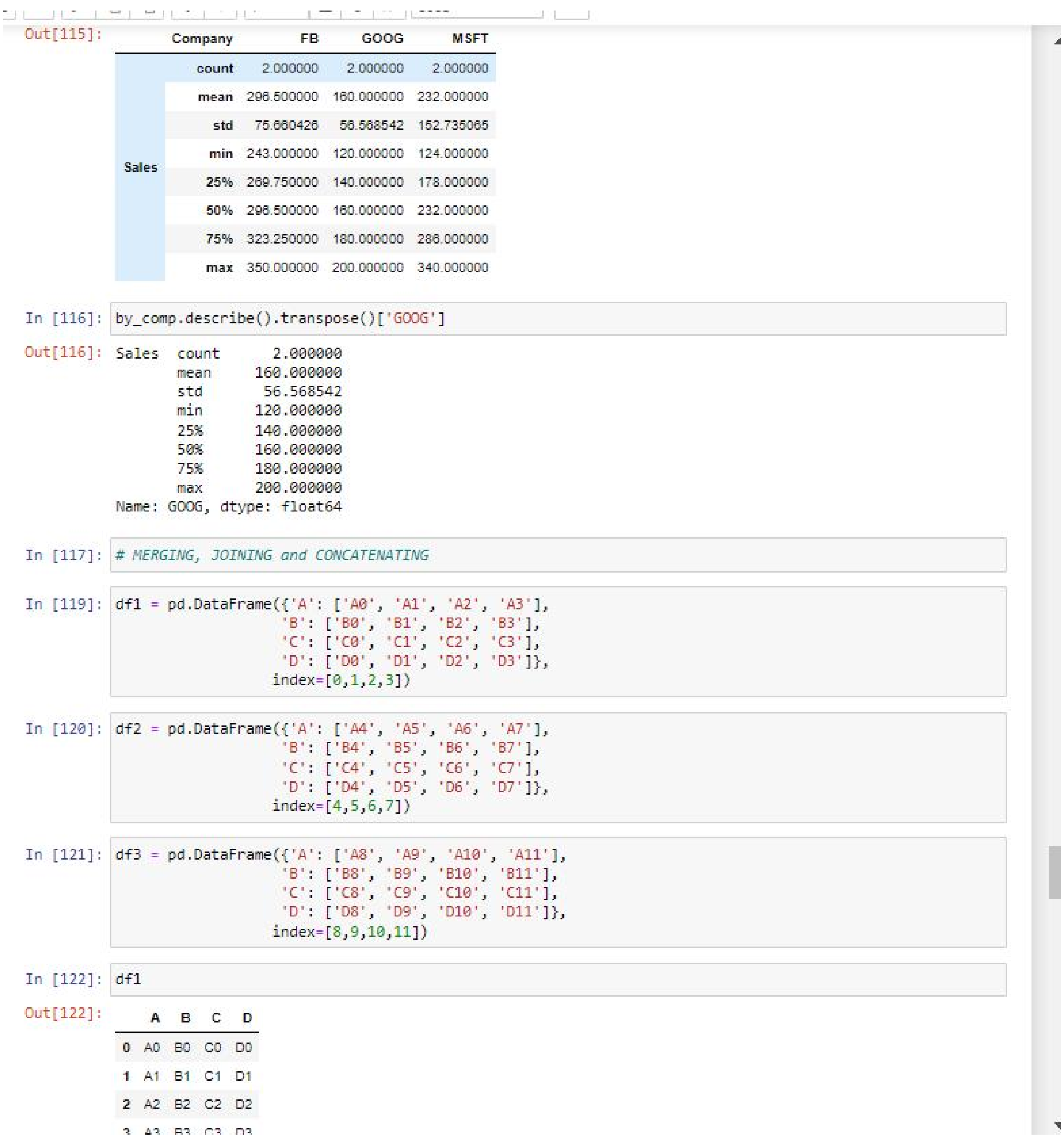
```
In [113]: by_comp.count()
```

```
Out[113]:
                 Person  Sales
         Company
              FB      2      2
            GOOG      2      2
            MSFT      2      2
```

```
In [114]: by_comp.describe()
```

```
Out[114]:
                                                                 Sales
                 count  mean         std    min     25%    50%     75%    max
         Company
              FB   2.0 296.5   75.660426  243.0  269.75  296.5  323.25  350.0
            GOOG   2.0 160.0   56.568542  120.0  140.00  160.0  180.00  200.0
            MSFT   2.0 232.0  152.735065  124.0  178.00  232.0  286.00  340.0
```

```
In [115]: by_comp.describe().transpose()
```



