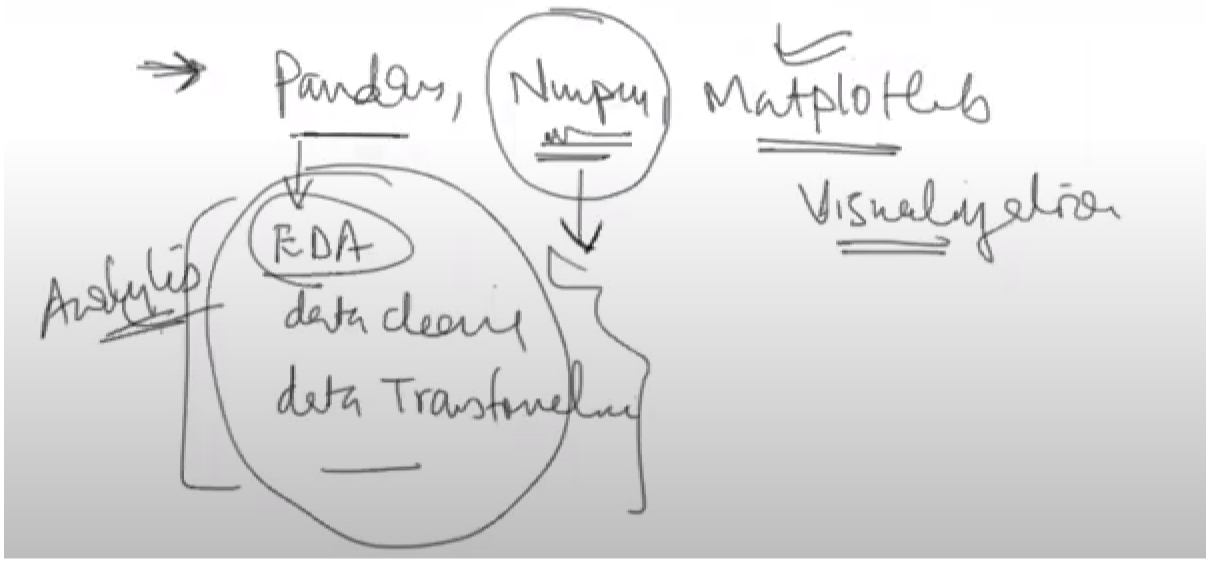


Installing Pandas

```
In [6]: !pip install pandas
```

```
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (1.1.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2.8.
1)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: numpy>=1.15.4 in c:\programdata\anaconda3\lib\site-packages (from pandas) (1.19.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pand
as) (1.15.0)
```

Pandas Series

```
In [ ]: A Pandas Series is like a column in a table.

        It is a one-dimensional array holding data of any type.
```

In [12]:

```
import pandas as pd

a = ['CS', 'IT', 'CSCE']

branch = pd.Series(a)

print(branch)
```

```
0      CS
1      IT
2    CSCE
dtype: object
```

Labels

In [ ]:

```
If nothing else is specified, the values are labeled with their index number. First value has index 0,
 second value has index 1 etc.

This label can be used to access a specified value.
```

In [13]:

```
print(branch[0])
```

```
CS
```

Create Labels

In [14]:

```
import pandas as pd

a = ['CS', 'IT', 'CSCE']

branch = pd.Series(a,index=['a','b','c'])

print(branch)
```

```
a      CS
b      IT
c    CSCE
dtype: object
```

In [15]:

```
print(branch["b"])
```

```
IT
```

DataFrames

In [ ]:

```
A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.
```

In [17]:

```
import pandas as pd

data = {
  "empid": [1,2,3],
  "salary": [50000, 40000, 45000]
}

df = pd.DataFrame(data)

print(df)
```

```
   empid  salary
0      1   50000
1      2   40000
2      3   45000
```

Locate Row

In [ ]:

```
Pandas use the loc attribute to return one or more specified row(s
```

