```
#Pandas is used for data manupulation,Analysis and cleaning.
#Library contain the many function.
```

```
import pandas as pd
```

```
import numpy as np
```

Data Structure in pandas:

Series, Dataframe, Panel

- A Pandas Series is like a column in a table
- It is a one-dimensional array holding data of any type.

Create Series

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)

type(myvar)
```

```
0    1
1    7
2    2
dtype: int64
pandas.core.series.Series
```

Change index

```
b = [1, 7, 2]

var = pd.Series(b,index=['a','b','c'],dtype=float)   #change index and data type

print(var)
```

```
a    1.0
b    7.0
c    2.0
dtype: float64
```

Create DataFrame : 2 dimensonal array

```
a=[1,'sunil','divy','best']

df=pd.DataFrame(a)

df
```

	0
0	1
1	sunil
2	divy
3	best

```
b={'a':[10,20,30],'b':[40,50,60],'c':70,}

dfn=pd.DataFrame(b)

dfn
```

	a	b	c
0	10	40	70
1	20	50	70
2	30	60	70

import csv file

```
df=pd.read_csv("/content/appl_stock.csv",nrows=10,usecols=["Date",'Open','High'])     #imp

df
```

	Date	Open	High
0	2010-01-04	213.429998	214.499996
1	2010-01-05	214.599998	215.589994
2	2010-01-06	214.379993	215.230000
3	2010-01-07	211.750000	212.000006
4	2010-01-08	210.299994	212.000006
5	2010-01-11	212.799997	213.000002
6	2010-01-12	209.189995	209.769995
7	2010-01-13	207.870005	210.929995
8	2010-01-14	210.110003	210.459997
9	2010-01-15	210.929995	211.599997

```
type(df)
```

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

Write csv file

```
df.to_csv("/content/new_Stock_file.csv")     #saved in csv file format
```

…

```
dfn=pd.read_csv("/content/appl_stock.csv",skiprows=[1,2,3])  #Skip rows 1,2,3

dfn
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
1	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034

```
df1=pd.read_csv("/content/appl_stock.csv",index_col=0)  #make index of perticular rows
```

```
df1
```

Date	Open	High	Low	Close	Volume	Adj Close
2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...
2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

1762 rows × 6 columns

```
#df2=pd.read_csv("/content/appl_stock.csv",header=0)  #make header to perticular rows
```

```
df2=pd.read_csv("/content/appl_stock.csv", names=['A','B','C','D','E','F','G'])  #if header not avaiable we can
```

```
df2
```

	A	B	C	D	E	F	G
0	Date	Open	High	Low	Close	Volume	Adj Close
1	2010-01-04	213.429998	214.499996	212.38000099999996	214.009998	123432400	27.727039
2	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976000000002
3	2010-01-06	214.379993	215.23	210.750004	210.969995	138040000	27.333178000000004
4	2010-01-07	211.75	212.000006	209.050005	210.58	119282800	27.28265
...	...	...	...	...	...	...	...
1758	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
1759	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.75380600000001
1760	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.25596499999999
1761	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1762	2016-12-30	116.650002	117.199997	115.43	115.82	30586300	115.32002

1763 rows × 7 columns

```
df2=pd.read_csv("/content/appl_stock.csv",dtype={'Open':'float64'})  #chnage data type of cloumn
```

```
df2
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650

check null value if avaiable it will write True

1757 2016-12-23 115.589996 116.519997 115.589996 116.519997 14249500 116.016995

```
df2.isnull()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1757	False	False	False	False	False	False	False
1758	False	False	False	False	False	False	False
1759	False	False	False	False	False	False	False
1760	False	False	False	False	False	False	False
1761	False	False	False	False	False	False	False

1762 rows × 7 columns

check total null value per column

```
df2.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Volume       0
Adj Close    0
dtype: int64
```

```
df2.notnull()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...
1757	True	True	True	True	True	True	True
1758	True	True	True	True	True	True	True
1759	True	True	True	True	True	True	True
1760	True	True	True	True	True	True	True
1761	True	True	True	True	True	True	True

1762 rows × 7 columns

check not null value

