Import Libraries

In [1]:
```
# import numpy
import numpy as np
```

In [2]:
```
#Importing Pandas library.
import pandas as pd
```

Create Pandas Series --> pandas.Series

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

Out[3]:
```
a    1
b    2
c    3
dtype: int64
```

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

Out[4]:
```
a    1.0
y    NaN
c    3.0
dtype: float64
```

In [5]:
```
l=['banana', 42]
s = pd.Series(data=l)
print(s)
```

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

In [6]:
```
s = pd.Series(['banana', 42], index=['a','b'])
print(s)
```

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

In [7]:
```
s = pd.Series(['Wes McKinney', 'Creator of Pandas'],index=['Person', 'Who'])
print(s)
```

```
Person         Wes McKinney
Who       Creator of Pandas
dtype: object
```

In [8]:
```
# Creating a list of prices.
pricelist=[100,200,300, 400]

#Creating a series.
productseries=pd.Series(pricelist)

#Displaying the series.
print(productseries)
```

```
0    100
1    200
2    300
3    400
dtype: int64
```

Create data Frame --> pandas.DataFrame

In [9]:
```
# Constructing a DataFrame from a Dictionary
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
df
```

Out[9]:

	col1	col2
0	1	3
1	2	4

In [10]:
```
type(df)
```

Out[10]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
# Constructing a DataFrame from a Dictionary
scientists = pd.DataFrame({
'Name': ['Rosaline Franklin','William Gosset'],
'Occupation': ['Chemist', 'Statistician'],
'Born': ['1920-07-25', '1876-06-13'],
'Died': ['1958-04-16', '1937-10-16'],
'Age': [37, 61]})

#Displaying the data frame
print(scientists)
```

```
                Name    Occupation        Born        Died  Age
0  Rosaline Franklin       Chemist  1920-07-25  1958-04-16   37
1     William Gosset  Statistician  1876-06-13  1937-10-16   61
```

In [12]:
```
# Creating DataFrame using a List
productdf = pd.DataFrame([[100,200, 300, 400], [4, 2,5,6]], columns=['Pen', 'Shir

#Displaying the data frame.
print(productdf)
```

```
   Pen  Shirt  Book  Mouse
0  100    200   300    400
1    4      2     5      6
```

Attributes

In [13]:
```
#Displaying the dimensions of the data frame using shape.
productdf.shape
```

Out[13]:
```
(2, 4)
```

In [14]:
```
#Displaying the size of the data frame using size.
productdf.size
```

Out[14]:
```
8
```

In [15]:
```
#Displaying the name of the columns using keys () function.
productdf.keys()
```

Out[15]:
```
Index(['Pen', 'Shirt', 'Book', 'Mouse'], dtype='object')
```

In [16]:
```
productdf.columns
```

Out[16]:
```
Index(['Pen', 'Shirt', 'Book', 'Mouse'], dtype='object')
```

adding column

In [17]:
```
#adding column named "Mobile" to the data frame.
productdf["Mobile"] =[15000,2]
productdf
```

Out[17]:

	Pen	Shirt	Book	Mouse	Mobile
0	100	200	300	400	15000
1	4	2	5	6	2

Drop columns from the data frame

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

In [18]:
```
#Deleting multiple columns from the data frame using drop() function.
df2 = productdf.drop("Mobile",axis=1)
df2
```

Out[18]:

	Pen	Shirt	Book	Mouse
0	100	200	300	400
1	4	2	5	6

In [19]:
```
#Deleting multiple columns from the data frame using drop() function.
df2 = productdf.drop(columns= ["Mobile"])
df2
```

Out[19]:

	Pen	Shirt	Book	Mouse
0	100	200	300	400
1	4	2	5	6

In [20]:
```
df2 = productdf.drop(columns= ["Mobile",'Book'])
df2
```

Out[20]:

	Pen	Shirt	Mouse
0	100	200	400
1	4	2	6

In [21]:
```
productdf = productdf.drop(columns= ["Mobile"])
productdf
```

Out[21]:

	Pen	Shirt	Book	Mouse
0	100	200	300	400
1	4	2	5	6

Drop rows from the data frame

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

In [22]:
```
#Deleting row from the data frame
df3=df2.drop(index=[0])
df3
```

Out[22]:

	Pen	Shirt	Mouse
1	4	2	6

Combining Dataframes

Append

In [23]:
```
#Creating a new data frame.
productdf2= pd.DataFrame ([[15, 16, 17,18],[5,6,7,8]],columns=['Pen','Shirt','Boo
productdf2
```

Out[23]:

	Pen	Shirt	Book	Mouse
0	15	16	17	18
1	5	6	7	8

In [24]:
```
# Adding rows to the data frame by adding other data frame.
df5 = productdf.append(productdf2)
df5
```

Out[24]:

	Pen	Shirt	Book	Mouse
0	100	200	300	400
1	4	2	5	6
0	15	16	17	18
1	5	6	7	8

In [25]:
```
#Creating a new data frame.
productdf3= pd.DataFrame([[15, 16, 17,18],[5,6,7,8]],columns=['Pens','Shirt','Boo
productdf3
```

Out[25]:

	Pens	Shirt	Book	Mouse
0	15	16	17	18
1	5	6	7	8

In [26]:
```
productdf.append(productdf3)
```

Out[26]:

	Pen	Shirt	Book	Mouse	Pens
0	100.0	200	300	400	NaN
1	4.0	2	5	6	NaN
0	NaN	16	17	18	15.0
1	NaN	6	7	8	5.0

Concatenate

In [27]:
```
india_weather = pd.DataFrame({
"city": ["mumbai","delhi","banglore"],
"temperature": [32,45,30],
"humidity": [80, 60, 78]
})