In [18]:

```
#refer to the row index:
print(df.loc[0])
```

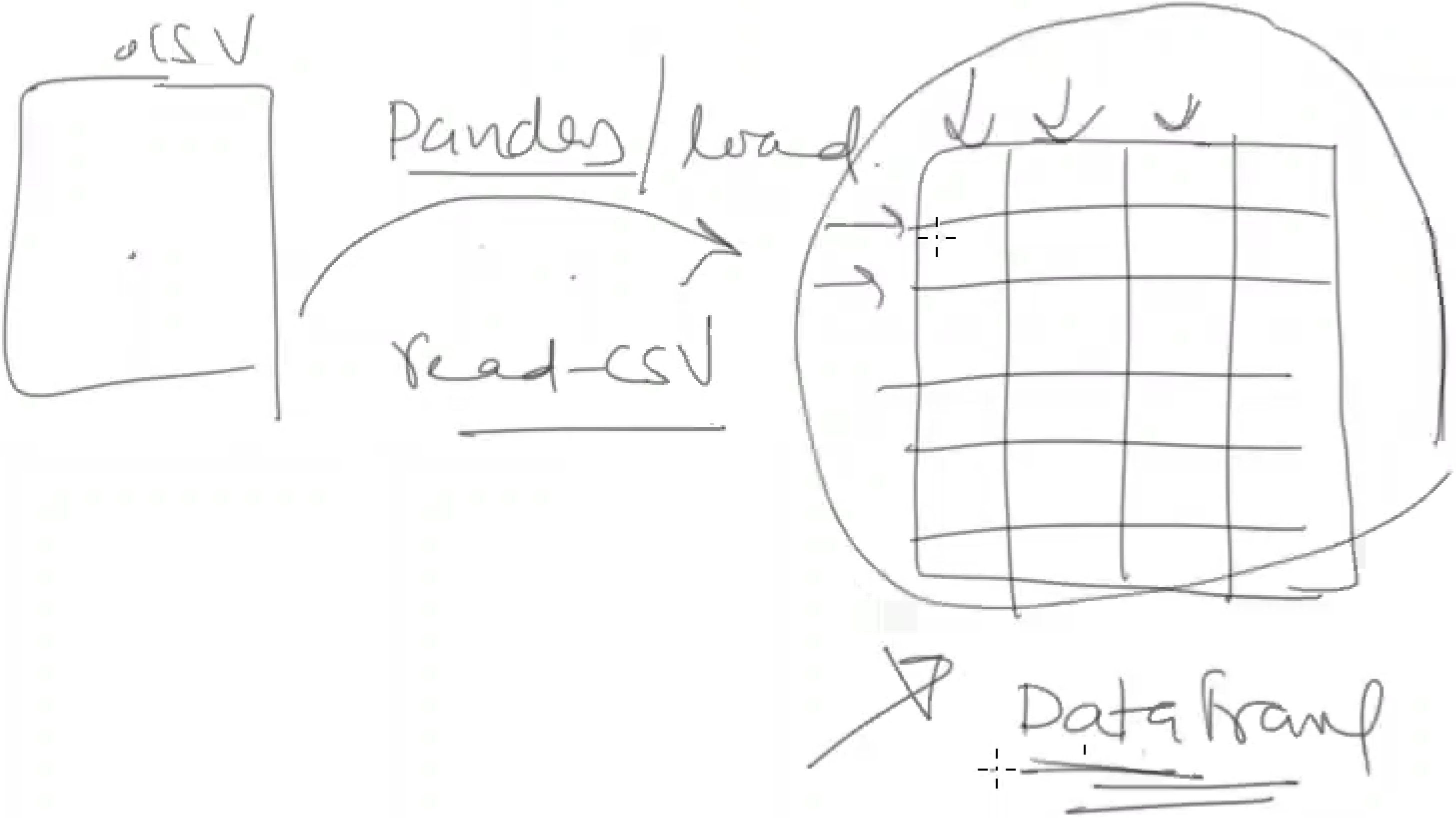
```
empid         1
salary    50000
Name: 0, dtype: int64
```

In [19]:

```
#use a list of indexes:
print(df.loc[[0, 1]])
```

```
   empid  salary
0      1   50000
1      2   40000
```

Load Files Into a DataFrame



In [ ]:

```
# salaries.csv-comma separated values
# Every entry or value of columns is separated by comma
# it is a text file

# Our objective:
# we want to analyze this text file, in later on also we will manipulate something in this text file.

# EDA Activities(Initial Investigation on Data):
# How many Assistant Professors/Professors etc..
# How many Male Professors etc..
# Who will answer these questions- that library in Python is called "Pandas".
```

In [4]:

```
import pandas as pd #alias name just like nick name given to people
```

In [3]:

```
pd.__version__
```

Out[3]:

```
'1.1.3'
```

In [25]:

```
dir(pd)
```

```
 'read_sql_query',
 'read_sql_table',
 'read_stata',
 'read_table',
 'reset_option',
 'set_eng_float_format',
 'set_option',
 'show_versions',
 'test',
 'testing',
 'timedelta_range',
 'to_datetime',
 'to_numeric',
 'to_pickle',
 'to_timedelta',
 'tseries',
 'unique',
 'util',
 'value_counts',
 'wide_to_long']
```

