Pandas

About Pandas :

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series

Let's start our tour

First Install Pandas And Import

- pip install pandas
- import pandas as pd

In [4]:
```
import pandas as pd
```


DataFrame And Series:

Series

The Pandas Series data structure is a one-dimensional labelled array. It is the primary building block for a DataFrame

Dictinary

In [6]:
```
ser={'a':1,'b':2,'c':3,'d':4,'e':5}
ser=pd.Series(data=ser)
ser
```

Out[6]:
```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

Now we can also get values based on Index

In [9]:
```
ser={'a':1,'b':2,'c':3,'d':4,'e':5}
ser=pd.Series(data=ser,index=["a","b"])
ser
```

Out[9]:
```
a    1
b    2
dtype: int64
```

List

In [12]:
```
ser1=[1,2,3,4,5]
ser1=pd.Series(data=ser1,index=['a','b','c','d','e'])
ser1
```

Out[12]:
```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

DataFrame

The Pandas DataFrame is a two-dimensional data structure composed of columns and rows. You can think of the DataFrame as similar to a CSV or relational database table

Dictinary

In [16]:
```
scientist={"scientist_Name":["Teja","Tswarup","Swaroop",'Thej'],
           "Domain":['ML','AI','NLP','DL'],
           'Age':[25,27,30,20]}
dfscience=pd.DataFrame(data=scientist)
dfscience
```

Out[16]:

	scientist_Name	Domain	Age
0	Teja	ML	25
1	Tswarup	AI	27
2	Swaroop	NLP	30
3	Thej	DL	20

List

In [19]:
```
scientist=pd.DataFrame([["Teja","ML",25],["Tswarup","AI",27],["Swaroop","NLP"
                        ["Thej","DL",20]],columns=["scientist_name","Domain",
scientist
```

Out[19]:

	scientist_name	Domain	Age
0	Teja	ML	25
1	Tswarup	AI	27
2	Swaroop	NLP	30
3	Thej	DL	20

Perform Some Operations

Lets

ADD

Salary Feature To Scientist Dataset

In [20]:
```
scientist
```

Out[20]:

	scientist_name	Domain	Age
0	Teja	ML	25
1	Tswarup	AI	27
2	Swaroop	NLP	30
3	Thej	DL	20

In [21]:
```
scientist["salary"]=[40000,70000,60000,65000]
```

In [22]:
```
scientist
```

Out[22]:

	scientist_name	Domain	Age	salary
0	Teja	ML	25	40000
1	Tswarup	AI	27	70000
2	Swaroop	NLP	30	60000
3	Thej	DL	20	65000

Lets

DROP

Salary Feature To Scientist Dataset

In [24]:
```
scientist=scientist.drop(columns=["salary"])
scientist
```

Out[24]:

	scientist_name	Domain	Age
0	Teja	ML	25
1	Tswarup	AI	27
2	Swaroop	NLP	30
3	Thej	DL	20

Lets learn New techinque drop Feature or Record By axis

I Don't Want Spoil Dataset I am Copying Dataset into New Dum_scientist

In [25]:
```
Dum_scientist=scientist
```

In [27]:
```
Dum_scientist
```


Out[27]:

	scientist_name	Domain	Age
0	Teja	ML	25
1	Tswarup	AI	27
2	Swaroop	NLP	30
3	Thej	DL	20

axis = 1

- axis =1 Refers Column

In [28]:
```
Dum_scientist=Dum_scientist.drop("Age",axis=1)
Dum_scientist
```

Out[28]:

	scientist_name	Domain
0	Teja	ML
1	Tswarup	AI
2	Swaroop	NLP
3	Thej	DL

axis = 0

- axis = 0 Refers ROW

In [29]:
```
Dum_scientist=Dum_scientist.drop(2,axis=0)
Dum_scientist
```

Out[29]:

	scientist_name	Domain
0	Teja	ML
1	Tswarup	AI
3	Thej	DL

Let's Create 2 Different DataFrame to do more operations

In [32]:
```
fruits1=pd.DataFrame([[3,0,7],[2,3,14],[0,7,6],[1,2,15]],
                     columns=["orange","apple","grapes"])
fruits1
```