india_weather
```

Out[27]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78

In [28]:
```
us_weather = pd.DataFrame({
"city": ["new york","chicago","orlando"],
"temperature": [21,14,35],
"humidity": [68, 65, 75]
})
us_weather
```

Out[28]:

	city	temperature	humidity
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

In [29]:
```
#concate two dataframes
df = pd.concat([india_weather, us_weather])
df
```

Out[29]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

In [30]:
```
#if you want continuous index
df = pd.concat([india_weather, us_weather], ignore_index=True)
df
```

Out[30]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78
3	new york	21	68
4	chicago	14	65
5	orlando	35	75

In [31]:
```
df = pd.concat([india_weather, us_weather],axis=0)
df
```

Out[31]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

Merging of DataFrame

In [32]:
```
temperature_df = pd.DataFrame({
"city": ["mumbai","delhi","banglore", 'hyderabad'],
"temperature": [32,45,30,40]})

temperature_df
```

Out[32]:

	city	temperature
0	mumbai	32
1	delhi	45
2	banglore	30
3	hyderabad	40

In [33]:
```
humidity_df = pd.DataFrame({
"city": ["delhi","mumbai","banglore"],
"humidity": [68, 65, 75]})
humidity_df
```

Out[33]:

	city	humidity
0	delhi	68
1	mumbai	65
2	banglore	75

In [34]:
```
# merge two dataframes with out explicitly mention index
df = pd.merge(temperature_df, humidity_df, on='city')
df
```

Out[34]:

	city	temperature	humidity
0	mumbai	32	65
1	delhi	45	68
2	banglore	30	75

In [35]:
```
# OUTER-JOIN
df = pd.merge(temperature_df, humidity_df, on='city', how='outer')
df
```

Out[35]:

	city	temperature	humidity
0	mumbai	32	65.0
1	delhi	45	68.0
2	banglore	30	75.0
3	hyderabad	40	NaN

Import data

In [36]:
```
df = pd.read_csv("Indian Liver Patient.csv")
df.head()
```

Out[36]:

	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 [37]:
```
# we use the head function so Python only shows us the first 5 rows
df.head()
```

Out[37]:

	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 [38]:
```
# show the last 5 observations
df.tail()
```

Out[38]:

	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

In [39]:
```
#shape attribute that will give us the number of rows and columns of the DataFram
df.shape
```

Out[39]: (583, 11)

In [40]:
```
#Return an int representing the number of elements in this object.
df.size
```

Out[40]: 6413

In [41]:
```
# The column labels of the DataFrame
df.columns
```

Out[41]:
```
Index(['Age', 'Gender', 'TB', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB',
       'AG', 'LiverPatient'],
      dtype='object')
```

In [42]:
```
# The data type of each column.
df.dtypes
```

Out[42]:
```
Age              int64
Gender          object
TB             float64
DB             float64
Alkphos          int64
Sgpt             int64
Sgot             int64
TP             float64
ALB            float64
AG             float64
LiverPatient     int64
dtype: object
```

In [43]:
```
#concise summary of a DataFrame
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
```

In [44]:
```
#total no. of Missing values in each column
df.isnull().sum()
```

Out[44]:
```
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 [45]:
```
#Displaying deseriptive statistical values of column using describe ()
df.describe()
```