Out[115]:

	Company	FB	GOOG	MSFT
Sales	count	2.000000	2.000000	2.000000
	mean	296.500000	160.000000	232.000000
	std	75.660426	56.568542	152.735065
	min	243.000000	120.000000	124.000000
	25%	269.750000	140.000000	178.000000
	50%	296.500000	160.000000	232.000000
	75%	323.250000	180.000000	286.000000
	max	350.000000	200.000000	340.000000

In [116]:
```
by_comp.describe().transpose()['GOOG']
```

Out[116]:
```
Sales  count      2.000000
       mean     160.000000
       std       56.568542
       min      120.000000
       25%      140.000000
       50%      160.000000
       75%      180.000000
       max      200.000000
Name: GOOG, dtype: float64
```

In [117]:
```
# MERGING, JOINING and CONCATENATING
```

In [119]:
```
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])
```

In [120]:
```
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])
```

In [121]:
```
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])
```

In [122]:
```
df1
```

Out[122]:

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

```
In [123]: df2
```

```
Out[123]:
     A   B   C   D
4   A4  B4  C4  D4
5   A5  B5  C5  D5
6   A6  B6  C6  D6
7   A7  B7  C7  D7
```

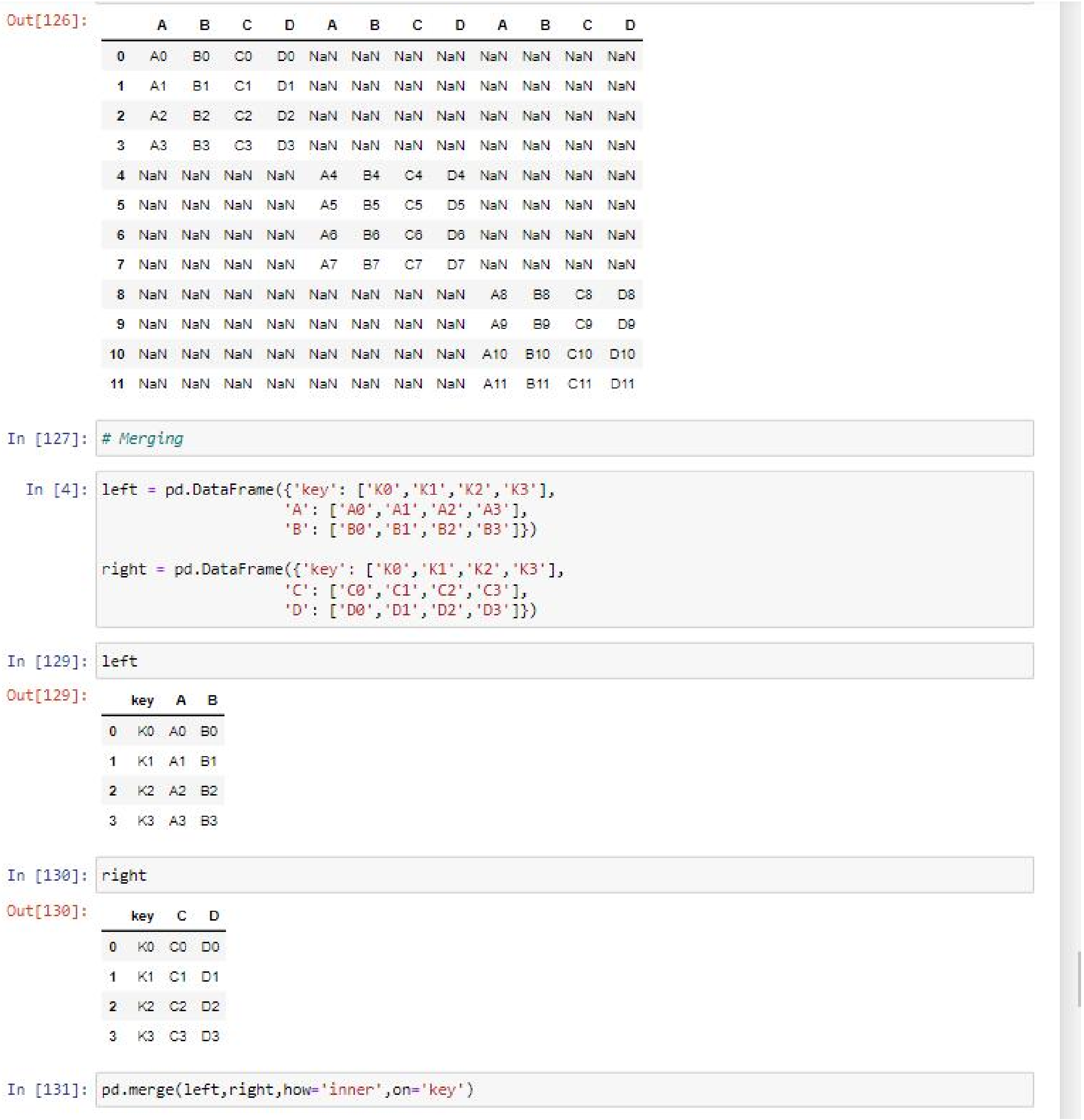
```
In [124]: df3
```

```
Out[124]:
      A    B    C    D
8    A8   B8   C8   D8
9    A9   B9   C9   D9
10  A10  B10  C10  D10
11  A11  B11  C11  D11
```

```
In [125]: pd.concat([df1,df2,df3])
```

```
Out[125]:
      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
```

```
In [126]: pd.concat([df1,df2,df3], axis=1)
```



