Name: Nguyễn Thanh Lộc

ID: 19521763

Link github: https://github.com/talo33/Data-Mining-Lab2

Lab 2

[9]:
```
%matplotlib inline
import numpy as np
import pandas as pd

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

[9]:

	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

[ ]:
```

```

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

[10]:

	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

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

[10]:

	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

[ ]:
```

```

[11]:
```
df.tail(4)
```

[11]:

	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]:

	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

[ ]:

[12]:
```
df.shape
```

[12]:
```
(13, 7)
```

[ ]:

[13]:
```
df.size
```

[13]:
```
91
```

[ ]:

[14]:
```
len(df)
```

[14]:
```
13
```

[ ]:

[15]:
```
df.columns
```

[15]:
```
df.columns
```

[15]:
```
Index(['Years Experience', 'Employed?', 'Previous employers',
       'Level of Education', 'Top-tier school', 'Interned', 'Hired'],
      dtype='object')
```

[ ]:

[16]:
```
df['Hired']
```

[16]:
```
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
```

[ ]:

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

[17]:
```
0    Y
1    Y
```

[ ]:

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

[17]:
```
0    Y
1    Y
2    N
3    Y
4    N
Name: Hired, dtype: object
```

[ ]:

[18]:
```
df['Hired'][5]
```

[18]:
```
'Y'
```

[ ]:

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

[19]:

	Years Experience	Hired
0	10	Y
1	0	Y
2	7	N
3	2	Y
4	20	N
5	0	Y
6	5	Y

4	20	N
5	0	Y
6	5	Y
7	3	Y
8	15	Y
9	0	N
10	1	N
11	4	Y
12	0	Y

[ ]:

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

[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

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

[ ]:

[21]:
```
degree_counts = df['Level of Education'].value_counts()
degree_counts
```

[21]:
```
BS     7
PhD    4
MS     2
Name: Level of Education, dtype: int64
```

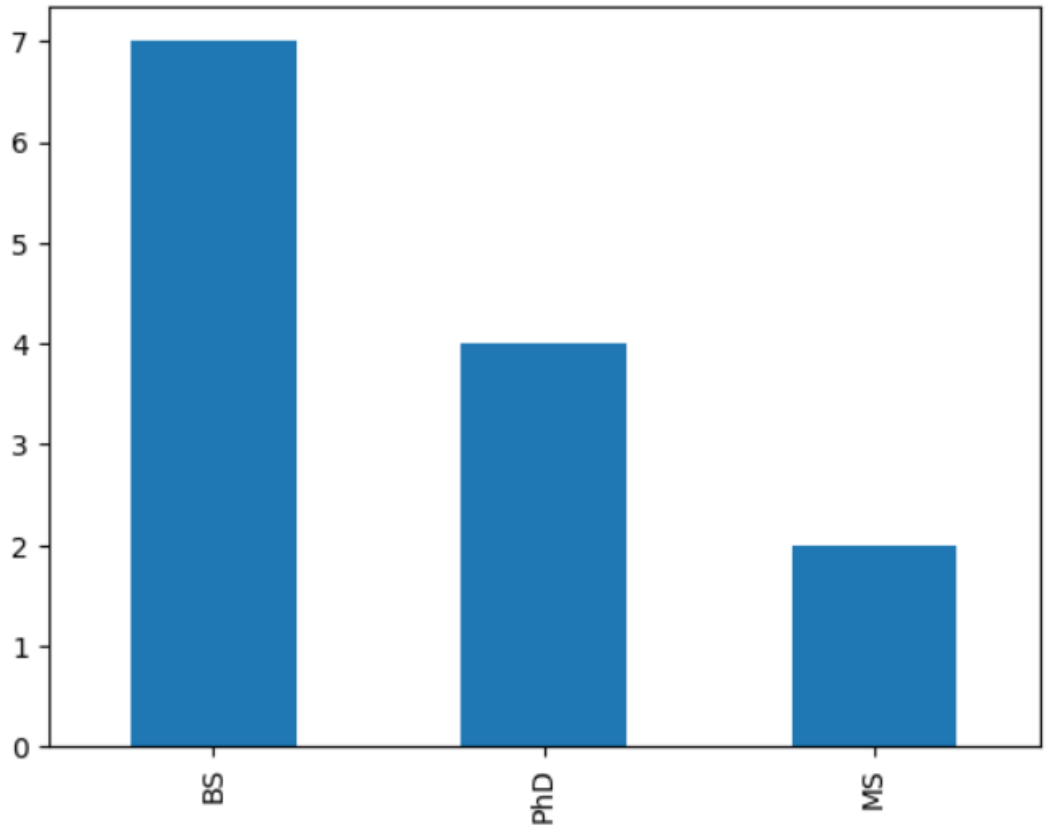
[ ]:

[22]:
```
degree_counts.plot(kind='bar')
```

[ ]:

[22]:
```
degree_counts.plot(kind='bar')
```

[22]: <AxesSubplot:>



[ ]:

[ ]:

[23]:
```
import numpy as np
import pandas as pd
```

[ ]:

[24]:
```
labels = ['a','b','c']
my_list = [10,20,30]
arr = np.array([10,20,30])
d = {'a':10,'b':20,'c':30}
```

[ ]:

[25]:
```
 pd.Series(data=my_list)
```

[25]:
```
0    10
1    20
2    30
dtype: int64
```

[ ]:

[26]:
```
pd.Series(data=my_list,index=labels)
```

[26]:
```
a    10
b    20
c    30
dtype: int64
```

[ ]:

[27]:
```
pd.Series(my_list,labels)
```