Out[45]:

	Age	TB	DB	Alkphos	Sgpt	Sgot	TP	
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5

In [46]:
```
df.describe(include="all")
```

Out[46]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	
count	583.000000	583	583.000000	583.000000	583.000000	583.000000	583.000000	583.0
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	6.4
std	16.189833	NaN	6.209522	2.808498	242.937989	182.620356	288.918529	1.0
min	4.000000	NaN	0.400000	0.100000	63.000000	10.000000	10.000000	2.7
25%	33.000000	NaN	0.800000	0.200000	175.500000	23.000000	25.000000	5.8
50%	45.000000	NaN	1.000000	0.300000	208.000000	35.000000	42.000000	6.6
75%	58.000000	NaN	2.600000	1.300000	298.000000	60.500000	87.000000	7.2
max	90.000000	NaN	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.6

In [47]:
```
print(df.describe())
```

```
              Age          TB          DB      Alkphos         Sgpt  \
count  583.000000  583.000000  583.000000   583.000000   583.000000
mean    44.746141    3.298799    1.486106   290.576329    80.713551
std     16.189833    6.209522    2.808498   242.937989   182.620356
min      4.000000    0.400000    0.100000    63.000000    10.000000
25%     33.000000    0.800000    0.200000   175.500000    23.000000
50%     45.000000    1.000000    0.300000   208.000000    35.000000
75%     58.000000    2.600000    1.300000   298.000000    60.500000
max     90.000000   75.000000   19.700000  2110.000000  2000.000000

              Sgot          TP         ALB          AG  LiverPatient
count   583.000000  583.000000  583.000000  579.000000    583.000000
mean    109.910806    6.483190    3.141852    0.947064      1.286449
std     288.918529    1.085451    0.795519    0.319592      0.452490
min      10.000000    2.700000    0.900000    0.300000      1.000000
25%      25.000000    5.800000    2.600000    0.700000      1.000000
50%      42.000000    6.600000    3.100000    0.930000      1.000000
75%      87.000000    7.200000    3.800000    1.100000      2.000000
max    4929.000000    9.600000    5.500000    2.800000      2.000000
```

Single column

In [48]:
```
# just get the TB column and save it to its own variable
df['TB']
```

Out[48]:
```
0       0.7
1      10.9
2       7.3
3       1.0
4       3.9
       ...
578     0.5
579     0.6
580     0.8
581     1.3
582     1.0
Name: TB, Length: 583, dtype: float64
```

In [49]:
```
#When subsetting a single column, you can use dot notation and call the column na
df.TB
```

Out[49]:
```
0       0.7
1      10.9
2       7.3
3       1.0
4       3.9
       ...
578     0.5
579     0.6
580     0.8
581     1.3
582     1.0
Name: TB, Length: 583, dtype: float64
```

Multiple Columns

In [50]:
```
#In order to specify multiple columns by the column name, we need to pass in a py
df[['TB','DB']]
```

Out[50]:

	TB	DB
0	0.7	0.1
1	10.9	5.5
2	7.3	4.1
3	1.0	0.4
4	3.9	2.0
...	...	...
578	0.5	0.1
579	0.6	0.1
580	0.8	0.2
581	1.3	0.5
582	1.0	0.3

583 rows × 2 columns

In [51]:
```
#Determining all the values of column only.
df['Alkphos'].values
```