In [5]:
```python
print(help(pd.read_csv))
```

```
            If callable, the callable function will be evaluated against the column
            names, returning names where the callable function evaluates to True. An
            example of a valid callable argument would be ``lambda x: x.upper() in
            ['AAA', 'BBB', 'DDD']``. Using this parameter results in much faster
            parsing time and lower memory usage.
        squeeze : bool, default False
            If the parsed data only contains one column then return a Series.
        prefix : str, optional
            Prefix to add to column numbers when no header, e.g. 'X' for X0, X1, ...
        mangle_dupe_cols : bool, default True
            Duplicate columns will be specified as 'X', 'X.1', ...'X.N', rather than
            'X'...'X'. Passing in False will cause data to be overwritten if there
            are duplicate names in the columns.
        dtype : Type name or dict of column -> type, optional
            Data type for data or columns. E.g. {'a': np.float64, 'b': np.int32,
            'c': 'Int64'}
            Use `str` or `object` together with suitable `na_values` settings
            to preserve and not interpret dtype.
            If converters are specified, they will be applied INSTEAD
```

In [21]:
```python
df=pd.read_csv('Salaries.csv')
```

```
In [22]: print(df.to_string()) # print the entire DataFrame.
```

```
    empid       rank discipline   phd  service  gender    salary
0     101       Prof          B  56.0       49    Male  186960.0
1     102       Prof          A  12.0        6    Male   93000.0
2     103       Prof          A  23.0       20    Male  110515.0
3     104       Prof          A  40.0       31    Male  131205.0
4     105       Prof          B  20.0       18    Male  104800.0
5     106       Prof          A  20.0       20    Male  122400.0
6     107  AssocProf          A  20.0       17    Male   81285.0
7     108       Prof          A  18.0       18    Male       NaN
8     109       Prof          A  29.0       19    Male   94350.0
9     110       Prof          A  51.0       51    Male   57800.0
10    111       Prof          B  39.0       33    Male  128250.0
11    112       Prof          B  23.0       23    Male  134778.0
12    113   AsstProf          B   1.0        0    Male   88000.0
13    114       Prof          B   NaN       33    Male  162200.0
14    115       Prof          B  25.0       19    Male  153750.0
15    116       Prof          B  17.0        3    Male  150480.0
16    117   AsstProf          B   8.0        3    Male   75044.0
17    118   AsstProf          B   4.0        0    Male   92000.0
18    119       Prof          A  19.0        7    Male  107300.0
19    120       Prof          A  29.0       27    Male  150500.0
20    121   AsstProf          B   4.0        4    Male   92000.0
21    122       Prof          A  33.0       30    Male  103106.0
22    123   AsstProf          A   4.0        2    Male   73000.0
23    124   AsstProf          A   2.0        0    Male   85000.0
24    125       Prof          A  30.0       23    Male   91100.0
25    126       Prof          B  35.0       31    Male   99418.0
26    127       Prof          A  38.0       19    Male  148750.0
27    128       Prof          A  45.0       43    Male  155865.0
28    129   AsstProf          B   7.0        2    Male       NaN
29    130       Prof          B  21.0       20    Male  123683.0
30    131  AssocProf          B   9.0        7    Male  107008.0
31    132       Prof          B  22.0       21    Male  155750.0
32    133       Prof          A  27.0       19    Male  103275.0
33    134       Prof          B  18.0       18    Male  120000.0
34    135  AssocProf          B   NaN        8    Male  119800.0
35    136       Prof          B  28.0       23    Male  126933.0
36    137       Prof          B  45.0       45    Male  146856.0
37    138       Prof          A  20.0        8    Male  102000.0
38    139   AsstProf          B   4.0        3    Male   91000.0
39    140       Prof          B  18.0       18  Female  129000.0
40    141       Prof          A  39.0       36  Female  137000.0
41    142  AssocProf          A  13.0        8  Female   74830.0
42    143   AsstProf          B   4.0        2  Female   80225.0
43    144   AsstProf          B   5.0        0  Female   77000.0
```

```
44    145       Prof          B  23.0         19  Female  151768.0
45    146       Prof          B  25.0         25  Female  140096.0
46    147   AsstProf          B  11.0          3  Female   74692.0
47    148  AssocProf          B  11.0         11  Female  103613.0
48    149       Prof          B  17.0         17  Female  111512.0
49    150       Prof          B  17.0         18  Female  122960.0
50    151   AsstProf          B  10.0          5  Female   97032.0
51    152       Prof          B  20.0         14  Female  127512.0
52    153       Prof          A  12.0          0  Female  105000.0
53    154   AsstProf          A   5.0          3  Female   73500.0
54    155  AssocProf          A  25.0         22  Female   62884.0
55    156   AsstProf          A   2.0          0  Female   72500.0
56    157  AssocProf          A  10.0          8  Female   77500.0
57    158   AsstProf          A   3.0          1  Female   72500.0
58    159       Prof          B  36.0         26  Female  144651.0
59    160  AssocProf          B  12.0         10  Female  103994.0
60    161   AsstProf          B   3.0          3  Female   92000.0
61    162  AssocProf          B  13.0         10  Female  103750.0
62    163  AssocProf          B  14.0          7  Female  109650.0
63    164       Prof          A  29.0         27  Female   91000.0
64    165  AssocProf          A  26.0         24  Female   73300.0
65    166       Prof          A  36.0         19  Female  117555.0
66    167   AsstProf          A   7.0          6  Female   63100.0
67    168       Prof          A  17.0         11  Female   90450.0
68    169   AsstProf          A   4.0          2  Female   77500.0
69    170       Prof          A  28.0          7  Female  116450.0
70    171   AsstProf          A   8.0          3  Female   78500.0
71    172  AssocProf          B  12.0          9  Female   71065.0
72    173       Prof          B  24.0         15  Female  161101.0
73    174       Prof          B  18.0         10  Female  105450.0
74    175  AssocProf          B  19.0          6  Female  104542.0
75    176       Prof          B  17.0         17  Female  124312.0
76    177       Prof          A  28.0         14  Female  109954.0
77    178       Prof          A  23.0         15  Female  109646.0
```

