In [1]:

```
import numpy as np
```

In [2]:

```
#import Pandas library
import pandas as pd
```

In [3]:

```
pd.__version__
```

Out[3]:

```
'1.4.2'
```

Pandas Series

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

1. Create Series using List

In [4]:

```
l=['banana', 42]

s = pd.Series(l)
print(s)
```

```
0    banana
1        42
dtype: object
```

In [5]:

```
type(s)
```

Out[5]:

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

In [6]:

```
s.shape
```

Out[6]:

```
(2,)
```

In [7]:

```
l=['banana', 42]

s = pd.Series(l, index=['a','b'])
print(s)
```

```
a    banana
b        42
dtype: object
```

2. Create Series using 1D Array

In [8]:

```
a=np.array(['banana', 42])

s = pd.Series(a, index=['a','b'])
print(s)
```

```
a    banana
b        42
dtype: object
```

3. Create Series using Dictionary

In [9]:

```
d = {'a': 1,'b': 2,'c': 3}
s = pd.Series(d)
s
```

Out[9]:

```
a    1
b    2
c    3
dtype: int64
```

In [10]:

```
d = {'col1': [1, 2], 'col2': [3, 4]}
s = pd.Series(d)
s
```

Out[10]:

```
col1    [1, 2]
col2    [3, 4]
dtype: object
```

In [11]:

```
type(s)
```

Out[11]:

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

In [12]:

```
s.shape
```

Out[12]:

```
(2,)
```

Pandas DataFrame

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

1. Create DataFrame using Dictionary

In [13]:

```
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(d)
df
```

Out[13]:

	col1	col2
0	1	3
1	2	4

In [14]:

```
type(df)
```

Out[14]:

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

In [15]:

```
df.shape
```

Out[15]:

```
(2, 2)
```

2. Create DataFrame using Nested List

In [16]:

```
data=[[1,2,3],[4,5,6]]
df1 = pd.DataFrame(data,columns=["col1","col2","col3"])
df1
```

Out[16]:

	col1	col2	col3
0	1	2	3
1	4	5	6

3. Create DataFrame using 2D array

In [17]:

```
arr_2d = np.array([[1,2,3],[4,5,6]])

df2 = pd.DataFrame(arr_2d,columns=["col1","col2","col3"])
df2
```

Out[17]:

	col1	col2	col3
0	1	2	3
1	4	5	6

Combining Dataframes

In [18]:

```
temperature_df = pd.DataFrame({
"city": ["mumbai","delhi","banglore", 'hyderabad'],
"temperature": [32,45,40,36]})

temperature_df
```

Out[18]:

	city	temperature
0	mumbai	32
1	delhi	45
2	banglore	40
3	hyderabad	36

In [19]:

```
humidity_df = pd.DataFrame({
"city": ["delhi","mumbai","banglore","Chennai"],
"humidity": [68, 65, 75, 70]})

humidity_df
```

Out[19]:

	city	humidity
0	delhi	68
1	mumbai	65
2	banglore	75
3	Chennai	70

option 1: **Append**

In [20]:

```
temperature_df.append(humidity_df)
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_3208\1270164571.py:1: FutureWarning: The frame.append method is deprecated and
will be removed from pandas in a future version. Use pandas.concat instead.
  temperature_df.append(humidity_df)
```

Out[20]:

	city	temperature	humidity
0	mumbai	32.0	NaN
1	delhi	45.0	NaN
2	banglore	40.0	NaN
3	hyderabad	36.0	NaN
0	delhi	NaN	68.0
1	mumbai	NaN	65.0
2	banglore	NaN	75.0
3	Chennai	NaN	70.0

option 2: **Concatenate**

In [21]:

```
pd.concat([temperature_df, humidity_df])
```

Out[21]:

	city	temperature	humidity
0	mumbai	32.0	NaN
1	delhi	45.0	NaN
2	banglore	40.0	NaN
3	hyderabad	36.0	NaN
0	delhi	NaN	68.0
1	mumbai	NaN	65.0
2	banglore	NaN	75.0
3	Chennai	NaN	70.0