Out[51]:
```
array([ 187,  699,  490,  182,  195,  208,  154,  202,  202,  290,  210,
        260,  310,  214,  145,  183,  342,  165,  293,  293,  610,  482,
        542,  231,  194,  289,  289,  240,  128,  188,  190,  156,  187,
        410,  410,  482,  145,  374,  263,  275,  168,  160,  630,  415,
        208,  275,  150,  230,  176,  206,  170,  161,  253,  198,  272,
        272,  198,  175,  367,  145,  158,  158,  158,  208,  259,  470,
        195,  215,  239,  215,  186,  188,  205,  171,  145,  162,  518,
       1620,  146,  670,  915,   75,  148,  258,  237,  269,  320,  298,
        538,  238,  214,  308,  298,  204,  168,  282,  298,  215,  265,
        312,  161,  243,  224,  225,  170,  145,  145,  158,  158,  486,
        188,  257,  179,  272,  661, 1580, 1630,  194,  280,  298,  300,
        290,  188,  178,  177,  201,  802,  248, 1896,  263,  512,  237,
        199,  238,  178, 1110,  310,  282,  282,  380,  186,  159,  332,
        332,  189,  201,  168,  392,  202,  286,  180,  218,  182,  178,
        290,  298,  462,  196,  196,  282,  750, 1050,  599,  180,  180,
        282,  332,  292,  962,  950,  200,  298,  750,  175,  175,  198,
        482, 1020,  562,  386,  250,  218,  170,  171,  201,  298,  750,
        191,  614,  218,  314,  257,  272,  206,  209, 1124,  664,  142,
        169, 1420,  218,  218,  145,  142,  135,  163,  285,  350,  220,
        189,  190,  219,  160,  401,  180,  100,  116,  159,  289,  125,
        147,  192,  265,  175,  400,  120,  173,  186,  202,  290,  196,
        282,  157, 2110,  285,  360,  300,  158,  190,  196,  165,  230,
        205,  316,  218,  290,  272,  190,  202,  498,  480,  680,  258,
        180,  152,  859,  901,  335,  182,  285,  245,  505,  228,  185,
        247,  348,  195,  140,  358,  110,  235,  460,  380,  262,  196,
        180,  190,  190,  209,  144,  123,  192,  188,  316,  300,  575,
        192,  155,  239,  315,  250,  174,  245,  191,  340,  202,  234,
        159,  190,  195,  180,  280,  430,  206,  155,  195,  588,  174,
        165,  527,  175,  574,  106,  158,  195,  179,  182,  198,  216,
        310,   63,  198,  205,  302,  171,  158,  358,  174,  192,  211,
        157,  210,  258,  152,  350,  182,  458,  375,  405,  215,  206,
        650,  198,  198,  195,  230,  115,  216,  358,  158,  145,  195,
        144,  621,  150,  178,  256,  205,  176,  146,  218,  182,  215,
        165,  183,  176,  418,  271,  182,  130,  558,  135,  326,  140,
        145,  206,  168,  202,  192,  185,  331,  188,  172,  159,  490,
        152,  105,  160,  160,  102,  148,  162,  149,  580,  310,  140,
        175,  152,  208,  205,  162,   92,  162,  199,  198,  215,  180,
        719,  554,  555,  215,  509,  190,  208,  260,  690,  862,  592,
        450, 1350, 1350,  163,  246,  178,  240,  100,  166,  170,  194,
       1750,  182,  236,  165,  201,  194,  206,  212,  157,  162,  168,
        198,  292,  298,  152,  163,  279,  181, 1550,  142,  173,  282,
        279, 1100,  224,  159,  186,  189,  192,  140,  686,  215,  309,
        110,  130,  164,  270,  137,   90,  190,  165,  167,  185,  197,
        154,  226,  310,  310,  220,  196,  186,  352,  282,   92,  182,
        103,  850,  195,  276,  171,  146,  193,  180,  805,  265,  185,
        165,  189,  198,  151,  349,  365,  305,  127,  238,  218,  219,
        239,  194,  450,  254,  205,  320,  195,  215,  230,  189,  168,
        215,  210,  108,  224,  230,  185,  137,  156,  210,  268,  298,
        315,  214,  138,  285,  162,  298,  230,  466,  227,  395,   97,
        406,  114,  198,  173,  204,  350,  153,  188,  380,  214,  768,
        172,  160,  196,  232,  220,  290,  180,  189,  275,  390,  356,
        315,  388,  298,  165,  143,  191,  251,  200,  268,  236,  215,
        134,  612,  515,  560,  289,  190,  500,   98,  245,  184,  216],
      dtype=int64)
```