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

Out[7]:
```
pandas.core.frame.DataFrame
```

```
In [26]: dir(pd.DataFrame)
```

```
 'to_records',
 'to_sql',
 'to_stata',
 'to_string',
 'to_timestamp',
 'to_xarray',
 'transform',
 'transpose',
 'truediv',
 'truncate',
 'tshift',
 'tz_convert',
 'tz_localize',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'where',
 'xs']
```

```
In [ ]: # EDA activities
        # couple of investigation
        # understanding the data sets by summarizing their main characteristics
```

```
In [8]: df.shape  #how many rows and how many columns
```

```
Out[8]: (78, 7)
```

```
In [28]: df.ndim
```

```
Out[28]: 2
```

```
In [29]: #total entries in dataframe
         df.size
```

```
Out[29]: 546
```

```
In [26]: df.columns
```

```
Out[26]: Index(['empid', 'rank', 'discipline', 'phd', 'service', 'gender', 'salary'], dtype='object')
```

In [27]:

```
df.columns.tolist()
```

Out[27]:

```
['empid', 'rank', 'discipline', 'phd', 'service', 'gender', 'salary']
```

In [9]:

```
df.head() #By deafult 5 rows
```

Out[9]:

	empid	rank	discipline	phd	service	gender	salary
0	101	Prof	B	56.0	49	Male	186960.0
1	102	Prof	A	12.0	6	Male	93000.0
2	103	Prof	A	23.0	20	Male	110515.0
3	104	Prof	A	40.0	31	Male	131205.0
4	105	Prof	B	20.0	18	Male	104800.0

In [10]:
```
df.head(20)
```

Out[10]:

	empid	rank	discipline	phd	service	gender	salary
0	101	Prof	B	56.0	49	Male	186960.0
1	102	Prof	A	12.0	6	Male	93000.0
2	103	Prof	A	23.0	20	Male	110515.0
3	104	Prof	A	40.0	31	Male	131205.0
4	105	Prof	B	20.0	18	Male	104800.0
5	106	Prof	A	20.0	20	Male	122400.0
6	107	AssocProf	A	20.0	17	Male	81285.0
7	108	Prof	A	18.0	18	Male	NaN
8	109	Prof	A	29.0	19	Male	94350.0
9	110	Prof	A	51.0	51	Male	57800.0
10	111	Prof	B	39.0	33	Male	128250.0
11	112	Prof	B	23.0	23	Male	134778.0
12	113	AsstProf	B	1.0	0	Male	88000.0
13	114	Prof	B	NaN	33	Male	162200.0
14	115	Prof	B	25.0	19	Male	153750.0
15	116	Prof	B	17.0	3	Male	150480.0
16	117	AsstProf	B	8.0	3	Male	75044.0
17	118	AsstProf	B	4.0	0	Male	92000.0
18	119	Prof	A	19.0	7	Male	107300.0
19	120	Prof	A	29.0	27	Male	150500.0

In [11]:
```
df.tail(10)
```

Out[11]:

	empid	rank	discipline	phd	service	gender	salary
68	169	AsstProf	A	4.0	2	Female	77500.0
69	170	Prof	A	28.0	7	Female	116450.0
70	171	AsstProf	A	8.0	3	Female	78500.0
71	172	AssocProf	B	12.0	9	Female	71065.0
72	173	Prof	B	24.0	15	Female	161101.0
73	174	Prof	B	18.0	10	Female	105450.0
74	175	AssocProf	B	19.0	6	Female	104542.0
75	176	Prof	B	17.0	17	Female	124312.0
76	177	Prof	A	28.0	14	Female	109954.0
77	178	Prof	A	23.0	15	Female	109646.0

In [12]:
```
df.sample() #take 1 row randomly
```

Out[12]:

	empid	rank	discipline	phd	service	gender	salary
58	159	Prof	B	36.0	26	Female	144651.0