Out[126]:

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

In [127]:

```
# Merging
```

In [4]:

```
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']})
```

In [129]:

```
left
```

Out[129]:

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

In [130]:

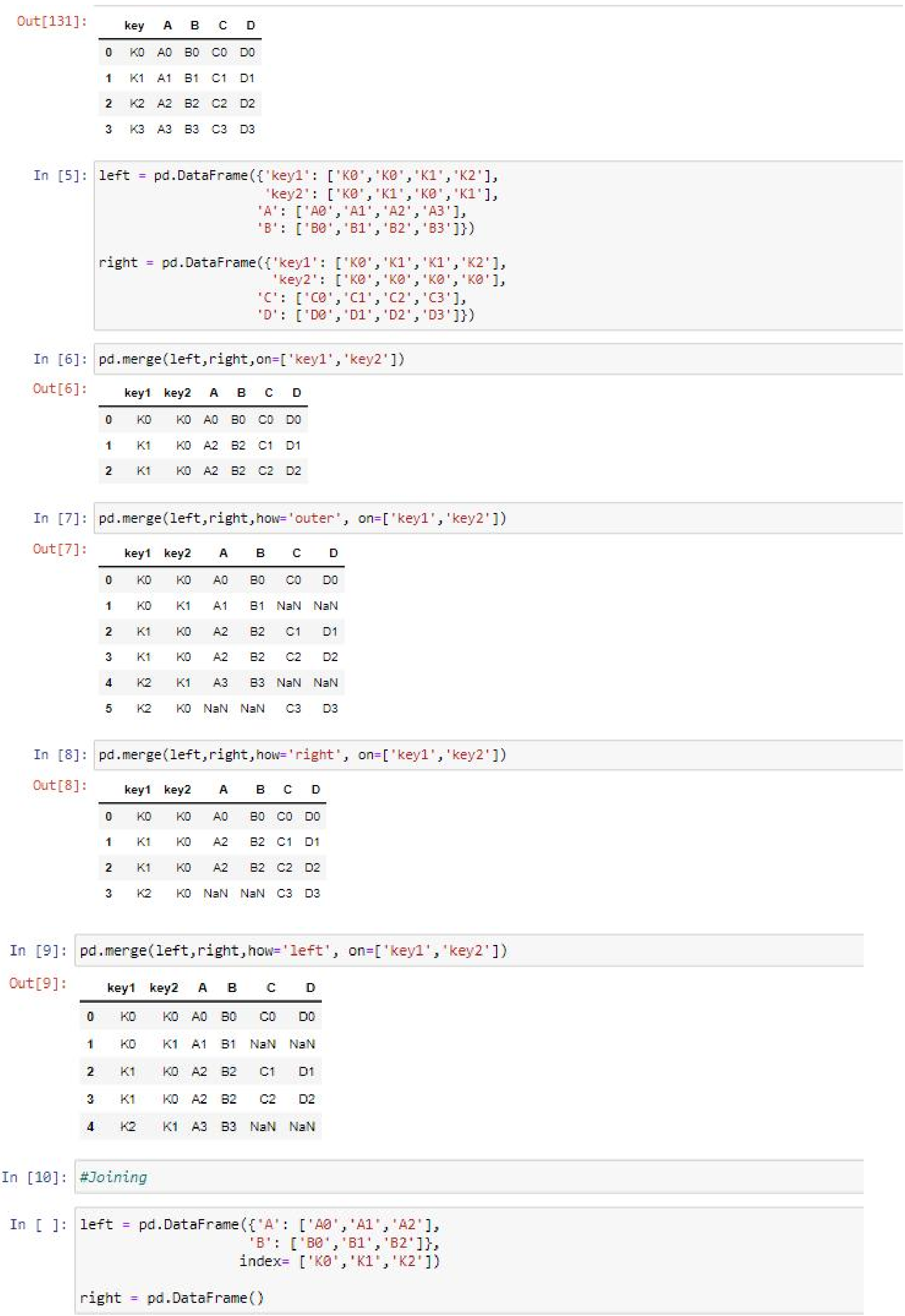
```
right
```

Out[130]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

In [131]:

```
pd.merge(left,right,how='inner',on='key')
```



```
Out[131]:
     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
```

```
In [5]: 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']})
```

```
In [6]: pd.merge(left,right,on=['key1','key2'])
```

```
Out[6]:
     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
```

```
In [7]: pd.merge(left,right,how='outer', on=['key1','key2'])
```

```
Out[7]:
     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
  5  K2    K0    NaN  NaN  C3   D3
```

```
In [8]: pd.merge(left,right,how='right', on=['key1','key2'])
```

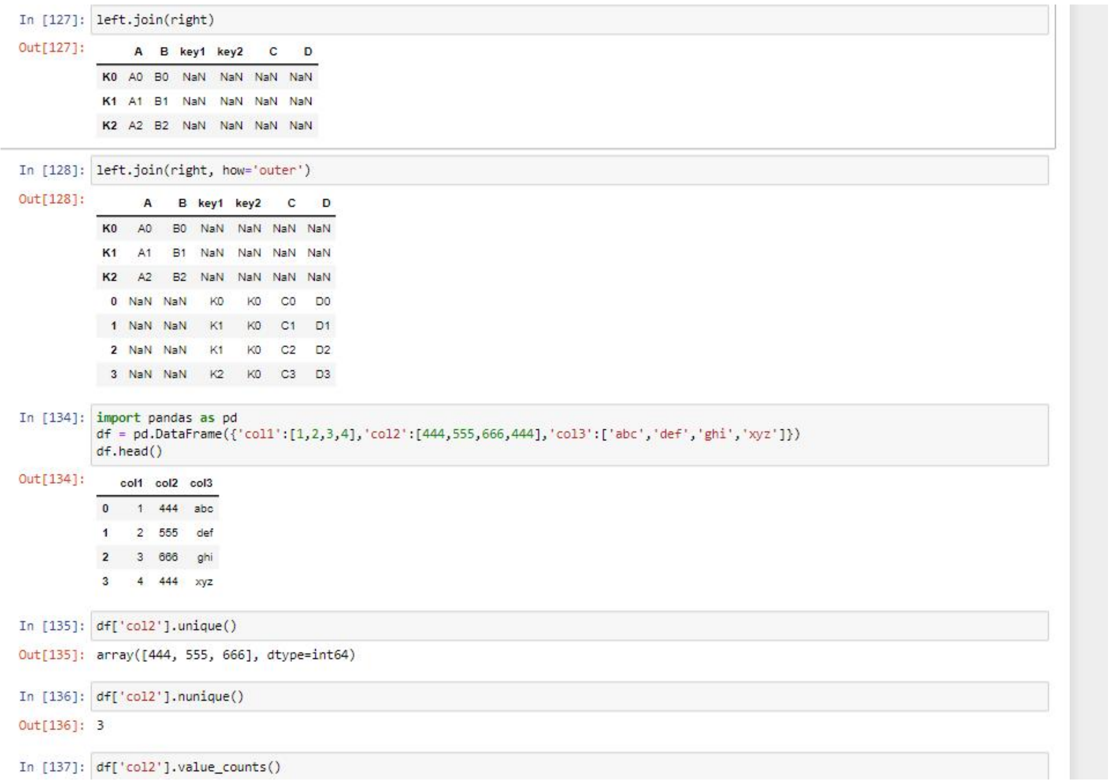
```
Out[8]:
     key1  key2  A    B    C   D
  0  K0    K0    A0   B0   C0  D0
  1  K1    K0    A2   B2   C1  D1
  2  K1    K0    A2   B2   C2  D2
  3  K2    K0    NaN  NaN  C3  D3
```

```
In [9]: pd.merge(left,right,how='left', on=['key1','key2'])
```

```
Out[9]:
     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
```

```
In [10]: #Joining
```