```
dtype: int64
```

[ ]:

[27]:
```
pd.Series(my_list,labels)
```

[27]:
```
a    10
b    20
c    30
dtype: int64
```

[ ]:

[28]:
```
pd.Series(arr)
```

[28]:
```
0    10
1    20
2    30
dtype: int32
```

[ ]:

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

[29]:
```
a    10
b    20
c    30
dtype: int32
```

[ ]:

[30]:
```
pd.Series(d)
```

[30]:
```
a    10
```

[ ]:

[30]:
```
pd.Series(d)
```

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

[ ]:

[31]:
```
pd.Series(data=labels)
```

[31]:
```
0    a
1    b
2    c
dtype: object
```

[ ]:

[32]:
```
#Even functions (although unlikely that you will use this)
pd.Series([sum,print,len])
```

[32]:
```
0      <built-in function sum>
1    <built-in function print>
2      <built-in function len>
dtype: object
```

[ ]:

[33]:
```
ser1=pd.Series([1,2,3,4],index=['USA','Germany','USSR','Japan'])
ser1
```

[33]:
```
USA        1
```

[33]:
```
ser1=pd.Series([1,2,3,4],index=['USA','Germany','USSR','Japan'])
ser1
```

[33]:
```
USA        1
Germany    2
USSR       3
Japan      4
dtype: int64
```

[ ]:

[34]:
```
ser2=pd.Series([1,2,5,4],index=['USA','Germany','Italy','Japan'])
ser2
```

[34]:
```
USA        1
Germany    2
Italy      5
Japan      4
dtype: int64
```

[ ]:

[35]:
```
ser1['USA']
```

[35]:
```
1
```

[ ]:

[36]:
```
ser1+ser2
```

[36]:
```
Germany    4.0
Italy      NaN
Japan      8.0
```

[36]:
```
ser1+ser2
```

[36]:
```
Germany    4.0
Italy      NaN
Japan      8.0
USA        2.0
USSR       NaN
dtype: float64
```

[ ]:

[37]:
```
# DataFrames
```

[37]:
```
from numpy.random import randn
np.random.seed(101)
```

[ ]:

[38]:
```
df=pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

[ ]:

[39]:
```
df
```

[39]:

	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

[41]:
```
# Selection and Indexing
```

[40]:
```
df['W']
```

[40]:
```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

[ ]:
```

```

[41]:
```
#Pass a list of column names
df[['W','Z']]
```

[41]:

	W	Z
A	2.706850	0.503826
B	0.651118	0.605965
C	-2.018168	-0.589001
D	0.188695	0.955057
E	0.190794	0.683509

[ ]:
```

```

[42]:
```
# SQL syntax (NOT RECOMMENDED!)
df.W
```

[ ]:

[42]:
```
# SQL syntax (NOT RECOMMENDED!)
df.W
```

[42]:
```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

[ ]:

[43]:
```
type(df['W'])
```

[43]:
```
pandas.core.series.Series
```

[ ]:

[44]:
```
df['new'] = df['W'] + df['Y']
```

[ ]:

[45]:
```
df
```

[45]:

	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

[ ]:

[46]:
```
#Return a new DataFrame with the 'new1'
#column dropped
df.drop('new',axis=1)
```

[46]:

	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

[ ]:

[47]:
```
# Not inplace unless specified!
df
```

[47]:

	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

[48]:
```
#Drop the 'new' column of DataFrame itself
df.drop('new',axis=1,inplace=True)
```

[ ]:

[49]:
```
df
```

[49]:

	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

[ ]:

[50]:
```
df.drop('E',axis=0)
```

[50]:

	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

[51]:

	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

[ ]:

[52]:
```
df.loc['A']
```

[52]:
```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

[ ]:

[53]:
```
df.iloc[2]
```

[53]:
```
W   -2.018168
X    0.740122
Y    0.528813
Z   -0.589001
Name: C, dtype: float64
```

[ ]:

[54]:
```
df.loc['B','Y']
```

[54]: -0.8480760834036315

[55]:
```
df.loc[['A','B'],['W','Y']]
```

[55]:

	W	Y
A	2.706850	0.907969
B	0.651118	-0.848077

[ ]:

[56]:
```
df
```

[56]:

	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

[ ]:

[57]:
```
df>0
```

[57]:

	W	X	Y	Z
A	True	True	True	True
B	True	False	False	True

[57]:

	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

[ ]:

[58]:
```
df[df>0]
```

[58]:

	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

[ ]:

[59]:
```
df[df['W']>0]
```

[59]:

	W	X	Y	Z

E	0.190794	1.978757	2.605967	0.683509

[ ]:

[60]:
```
df[df['W']>0]['Y']
```

[60]:
```
A    0.907969
B   -0.848077
D   -0.933237
E    2.605967
Name: Y, dtype: float64
```

[ ]:

[61]:
```
df[df['W']>0][['Y','X']]
```

[61]:

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

[ ]:

[62]:
```
df[(df['W']>0) & (df['Y'] > 1)]
```

[62]:

	W	X	Y	Z

4	E	0.190794	1.978757	2.605967	0.683509

[ ]:

[65]:
```
newind = 'CA NY WY OR CO'.split()
```

[ ]:

[66]:
```
df['States'] = newind
```

[ ]:

[67]:
```
df
```

[67]:

	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

[ ]:

[68]:
```
df.set_index('States')
```

[68]:

	W	X	Y	Z

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

[ ]:

[69]:
```
df
```

[69]:

	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

[ ]:

[70]:
```
df.set_index('States',inplace=True)
```

[ ]:

[71]:
```
df
```

[71]:

	W	X	Y	Z

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

[ ]:

[72]:
```
# 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)
```

[ ]:

[73]:
```
hier_index
```

[73]:
```
MultiIndex([('G1', 1),
            ('G1', 2),
            ('G1', 3),
            ('G2', 1),
            ('G2', 2),
            ('G2', 3)],
           )
```

[ ]:

[74]:
```
df = pd.DataFrame(np.random.randn(6,2),index=hier_index,columns=['A','B'])
```

[ ]:

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

[ ]:

[76]:
```
df.loc['G1'].loc[1]
```

[76]:
```
A    0.302665
B    1.693723
Name: 1, dtype: float64
```

[ ]:

[77]:
```
df.index.names
```

[77]:
```
FrozenList([None, None])
```

[ ]:

[78]:
```
df.index.names = ['Group','Num']
```

[ ]:

[79]:
```
df
```

[79]:

		A	B
Group	**Num**		
G1	**1**	0.302665	1.693723

G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

[ ]:

[80]:
```
df.xs('G1')
```

[80]:

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

[ ]:

[81]:
```
df.xs(['G1',1])
```