In [13]:
```
df.sample() #take 1 row randomly
```

Out[13]:

	empid	rank	discipline	phd	service	gender	salary
56	157	AssocProf	A	10.0	8	Female	77500.0

In [14]:
```
df.sample(5) #take 5 rows randomly
```

Out[14]:

	empid	rank	discipline	phd	service	gender	salary
73	174	Prof	B	18.0	10	Female	105450.0
50	151	AsstProf	B	10.0	5	Female	97032.0
7	108	Prof	A	18.0	18	Male	NaN
47	148	AssocProf	B	11.0	11	Female	103613.0
40	141	Prof	A	39.0	36	Female	137000.0

In [15]:
```
# when we want to access only column values let's say "salary"column
df['salary']
```

Out[15]:
```
0     186960.0
1      93000.0
2     110515.0
3     131205.0
4     104800.0
        ...
73    105450.0
74    104542.0
75    124312.0
76    109954.0
77    109646.0
Name: salary, Length: 78, dtype: float64
```

In [31]:
```
df.rank #rank is the internal property of dataframe
```

Out[31]:
```
<bound method NDFrame.rank of     empid       rank discipline   phd  service  gender    salary
0     101       Prof          B  56.0       49    Male  186960.0
1     102       Prof          A  12.0        6    Male   93000.0
2     103       Prof          A  23.0       20    Male  110515.0
3     104       Prof          A  40.0       31    Male  131205.0
4     105       Prof          B  20.0       18    Male  104800.0
..    ...        ...        ...   ...      ...     ...       ...
73    174       Prof          B  18.0       10  Female  105450.0
74    175  AssocProf          B  19.0        6  Female  104542.0
75    176       Prof          B  17.0       17  Female  124312.0
76    177       Prof          A  28.0       14  Female  109954.0
77    178       Prof          A  23.0       15  Female  109646.0

[78 rows x 7 columns]>
```

```
In [33]: dir(pd)
```

```
Out[33]: ['BooleanDtype',
          'Categorical',
          'CategoricalDtype',
          'CategoricalIndex',
          'DataFrame',
          'DateOffset',
          'DatetimeIndex',
          'DatetimeTZDtype',
          'ExcelFile',
          'ExcelWriter',
          'Float64Index',
          'Grouper',
          'HDFStore',
          'Index',
          'IndexSlice',
          'Int16Dtype',
          'Int32Dtype',
          'Int64Dtype',
          'Int64Index',
```

```
In [32]: df['rank']
```

```
Out[32]: 0           Prof
         1           Prof
         2           Prof
         3           Prof
         4           Prof
                 ...
         73          Prof
         74     AssocProf
         75          Prof
         76          Prof
         77          Prof
         Name: rank, Length: 78, dtype: object
```

```
In [16]: df[['empid','salary']]
```

```
Out[16]:
       empid     salary
  0      101   186960.0
  1      102    93000.0
  2      103   110515.0
  3      104   131205.0
  4      105   104800.0
...      ...        ...
 73      174   105450.0
 74      175   104542.0
 75      176   124312.0
 76      177   109954.0
 77      178   109646.0

78 rows × 2 columns
```

```
In [17]: df[['empid','salary','gender']]
```

Out[17]:

	empid	salary	gender
0	101	186960.0	Male
1	102	93000.0	Male
2	103	110515.0	Male
3	104	131205.0	Male
4	105	104800.0	Male
...	...	...	...
73	174	105450.0	Female
74	175	104542.0	Female
75	176	124312.0	Female
76	177	109954.0	Female
77	178	109646.0	Female

78 rows × 3 columns

Access only unique column values-unique()

```
In [18]: df['rank'].unique()
```

```
Out[18]: array(['Prof', 'AssocProf', 'AsstProf'], dtype=object)
```

```
In [19]: df['gender'].unique() #unique entries
```

```
Out[19]: array(['Male', 'Female'], dtype=object)
```

```
In [20]: df['rank'].value_counts() #unique_entries _count
```

```
Out[20]: Prof         46
         AsstProf     19
         AssocProf    13
         Name: rank, dtype: int64
```

```
In [21]: df['gender'].value_counts()
```

```
Out[21]: Female    39
         Male      39
         Name: gender, dtype: int64
```

```
In [22]: df['rank'].value_counts(normalize=True) #out of total data, which data in how many (%)
```

```
Out[22]: Prof         0.589744
         AsstProf     0.243590
         AssocProf    0.166667
         Name: rank, dtype: float64
```

```
In [23]: df['salary'].max()
```

```
Out[23]: 186960.0
```

```
In [25]: df['salary'].min()
```

```
Out[25]: 57800.0
```