Out[32]:

	orange	apple	grapes
0	3	0	7
1	2	3	14
2	0	7	6
3	1	2	15

In [33]:
```
fruits2=pd.DataFrame([[13,10,20,21,31],[12,13,23,24,33],[2,2,4,51,30],[55,9,0
                     ,[96,76,9,25,32]],columns=["grapes","mango","banana","pe
                                                "pineapple"])
fruits2
```

Out[33]:

	grapes	mango	banana	pear	pineapple
0	13	10	20	21	31
1	12	13	23	24	33
2	2	2	4	51	30
3	55	9	0	22	36
4	96	76	9	25	32

Let's Combine 2 DataFrames

Lets

Append

friuts1 to fruits 2

In [35]:
```
AppendOperation=fruits1.append(fruits2)
AppendOperation
```

```
C:\Users\tswar\AppData\Local\Temp/ipykernel_9412/2529817994.py:1: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas i
n a future version. Use pandas.concat instead.
  AppendOperation=fruits1.append(fruits2)
```

Out[35]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	3.0	0.0	7	NaN	NaN	NaN	NaN
1	2.0	3.0	14	NaN	NaN	NaN	NaN
2	0.0	7.0	6	NaN	NaN	NaN	NaN
3	1.0	2.0	15	NaN	NaN	NaN	NaN
0	NaN	NaN	13	10.0	20.0	21.0	31.0
1	NaN	NaN	12	13.0	23.0	24.0	33.0
2	NaN	NaN	2	2.0	4.0	51.0	30.0
3	NaN	NaN	55	9.0	0.0	22.0	36.0
4	NaN	NaN	96	76.0	9.0	25.0	32.0

Lets

Append

friuts1 to fruits 2 Lets

Concat

friuts1 to fruits 2

In [39]:
```
ConcatOperation=pd.concat([fruits1,fruits2])
ConcatOperation
```

Out[39]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	3.0	0.0	7	NaN	NaN	NaN	NaN
1	2.0	3.0	14	NaN	NaN	NaN	NaN
2	0.0	7.0	6	NaN	NaN	NaN	NaN
3	1.0	2.0	15	NaN	NaN	NaN	NaN
0	NaN	NaN	13	10.0	20.0	21.0	31.0
1	NaN	NaN	12	13.0	23.0	24.0	33.0
2	NaN	NaN	2	2.0	4.0	51.0	30.0
3	NaN	NaN	55	9.0	0.0	22.0	36.0
4	NaN	NaN	96	76.0	9.0	25.0	32.0

Let's Play small game with axis

In [40]:
```
ConcatOperation=pd.concat([fruits1,fruits2],axis=1)
ConcatOperation
```


Out[40]:

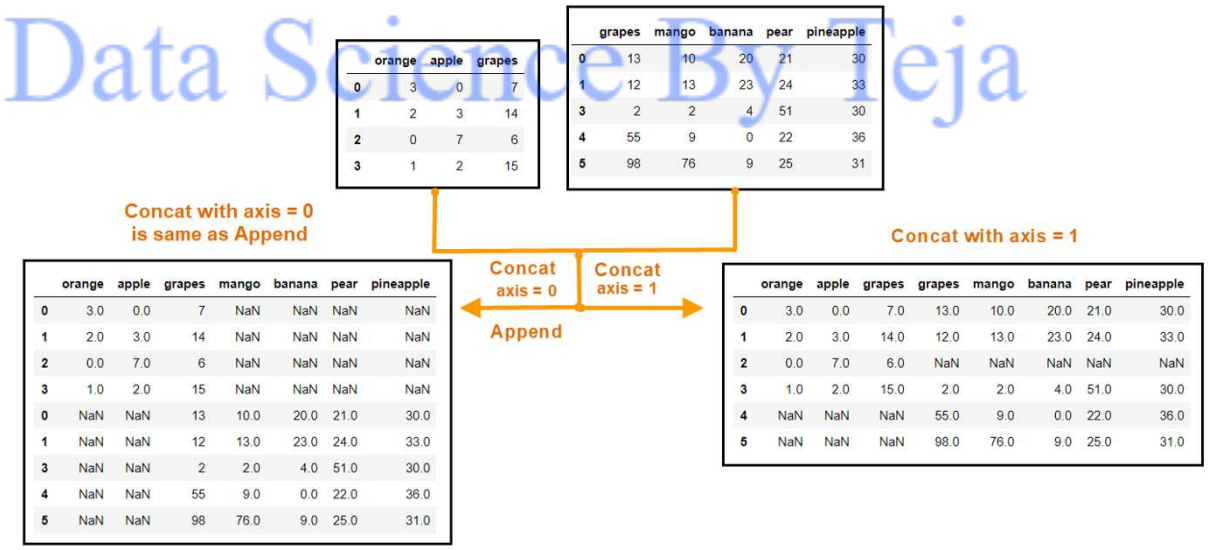
	orange	apple	grapes	grapes	mango	banana	pear	pineapple
0	3.0	0.0	7.0	13	10	20	21	31
1	2.0	3.0	14.0	12	13	23	24	33
2	0.0	7.0	6.0	2	2	4	51	30
3	1.0	2.0	15.0	55	9	0	22	36
4	NaN	NaN	NaN	96	76	9	25	32

In [41]:
```
ConcatOperation=pd.concat([fruits1,fruits2],axis=0)
ConcatOperation
```

Out[41]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	3.0	0.0	7	NaN	NaN	NaN	NaN
1	2.0	3.0	14	NaN	NaN	NaN	NaN
2	0.0	7.0	6	NaN	NaN	NaN	NaN
3	1.0	2.0	15	NaN	NaN	NaN	NaN
0	NaN	NaN	13	10.0	20.0	21.0	31.0
1	NaN	NaN	12	13.0	23.0	24.0	33.0
2	NaN	NaN	2	2.0	4.0	51.0	30.0
3	NaN	NaN	55	9.0	0.0	22.0	36.0
4	NaN	NaN	96	76.0	9.0	25.0	32.0

Let's below image for better understanding



Lets

Merge

friuts1 and fruits 2

Parameter	Value	Description
right		Required. A DataFrame, a Series to merge with
how	'left' 'right' 'outer' 'inner' 'cross'	Optional. Default 'inner'. Specifies how to merge
on	String List	Optional. Specifies in what level to do the merging
left_on	String List	Optional. Specifies in what level to do the merging on the DataFrame to the left
right_on	String List	Optional. Specifies in what level to do the merging on the DataFrame to the right
left_index	True False	Optional. Default False. Whether to use the index from the left DataFrame as join key or not
right_index	True False	Optional. Default False. Whether to use the index from the right DataFrame as join key or not
sort	True False	Optional. Default False. Specifies whether to sort the DataFrame by the join key or not
suffixes	List	Optional. Default '_x', '_y''. Specifies a list of strings to add for overlapping columns
copy	True False	Optional. Default True. Specifies whether to keep copies or not
indicator	True False String	Optional. Default False. Specifies whether to add a column in the DataFrame with information about the source of each row

In [48]:
```
MergeOperation=pd.merge(fruits1,fruits2,how='right')
MergeOperation
```

Out[48]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	NaN	NaN	13	10	20	21	31
1	NaN	NaN	12	13	23	24	33
2	NaN	NaN	2	2	4	51	30
3	NaN	NaN	55	9	0	22	36
4	NaN	NaN	96	76	9	25	32

In [49]:
```
MergeOperation=pd.merge(fruits1,fruits2,how='left')
MergeOperation
```