```
C:\Users\phamk\AppData\Local\Temp\ipykernel_7044\580597333.py:1: FutureWarning: Passing lists as k
tead.
  df.xs(['G1',1])
```

[81]:
```
A    0.302665
B    1.693723
Name: (G1, 1), dtype: float64
```

[ ]:

```
import pandas as pd
```

[ ]:

[84]:
```
df = pd.DataFrame({'A':[1,2,np.nan],
                   'B':[5,np.nan,np.nan],
                   'C':[1,2,3]})
```

[ ]:

[85]:
```
df
```

[85]:

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

[ ]:

[86]:
```
df.dropna()
```

[86]:

	A	B	C
0	1.0	5.0	1

[ ]:

[87]:
```
df.dropna(axis=1)
```

1	2
2	3

[ ]:

[88]:
```
df.dropna(thresh=2)
```

[88]:

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

[ ]:

[89]:
```
df.fillna(value='FILL VALUE')
```

[89]:

	A	B	C
0	1.0	5.0	1
1	2.0	FILL VALUE	2
2	FILL VALUE	FILL VALUE	3

[ ]:

[90]:
```
df['A'].fillna(value=df['A'].mean())
```

[90]:
```
0    1.0
1    2.0
```

[91]:
```
import pandas as pd
# Create dataframe
data = {'Company':['GOOG','GOOG','MSFT','MSFT','FB','FB'],
       'Person':['Sam','Charlie','Amy','vanessa','Carl','Sarah'],
       'Sales':[200,120,340,124,243,350]}
```

[ ]:

[92]:
```
df = pd.DataFrame(data)
```

[ ]:

[93]:
```
df
```

[93]:

	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

[ ]:

[94]:
```
df.groupby('Company')
```

[94]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001824583D220>
```

[ ]:

[96]:
```
by_comp.mean()
```

[96]:

Company	Sales
FB	296.5
GOOG	160.0
MSFT	232.0

[ ]:

[97]:
```
df.groupby('Company').mean()
```

[97]:

Company	Sales
FB	296.5
GOOG	160.0
MSFT	232.0

[ ]:

[98]:
```
by_comp.std()
```

FB	75.660426
GOOG	56.568542
MSFT	152.735065

[ ]:

[99]:
```
by_comp.min()
```

[99]:

	Person	Sales
Company		
FB	Carl	243
GOOG	Charlie	120
MSFT	Amy	124

[ ]:

[100]:
```
by_comp.max()
```

[100]:

	Person	Sales
Company		
FB	Sarah	350
GOOG	Sam	200
MSFT	vanessa	340

[101]:

Company	Person	Sales
FB	2	2
GOOG	2	2
MSFT	2	2

[ ]:

[102]:
```
by_comp.describe()
```

[102]:

		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

[ ]:

[103]:
```
by_comp.describe().transpose()
```

[103]:

Company		FB	GOOG	MSFT
Sales	**count**	2.000000	2.000000	2.000000

	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

[ ]:

[104]:
```
by_comp.describe().transpose()
```

[104]:

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

[ ]:

[105]:
```
by_comp.describe().transpose()['GOOG']
```

```
min      120.000000
25%      140.000000
50%      160.000000
75%      180.000000
max      200.000000
Name: GOOG, dtype: float64
```

[ ]:

[2]:
```
import pandas as pd
```

[ ]:

[107]:
```
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])
```

[ ]:

[108]:
```
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])
```

[ ]:

[109]:
```
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                    'B': ['B8', 'B9', 'B10', 'B11'],
                    'C': ['C8', 'C9', 'C10', 'C11'],
                    'D': ['D8', 'D9', 'D10', 'D11']},
```

3	A3	B3	C3	D3

[ ]:

[111]:
```
df2
```

[111]:

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

[ ]:

[112]:
```
df3
```

[112]:

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

[ ]:

[113]:
```
pd.concat([df1,df2,df3])
```

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

[ ]:

[114]:
```
pd.concat([df1,df2,df3],axis=1)
```

[114]:

	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

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

[ ]:

[ ]:

[115]:
```
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']})
```

[ ]:

[116]:
```
left
```

[116]:

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2

[ ]:

[ ]:

[117]:
```
right
```

[117]:

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

[118]:
```
pd.merge(left,right,how='inner',on='key')
```

[118]:

	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

[ ]:

[ ]:

[119]:
```
left = pd.DataFrame({'key1': ['K0','K0','K1','K2'],
```

0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

[ ]:

[ ]:

[121]:
```
pd.merge(left,right,how='outer',on=['key1','key2'])
```

[121]:

	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

[122]:
```
pd.merge(left,right,how='right',on=['key1','key2'])
```

[122]:

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

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

[125]:
```
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'])
```

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

[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

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

[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

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

[134]:

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

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

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

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

[136]:
```
3
```

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

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

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

[143]:
```
newdf
```

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

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

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

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

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

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

[148]:
```
10
```

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

[150]:
```
df
```

[150]:

	col2	col3
0	444	abc
1	555	def

[151]:
```
df.columns
```

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

[152]:
```
df.index
```

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

[153]:
```
df
```

[153]:

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

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

[154]:

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

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

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

[157]:
```
import numpy as np
```

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

[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

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

[159]:

	col1	col2	col3
0	False	True	False
1	False	False	False

[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)
```

[165]:
```
df
```

[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

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

[202]:

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

```
import pandas as pd
```

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

[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

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

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

[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

2	8	9	10	11
3	12	13	14	15

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

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

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

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

[18]:
```
4
```

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

[20]:
```
sql_df
```

[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

Exercise:

[3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn
import re
```

Import the data and get a high-level picture

[6]:
```
df = pd.read_csv('sales.csv')
df.head()
```

[6]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	$10.50
1	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	1	$3.50
2	10001	"SORBET" Raspberry	2018-01-01 12:14:54	$2.50	2	$5.00
3	10001	NaN	2018-01-01 12:14:54	$1.50	1	$1.50
4	10001	"CONE" Dipped Waffle Cone	2018-01-01 12:14:54	$3.50	1	$3.50

[7]:
```
df.shape
```

[7]:
```
(29922, 6)
```

[8]:
```
df.dtypes
```

[8]:
```
order_id       int64
name          object
ordered_at    object
price         object
quantity       int64
```

TODO: Fix column datatypes

Change ordered_at to datetime

Change price and line_total to float

[5]:

[6]:

[9]:
```
df.dtypes
```

[9]:
```
order_id       int64
name          object
ordered_at    object
price         object
quantity       int64
line_total    object
dtype: object
```

TODO: drop if duplicated or null

[10]:
```
df[df.duplicated()].shape[0]
```

[10]:
```
538
```

[9]:

[11]:
```
df.isnull().sum()
```

[11]:
```
order_id         0
name          1488
ordered_at       0
```

[12]:
```
df[df['name'].isnull()].head()
```

[12]:

	order_id	name	ordered_at	price	quantity	line_total
3	10001	NaN	2018-01-01 12:14:54	$1.50	1	$1.50
6	10002	NaN	2018-01-01 12:23:09	$3.00	3	$9.00
27	10007	NaN	2018-01-01 15:03:17	$2.50	1	$2.50
77	10026	NaN	2018-01-02 03:25:40	$0.50	2	$1.00
88	10031	NaN	2018-01-02 05:45:48	$3.50	3	$10.50

[12]:

Sanity check for value ranges and to check assumptions

[15]:
```
df[(df['price'] * df['quantity']) != df['line_total']].shape[0]
```

[15]:
```
19924
```

[17]:
```
# df[df['line_total'] < 0].shape[0]
df['line_total'] = pd.to_numeric(df['line_total'], errors='coerce')
```

TODO:

Set line_total = price * quantity if different Remove if line total < 0

[15]:

[18]:

	order_id	quantity	line_total
count	29922.000000	29922.000000	0.0
mean	14992.538701	2.002105	NaN
std	2889.466576	0.819225	NaN
min	10000.000000	1.000000	NaN
25%	12498.000000	1.000000	NaN
50%	14972.000000	2.000000	NaN
75%	17506.750000	3.000000	NaN
max	19999.000000	3.000000	NaN

TODO: Get value between "" in name and put it in category column

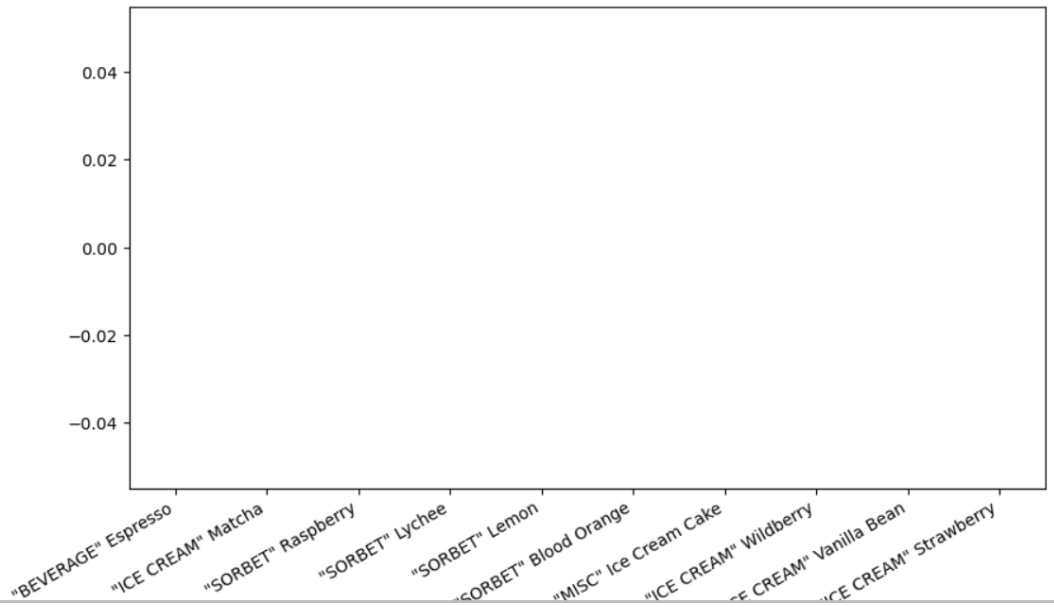
[18]:

[19]:
```
df.head()
```

[19]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	NaN
1	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	1	NaN
2	10001	"SORBET" Raspberry	2018-01-01 12:14:54	$2.50	2	NaN
3	10001	NaN	2018-01-01 12:14:54	$1.50	1	NaN
4	10001	"CONE" Dipped Waffle Cone	2018-01-01 12:14:54	$3.50	1	NaN

[20]:
```
f, ax = plt.subplots(figsize=(10, 6))
df.groupby('name')['line_total'].sum().sort_values(ascending=False).head(10).plot(kind='bar')
f.autofmt_xdate()
plt.show()
```




```
dtype: object
```

[25]:
```
df['col1'] = df['order_id'].astype(int)
```

[26]:
```
print(df.dtypes)
```

```
order_id       int64
name          object
ordered_at    object
price         object
quantity       int64
line_total    object
col1           int32
dtype: object
```

[27]:
```
df['col1'] = pd.to_numeric(df['col1'], errors='coerce')
```

[28]:
```
print(df.dtypes)
```

```
order_id       int64
name          object
ordered_at    object
price         object
quantity       int64
line_total    object
col1           int32
dtype: object
```