```
df2.notnull().sum()
```

```
Date         1762
Open         1762
High         1762
Low          1762
Close        1762
Volume       1762
Adj Close    1762
dtype: int64
```

```
df2.notnull().sum().sum()
```

```
12334
```

drop the blank rows

```
df2.dropna()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
4	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...	...
1757	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
1758	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
1759	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
1760	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1761	2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

1762 rows × 7 columns

drop the null value rows from perticular column

```
df2.dropna(subset=['Date'])
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
4	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...	...
1757	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
1758	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
1759	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
1760	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1761	2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

1762 rows × 7 columns

fill null value with 0

```
df2.fillna(0)
```

fill null value with particular column

```
df2.fillna({'Date':0,'Open':2})
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
4	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...	...
1757	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
1758	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
1759	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
1760	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1761	2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

1762 rows × 7 columns

replace old value with new value

```
df2.replace('2010-01-04',0)
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	0	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
4	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...	...
1757	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995
1758	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
1759	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
1760	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1761	2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

1762 rows × 7 columns

replace old value with new value on particular column

```
df2.replace({'Date':'2010-01-05'},0)
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	0	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650
4	2010-01-08	210.299994	212.000006	209.060005	211.980005	111902700	27.464034
...	...	...	...	...	...	...	...
1757	2016-12-23	115.589996	116.519997	115.589996	116.519997	14249500	116.016995

Group by

```
df2=pd.read_csv("/content/sales_info.csv",nrows=10)

df2
```

	Company	country	Person	Sales
0	GOOG	IND	Sam	200
1	GOOG	USA	Charlie	120
2	GOOG	IND	Frank	340
3	MSFT	USA	Tina	600
4	MSFT	PAK	Amy	124
5	MSFT	IND	Vanessa	243
6	FB	USA	Carl	870
7	FB	PAK	Sarah	350
8	APPL	IND	John	250
9	APPL	PAK	Linda	130

```
b=df2.groupby('Company').get_group('GOOG')

b
```

	Company	country	Person	Sales
0	GOOG	IND	Sam	200
1	GOOG	USA	Charlie	120
2	GOOG	IND	Frank	340

Merge join

```
m1=pd.DataFrame({'id':[1,2,3],'Name':['A','B','C']})

m1
```

	id	Name
0	1	A
1	2	B
2	3	C

```
m2=pd.DataFrame({'id':[1,2,3,4,6],'CLASS':[7,8,8,9,7]})
m2
```

	id	CLASS
0	1	7
1	2	8
2	3	8
3	4	9

```
pd.merge(m2,m1, on='id')   #merge
```

	id	CLASS	Name
0	1	7	A
1	2	8	B
2	3	8	C

```
pd.merge(m1,m2, on='id')
```

	id	Name	CLASS
0	1	A	7
1	2	B	8
2	3	C	8

```
pd.merge(m1,m2, on='id',how='inner')    #inner
```

	id	Name	CLASS
0	1	A	7
1	2	B	8
2	3	C	8

```
pd.merge(m2,m1, on='id',how='left')    #left
```

	id	CLASS	Name
0	1	7	A
1	2	8	B
2	3	8	C
3	4	9	NaN
4	6	7	NaN

concat union

```
pd.concat([m1,m2])
```

	id	Name	CLASS
0	1	A	NaN
1	2	B	NaN
2	3	C	NaN
0	1	NaN	7.0
1	2	NaN	8.0
2	3	NaN	8.0
3	4	NaN	9.0
4	6	NaN	7.0

```
pd.concat([m1,m2],axis=1)
```

	id	Name	id	CLASS
0	1.0	A	1	7
1	2.0	B	2	8
2	3.0	C	3	8

Pandas Function

```
df.columns
```

```
Index(['Date', 'Open', 'High'], dtype='object')
```

```
df.iloc[0:5]
```

	Date	Open	High
0	2010-01-04	213.429998	214.499996
1	2010-01-05	214.599998	215.589994
2	2010-01-06	214.379993	215.230000
3	2010-01-07	211.750000	212.000006
4	2010-01-08	210.299994	212.000006

Get Top value

```
df.head(4)
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178
3	2010-01-07	211.750000	212.000006	209.050005	210.580000	119282800	27.282650

Get LAST value