In [22]:

```
pd.concat([temperature_df, humidity_df], ignore_index=True)
```

Out[22]:

	city	temperature	humidity
0	mumbai	32.0	NaN
1	delhi	45.0	NaN
2	banglore	40.0	NaN
3	hyderabad	36.0	NaN
4	delhi	NaN	68.0
5	mumbai	NaN	65.0
6	banglore	NaN	75.0
7	Chennai	NaN	70.0

In [23]:

```
pd.concat([temperature_df, humidity_df],axis=1)
```

Out[23]:

	city	temperature	city	humidity
0	mumbai	32	delhi	68
1	delhi	45	mumbai	65
2	banglore	40	banglore	75
3	hyderabad	36	Chennai	70

Merging of DataFrame

In [24]:

```
#Outer Join -all te records of both dataframes
pd.merge(temperature_df, humidity_df, on='city', how='outer')
```

Out[24]:

	city	temperature	humidity
0	mumbai	32.0	65.0
1	delhi	45.0	68.0
2	banglore	40.0	75.0
3	hyderabad	36.0	NaN
4	Chennai	NaN	70.0

In [25]:

```
#Inner Join -- common data records
pd.merge(temperature_df, humidity_df, on='city',how="inner")
```

Out[25]:

	city	temperature	humidity
0	mumbai	32	65
1	delhi	45	68
2	banglore	40	75

In [26]:

```
#Left Join
pd.merge(temperature_df, humidity_df, on='city', how='left')
```

Out[26]:

	city	temperature	humidity
0	mumbai	32	65.0
1	delhi	45	68.0
2	banglore	40	75.0
3	hyderabad	36	NaN

In [27]:

```
#Right Join
pd.merge(temperature_df, humidity_df, on='city', how='right')
```

Out[27]:

	city	temperature	humidity
0	delhi	45.0	68
1	mumbai	32.0	65
2	banglore	40.0	75
3	Chennai	NaN	70

Import data/ Load data

- **Option1 :** path + (file name with extension)

In [28]:

```
df = pd.read_excel("D:\\NIT\\02. Python for Data Science\\liver.xlsx")
df
```

Out[28]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

In [29]:

```
df = pd.read_csv("D:\\NIT\\02. Python for Data Science\\liver.csv")
df
```

Out[29]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

- **Option2 :** "file name with extension"

Note: this will work, when dataset is in your working directory (means python file and dataset both in same folder)

In [30]:

```
df=pd.read_excel("liver.xlsx")
df
```

Out[30]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

In [31]:

```
df = pd.read_csv("Liver.csv")
df
```

Out[31]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

In [32]:

```
#first 5 observations
df.head()
```

Out[32]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes

In [33]:

```
#last 5 observations
df.tail()
```

Out[33]:

	Age	Gender	TB	DB	TP	LiverPatient
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

In [34]:

```
#dimensions of the data frame
df.shape
```

Out[34]:

```
(10, 6)
```

In [35]:

```
#Return an int representing the number of elements.
df.size
```

Out[35]:

```
60
```

In [36]:

```
#column names/Variable names
df.columns
```

Out[36]:

```
Index(['Age', 'Gender', 'TB', 'DB', 'TP', 'LiverPatient'], dtype='object')
```

In [37]:

```
#column names/Variables
df.keys()                          #option-2
```

Out[37]:

```
Index(['Age', 'Gender', 'TB', 'DB', 'TP', 'LiverPatient'], dtype='object')
```

In [38]:

```
#Data type of each column
df.dtypes
```

Out[38]:

```
Age              int64
Gender          object
TB             float64
DB             float64
TP             float64
LiverPatient    object
dtype: object
```

concise summary of a DataFrame

In [39]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           10 non-null     int64  
 1   Gender        10 non-null     object 
 2   TB            10 non-null     float64
 3   DB            10 non-null     float64
 4   TP            8 non-null      float64
 5   LiverPatient  10 non-null     object 
dtypes: float64(3), int64(1), object(2)
memory usage: 608.0+ bytes
```

Accessing/Selecting Data

by using iloc()

- works on the row index numder & column index number

In [40]:

```
df.iloc[0]
```

Out[40]:

```
Age               65
Gender        Female
TB               0.7
DB               0.1
TP               6.8
LiverPatient     Yes
Name: 0, dtype: object
```

In [41]:

```
df.iloc[-1]
```

Out[41]:

```
Age              29
Gender         Male
TB              3.6
DB              0.1
TP              NaN
LiverPatient     No
Name: 9, dtype: object
```

In [42]:

```
df.iloc[[1,6]]
```

Out[42]:

	Age	Gender	TB	DB	TP	LiverPatient
1	62	Male	7.3	4.1	7.0	Yes
6	26	Female	0.9	0.2	4.5	No

In [43]:

```
df.iloc[2,1]
```

Out[43]:

```
'Male'
```

In [44]:

```
df.iloc[[5,8],[1,2,3,4]]
```

Out[44]:

	Gender	TB	DB	TP
5	Male	1.8	0.7	7.6
8	Male	0.9	0.3	5.4

In [45]:

```
df.iloc[[5,8],1:5]
```

Out[45]:

	Gender	TB	DB	TP
5	Male	1.8	0.7	7.6
8	Male	0.9	0.3	5.4

In [46]:

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

Out[46]:

	Age	Gender
0	65	Female
1	62	Male
2	62	Male

by using loc()

- Locate Named Indexes (Use the named index in the loc attribute)

In [47]:

```
df.loc[0]
```

Out[47]:

```
Age                65
Gender         Female
TB                0.7
DB                0.1
TP                6.8
LiverPatient      Yes
Name: 0, dtype: object
```

In [48]:

```
df.loc[[2,3]]
```

Out[48]:

	Age	Gender	TB	DB	TP	LiverPatient
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes

In [49]:

```
df.loc[:,"Age"]
```

Out[49]:

```
0    65
1    62
2    62
3    58
4    72
5    46
6    26
7    29
8    17
9    29
Name: Age, dtype: int64
```

In [50]:

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

Out[50]:

```
0    65
1    62
2    62
3    58
4    72
5    46
6    26
7    29
8    17
9    29
Name: Age, dtype: int64
```

In [51]:

```
df.loc[[2,3],"Age":"TP"]
```

Out[51]:

	Age	Gender	TB	DB	TP
2	62	Male	7.3	4.1	7.0
3	58	Male	1.0	0.4	6.8

In [52]:

```
df.iloc[[2,3],0:5]
```

Out[52]:

	Age	Gender	TB	DB	TP
2	62	Male	7.3	4.1	7.0
3	58	Male	1.0	0.4	6.8

Just make sure you don't confuse the differences between loc and iloc

In [53]:

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

Out[53]:

```
62
```

In [54]:

```
df.loc[2,"Age"]
```

Out[54]:

```
62
```

In [55]:

```
df.iloc[0]
```

Out[55]:

```
Age                65
Gender         Female
TB                0.7
DB                0.1
TP                6.8
LiverPatient      Yes
Name: 0, dtype: object
```

Accessing Single column

In [56]:

```
df.loc[:,"TB"]
```

Out[56]:

```
0    0.7
1    7.3
2    7.3
3    1.0
4    3.9
5    1.8
6    0.9
7    4.5
8    0.9
9    3.6
Name: TB, dtype: float64
```

In [57]:

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

Out[57]:

```
0    0.7
1    7.3
2    7.3
3    1.0
4    3.9
5    1.8
6    0.9
7    4.5
8    0.9
9    3.6
Name: TB, dtype: float64
```

In [58]:

```
df.TB
```

Out[58]:

```
0    0.7
1    7.3
2    7.3
3    1.0
4    3.9
5    1.8
6    0.9
7    4.5
8    0.9
9    3.6
Name: TB, dtype: float64
```

In [59]:

```
df['TB']
```

Out[59]:

```
0    0.7
1    7.3
2    7.3
3    1.0
4    3.9
5    1.8
6    0.9
7    4.5
8    0.9
9    3.6
Name: TB, dtype: float64
```

In [60]:

```
df['TB'].values                    #output as array
```

Out[60]:

```
array([0.7, 7.3, 7.3, 1. , 3.9, 1.8, 0.9, 4.5, 0.9, 3.6])
```

Accessing Multiple Columns

In [61]:

```
df.loc[:,['TB','DB']]
```

Out[61]:

	TB	DB
0	0.7	0.1
1	7.3	4.1
2	7.3	4.1
3	1.0	0.4
4	3.9	2.0
5	1.8	0.7
6	0.9	0.2
7	4.5	0.3
8	0.9	0.3
9	3.6	0.1

In [62]:

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

Out[62]:

	TB	DB
0	0.7	0.1
1	7.3	4.1
2	7.3	4.1
3	1.0	0.4
4	3.9	2.0
5	1.8	0.7
6	0.9	0.2
7	4.5	0.3
8	0.9	0.3
9	3.6	0.1

In [63]:

```
#we need to pass, list of multiple columns by the column names
df[['TB','DB']]
```

Out[63]:

	TB	DB
0	0.7	0.1
1	7.3	4.1
2	7.3	4.1
3	1.0	0.4
4	3.9	2.0
5	1.8	0.7
6	0.9	0.2
7	4.5	0.3
8	0.9	0.3
9	3.6	0.1

Duplicates

check for duplicates

In [64]:

```
df.duplicated()
```

Out[64]:

```
0    False
1    False
2     True
3    False
4    False
5    False
6    False
7    False
8    False
9    False
dtype: bool
```

drop duplicates

In [65]:

```
df.drop_duplicates()
```

Out[65]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

Missing Values

Check for missing values

In [66]:

```
df.isnull().sum()
```

Out[66]:

```
Age             0
Gender          0
TB              0
DB              0
TP              2
LiverPatient    0
dtype: int64
```

In [67]:

```
df.isna().sum()
```

Out[67]:

```
Age             0
Gender          0
TB              0
DB              0
TP              2
LiverPatient    0
dtype: int64
```

drop missing values

In [68]:

```
df.dropna()
```

Out[68]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No

In [69]:

```
df
```

Out[69]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	NaN	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	NaN	No

fill missing values

In [70]:

```
df['TP'] = df["TP"].fillna(4.5)      #df["TP"] = df["TP"].fillna(4.5)
df
```

Out[70]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	4.5	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	4.5	No

In [71]:

```
df = pd.read_csv("Liver.csv")
df["TP"].fillna(4.5,inplace=True)
df
```

Out[71]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	4.5	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	4.5	No

Drop a Column

In [72]:

```
df.drop("DB",axis=1)
```

Out[72]:

	Age	Gender	TB	TP	LiverPatient
0	65	Female	0.7	6.8	Yes
1	62	Male	7.3	7.0	Yes
2	62	Male	7.3	7.0	Yes
3	58	Male	1.0	6.8	Yes
4	72	Male	3.9	4.5	Yes
5	46	Male	1.8	7.6	Yes
6	26	Female	0.9	4.5	No
7	29	Female	4.5	6.3	Yes
8	17	Male	0.9	5.4	No
9	29	Male	3.6	4.5	No

Drop multiple columns

In [73]:

```
df.drop(["TB","DB"],axis=1)
```

Out[73]:

	Age	Gender	TP	LiverPatient
0	65	Female	6.8	Yes
1	62	Male	7.0	Yes
2	62	Male	7.0	Yes
3	58	Male	6.8	Yes
4	72	Male	4.5	Yes
5	46	Male	7.6	Yes
6	26	Female	4.5	No
7	29	Female	6.3	Yes
8	17	Male	5.4	No
9	29	Male	4.5	No

Drop a row

- Drop (Need to specify that axis = 0 or 'index', axis=1 or 'columns')

In [74]:

```
df.drop(0,axis=0)
```

Out[74]:

	Age	Gender	TB	DB	TP	LiverPatient
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	4.5	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	4.5	No

Drop multiple rows

In [75]:

```
df.drop([0,1,2],axis=0)
```

Out[75]:

	Age	Gender	TB	DB	TP	LiverPatient
3	58	Male	1.0	0.4	6.8	Yes
4	72	Male	3.9	2.0	4.5	Yes
5	46	Male	1.8	0.7	7.6	Yes
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
9	29	Male	3.6	0.1	4.5	No

Sort Function

In [76]:

```
#Sorting in ascending order.
df.sort_values(by='DB',ascending=True)
```

Out[76]:

	Age	Gender	TB	DB	TP	LiverPatient
0	65	Female	0.7	0.1	6.8	Yes
9	29	Male	3.6	0.1	4.5	No
6	26	Female	0.9	0.2	4.5	No
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
3	58	Male	1.0	0.4	6.8	Yes
5	46	Male	1.8	0.7	7.6	Yes
4	72	Male	3.9	2.0	4.5	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes

In [77]:

```
# Sorting in descending order.
df.sort_values(by='TP',ascending=False)
```

Out[77]:

	Age	Gender	TB	DB	TP	LiverPatient
5	46	Male	1.8	0.7	7.6	Yes
1	62	Male	7.3	4.1	7.0	Yes
2	62	Male	7.3	4.1	7.0	Yes
0	65	Female	0.7	0.1	6.8	Yes
3	58	Male	1.0	0.4	6.8	Yes
7	29	Female	4.5	0.3	6.3	Yes
8	17	Male	0.9	0.3	5.4	No
4	72	Male	3.9	2.0	4.5	Yes
6	26	Female	0.9	0.2	4.5	No
9	29	Male	3.6	0.1	4.5	No

Add a new Column

In [78]:

```
df['IB'] = df["TB"]-df["DB"]
df
```

Out[78]:

	Age	Gender	TB	DB	TP	LiverPatient	IB
0	65	Female	0.7	0.1	6.8	Yes	0.6
1	62	Male	7.3	4.1	7.0	Yes	3.2
2	62	Male	7.3	4.1	7.0	Yes	3.2
3	58	Male	1.0	0.4	6.8	Yes	0.6
4	72	Male	3.9	2.0	4.5	Yes	1.9
5	46	Male	1.8	0.7	7.6	Yes	1.1
6	26	Female	0.9	0.2	4.5	No	0.7
7	29	Female	4.5	0.3	6.3	Yes	4.2
8	17	Male	0.9	0.3	5.4	No	0.6
9	29	Male	3.6	0.1	4.5	No	3.5

In [79]:

```
df["Name"] =["P1","P2","P3","P4","P5","P6","P7","P8","P9","P10"]
df
```

Out[79]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
0	65	Female	0.7	0.1	6.8	Yes	0.6	P1
1	62	Male	7.3	4.1	7.0	Yes	3.2	P2
2	62	Male	7.3	4.1	7.0	Yes	3.2	P3
3	58	Male	1.0	0.4	6.8	Yes	0.6	P4
4	72	Male	3.9	2.0	4.5	Yes	1.9	P5
5	46	Male	1.8	0.7	7.6	Yes	1.1	P6
6	26	Female	0.9	0.2	4.5	No	0.7	P7
7	29	Female	4.5	0.3	6.3	Yes	4.2	P8
8	17	Male	0.9	0.3	5.4	No	0.6	P9
9	29	Male	3.6	0.1	4.5	No	3.5	P10

set Index

In [80]:

```
l1= df.set_index("TB")
l1
```

Out[80]:

TB	Age	Gender	DB	TP	LiverPatient	IB	Name
0.7	65	Female	0.1	6.8	Yes	0.6	P1
7.3	62	Male	4.1	7.0	Yes	3.2	P2
7.3	62	Male	4.1	7.0	Yes	3.2	P3
1.0	58	Male	0.4	6.8	Yes	0.6	P4
3.9	72	Male	2.0	4.5	Yes	1.9	P5
1.8	46	Male	0.7	7.6	Yes	1.1	P6
0.9	26	Female	0.2	4.5	No	0.7	P7
4.5	29	Female	0.3	6.3	Yes	4.2	P8
0.9	17	Male	0.3	5.4	No	0.6	P9
3.6	29	Male	0.1	4.5	No	3.5	P10

In [81]:

```
l1.loc[0.9]
```

Out[81]:

TB	Age	Gender	DB	TP	LiverPatient	IB	Name
0.9	26	Female	0.2	4.5	No	0.7	P7
0.9	17	Male	0.3	5.4	No	0.6	P9

In [82]:

```
l1.loc[0.9, 'Gender':'TP']
```

Out[82]:

TB	Gender	DB	TP
0.9	Female	0.2	4.5
0.9	Male	0.3	5.4

reset index

In [83]:

```
l1.reset_index()
```

Out[83]:

	TB	Age	Gender	DB	TP	LiverPatient	IB	Name
0	0.7	65	Female	0.1	6.8	Yes	0.6	P1
1	7.3	62	Male	4.1	7.0	Yes	3.2	P2
2	7.3	62	Male	4.1	7.0	Yes	3.2	P3
3	1.0	58	Male	0.4	6.8	Yes	0.6	P4
4	3.9	72	Male	2.0	4.5	Yes	1.9	P5
5	1.8	46	Male	0.7	7.6	Yes	1.1	P6
6	0.9	26	Female	0.2	4.5	No	0.7	P7
7	4.5	29	Female	0.3	6.3	Yes	4.2	P8
8	0.9	17	Male	0.3	5.4	No	0.6	P9
9	3.6	29	Male	0.1	4.5	No	3.5	P10

Filtering

- Selecting/extracting the data based on condition(s)

using 1 condition

Syntax: df[condition]

In [84]:

```
df[df["Gender"]=="Male"]
```

Out[84]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
1	62	Male	7.3	4.1	7.0	Yes	3.2	P2
2	62	Male	7.3	4.1	7.0	Yes	3.2	P3
3	58	Male	1.0	0.4	6.8	Yes	0.6	P4
4	72	Male	3.9	2.0	4.5	Yes	1.9	P5
5	46	Male	1.8	0.7	7.6	Yes	1.1	P6
8	17	Male	0.9	0.3	5.4	No	0.6	P9
9	29	Male	3.6	0.1	4.5	No	3.5	P10

In [85]:

```
df[df['Age']>=70]
```

Out[85]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
4	72	Male	3.9	2.0	4.5	Yes	1.9	P5

using multiple conditions

In [86]:

```
filter1=df[(df['Age']==65) | (df['Gender']=="Female")]
filter1
```

Out[86]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
0	65	Female	0.7	0.1	6.8	Yes	0.6	P1
6	26	Female	0.9	0.2	4.5	No	0.7	P7
7	29	Female	4.5	0.3	6.3	Yes	4.2	P8

In [87]:

```
filter2=df[(df['LiverPatient']=="Yes") & (df.Age>=70)]
filter2
```

Out[87]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
4	72	Male	3.9	2.0	4.5	Yes	1.9	P5

In [88]:

```
filter3=df[(df['Gender']=="Female") | (df['Age']>=35) & (df['DB']<=6)]
filter3
```

Out[88]:

	Age	Gender	TB	DB	TP	LiverPatient	IB	Name
0	65	Female	0.7	0.1	6.8	Yes	0.6	P1
1	62	Male	7.3	4.1	7.0	Yes	3.2	P2
2	62	Male	7.3	4.1	7.0	Yes	3.2	P3
3	58	Male	1.0	0.4	6.8	Yes	0.6	P4
4	72	Male	3.9	2.0	4.5	Yes	1.9	P5
5	46	Male	1.8	0.7	7.6	Yes	1.1	P6
6	26	Female	0.9	0.2	4.5	No	0.7	P7
7	29	Female	4.5	0.3	6.3	Yes	4.2	P8

using loc & condition

In [89]:

```
df1= df.loc[df['DB']>=3.7,['Age','DB','LiverPatient']]
df1
```

Out[89]:

	Age	DB	LiverPatient
1	62	4.1	Yes
2	62	4.1	Yes

value_counts() -- for each category, there respective frequency/count

- This is applicate for dicrete data only

In [90]:

```
df["Gender"].value_counts()
```

Out[90]:

```
Male      7
Female    3
Name: Gender, dtype: int64
```

In [91]:

```
df["LiverPatient"].value_counts()
```

Out[91]:

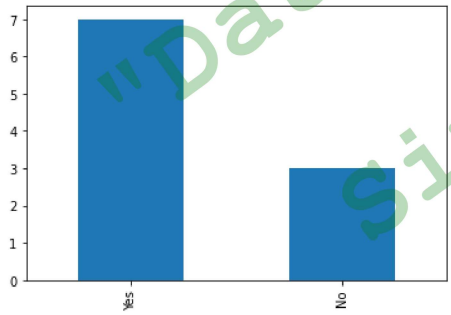
```
Yes    7
No     3
Name: LiverPatient, dtype: int64
```

In [92]:

```
df["LiverPatient"].value_counts().plot(kind='bar')
```

Out[92]:

```
<AxesSubplot:>
```



In [93]:

```
df["LiverPatient"].value_counts()/len(df["LiverPatient"])
```

Out[93]:

```
Yes    0.7
No     0.3
Name: LiverPatient, dtype: float64
```

In [94]:

```
df["LiverPatient"].value_counts(normalize=True)
```

Out[94]:

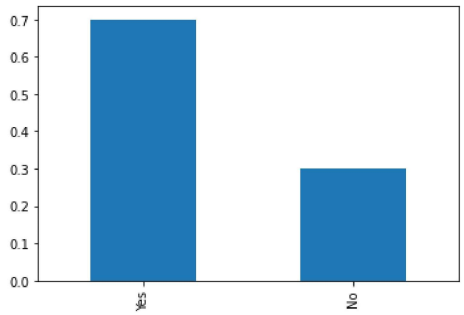
```
Yes    0.7
No     0.3
Name: LiverPatient, dtype: float64
```

In [95]:

```
df["LiverPatient"].value_counts(normalize=True).plot(kind='bar')
```

Out[95]:

<AxesSubplot:>



In [96]:

```
df["LiverPatient"].value_counts(normalize=True).plot(kind='pie')
```

Out[96]:

<AxesSubplot:ylabel='LiverPatient'>

In [97]:

```
male=df[df["Gender"]=="Male"]
male['LiverPatient'].value_counts()
```

Out[97]:

```
Yes    5
No     2
Name: LiverPatient, dtype: int64
```

In [98]:

```
female=df[df["Gender"]=="Female"]
female['LiverPatient'].value_counts()
```

Out[98]:

```
Yes    2
No     1
Name: LiverPatient, dtype: int64
```

crosstab

In [99]:

```
pd.crosstab(df.LiverPatient,df.Gender)
```

Out[99]:

Gender	Female	Male
LiverPatient		
No	1	2
Yes	2	5

In [100]:

```
pd.crosstab(df.LiverPatient,df.Gender,margins=True)
```

Out[100]:

Gender	Female	Male	All
LiverPatient			
No	1	2	3
Yes	2	5	7
All	3	7	10

groupby

In [101]:

```
#Grouping on basis of "Gender" and using sum () function for "TB".
df.groupby('Gender')["TB"].mean()
```

Out[101]:

```
Gender
Female    2.033333
Male      3.685714
Name: TB, dtype: float64
```

In [102]:

```
female = df[df["Gender"]=="Female"]
female['TB'].mean()
```

Out[102]:

```
2.033333333333333
```

In [103]:

```
male = df[df["Gender"]=="Male"]
male['TB'].mean()
```

Out[103]:

```
3.685714285714286
```