[29]:
```
df = pd.read_csv('sales.csv')
```

[30]:
```
df
```

[30]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	$10.50

[30]:
```
df
```

[30]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	$10.50
1	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	1	$3.50
2	10001	"SORBET" Raspberry	2018-01-01 12:14:54	$2.50	2	$5.00
3	10001	NaN	2018-01-01 12:14:54	$1.50	1	$1.50
4	10001	"CONE" Dipped Waffle Cone	2018-01-01 12:14:54	$3.50	1	$3.50
...	...	...	...	...	...	...
29917	18452	"ICE CREAM" Dulce De Leche	2018-06-26 03:56:13	$-1.50	2	$-3.00
29918	12889	"ICE CREAM" Dark Chocolate	2018-03-03 10:06:21	$4.00	3	$12.00
29919	14526	"ICE CREAM" Peanut Fudge	2018-04-05 17:33:24	$3.50	3	$10.50
29920	19589	"CONE" Dipped Waffle Cone	2018-07-20 09:10:01	$3.50	2	$7.00
29921	19270	"ICE CREAM" Earl Gray	2018-07-13 09:20:21	$0.50	2	$1.00

29922 rows × 6 columns

[31]:
```
df = df.drop_duplicates()
```

[32]:
```
df
```

[32]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	$10.50

[32]:
```
df
```

[32]:

	order_id	name	ordered_at	price	quantity	line_total
0	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	3	$10.50
1	10000	"ICE CREAM" Peanut Fudge	2018-01-01 11:30:00	$3.50	1	$3.50
2	10001	"SORBET" Raspberry	2018-01-01 12:14:54	$2.50	2	$5.00
3	10001	NaN	2018-01-01 12:14:54	$1.50	1	$1.50
4	10001	"CONE" Dipped Waffle Cone	2018-01-01 12:14:54	$3.50	1	$3.50
...	...	...	...	...	...	...
29817	19997	"CONE" Waffle Cone	2018-07-28 17:40:40	$4.00	3	$12.00
29818	19997	"SORBET" Blood Orange	2018-07-28 17:40:40	$2.50	3	$7.50
29819	19998	"SORBET" Lychee	2018-07-28 18:21:44	$3.00	1	$3.00
29820	19998	"ICE CREAM" Rocky Road	2018-07-28 18:21:44	$3.50	1	$3.50
29821	19999	"SORBET" Blood Orange	2018-07-28 18:51:57	$2.50	2	$5.00

29384 rows × 6 columns

[33]:
```
df = df.dropna()
```

[ ]:
```

```

[34]:
```
import re
```

[38]:
```
df = pd.read_csv('sales.csv')
```

[59]:
```
def extract_order_id(text):
    order_regex = r"ORDER-\d+"
    order_ids = re.findall(order_regex, text)
    if order_ids:
        return order_ids[0]
    else:
        return None
```

[47]:
```
print(df['order_id'].isna().sum())
```

```
0
```

[48]:
```
df['order_id'] = df['order_id'].astype(int)
```

[50]:
```
print(df['order_id'].unique())
```

```
[10000 10001 10002 ... 19997 19998 19999]
```

[51]:
```
df = df.dropna(subset=['order_id'])
```

[52]:
```
df['order_id'] = df['order_id'].fillna(0)
```

[53]:
```
df['order_id'] = df['order_id'].astype(int)
```

[50]:
```
print(df['order_id'].unique())
```

```
[10000 10001 10002 ... 19997 19998 19999]
```

[51]:
```
df = df.dropna(subset=['order_id'])
```

[52]:
```
df['order_id'] = df['order_id'].fillna(0)
```

[53]:
```
df['order_id'] = df['order_id'].astype(int)
```

[55]:
```
df['order_id'] = df['order_id'].astype(str)
```

[56]:
```
df['order_id'] = df['order_id'].apply(lambda x: extract_order_id(x))
```

[57]:
```
print(df[df['order_id'].str.contains(r"ORDER-\d+", na=False)])
```

```
Empty DataFrame
Columns: [order_id, name, ordered_at, price, quantity, line_total]
Index: []
```