In [52]:
```
df
```

Out[52]:

	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

Create a new Column

In [53]:
```
df["m"] = df["TB"]/2
df
```

Out[53]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient	m
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1	0.35
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1	5.45
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1	3.65
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1	0.50
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1	1.95
...	...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2	0.25
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1	0.30
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1	0.40
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1	0.65
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2	0.50

583 rows × 12 columns

In [54]:
```
df.drop("m",axis=1,inplace=True)
df
```

Out[54]:

	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

Selecting particular rows & columns

iloc

In [55]:
```
# get the first row
df.iloc[0]
```

Out[55]:
```
Age                65
Gender         Female
TB                0.7
DB                0.1
Alkphos           187
Sgpt               16
Sgot               18
TP                6.8
ALB               3.3
AG                0.9
LiverPatient        1
Name: 0, dtype: object
```

In [56]:
```
## get the 100th row
df.iloc[99]
```

Out[56]:
```
Age               18
Gender          Male
TB               0.7
DB               0.1
Alkphos          312
Sgpt             308
Sgot             405
TP               6.9
ALB              3.7
AG               1.1
LiverPatient       1
Name: 99, dtype: object
```

In [57]:
```
# select the first, 100th, and 500th row
df.iloc[[0, 99, 499]]
```

Out[57]:

	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	0.9	1
99	18	Male	0.7	0.1	312	308	405	6.9	3.7	1.1	1
499	55	Male	3.6	1.6	349	40	70	7.2	2.9	0.6	1

In [58]:
```
# select the 4th row & 3rd Column
df.iloc[3,2]
```

Out[58]:
```
1.0
```

In [59]:
```
#Displaying multiple specified columns and rows.
df.iloc[[5,9],[1,4]]
```

Out[59]:

	Gender	Alkphos
5	Male	208
9	Male	290

In [60]:
```
#Displaying specific column of range of rows.
df.iloc[7:15,[5]]
```

Out[60]:

	Sgpt
7	14
8	22
9	53
10	51
11	31
12	61
13	22
14	53

In [61]:
```
# get the first row
print(df.loc[0])
```

```
Age                65
Gender         Female
TB                0.7
DB                0.1
Alkphos           187
Sgpt               16
Sgot               18
TP                6.8
ALB               3.3
AG                0.9
LiverPatient        1
Name: 0, dtype: object
```

In [62]:
```
# get the 100th row
# recall that values start with 0
print(df.loc[99])
```

```
Age               18
Gender          Male
TB               0.7
DB               0.1
Alkphos          312
Sgpt             308
Sgot             405
TP               6.9
ALB              3.7
AG               1.1
LiverPatient       1
Name: 99, dtype: object
```

In [63]:
```
# get the last row
print(df.loc[-1])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(self, k
ey, method, tolerance)
    349             try:
--> 350                 return self._range.index(new_key)
    351             except ValueError:

ValueError: -1 is not in range

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-63-2d6dcc44a67f> in <module>
      1 # get the last row
----> 2 print(df.loc[-1])

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, ke
y)
   1766
   1767             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1768             return self._getitem_axis(maybe_callable, axis=axis)
   1769
   1770     def _is_scalar_access(self, key: Tuple):

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self,
 key, axis)
   1963         # fall thru to straight lookup
   1964         self._validate_key(key, axis)
-> 1965         return self._get_label(key, axis=axis)
   1966
   1967

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_label(self, lab
el, axis)
    623             raise IndexingError("no slices here, handle elsewhere")
    624
--> 625         return self.obj._xs(label, axis=axis)
    626
    627     def _get_loc(self, key: int, axis: int):

~\Anaconda3\lib\site-packages\pandas\core\generic.py in xs(self, key, axis, l
evel, drop_level)
   3535             loc, new_index = self.index.get_loc_level(key, drop_level
=drop_level)
   3536         else:
-> 3537             loc = self.index.get_loc(key)
   3538
   3539             if isinstance(loc, np.ndarray):

~\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(self, k
ey, method, tolerance)
    350                 return self._range.index(new_key)
    351             except ValueError:
```

```
--> 352                 raise KeyError(key)
    353         return super().get_loc(key, method=method, tolerance=toleranc
e)
    354

KeyError: -1
```