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

        right = pd.DataFrame()
```



In [127]:
```
left.join(right)
```

Out[127]:

	A	B	key1	key2	C	D
K0	A0	B0	NaN	NaN	NaN	NaN
K1	A1	B1	NaN	NaN	NaN	NaN
K2	A2	B2	NaN	NaN	NaN	NaN

In [128]:
```
left.join(right, how='outer')
```

Out[128]:

	A	B	key1	key2	C	D
K0	A0	B0	NaN	NaN	NaN	NaN
K1	A1	B1	NaN	NaN	NaN	NaN
K2	A2	B2	NaN	NaN	NaN	NaN
0	NaN	NaN	K0	K0	C0	D0
1	NaN	NaN	K1	K0	C1	D1
2	NaN	NaN	K1	K0	C2	D2
3	NaN	NaN	K2	K0	C3	D3

In [134]:
```
import pandas as pd
df = pd.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,444],'col3':['abc','def','ghi','xyz']})
df.head()
```

Out[134]:

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

In [135]:
```
df['col2'].unique()
```

Out[135]:
```
array([444, 555, 666], dtype=int64)
```

In [136]:
```
df['col2'].nunique()
```

Out[136]:
```
3
```

In [137]:
```
df['col2'].value_counts()
```

Out[137]:
```
444    2
555    1
666    1
Name: col2, dtype: int64
```

In [142]:
```
newdf = df[(df['col1']>2) & (df['col2']==444)]
```

In [143]:
```
newdf
```

Out[143]:

	col1	col2	col3
3	4	444	xyz

In [145]:
```
def times2(x):
    return x*2
```

In [146]:
```
df['col1'].apply(times2)
```

Out[146]:
```
0    2
1    4
2    6
3    8
Name: col1, dtype: int64
```

In [147]:
```
df['col3'].apply(len)
```

Out[147]:
```
0    3
1    3
2    3
3    3
Name: col3, dtype: int64
```

In [148]:
```
df['col1'].sum()
```

Out[148]:
```
10
```

In [149]:
```
del df['col1']
```

In [150]:
```
df
```

Out[150]:

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
In [151]: df.columns

Out[151]: Index(['col2', 'col3'], dtype='object')
```

```
In [152]: df.index

Out[152]: RangeIndex(start=0, stop=4, step=1)
```

```
In [153]: df

Out[153]:
             col2  col3
          0   444   abc
          1   555   def
          2   666   ghi
          3   444   xyz
```

```
In [154]: df.sort_values(by='col2')

Out[154]:
             col2  col3
          0   444   abc
          3   444   xyz
          1   555   def
          2   666   ghi
```

```
In [155]: df.isnull()

Out[155]:
              col2   col3
          0  False  False
          1  False  False
          2  False  False
          3  False  False
```

```
In [156]: df.dropna()

Out[156]:
             col2  col3
          0   444   abc
          1   555   def
          2   666   ghi
          3   444   xyz
```

In [158]:
```
df = pd.DataFrame({'col1':[1,2,3,np.nan],
                   'col2':[np.nan,555,666,444],
                   'col3':['abc','def','ghi','xyz']})
df.head()
```

Out[158]:

	col1	col2	col3
0	1.0	NaN	abc
1	2.0	555.0	def
2	3.0	666.0	ghi
3	NaN	444.0	xyz

In [159]:
```
df.isnull()
```

Out[159]:

	col1	col2	col3
0	False	True	False
1	False	False	False
2	False	False	False
3	True	False	False

In [160]:
```
df.dropna()
```

Out[160]:

	col1	col2	col3
1	2.0	555.0	def
2	3.0	666.0	ghi

In [161]:
```
df.fillna('FILL')
```

Out[161]:

	col1	col2	col3
0	1.0	FILL	abc
1	2.0	555.0	def
2	3.0	666.0	ghi
3	FILL	444.0	xyz

In [164]:
```
data = {'A':['foo','foo','foo','bar','bar','bar'],
       'B':['one','one','two','two','one','one'],
       'C':['x','y','x','y','x','y'],
       'D':[1,3,2,5,4,1]}
df = pd.DataFrame(data)
```

```
In [165]: df
```

Out[165]:

	A	B	C	D
0	foo	one	x	1
1	foo	one	y	3
2	foo	two	x	2
3	bar	two	y	5
4	bar	one	x	4
5	bar	one	y	1

```
In [202]: df.pivot_table(values='D',index=['A','B'],columns=['C'])
```