```
df.tail(4)
```

	Date	Open	High	Low	Close	Volume	Adj Close
1758	2016-12-27	116.519997	117.800003	116.489998	117.260002	18296900	116.753806
1759	2016-12-28	117.519997	118.019997	116.199997	116.760002	20905900	116.255965
1760	2016-12-29	116.449997	117.110001	116.400002	116.730003	15039500	116.226096
1761	2016-12-30	116.650002	117.199997	115.430000	115.820000	30586300	115.320020

Check number of row and column

```
df.shape
```

```
(1762, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1762 entries, 0 to 1761
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1762 non-null   object
 1   Open       1762 non-null   float64
```

```
 2   High       1762 non-null   float64
 3   Low        1762 non-null   float64
 4   Close      1762 non-null   float64
 5   Volume     1762 non-null   int64
 6   Adj Close  1762 non-null   float64
dtypes: float64(5), int64(1), object(1)
memory usage: 96.5+ KB
```

```
df.describe()
```

	Open	High	Low	Close	Volume	Adj Close
count	1762.000000	1762.000000	1762.000000	1762.000000	1.762000e+03	1762.000000
mean	313.076311	315.911288	309.828241	312.927066	9.422578e+07	75.001741
std	185.299468	186.898177	183.383917	185.147104	6.020519e+07	28.574930
min	90.000000	90.699997	89.470001	90.279999	1.147590e+07	24.881912
25%	115.222498	116.362499	114.002500	115.190002	4.917478e+07	50.288540
50%	318.230007	320.600008	316.545002	318.240008	8.050385e+07	72.983145
75%	470.880017	478.110008	467.972513	472.592512	1.210816e+08	100.207243
max	702.409988	705.070023	699.569977	702.100021	4.702495e+08	127.966091

```
a=df["Open"]
```

```
type(a)
```

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

```
df[["Open","High"]]
```

	Open	High
0	213.429998	214.499996
1	214.599998	215.589994
2	214.379993	215.230000
3	211.750000	212.000006
4	210.299994	212.000006
...	...	...
1757	115.589996	116.519997
1758	116.519997	117.800003
1759	117.519997	118.019997
1760	116.449997	117.110001
1761	116.650002	117.199997

1762 rows × 2 columns

fetch particular row

```
df.iloc[0:3]           #First row from dataframe
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	2010-01-04	213.429998	214.499996	212.380001	214.009998	123432400	27.727039
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976
2	2010-01-06	214.379993	215.230000	210.750004	210.969995	138040000	27.333178

```
df.iloc[[1,2,4]]       #Fancy Indexing
```

```
            Date       Open       High        Low      Close     Volume  Adi Close
date=df['Date']=='2010-01-05'

df[date]
```

	Date	Open	High	Low	Close	Volume	Adj Close
1	2010-01-05	214.599998	215.589994	213.249994	214.379993	150476200	27.774976

```
df.iloc[:,[1,2]]      #few number of column
```

	Open	High
0	213.429998	214.499996
1	214.599998	215.589994
2	214.379993	215.230000
3	211.750000	212.000006
4	210.299994	212.000006
...	...	...
1757	115.589996	116.519997
1758	116.519997	117.800003
1759	117.519997	118.019997
1760	116.449997	117.110001
1761	116.650002	117.199997

1762 rows × 2 columns

Filtering data

```
filter_data=df["Open"]<220

filter_data        #show output in boolen
```

```
0       True
1       True
2       True
3       True
4       True
        ...
1757    True
1758    True
1759    True
1760    True
1761    True
Name: Open, Length: 1762, dtype: bool
```

```
df[filter_data]
```

```
(692, 7)
```

```
def value_on_date(date):
   dat=df['Date']==date
   return df[dat]

value_on_date('2010-01-06')
```