In [64]:
```
df.tail()
```

Out[64]:

	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

In [65]:
```
df.loc[582]
```

Out[65]:
```
Age               38
Gender          Male
TB                 1
DB               0.3
Alkphos          216
Sgpt              21
Sgot              24
TP               7.3
ALB              4.4
AG               1.5
LiverPatient       2
Name: 582, dtype: object
```

In [66]:
```
df.loc[1:5,]
```

Out[66]:

	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

In [67]:
```
#Retrieving selected rows with range of columns between 'TB' and 'TP
df.loc[[5,6],'TB':'TP']
```

Out[67]:

	TB	DB	Alkphos	Sgpt	Sgot	TP
5	1.8	0.7	208	19	14	7.6
6	0.9	0.2	154	16	12	7.0

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

In [68]:
```
df.loc[42,'Age']
```

Out[68]: 42

In [69]:
```
df.iloc[42, 0]
```

Out[69]: 42

In [70]:
```
df.iloc[0,0]
```

Out[70]: 65

In [71]:
```
df.iloc[[0, 99, 499], [0, 3, 5]]
```

Out[71]:

	Age	DB	Sgpt
0	65	0.1	16
99	18	0.1	308
499	55	1.6	40

SetIndex

In [72]:
```
l1= df.set_index("Age")
l1.head()
```

Out[72]:

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

In [73]:
```
l1.shape
```

Out[73]: (583, 10)

In [74]:
```
l1.loc[40]
```

Out[74]:

Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
40	Female	0.9	0.3	293	232	245	6.8	3.1	0.8	1
40	Female	0.9	0.3	293	232	245	6.8	3.1	0.8	1
40	Male	1.9	1.0	231	16	55	4.3	1.6	0.6	1
40	Male	1.1	0.3	230	1630	960	4.9	2.8	1.3	1
40	Male	0.6	0.1	171	20	17	5.4	2.5	0.8	1
40	Male	3.6	1.8	285	50	60	7.0	2.9	0.7	1
40	Male	3.9	1.7	350	950	1500	6.7	3.8	1.3	1
40	Female	0.9	0.2	285	32	27	7.7	3.5	0.8	1
40	Male	0.9	0.3	196	69	48	6.8	3.1	0.8	1
40	Male	0.7	0.1	202	37	29	5.0	2.6	1.0	1
40	Male	14.5	6.4	358	50	75	5.7	2.1	0.5	1
40	Male	0.7	0.2	176	28	43	5.3	2.4	0.8	2
40	Male	3.5	1.6	298	68	200	7.1	3.4	0.9	1
40	Male	30.8	18.3	285	110	186	7.9	2.7	0.5	1
40	Male	1.2	0.6	204	23	27	7.6	4.0	1.1	1
40	Female	2.1	1.0	768	74	141	7.8	4.9	1.6	1
40	Male	0.6	0.1	98	35	31	6.0	3.2	1.1	1

In [75]:
```
l1.loc[60, 'TB':'ALB']
```

Out[75]:

Age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB
60	0.8	0.2	215	24	17	6.3	3.0
60	4.0	1.9	238	119	350	7.1	3.3
60	5.7	2.8	214	412	850	7.3	3.2
60	6.8	3.2	308	404	794	6.8	3.0
60	8.6	4.0	298	412	850	7.4	3.0
60	5.8	2.7	204	220	400	7.0	3.0
60	5.2	2.4	168	126	202	6.8	2.9
60	1.8	0.5	201	45	25	3.9	1.7
60	0.6	0.1	186	20	21	6.2	3.3
60	0.8	0.2	286	21	27	7.1	4.0
60	11.0	4.9	750	140	350	5.5	2.1
60	11.5	5.0	1050	99	187	6.2	2.8
60	5.8	2.7	599	43	66	5.4	1.8
60	22.8	12.6	962	53	41	6.9	3.3
60	8.9	4.0	950	33	32	6.8	3.1
60	3.2	1.8	750	79	145	7.8	3.2
60	2.1	1.0	191	114	247	4.0	1.6
60	1.9	0.8	614	42	38	4.5	1.8
60	6.3	3.2	314	118	114	6.6	3.7
60	5.8	3.0	257	107	104	6.6	3.5
60	2.3	0.6	272	79	51	6.6	3.5
60	2.4	1.0	1124	30	54	5.2	1.9
60	2.0	1.1	664	52	104	6.0	2.1
60	1.5	0.6	360	230	298	4.5	2.0
60	2.0	0.8	190	45	40	6.0	2.8
60	0.7	0.2	174	32	14	7.8	4.2
60	2.6	1.2	171	42	37	5.4	2.7
60	2.9	1.3	230	32	44	5.6	2.0
60	2.2	1.0	271	45	52	6.1	2.9
60	1.4	0.7	159	10	12	4.9	2.5
60	0.7	0.2	171	31	26	7.0	3.5
60	0.9	0.3	168	16	24	6.7	3.0
60	19.6	9.5	466	46	52	6.1	2.0

Age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB
60	0.5	0.1	500	20	34	5.9	1.6

In [76]:
```
l1.reset_index()
```

Out[76]:

	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

In [77]:
```
#Determine number of records of each category in column
df['Gender'].value_counts()
```

Out[77]:
```
Male      441
Female    142
Name: Gender, dtype: int64
```

In [78]:
```
#Determine number of records based on gender in percentage form.
df['Gender'].value_counts()/len(df['Gender'])
```

Out[78]:
```
Male      0.756432
Female    0.243568
Name: Gender, dtype: float64
```

In [79]:
```
#Determine number of records based on gender in percentage form.
df['Gender'].value_counts(normalize=True)
```

Out[79]:
```
Male      0.756432
Female    0.243568
Name: Gender, dtype: float64
```

In [80]:
```
#Determining mean of Age column using mean () function.
df['Age'].mean()
```

Out[80]:
```
44.74614065180103
```

In [81]:
```
#Determining median of Age column using median () function.
df['Age'].median()
```

Out[81]:
```
45.0
```

In [82]:
```
# Determining maximum of Akphos column using max () function.
df['Alkphos'].max()
```

Out[82]:
```
2110
```

In [83]:
```
#Detemining minimum of Alkphos column using min () function.
df['Alkphos'].min()
```

Out[83]:
```
63
```

In [84]:
```
#Determining sum of 'TB' column using sum () function.
df['TB'].sum()
```

Out[84]:
```
1923.1999999999998
```

In [85]:
```
# Determining product of 'TB' column using product () function.
df['TB'].product()
```

Out[85]:
```
2.163950516974104e+117
```

In [86]:
```
#Determining 5 smallest values of 'DB' using nsmalllest () function.
df['DB'].nsmallest()
```

Out[86]:
```
0     0.1
10    0.1
15    0.1
41    0.1
48    0.1
Name: DB, dtype: float64
```

In [87]:
```
#Determining 5 largest values of 'DB' using nlargest () function.
df['DB'].nlargest()
```

Out[87]:
```
559    19.7
531    18.3
504    17.1
259    14.2
505    14.1
Name: DB, dtype: float64
```

Sort Functions

In [88]:
```
# Sorting in descending order.
df.sort_values(by='TP',ascending=False)
```

Out[88]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
273	30	Male	0.7	0.2	262	15	18	9.6	4.7	1.2	1
270	37	Male	0.7	0.2	235	96	54	9.5	4.9	1.0	1
519	35	Male	26.3	12.1	108	168	630	9.2	2.0	0.3	1
510	37	Female	0.8	0.2	205	31	36	9.2	4.6	1.0	2
335	13	Female	0.7	0.1	182	24	19	8.9	4.9	1.2	1
...	...	...	...	...	...	...	...	...	...	...	...
533	46	Female	1.4	0.4	298	509	623	3.6	1.0	0.3	1
458	26	Male	6.8	3.2	140	37	19	3.6	0.9	0.3	1
181	75	Male	2.9	1.3	218	33	37	3.0	1.5	1.0	1
269	26	Male	0.6	0.1	110	15	20	2.8	1.6	1.3	1
180	75	Male	2.8	1.3	250	23	29	2.7	0.9	0.5	1