Out[49]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	3	0	7	NaN	NaN	NaN	NaN
1	2	3	14	NaN	NaN	NaN	NaN
2	0	7	6	NaN	NaN	NaN	NaN
3	1	2	15	NaN	NaN	NaN	NaN

In [50]:
```
MergeOperation=pd.merge(fruits1,fruits2,how='outer')
MergeOperation
```

Out[50]:

	orange	apple	grapes	mango	banana	pear	pineapple
0	3.0	0.0	7	NaN	NaN	NaN	NaN
1	2.0	3.0	14	NaN	NaN	NaN	NaN
2	0.0	7.0	6	NaN	NaN	NaN	NaN
3	1.0	2.0	15	NaN	NaN	NaN	NaN
4	NaN	NaN	13	10.0	20.0	21.0	31.0
5	NaN	NaN	12	13.0	23.0	24.0	33.0
6	NaN	NaN	2	2.0	4.0	51.0	30.0
7	NaN	NaN	55	9.0	0.0	22.0	36.0
8	NaN	NaN	96	76.0	9.0	25.0	32.0

Ok Let's Import Data and Perform Some Operations on it

In [79]:
```
df=pd.read_csv("Indian Liver Patient.csv")
df
```

Out[79]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

583 rows × 11 columns

Head & Tail

- Head display first 10 records
- Tail display last 10 records

In [58]:
```
df.head()
```

Out[58]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

In [59]:
```
df.tail()
```

Out[59]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

Shape

- It describe how many rows and records

In [61]:
```
df.shape
```

Out[61]:
```
(583, 11)
```

583 Records and 11 Columns

Size

In [62]:
```
df.size
```

Out[62]:
```
6413
```

583*11 = 6413

columns

- Display Columns

In [64]:
```
df.columns
```

Out[64]:
```
Index(['Age', 'Gender', 'TB', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB',
       'AG', 'LiverPatient'],
      dtype='object')
```

nunique

- Display the No of Unique Values in Features

In [65]:
```
df.nunique()
```

Out[65]:
```
Age              72
Gender            2
TB              113
DB               80
Alkphos         263
Sgpt            152
Sgot            177
TP               58
ALB              40
AG               69
LiverPatient      2
dtype: int64
```

dtypes

- Dispaly the type of the Datatype

In [66]:
```
df.dtypes
```

Out[66]:
```
Age               int64
Gender           object
TB              float64
DB              float64
Alkphos           int64
Sgpt              int64
Sgot              int64
TP              float64
ALB             float64
AG              float64
LiverPatient      int64
dtype: object
```


info

- Display the total Information about dataset

In [67]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           583 non-null    int64  
 1   Gender        583 non-null    object 
 2   TB            583 non-null    float64
 3   DB            583 non-null    float64
 4   Alkphos       583 non-null    int64  
 5   Sgpt          583 non-null    int64  
 6   Sgot          583 non-null    int64  
 7   TP            583 non-null    float64
 8   ALB           583 non-null    float64
 9   AG            579 non-null    float64
 10  LiverPatient  583 non-null    int64  
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

describe

- display statical values of all Numerical data

In [69]:
```
df.describe()
```


Out[69]:

	Age	TB	DB	Alkphos	Sgpt	Sgot	TP
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000

- Let's include categorial data also

In [70]:
```
df.describe(include="all")
```

Out[70]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	
count	583.000000	583	583.000000	583.000000	583.000000	583.000000	583.000000	583
unique	NaN	2	NaN	NaN	NaN	NaN	NaN	
top	NaN	Male	NaN	NaN	NaN	NaN	NaN	
freq	NaN	441	NaN	NaN	NaN	NaN	NaN	
mean	44.746141	NaN	3.298799	1.486106	290.576329	80.713551	109.910806	6
std	16.189833	NaN	6.209522	2.808498	242.937989	182.620356	288.918529	1
min	4.000000	NaN	0.400000	0.100000	63.000000	10.000000	10.000000	2
25%	33.000000	NaN	0.800000	0.200000	175.500000	23.000000	25.000000	5
50%	45.000000	NaN	1.000000	0.300000	208.000000	35.000000	42.000000	6
75%	58.000000	NaN	2.600000	1.300000	298.000000	60.500000	87.000000	7
max	90.000000	NaN	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9