	Date	Open	High	Low	Close	Volume	Adj Close
2	2010-01-06	214.379993	215.23	210.750004	210.969995	138040000	27.333178

```
df1=pd.read_csv('/content/flightdata.csv')

df1.head()
```

```
df1['ORIGIN_COUNTRY_NAME'].value_counts()
```

```
United States             125
Romania                     1
Curacao                     1
Saint Lucia                 1
British Virgin Islands      1
                         ...
Egypt                       1
Dominican Republic          1
Kuwait                      1
Vietnam                     1
Uganda                      1
Name: ORIGIN_COUNTRY_NAME, Length: 131, dtype: int64
```

```
df1['count']+100
```

sort

```
df1.sort_values('count',ascending=False,inplace=True)      #inplace=True  make changes in original df permanent
```

```
df1
```

```
#sort based on multiple column
```

```
df1.sort_values(['count','ORIGIN_COUNTRY_NAME'])
```

	DEST_COUNTRY_NAME	ORIGIN_COUNTRY_NAME	count
60	United States	Algeria	1
91	United States	Azerbaijan	1
168	United States	Bahrain	1
45	United States	Bosnia and Herzegovina	1
212	United States	Burkina Faso	1
...	...	...	...
42	Mexico	United States	6200
139	United States	Mexico	6220
115	Canada	United States	8271
209	United States	Canada	8305
81	United States	United States	348113

255 rows × 3 columns

Drop_duplicates()

```
dff=pd.read_csv('/content/sales_info.csv')
```

```
dff
```

	Company	country	Person	Sales
0	GOOG	IND	Sam	200
1	GOOG	USA	Charlie	120
2	GOOG	IND	Frank	340
3	MSFT	USA	Tina	600

```
dff.drop_duplicates('Company')
```

	Company	country	Person	Sales
0	GOOG	IND	Sam	200
3	MSFT	USA	Tina	600
6	FB	USA	Carl	870
8	APPL	IND	John	250

Group By

```
grp=dff.groupby('Company')
```

```
grp.groups
```

```
{'APPL': [8, 9, 10, 11], 'FB': [6, 7], 'GOOG': [0, 1, 2], 'MSFT': [3, 4, 5]}
```

```
grp.get_group('APPL')
```

	Company	country	Person	Sales
8	APPL	IND	John	250
9	APPL	PAK	Linda	130
10	APPL	PAK	Mike	750
11	APPL	USA	Chris	350

Mising value Handle

```
##dropna   :drop missing data
```

```
ms=pd.read_csv("/content/Missing_Data.csv")
```

```
ms
```

	Company	Person	Sales
0	GOOG	Sam	200.0
1	GOOG	Charlie	120.0
2	GOOG	Frank	340.0
3	MSFT	Tina	NaN
4	MSFT	Amy	124.0
5	MSFT	Vanessa	243.0
6	FB	NaN	870.0
7	FB	Sarah	350.0
8	APPL	John	NaN
9	APPL	Linda	130.0
10	APPL	Mike	NaN
11	APPL	Chris	350.0

```
df=pd.DataFrame(np.random.randn(12,4),columns=['A','B','C','D'])
```

```
df
```

	A	B	C	D
0	0.196617	-0.602811	-1.982075	0.794597
1	-0.111764	2.328204	0.182549	-0.470315
2	0.978815	-0.746215	-0.119509	-0.179377
3	0.374885	-0.135870	-2.167354	-1.003264
4	-0.293240	1.029291	-0.197197	0.304553
5	-0.297673	-0.562963	-0.189475	-0.869380
6	2.309035	0.879439	0.716171	-0.314853
7	-0.499882	-0.903105	0.218377	0.759849
8	0.600505	-1.455509	-0.378238	1.360625
9	-0.556043	0.011516	1.155717	-0.019206
10	0.138658	-1.148449	0.159540	1.367617
11	-0.494426	0.473608	-0.126364	-0.375567

```
df[['A','B']]  #SELCT COLUMN
```

	A	B
0	0.196617	-0.602811
1	-0.111764	2.328204
2	0.978815	-0.746215
3	0.374885	-0.135870
4	-0.293240	1.029291
5	-0.297673	-0.562963
6	2.309035	0.879439
7	-0.499882	-0.903105
8	0.600505	-1.455509
9	-0.556043	0.011516
10	0.138658	-1.148449
11	-0.494426	0.473608

```
df['new']=df['A']+df['B']     #create new column
```

```
df
```