Out[202]:

	C	x	y
A	**B**		
bar	**one**	4.0	1.0
	two	NaN	5.0
foo	**one**	1.0	3.0
	two	2.0	NaN

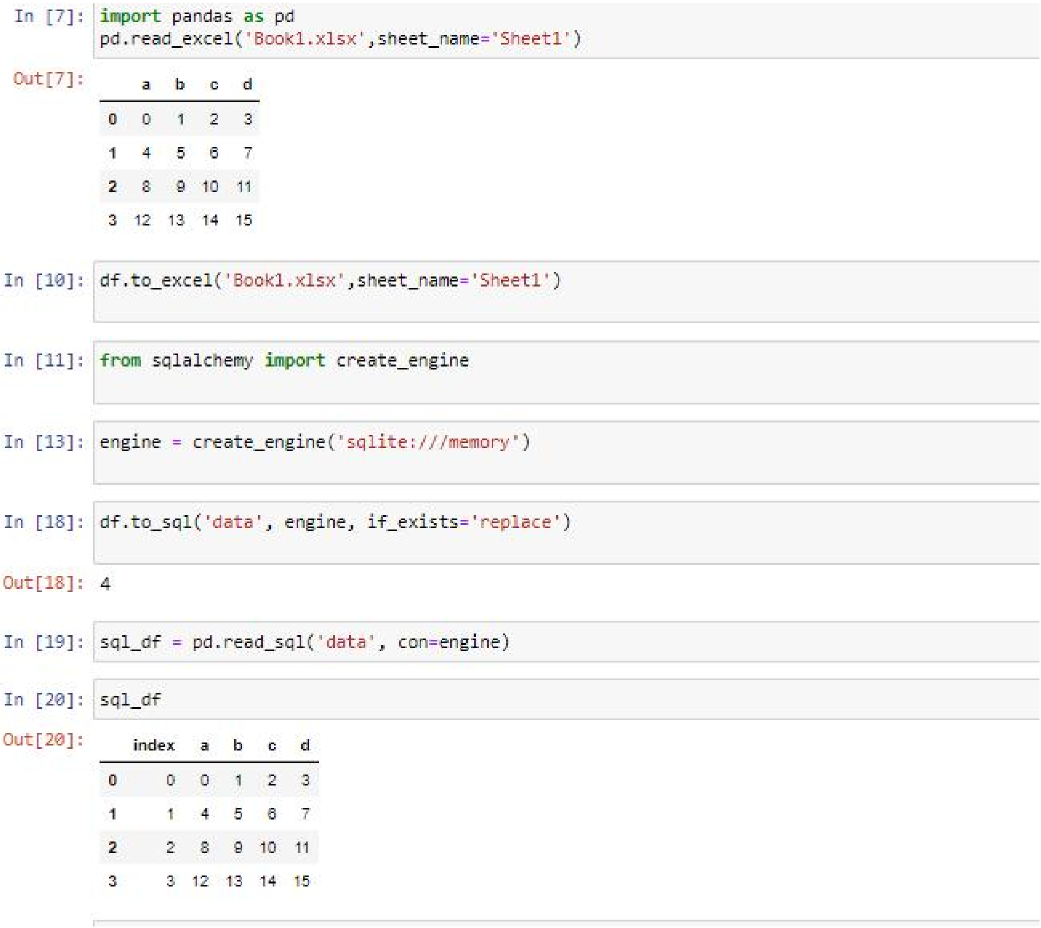
```
In [204]: import numpy as np
          import pandas as pd
```

```
In [205]: df = pd.read_csv('example.csv')
          df
```

Out[205]:

	A	B	C	D
0	foo	one	x	1
1	foo	one	y	3
2	foo	two	x	2
3	bar	two	y	5
4	bar	one	x	4
5	bar	one	y	1

```
In [190]: df.to_csv('example.csv',index=False)
```



In [7]:
```
import pandas as pd
pd.read_excel('Book1.xlsx',sheet_name='Sheet1')
```

Out[7]:

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

In [10]:
```
df.to_excel('Book1.xlsx',sheet_name='Sheet1')
```

In [11]:
```
from sqlalchemy import create_engine
```

In [13]:
```
engine = create_engine('sqlite:///memory')
```

In [18]:
```
df.to_sql('data', engine, if_exists='replace')
```

Out[18]:
```
4
```

In [19]:
```
sql_df = pd.read_sql('data', con=engine)
```

In [20]:
```
sql_df
```

Out[20]:

	index	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15