isnull

- Let's for the null values

In [71]:
```
df.isnull()
```

Out[71]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
578	False	False	False	False	False	False	False	False	False	False	False
579	False	False	False	False	False	False	False	False	False	False	False
580	False	False	False	False	False	False	False	False	False	False	False
581	False	False	False	False	False	False	False	False	False	False	False
582	False	False	False	False	False	False	False	False	False	False	False

583 rows × 11 columns

- Above we can see all True and False very borning lets check numberical

In [72]:
```
df.isnull().sum()
```

Out[72]:
```
Age             0
Gender          0
TB              0
DB              0
Alkphos         0
Sgpt            0
Sgot            0
TP              0
ALB             0
AG              4
LiverPatient    0
dtype: int64
```

```
 In Ag is there is 4 null values , There less null values so i drop those records
```

dropna

- Drop all null values

In [96]:
```
df=df.dropna()
```

In [97]:
```
df.isnull().sum()
```

Out[97]:
```
Age             0
Gender          0
TB              0
DB              0
Alkphos         0
Sgpt            0
Sgot            0
TP              0
ALB             0
AG              0
LiverPatient    0
dtype: int64
```

Select 1 Feature

In [98]:
```
df.Gender
```

Out[98]:
```
0      Female
1        Male
2        Male
3        Male
4        Male
       ...
578      Male
579      Male
580      Male
581      Male
582      Male
Name: Gender, Length: 579, dtype: object
```


- Another way

In [99]:
```
df["Gender"]
```

Out[99]:
```
0      Female
1        Male
2        Male
3        Male
4        Male
       ...
578      Male
579      Male
580      Male
581      Male
582      Male
Name: Gender, Length: 579, dtype: object
```

Select More Features

In [100]:
```
df.columns
```

Out[100]:
```
Index(['Age', 'Gender', 'TB', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB',
       'AG', 'LiverPatient'],
      dtype='object')
```

In [101]:
```
df[['Age', 'Gender', 'TB', 'DB']]
```

Out[101]:

	Age	Gender	TB	DB
0	65	Female	0.7	0.1
1	62	Male	10.9	5.5
2	62	Male	7.3	4.1
3	58	Male	1.0	0.4
4	72	Male	3.9	2.0
...	...	...	...	...
578	60	Male	0.5	0.1
579	40	Male	0.6	0.1
580	52	Male	0.8	0.2
581	31	Male	1.3	0.5
582	38	Male	1.0	0.3

579 rows × 4 columns

iloc

- iloc is index based on index it will work
- first -> Rows[Records] and Second -> columns[Features]
- [Rows:columns]

In [107]:
```
df.iloc[:]
```

Out[107]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

579 rows × 11 columns

In [109]:
```
df.iloc[:,:4]
```


Out[109]:

	Age	Gender	TB	DB
0	65	Female	0.7	0.1
1	62	Male	10.9	5.5
2	62	Male	7.3	4.1
3	58	Male	1.0	0.4
4	72	Male	3.9	2.0
...	...	...	...	...
578	60	Male	0.5	0.1
579	40	Male	0.6	0.1
580	52	Male	0.8	0.2
581	31	Male	1.3	0.5
582	38	Male	1.0	0.3

579 rows × 4 columns

In [110]:
```
df.iloc[0:5,0:5]
```

Out[110]:

	Age	Gender	TB	DB	Alkphos
0	65	Female	0.7	0.1	187
1	62	Male	10.9	5.5	699
2	62	Male	7.3	4.1	490
3	58	Male	1.0	0.4	182
4	72	Male	3.9	2.0	195

Above I display 5 Records and 4 columns

loc

- loc is used location

In [118]:
```
df.loc[[1,5],'Age':'TB']
```

Out[118]:

	Age	Gender	TB
1	62	Male	10.9
5	46	Male	1.8

In [120]:
```
df.loc[1:5,'Age':'TB']
```

Out[120]:

	Age	Gender	TB
1	62	Male	10.9
2	62	Male	7.3
3	58	Male	1.0
4	72	Male	3.9
5	46	Male	1.8

In [121]:
```
df.loc[1:,'Age':'TB']
```

Out[121]:

	Age	Gender	TB
1	62	Male	10.9
2	62	Male	7.3
3	58	Male	1.0
4	72	Male	3.9
5	46	Male	1.8
...	...	...	...
578	60	Male	0.5
579	40	Male	0.6
580	52	Male	0.8
581	31	Male	1.3
582	38	Male	1.0

578 rows × 3 columns

set_index

- set_index is used to change the Index

In [122]:
```
dumie=df.set_index("Age")
dumie
```

Out[122]:

Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...
60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

579 rows × 10 columns

I need Peoples who's age is 25

In [124]:
```
dumie.loc[25]
```

Out[124]:

Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
25	Male	0.6	0.1	183	91	53	5.5	2.3	0.7	2
25	Female	0.9	0.3	159	24	25	6.9	4.4	1.7	2
25	Female	0.7	0.1	140	32	25	7.6	4.3	1.3	2
25	Male	0.8	0.1	130	23	42	8.0	4.0	1.0	1
25	Male	0.7	0.2	185	196	401	6.5	3.9	1.5	1

value_counts

- value_counts used for to know the how many unique values in feataures

In [126]:
```
df["Gender"].value_counts()
```

Out[126]:
```
Male      439
Female    140
Name: Gender, dtype: int64
```

In Gender there are 2 categories

- Male :439
- Female:140

Percentage

In [127]:
```
df["Gender"].value_counts()/len(df['Gender'])
```

Out[127]:
```
Male      0.758204
Female    0.241796
Name: Gender, dtype: float64
```

In our Dataset 75% Male and 24% Female

Mean

In [128]:
```
df["Age"].mean()
```

Out[128]:
```
44.78238341968912
```

Median

In [129]:
```
df["Age"].median()
```

Out[129]:
```
45.0
```

Mode

In [130]:
```
df["Age"].mode()
```

Out[130]:
```
0    60
Name: Age, dtype: int64
```

Min

In [131]:
```
df["Age"].min()
```

Out[131]:
```
4
```

Max

In [132]:
```
df["Age"].max()
```

Out[132]:
```
90
```

sum

In [133]:
```
df["Age"].sum()
```

Out[133]:
```
25929
```

nsmallest

- print the 5 smallest values

In [134]:
```
df["Age"].nsmallest()
```

Out[134]:
```
265    4
271    4
218    6
199    7
480    7
Name: Age, dtype: int64
```


nlargest

- Print 5 largest Number

In [135]:
```
df["Age"].nlargest()
```

Out[135]:
```
571    90
44     85
29     84
397    78
71     75
Name: Age, dtype: int64
```

sort

In [136]:
```
df.sort_values(by='Age')
```

Out[136]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
265	4	Male	0.9	0.2	348	30	34	8.0	4.0	1.0	2
271	4	Male	0.8	0.2	460	152	231	6.5	3.2	0.9	2
218	6	Male	0.6	0.1	289	38	30	4.8	2.0	0.7	2
480	7	Male	0.5	0.1	352	28	51	7.9	4.2	1.1	2
199	7	Female	27.2	11.8	1420	790	1050	6.1	2.0	0.4	1
...	...	...	...	...	...	...	...	...	...	...	...
177	75	Male	14.8	9.0	1020	71	42	5.3	2.2	0.7	1
397	78	Male	1.0	0.3	152	28	70	6.3	3.1	0.9	1
29	84	Female	0.7	0.2	188	13	21	6.0	3.2	1.1	2
44	85	Female	1.0	0.3	208	17	15	7.0	3.6	1.0	2
571	90	Male	1.1	0.3	215	46	134	6.9	3.0	0.7	1

