In [ ]:

```
# PANDAS
## Pandas is an open source.
## BSD-licensed library providing high-performance.
## EAsy to use  data structures and data analysis tools for the python programming la
```

In [1]:

```
# What is a Data Frames?
# What is DAta SEries?
# Different Operation in Pandas
```

In [2]:

```
## step- one
import pandas as pd
import numpy as np
```

In [3]:

```
# Data Frame
df=pd.DataFrame(np.arange(0,20).reshape(5,4),
index=['Row1','Row2','Row3','Row4','Row5'],
columns=["column1","column2","column3","column4"])

```

In [4]:

```
df
```

Out[4]:

	column1	column2	column3	column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [34]:

```
df['column1'] # one column
```

Out[34]:

```
Row1     0
Row2     4
Row3     8
Row4    12
Row5    16
Name: column1, dtype: int32
```

In [35]:

```
1 type(df['column3'])
```

Out[35]:

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

In [36]:

```
1 df)
```

```
  File "<ipython-input-36-c993fb781b14>", line 1
    df)
      ^
SyntaxError: invalid syntax
```

In [37]:

```
1 df[['column2','column3']]
```

Out[37]:

	column2	column3
Row1	1	2
Row2	5	6
Row3	9	10
Row4	13	14
Row5	17	18

In [38]:

```
1 df.head()
```

Out[38]:

	column1	column2	column3	column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [40]:

```
1 df.to_csv('test6.csv')
```

In [14]:

```
# Accessing the Elements

# 1  .loc (type of Row index)  2 .iloc(index location)

df.loc['Row1']
```

Out[14]:

```
column1    0
column2    1
column3    2
column4    3
Name: Row1, dtype: int32
```

In [15]:

```
type(df.loc['Row1'])
```

Out[15]:

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

In [16]:

```
df.iloc[:,:]  # left side is an row index,  right side is an column index
```

Out[16]:

	column1	column2	column3	column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [17]:

```
df.iloc[0:3,0:2]
```

Out[17]:

	column1	column2
Row1	0	1
Row2	4	5
Row3	8	9

In [18]:

```
type(df.iloc[0:3,0:2])
```

Out[18]:

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

In [19]:

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

Out[19]:

	column2	column3	column4
Row1	1	2	3
Row2	5	6	7
Row3	9	10	11
Row4	13	14	15
Row5	17	18	19

In [20]:

```
df.iloc[0:3,0:2]
```

Out[20]:

	column1	column2
Row1	0	1
Row2	4	5
Row3	8	9

In [21]:

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

Out[21]:

```
Row1    0
Row2    4
Row3    8
Name: column1, dtype: int32
```

type(df.iloc[0:3,0]) # if we have more then one column it is an data frame

In [22]:

```
## Take some elements from the column
df.iloc[:,1:]
```

Out[22]:

	column2	column3	column4
Row1	1	2	3
Row2	5	6	7
Row3	9	10	11
Row4	13	14	15
Row5	17	18	19

In [23]:

```
# Converting DAta frame into an Array
df.iloc[:,1:].values # if we converting our data frame into an
                     #array just skip the column name or
                     #row indexes or column indexes
```

Out[23]:

```
array([[ 1,  2,  3],
       [ 5,  6,  7],
       [ 9, 10, 11],
       [13, 14, 15],
       [17, 18, 19]])
```

In [24]:

```
df.iloc[:,1:].values.shape
```

Out[24]:

```
(5, 3)
```

In [25]:

```
df['column1'].value_counts()
```

Out[25]:

```
12    1
4     1
16    1
8     1
0     1
Name: column1, dtype: int64
```

In [26]:

```
# operations with data frames
# How to check null condition
df.isnull().sum()
```

Out[26]:

```
column1    0
column2    0
column3    0
column4    0
dtype: int64
```

In [45]:

```
# i want to find how many unique categories in this case i can gor for:
df.loc['Row1'].value_counts()
```

Out[45]:

```
3    1
2    1
1    1
0    1
Name: Row1, dtype: int64
```

In [28]:

```
df['column1'].unique() # unique value
```

Out[28]:

```
array([ 0,  4,  8, 12, 16], dtype=int64)
```

In [29]:

```
test_df=pd.read_csv('test1.csv')
```

In [30]:

```
test_df.head()
```

Out[30]:

	Unnamed: 0	column1	column2	column3	column4
0	Row1	0	1	2	3
1	Row2	4	5	6	7
2	Row3	8	9	10	11
3	Row4	12	13	14	15
4	Row5	16	17	18	19

In [31]:

```
test_df=pd.read_csv('test1.csv',sep=';')
```

In [32]:

```
test_df.head()
```

Out[32]:

	,column1,column2,column3,column4
0	Row1,0,1,2,3
1	Row2,4,5,6,7
2	Row3,8,9,10,11
3	Row4,12,13,14,15
4	Row5,16,17,18,19

In [3]:

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

In [6]:

```
df
```

Out[6]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X3
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0	
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	

4209 rows × 378 columns

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

In [8]:

```
1 df.describe()
```

Out[8]:

	ID	y	X10	X11	X12	X13	X14
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000