	A	B	C	D	new
0	0.196617	-0.602811	-1.982075	0.794597	-0.406195
1	-0.111764	2.328204	0.182549	-0.470315	2.216440
2	0.978815	-0.746215	-0.119509	-0.179377	0.232601
3	0.374885	-0.135870	-2.167354	-1.003264	0.239015
4	-0.293240	1.029291	-0.197197	0.304553	0.736051
5	-0.297673	-0.562963	-0.189475	-0.869380	-0.860636
6	2.309035	0.879439	0.716171	-0.314853	3.188474
7	-0.499882	-0.903105	0.218377	0.759849	-1.402987
8	0.600505	-1.455509	-0.378238	1.360625	-0.855004
9	-0.556043	0.011516	1.155717	-0.019206	-0.544527
10	0.138658	-1.148449	0.159540	1.367617	-1.009791
11	-0.494426	0.473608	-0.126364	-0.375567	-0.020818

```
df.drop('A',axis=1,inplace=True)   #drop column
```

```
df
```

	B	C	D	new
0	-0.602811	-1.982075	0.794597	-0.406195
1	2.328204	0.182549	-0.470315	2.216440
2	-0.746215	-0.119509	-0.179377	0.232601
3	-0.135870	-2.167354	-1.003264	0.239015
4	1.029291	-0.197197	0.304553	0.736051
5	-0.562963	-0.189475	-0.869380	-0.860636
6	0.879439	0.716171	-0.314853	3.188474
7	-0.903105	0.218377	0.759849	-1.402987
8	-1.455509	-0.378238	1.360625	-0.855004
9	0.011516	1.155717	-0.019206	-0.544527
10	-1.148449	0.159540	1.367617	-1.009791
11	0.473608	-0.126364	-0.375567	-0.020818

```
df[df['B']>0]
```

	B	C	D	new
1	2.328204	0.182549	-0.470315	2.216440
4	1.029291	-0.197197	0.304553	0.736051
6	0.879439	0.716171	-0.314853	3.188474
9	0.011516	1.155717	-0.019206	-0.544527
11	0.473608	-0.126364	-0.375567	-0.020818

```
#and &     or |

result = df[(df['B'] > 0) & (df['C'] > 0)]

result
```

	B	C	D	new
1	2.328204	0.182549	-0.470315	2.216440
6	0.879439	0.716171	-0.314853	3.188474
9	0.011516	1.155717	-0.019206	-0.544527

Pivot

fillna

```
dff['country'].fillna("data missing")
```

Arithmetic Operation

```
var=pd.DataFrame({'A':[1,2,3],'B':[4,5,6]})

var
```

	A	B
0	1	4
1	2	5

```
var['C']=var['A']+var['B']
```

```
var
```

	A	B	C
0	1	4	5
1	2	5	7
2	3	6	9

```
var
```

	A	B	C
0	1	4	5
1	2	5	7
2	3	6	9

```
var["N"]=var["B"]<6
```

```
var
```

	A	B	C	N
0	1	4	5	True
1	2	5	7	True
2	3	6	9	False

Insert

```
var.insert(3,"new_column",var["A"])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-161f396c2fd6> in <cell line: 1>()
----> 1 var.insert(3,"new_column",var["A"])
      2

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in insert(self, loc, column, value, allow_duplicates)
   4815         if not allow_duplicates and column in self.columns:
   4816             # Should this be a different kind of error??
-> 4817             raise ValueError(f"cannot insert {column}, already exists")
   4818         if not isinstance(loc, int):
   4819             raise TypeError("loc must be int")

ValueError: cannot insert new_column, already exists
```

SEARCH STACK OVERFLOW

```
var
```

	A	B	C	N	new_column
0	1	4	5	True	1
1	2	5	7	True	2
2	3	6	9	False	3

```
var.insert(4,"new_column_1",var["A"][0:1])
```

```
var
```

	A	B	C	N	new_column_1	new_column
0	1	4	5	True	1.0	1
1	2	5	7	True	NaN	2
2	3	6	9	False	NaN	3

Delete

```
var1=var.pop("N")
```

```
var1
```

```
0     True
1     True
2    False
Name: N, dtype: bool
```

```
var
```