579 rows × 11 columns

In [137]:
```
df.sort_values(by='Age',ascending=False)
```

Out[137]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
571	90	Male	1.1	0.3	215	46	134	6.9	3.0	0.7	1
44	85	Female	1.0	0.3	208	17	15	7.0	3.6	1.0	2
29	84	Female	0.7	0.2	188	13	21	6.0	3.2	1.1	2
397	78	Male	1.0	0.3	152	28	70	6.3	3.1	0.9	1
340	75	Male	0.9	0.2	206	44	33	6.2	2.9	0.8	1
...	...	...	...	...	...	...	...	...	...	...	...
199	7	Female	27.2	11.8	1420	790	1050	6.1	2.0	0.4	1
480	7	Male	0.5	0.1	352	28	51	7.9	4.2	1.1	2
218	6	Male	0.6	0.1	289	38	30	4.8	2.0	0.7	2
271	4	Male	0.8	0.2	460	152	231	6.5	3.2	0.9	2
265	4	Male	0.9	0.2	348	30	34	8.0	4.0	1.0	2

579 rows × 11 columns

Data Extraction`

==

In [138]:
```
male_data=df[df["Gender"]=="Male"]
male_data
```

Out[138]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

439 rows × 11 columns

less then or equal

In [139]:
```
df[df["Age"]<=20]
```

Out[139]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	2
28	20	Male	1.1	0.5	128	20	30	3.9	1.9	0.95	2
36	17	Female	0.7	0.2	145	18	36	7.2	3.9	1.18	2
70	19	Female	0.7	0.2	186	166	397	5.5	3.0	1.20	1
85	14	Male	1.4	0.5	269	58	45	6.7	3.9	1.40	1
86	13	Male	0.6	0.1	320	28	56	7.2	3.6	1.00	2
88	18	Male	0.6	0.2	538	33	34	7.5	3.2	0.70	1
98	18	Male	0.6	0.1	265	97	161	5.9	3.1	1.10	1
99	18	Male	0.7	0.1	312	308	405	6.9	3.7	1.10	1
102	17	Male	0.9	0.2	224	36	45	6.9	4.2	1.55	1
132	18	Female	0.8	0.2	199	34	31	6.5	3.5	1.16	2
134	18	Male	1.8	0.7	178	35	36	6.8	3.6	1.10	1
137	18	Male	0.8	0.2	282	72	140	5.5	2.5	0.80	1
138	18	Male	0.8	0.2	282	72	140	5.5	2.5	0.80	1
139	15	Male	0.8	0.2	380	25	66	6.1	3.7	1.50	1
199	7	Female	27.2	11.8	1420	790	1050	6.1	2.0	0.40	1
213	8	Female	0.9	0.2	401	25	58	7.5	3.4	0.80	1
218	6	Male	0.6	0.1	289	38	30	4.8	2.0	0.70	2
262	18	Male	0.8	0.2	228	55	54	6.9	4.0	1.30	1
265	4	Male	0.9	0.2	348	30	34	8.0	4.0	1.00	2
271	4	Male	0.8	0.2	460	152	231	6.5	3.2	0.90	2
283	18	Male	1.3	0.7	316	10	21	6.0	2.1	0.50	2
284	18	Male	0.9	0.3	300	30	48	8.0	4.0	1.00	1
285	13	Male	1.5	0.5	575	29	24	7.9	3.9	0.90	1
319	14	Male	0.9	0.3	310	21	16	8.1	4.2	1.00	2
323	12	Male	0.8	0.2	302	47	67	6.7	3.5	1.10	2
334	13	Female	0.7	0.2	350	17	24	7.4	4.0	1.10	1
335	13	Female	0.7	0.1	182	24	19	8.9	4.9	1.20	1
355	19	Male	1.4	0.8	178	13	26	8.0	4.6	1.30	2
366	16	Male	0.7	0.2	418	28	35	7.2	4.1	1.30	2
377	20	Female	0.6	0.2	202	12	13	6.1	3.0	0.90	2
407	12	Male	1.0	0.2	719	157	108	7.2	3.7	1.00	1
410	18	Male	1.4	0.6	215	440	850	5.0	1.9	0.60	1