8 rows × 370 columns

In [40]:

```
1 df['X0'].value_counts()# get category count
```

Out[40]:

```
z     360
ak    349
y     324
ay    313
t     306
x     300
o     269
f     227
n     195
w     182
j     181
az    175
aj    151
s     106
ap    103
h      75
d      73
al     67
v      36
af     35
m      34
ai     34
e      32
ba     27
at     25
a      21
ax     19
aq     18
am     18
i      18
u      17
aw     16
l      16
ad     14
au     11
k      11
b      11
r      10
as     10
bc      6
ao      4
c       3
aa      2
q       2
ac      1
g       1
ab      1
Name: X0, dtype: int64
```

In [41]:

```
df[df['y']>100]
```

Out[41]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X3
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	
6	24	128.76	al	r	e	f	d	f	h	s	...	0	0	0	0	0	
8	27	108.67	w	s	as	e	d	f	i	h	...	1	0	0	0	0	
9	30	126.99	j	b	aq	c	d	f	a	e	...	0	0	1	0	0	
10	31	102.09	h	r	r	f	d	f	h	p	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4202	8402	123.34	ap	l	s	c	d	aa	d	r	...	0	0	0	0	0	
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	

2004 rows × 378 columns

In [5]:

```
# CSV
from io import StringIO,BytesIO
import pandas as pd
```

In [2]:

```
data=('col1,col2,col3\n'
              'x,y,1\n'
              'a,b,2\n'
              'c,d,3')
```

In [3]:

```
type(data)
```

Out[3]:

```
str
```

In [6]:

```
pd.read_csv(StringIO(data))
```

Out[6]:

	col1	col2	col3
0	x	y	1
1	a	b	2
2	c	d	3

In [22]:

```
## Read from specific columns
df=pd.read_csv(StringIO(data),usecols=['col1','col3'])
```

In [23]:

```
df
```

Out[23]:

	col1	col3
0	x	1
1	a	2
2	c	3

In [24]:

```
df.to_csv('Test.csv')
```

In [32]:

```
# secifying columns data types

data=('a,b,c,d\n'
          '1,2,3,4\n'
           '5,6,7,8\n'
            '9,10,11,12')

```

In [33]:

```
print(data)
```

```
a,b,c,d
1,2,3,4
5,6,7,8
9,10,11,12
```

In [36]:

```
df=pd.read_csv(StringIO(data),dtype=object) # int,float
```

In [37]:

```
df
```

Out[37]:

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

In [29]:

```
df['a']
```

Out[29]:

```
0    1
1    5
2    9
Name: a, dtype: object
```

In [34]:

```
df['a'][1]
```

Out[34]:

```
'5'
```

In [39]:

```
# i can specify differrnt data in each differetn column
df=pd.read_csv(StringIO(data),dtype={'b':int,'c':float,'a':'Int64'})
```

In [40]:

```
df
```

Out[40]:

	a	b	c	d
0	1	2	3.0	4
1	5	6	7.0	8
2	9	10	11.0	12

In [45]:

```
df['a'][2]
```

Out[45]:

```
9
```

In [46]:

```
## Check the datatype
df.dtypes
```

Out[46]:

```
a      Int64
b      int32
c    float64
d      int64
dtype: object
```

In [5]:

```
## Index columns and training delimiters

data=('index,a,b,c\n'
       '4,apple,bat,5.7\n'
         '8,orange,cow,10')
```

In [9]:

```
pd.read_csv(StringIO(data))
```

Out[9]:

	index	a	b	c
0	4	apple	bat	5.7
1	8	orange	cow	10.0

In [10]:

```
pd.read_csv(StringIO(data),index_col=0) # we have a parameter to make  parameter as c
```

Out[10]:

	a	b	c
index			
4	apple	bat	5.7
8	orange	cow	10.0

In [15]:

```
data=('a,b,c\n'
       '4,apple,bat,\n'
         '8,orange,cow,')
```

In [16]:

```
pd.read_csv(StringIO(data)) # to gandle the NAN Issue we have index_cols
```

Out[16]:

	a	b	c
4	apple	bat	NaN
8	orange	cow	NaN

In [20]:

```
pd.read_csv(StringIO(data),index_col=False)
```

Out[20]:

	a	b	c
0	4	apple	bat
1	8	orange	cow

In [21]:

```
# I can also Combine usecols and index_col
data=('a,b,c\n'
        '4,apple,b at,\n'
         '8,orange,cow,')
```

In [24]:

```
pd.read_csv(StringIO(data),usecols=['b','c'],index_col=False)
```

Out[24]:

	a	b	c
0	4	apple	b at
1	8	orange	cow

In [29]:

```
## quoting and Escape Characters, Very useful in NLP
data='a,b\n"hello, \\"Bob\\", nice to see you",5'
```

In [33]:

```
pd.read_csv(StringIO(data),escapechar='\\') # in built parameter using fo text charac
```

Out[33]:

	a	b
0	hello, "Bob", nice to see you	5

In [10]:

```
# in some of our data set '/t' as a separater
## URL to CSV

#df=pd.read_csv('https://github.com/cs109/2014_data/blob/master/countries.item',sep=
df=pd.read_csv('https://download.bls.gov/pub/time.series/cu/cu.item',sep='\t')
```

In [11]:

```
df.head()
```

Out[11]:

	item_code	item_name	display_level	selectable	sort_sequence
0	AA0	All items - old base	0	T	2
1	AA0R	Purchasing power of the consumer dollar - old ...	0	T	400
2	SA0	All items	0	T	1
3	SA0E	Energy	1	T	375
4	SA0L1	All items less food	1	T	359

In [ ]:

```

```