	A	B	C	new_column_1	new_column
0	1	4	5	1.0	1
1	2	5	7	NaN	2
2	3	6	9	NaN	3

```
del var["new_column_1"]
```

```
var
```

	A	B	C	new_column
0	1	4	5	1
1	2	5	7	2
2	3	6	9	3

Write Into CSV File

```
 var.to_csv("New")
```

Python project

```
a=pd.Series([1,2,3,4,5,6,'ss'])    #create series
```

```
a
```

```
0     1
1     2
2     3
3     4
4     5
5     6
6    ss
dtype: object
```

```
type(a)
```

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

```
a[6]
```

```
'ss'
```

```
b=pd.Series([1,2,3],index=["a","b","c"],dtype=float)
```

```
b
```

```
a    1.0
b    2.0
c    3.0
dtype: float64
```

```
di={"name":["sunil","Divya","Pillu","Baby"],"rank":[1,2,3,4]}
```

```
di
```

```
{'name': ['sunil', 'Divya', 'Pillu', 'Baby'], 'rank': [1, 2, 3, 4]}
```

```
type(di)
```

```
dict
```

```
ab=pd.Series(di)
```

```
ab
```

```
name    [sunil, Divya, Pillu, Baby]
rank                  [1, 2, 3, 4]
dtype: object
```

.....

```
s=[1,2,3,4]
```

```
v=pd.DataFrame(s)
```

```
v
```

	0
0	1
1	2
2	3
3	4

```
di={"name":["sunil","Divya","Pillu","Baby"],"rank":[1,2,3,4]}
```

```
s=pd.DataFrame(di)
```

```
s
```

	name	rank
0	sunil	1
1	Divya	2
2	Pillu	3
3	Baby	4

```
d=pd.date_range('20201201',periods=12)
```

```
d
```

```
DatetimeIndex(['2020-12-01', '2020-12-02', '2020-12-03', '2020-12-04',
               '2020-12-05', '2020-12-06', '2020-12-07', '2020-12-08',
               '2020-12-09', '2020-12-10', '2020-12-11', '2020-12-12'],
              dtype='datetime64[ns]', freq='D')
```

```
df=pd.DataFrame(np.random.randn(12,4), index=d,columns=['A','B','C','D'])
```

```
df
```

	A	B	C	D
2020-12-01	-0.975400	0.407957	1.951855	0.002358
2020-12-02	0.571167	0.645069	0.415742	-0.486392
2020-12-03	0.346904	1.563464	0.573322	-1.347443
2020-12-04	-1.389661	0.435281	0.725327	0.840646
2020-12-05	-0.469914	1.352876	0.093389	-0.659755
2020-12-06	0.821686	-0.922762	1.449260	0.553344
2020-12-07	-0.764062	1.511144	0.093776	0.600055
2020-12-08	1.328048	0.000159	-0.858375	-0.079920
2020-12-09	1.639707	-0.259061	-0.803596	-0.564940
2020-12-10	0.454399	0.219602	0.787052	0.986444
2020-12-11	-0.021031	0.572862	-1.578101	-0.782173
2020-12-12	-0.346258	-1.889512	0.007190	-0.097038

```
df1=pd.DataFrame({'A':[1,2,3,4],'B':[45,6,7,8],'C':[9,10,11,12]})
```

```
df1
```

	A	B	C
0	1	45	9
1	2	6	10
2	3	7	11
3	4	8	12

```
df1.dtypes
```

```
A    int64
B    int64
C    int64
dtype: object
```

```
df1.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
df.head()
```

	A	B	C	D
2020-12-01	-0.975400	0.407957	1.951855	0.002358
2020-12-02	0.571167	0.645069	0.415742	-0.486392
2020-12-03	0.346904	1.563464	0.573322	-1.347443
2020-12-04	-1.389661	0.435281	0.725327	0.840646

```
df1.tail()
```

	A	B	C
0	1	45	9
1	2	6	10
2	3	7	11
3	4	8	12

```
df.columns
```

```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

```
df1.to_numpy()           #array
```

```
array([[ 1, 45,  9],
       [ 2,  6, 10],
       [ 3,  7, 11],
       [ 4,  8, 12]])
```

```
#Viewing data
```