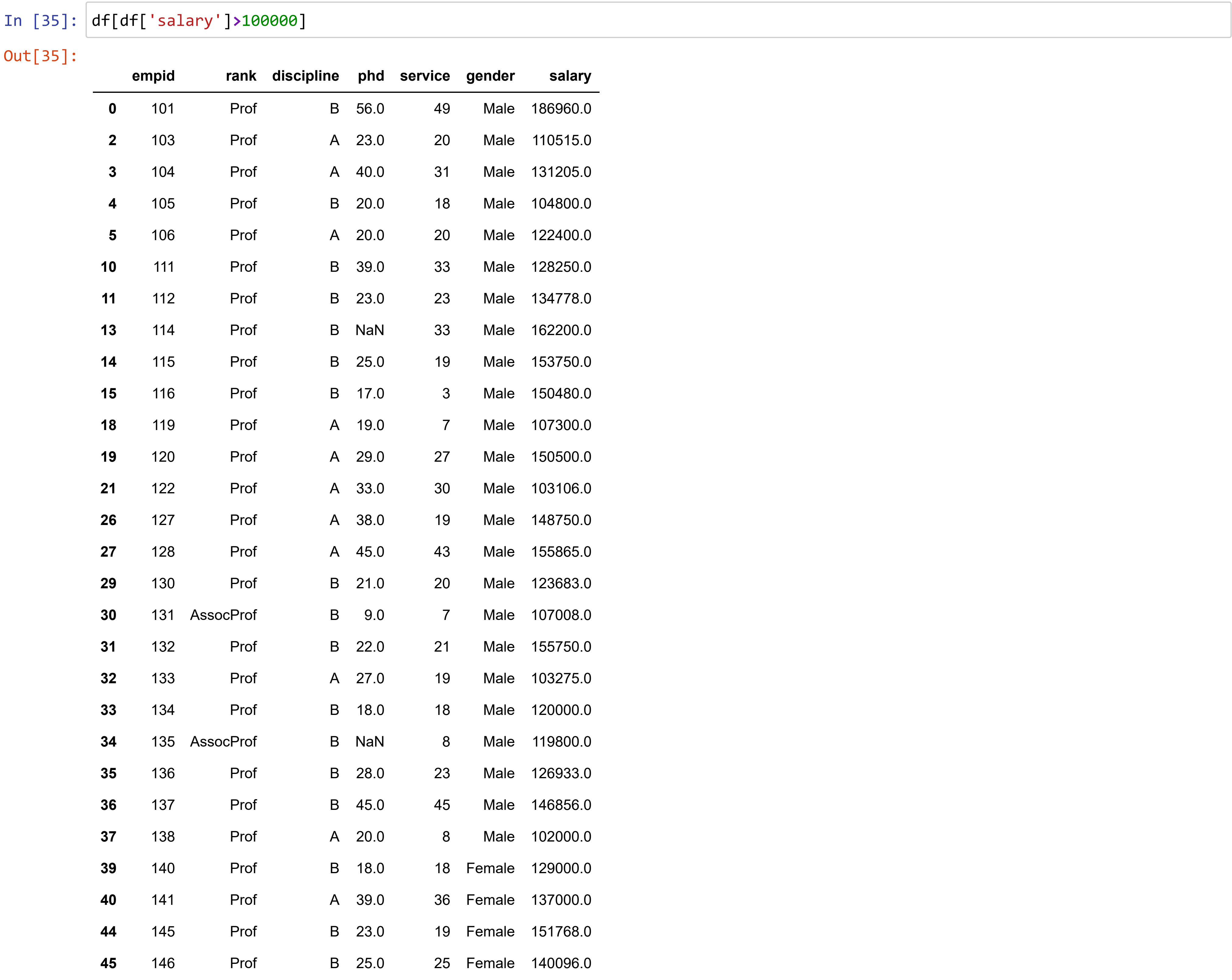
```
In [24]: df['salary'].mean()
```

```
Out[24]: 108003.3552631579
```

```
In [34]: df['salary']>100000
```

```
Out[34]: 0      True
         1     False
         2      True
         3      True
         4      True
               ...
         73     True
         74     True
         75     True
         76     True
         77     True
         Name: salary, Length: 78, dtype: bool
```

```
In [ ]: #filter
```



```
In [35]: df[df['salary']>100000]
```