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
417	11	Male	0.7	0.1	592	26	29	7.1	4.2	1.40	2
435	17	Female	0.5	0.1	206	28	21	7.1	4.5	1.70	2
445	17	Male	0.9	0.2	279	40	46	7.3	4.0	1.20	2
480	7	Male	0.5	0.1	352	28	51	7.9	4.2	1.10	2
537	10	Female	0.8	0.1	395	25	75	7.6	3.6	0.90	1
568	20	Female	16.7	8.4	200	91	101	6.9	3.5	1.02	1
569	16	Male	7.7	4.1	268	213	168	7.1	4.0	1.20	1
570	16	Male	2.6	1.2	236	131	90	5.4	2.6	0.90	1

Logical Operator

In [141]:
```
df[(df["Age"]>25)&(df["TB"]>0.7)]
```

Out[141]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
...	...	...	...	...	...	...	...	...	...	...	...
576	32	Male	15.0	8.2	289	58	80	5.3	2.2	0.70	1
577	32	Male	12.7	8.4	190	28	47	5.4	2.6	0.90	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

404 rows × 11 columns

In [143]:
```
df[(df["Age"]>25) | (df["TB"]>0.7)]
```

Out[143]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

556 rows × 11 columns

groupby

In [153]:
```
group1=df.groupby("Age")
group1
```

Out[153]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001E32E9C65B0>
```

In [155]:
```
group1.first()
```

Out[155]:

Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
4	Male	0.9	0.2	348	30	34	8.0	4.0	1.0	2
6	Male	0.6	0.1	289	38	30	4.8	2.0	0.7	2
7	Female	27.2	11.8	1420	790	1050	6.1	2.0	0.4	1
8	Female	0.9	0.2	401	25	58	7.5	3.4	0.8	1
10	Female	0.8	0.1	395	25	75	7.6	3.6	0.9	1
...	...	...	...	...	...	...	...	...	...	...
75	Female	0.8	0.2	188	20	29	4.4	1.8	0.6	1
78	Male	1.0	0.3	152	28	70	6.3	3.1	0.9	1
84	Female	0.7	0.2	188	13	21	6.0	3.2	1.1	2
85	Female	1.0	0.3	208	17	15	7.0	3.6	1.0	2
90	Male	1.1	0.3	215	46	134	6.9	3.0	0.7	1

72 rows × 10 columns

In [157]:
```
group2=df.groupby(["Gender","Age"])
```

In [158]:
```
group2.first()
```

Out[158]:

Gender	Age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
Female	**7**	27.2	11.8	1420	790	1050	6.1	2.0	0.40	1
	8	0.9	0.2	401	25	58	7.5	3.4	0.80	1
	10	0.8	0.1	395	25	75	7.6	3.6	0.90	1
	13	0.7	0.2	350	17	24	7.4	4.0	1.10	1
	17	0.7	0.2	145	18	36	7.2	3.9	1.18	2
...	**...**	...	...	...	...	...	...	...	...	...
Male	**73**	1.9	0.7	1750	102	141	5.5	2.0	0.50	1
	74	0.6	0.1	272	24	98	5.0	2.0	0.60	1
	75	0.9	0.2	282	25	23	4.4	2.2	1.00	1
	78	1.0	0.3	152	28	70	6.3	3.1	0.90	1
	90	1.1	0.3	215	46	134	6.9	3.0	0.70	1

121 rows × 9 columns

Crosstab

In [161]:
```
pd.crosstab(df.Gender,df.LiverPatient)
```

Out[161]:

Gender	1	2
Female	91	49
Male	323	116

- Only Two features can be used

In [ ]:
```

```