[58]:
```
print(len(df))
```

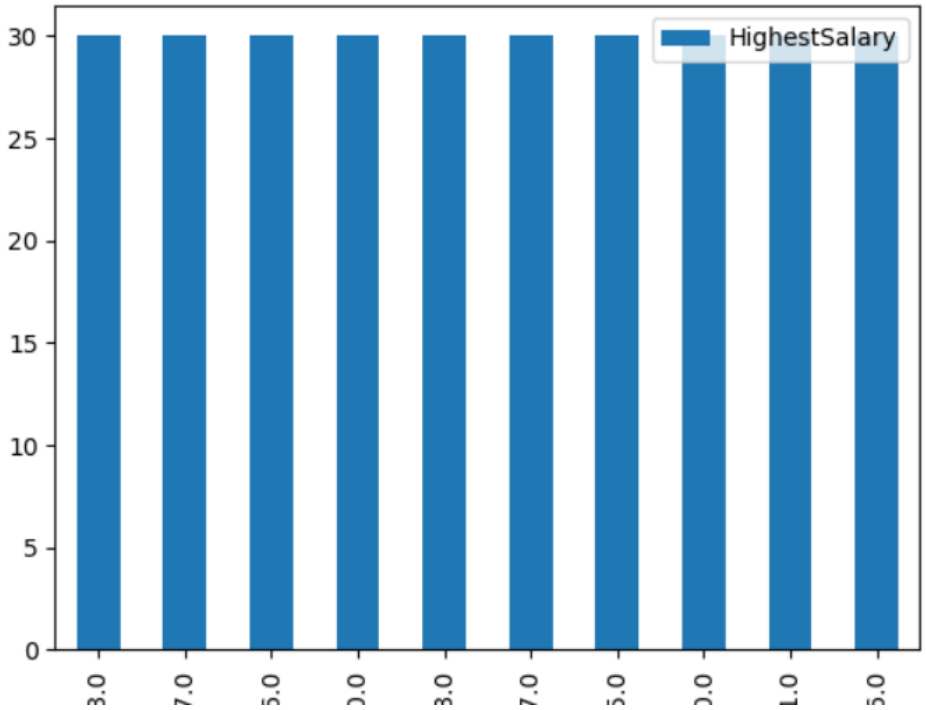
```
29922
```

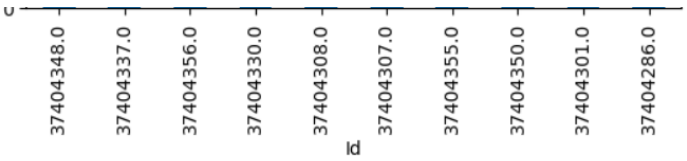
With file job-market

[4]:
```
import pandas as pd
df = pd.read_csv('job-market.csv')
top_10 = df.head(10)
```

[10]:
```
top_10.plot(kind='bar', x='Id', y='HighestSalary')
```

[10]:
```
<AxesSubplot:xlabel='Id'>
```





[13]:
```
df = df.dropna(subset=['HighestSalary'])
```

[14]:
```
df['HighestSalary'] = df['HighestSalary'].astype(int)
```

[15]:
```
print(df.isnull().sum())
```

```
Id                     0
Title                  0
Company              616
Date                   0
Location               0
Area                3707
Classification         0
SubClassification      0
Requirement            0
FullDescription      256
LowestSalary           0
HighestSalary          0
JobType              247
dtype: int64
```

[16]:
```
df = df.dropna()
```

```
HighestSalary          0
JobType              247
dtype: int64
```

[16]:
```
df = df.dropna()
```

[*]:
```
df = df.fillna(df.mean())
```

[18]:
```
print(df.duplicated().sum())
```

```
0
```

[19]:
```
df = df.drop_duplicates()
```

[*]:
```
import seaborn as
```

[ ]:
```
 sns.boxplot(x='col1', data=df)
```

[ ]:
```
df = df[(df['Id'] >= df['Id'].quantile(0.05)) & (df['Id'] <= df['Id'].quantile(0.95))]
```

[*]:
```
print(df.head())

print(df.describe())

print(df.isnull().sum())

print(df.duplicated().sum())
```

[ ]:
```

```