Out[35]:

	empid	rank	discipline	phd	service	gender	salary
0	101	Prof	B	56.0	49	Male	186960.0
2	103	Prof	A	23.0	20	Male	110515.0
3	104	Prof	A	40.0	31	Male	131205.0
4	105	Prof	B	20.0	18	Male	104800.0
5	106	Prof	A	20.0	20	Male	122400.0
10	111	Prof	B	39.0	33	Male	128250.0
11	112	Prof	B	23.0	23	Male	134778.0
13	114	Prof	B	NaN	33	Male	162200.0
14	115	Prof	B	25.0	19	Male	153750.0
15	116	Prof	B	17.0	3	Male	150480.0
18	119	Prof	A	19.0	7	Male	107300.0
19	120	Prof	A	29.0	27	Male	150500.0
21	122	Prof	A	33.0	30	Male	103106.0
26	127	Prof	A	38.0	19	Male	148750.0
27	128	Prof	A	45.0	43	Male	155865.0
29	130	Prof	B	21.0	20	Male	123683.0
30	131	AssocProf	B	9.0	7	Male	107008.0
31	132	Prof	B	22.0	21	Male	155750.0
32	133	Prof	A	27.0	19	Male	103275.0
33	134	Prof	B	18.0	18	Male	120000.0
34	135	AssocProf	B	NaN	8	Male	119800.0
35	136	Prof	B	28.0	23	Male	126933.0
36	137	Prof	B	45.0	45	Male	146856.0
37	138	Prof	A	20.0	8	Male	102000.0
39	140	Prof	B	18.0	18	Female	129000.0
40	141	Prof	A	39.0	36	Female	137000.0
44	145	Prof	B	23.0	19	Female	151768.0
45	146	Prof	B	25.0	25	Female	140096.0

	empid	rank	discipline	phd	service	gender	salary
47	148	AssocProf	B	11.0	11	Female	103613.0
48	149	Prof	B	17.0	17	Female	111512.0
49	150	Prof	B	17.0	18	Female	122960.0
51	152	Prof	B	20.0	14	Female	127512.0
52	153	Prof	A	12.0	0	Female	105000.0
58	159	Prof	B	36.0	26	Female	144651.0
59	160	AssocProf	B	12.0	10	Female	103994.0
61	162	AssocProf	B	13.0	10	Female	103750.0
62	163	AssocProf	B	14.0	7	Female	109650.0
65	166	Prof	A	36.0	19	Female	117555.0
69	170	Prof	A	28.0	7	Female	116450.0
72	173	Prof	B	24.0	15	Female	161101.0
73	174	Prof	B	18.0	10	Female	105450.0
74	175	AssocProf	B	19.0	6	Female	104542.0
75	176	Prof	B	17.0	17	Female	124312.0
76	177	Prof	A	28.0	14	Female	109954.0
77	178	Prof	A	23.0	15	Female	109646.0

```
In [49]: df[(df['salary']>100000) & (df['gender']=='Male')]
```

Out[49]:

	empid	rank	discipline	phd	service	gender	salary
0	101	Prof	B	56.0	49	Male	186960.0
2	103	Prof	A	23.0	20	Male	110515.0
3	104	Prof	A	40.0	31	Male	131205.0
4	105	Prof	B	20.0	18	Male	104800.0
5	106	Prof	A	20.0	20	Male	122400.0
10	111	Prof	B	39.0	33	Male	128250.0
11	112	Prof	B	23.0	23	Male	134778.0
13	114	Prof	B	NaN	33	Male	162200.0
14	115	Prof	B	25.0	19	Male	153750.0
15	116	Prof	B	17.0	3	Male	150480.0
18	119	Prof	A	19.0	7	Male	107300.0
19	120	Prof	A	29.0	27	Male	150500.0
21	122	Prof	A	33.0	30	Male	103106.0
26	127	Prof	A	38.0	19	Male	148750.0
27	128	Prof	A	45.0	43	Male	155865.0
29	130	Prof	B	21.0	20	Male	123683.0
30	131	AssocProf	B	9.0	7	Male	107008.0
31	132	Prof	B	22.0	21	Male	155750.0
32	133	Prof	A	27.0	19	Male	103275.0
33	134	Prof	B	18.0	18	Male	120000.0
34	135	AssocProf	B	NaN	8	Male	119800.0
35	136	Prof	B	28.0	23	Male	126933.0
36	137	Prof	B	45.0	45	Male	146856.0
37	138	Prof	A	20.0	8	Male	102000.0

In [50]:
```
# How to know which column having missing value
# false- column has no missing values
# true- column has missing values
#boolean indexing
df.isnull().any(axis=0)
```

Out[50]:
```
empid         False
rank          False
discipline    False
phd            True
service       False
gender        False
salary         True
dtype: bool
```

In [53]:
```
# How to know which row having missing value
#boolean indexing
df.isnull().any(axis=1)
```

Out[53]:
```
0     False
1     False
2     False
3     False
4     False
      ...
73    False
74    False
75    False
76    False
77    False
Length: 78, dtype: bool
```

In [54]:
```
# use filter
# which particular row has missing data in column
df[df.isnull().any(axis=1)]
```

Out[54]:

	empid	rank	discipline	phd	service	gender	salary
7	108	Prof	A	18.0	18	Male	NaN
13	114	Prof	B	NaN	33	Male	162200.0
28	129	AsstProf	B	7.0	2	Male	NaN
34	135	AssocProf	B	NaN	8	Male	119800.0

Handling missing data

In [56]:

```
#Right strategies to handle missing data
# 1.average value of that column
df['phd'].mean()
```

Out[56]:

```
19.605263157894736
```

In [57]:

```
df['phd'].fillna(df['phd'].mean())
```

Out[57]:

```
0     56.0
1     12.0
2     23.0
3     40.0
4     20.0
      ...
73    18.0
74    19.0
75    17.0
76    28.0
77    23.0
Name: phd, Length: 78, dtype: float64
```

In [58]:

```
df[df.isnull().any(axis=1)]
```

Out[58]:

	empid	rank	discipline	phd	service	gender	salary
7	108	Prof	A	18.0	18	Male	NaN
13	114	Prof	B	NaN	33	Male	162200.0
28	129	AsstProf	B	7.0	2	Male	NaN
34	135	AssocProf	B	NaN	8	Male	119800.0

In [85]:

```
df['phd']=df['phd'].fillna(df['phd'].mean())
```

In [86]:

```
df[df.isnull().any(axis=1)]
```

Out[86]:

	empid	rank	discipline	phd	service	gender	salary
7	108	Prof	A	18.0	18	Male	NaN
28	129	AsstProf	B	7.0	2	Male	NaN

In [88]:

```
df[df.isnull().any(axis=1)]
```

Out[88]:

	empid	rank	discipline	phd	service	gender	salary
7	108	Prof	A	18.0	18	Male	NaN
28	129	AsstProf	B	7.0	2	Male	NaN

In [91]:

```
# 2. Delete the rows which has missing values
By default, the dropna() method returns a new DataFrame, and will not change the original.

If you want to change the original DataFrame, use the inplace = True argument:
df.dropna(inplace=True)
```

In [92]:

```
df[df.isnull().any(axis=1)]
```

Out[92]:

empid	rank	discipline	phd	service	gender	salary

In [93]:

```
df.shape
```

Out[93]:

```
(76, 7)
```

How to add rows into dataframe

In [94]:

```
from IPython.display import display
df = pd.DataFrame(df)
display(df)
```

	empid	rank	discipline	phd	service	gender	salary
0	101	Prof	B	56.0	49	Male	186960.0
1	102	Prof	A	12.0	6	Male	93000.0
2	103	Prof	A	23.0	20	Male	110515.0
3	104	Prof	A	40.0	31	Male	131205.0
4	105	Prof	B	20.0	18	Male	104800.0
...	...	...	...	...	...	...	...
73	174	Prof	B	18.0	10	Female	105450.0
74	175	AssocProf	B	19.0	6	Female	104542.0
75	176	Prof	B	17.0	17	Female	124312.0
76	177	Prof	A	28.0	14	Female	109954.0
77	178	Prof	A	23.0	15	Female	109646.0

76 rows × 7 columns

In [80]:

```
df2={'empid':1001,'rank':'Prof','discipline':'B','phd':13,'service':5,'gender':'Male','salary':'NaN'}
df = df.append(df2, ignore_index = True)
display(df)
```

	empid	rank	discipline	phd	service	gender	salary	emp
0	1001.0	Prof	B	13.0	5	Male	NaN	NaN

How to delete columns in dataframe

In [96]:

```
del df['phd']
```

In [97]:

```
df.head()
```

Out[97]:

	empid	rank	discipline	service	gender	salary
0	101	Prof	B	49	Male	186960.0
1	102	Prof	A	6	Male	93000.0
2	103	Prof	A	20	Male	110515.0
3	104	Prof	A	31	Male	131205.0
4	105	Prof	B	18	Male	104800.0

How to access partial dataframe

In [99]:

```
"""
iloc is integer index based, so you have to specify rows and columns
by their integer index
"""
#df.iloc[row_selection,col_selection]

print(df.iloc[0:10,0:2])
```

```
    empid       rank
0     101       Prof
1     102       Prof
2     103       Prof
3     104       Prof
4     105       Prof
5     106       Prof
6     107  AssocProf
8     109       Prof
9     110       Prof
10    111       Prof
```

In [28]:

```
df=pd.read_csv('Salaries.csv')
```

In [29]:
```
print(df.iloc[0:10,0:2])
```

```
   empid       rank
0    101       Prof
1    102       Prof
2    103       Prof
3    104       Prof
4    105       Prof
5    106       Prof
6    107  AssocProf
7    108       Prof
8    109       Prof
9    110       Prof
```

In [103]:
```
print(df.iloc[10,:])
```

```
empid            111
rank            Prof
discipline         B
phd               39
service           33
gender          Male
salary        128250
Name: 10, dtype: object
```

In [104]:
```
print(df.iloc[[10,15],:])
```

```
    empid  rank discipline   phd  service gender    salary
10    111  Prof          B  39.0       33   Male  128250.0
15    116  Prof          B  17.0        3   Male  150480.0
```