583 rows × 11 columns

In [89]:
```
#Sorting in ascending order.
df.sort_values(by='TP',ascending=True)
```

Out[89]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
180	75	Male	2.8	1.3	250	23	29	2.7	0.9	0.5	1
269	26	Male	0.6	0.1	110	15	20	2.8	1.6	1.3	1
181	75	Male	2.9	1.3	218	33	37	3.0	1.5	1.0	1
458	26	Male	6.8	3.2	140	37	19	3.6	0.9	0.3	1
533	46	Female	1.4	0.4	298	509	623	3.6	1.0	0.3	1
...	...	...	...	...	...	...	...	...	...	...	...
335	13	Female	0.7	0.1	182	24	19	8.9	4.9	1.2	1
519	35	Male	26.3	12.1	108	168	630	9.2	2.0	0.3	1
510	37	Female	0.8	0.2	205	31	36	9.2	4.6	1.0	2
270	37	Male	0.7	0.2	235	96	54	9.5	4.9	1.0	1
273	30	Male	0.7	0.2	262	15	18	9.6	4.7	1.2	1

583 rows × 11 columns

Data Extraction

In [90]:
```
#Displaying the records wheré gender is male.
male_data=df[df["Gender"]=="Male"]
male_data
```

Out[90]:

	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

441 rows × 11 columns

In [91]:
```
#Displaying the records where Age is greater than equal to 50
df[df['Age']>=50]
```

Out[91]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
566	50	Female	1.0	0.3	191	22	31	7.8	4.0	1.00	2
567	52	Male	2.7	1.4	251	20	40	6.0	1.7	0.39	1
571	90	Male	1.1	0.3	215	46	134	6.9	3.0	0.70	1
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1

230 rows × 11 columns

In [92]:
```
#Displaying the records where ALB is less than or equal to 1.
df[df['ALB']<=1]
```

Out[92]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
180	75	Male	2.8	1.3	250	23	29	2.7	0.9	0.5	1
458	26	Male	6.8	3.2	140	37	19	3.6	0.9	0.3	1
533	46	Female	1.4	0.4	298	509	623	3.6	1.0	0.3	1

LogicalOperators

In [93]:
```
#Creating a new subset using "and" for multiple conditions.
filter1=df[(df['Age']>=35) & (df['DB']<=6)]
print("Shape of new dataset using and is:", filter1.shape)
```

```
Shape of new dataset using and is: (390, 11)
```

In [94]:
```
filter1
```

Out[94]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
571	90	Male	1.1	0.3	215	46	134	6.9	3.0	0.70	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
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

390 rows × 11 columns

In [95]:
```
#Creating a new subset using "or" operator for multiple conditions
filter2=df[(df['Gender']=="Female") | (df['Age']>=35) | (df['DB']<=6)]
print("Shape of new dataset using or is:", filter2.shape)
```

```
Shape of new dataset using or is: (569, 11)
```

In [96]:
```
#Using both "and" and "or" together for multiple conditions.
filter3=df[(df['LiverPatient']==1) & (df.Age>=50) | (df['TP']>=2) |(df.ALB>2)]
print("Shape of new dataset using and & or is:",filter3.shape)
```

```
Shape of new dataset using and & or is: (583, 11)
```

In [97]:
```
#Using= condition for selected columns.
df.loc[df['DB']==2,'Age':'DB']
```

Out[97]:

	Age	Gender	TB	DB
4	72	Male	3.9	2.0
25	34	Male	4.1	2.0
26	34	Male	4.1	2.0

In [98]:
```
#Using < condition for selected columns
df.loc[df['TB']<0.5,'Gender':'DB']
```

Out[98]:

	Gender	TB	DB
425	Male	0.4	0.1

In [99]:
```
#Using > condition for selected columns
df.loc[df['Age']>80,'Age':'DB']
```

Out[99]:

	Age	Gender	TB	DB
29	84	Female	0.7	0.2
44	85	Female	1.0	0.3
571	90	Male	1.1	0.3

In [100]:
```
#Using > and < conditions together (using &) for selected columns.
df.loc[(df['Sgpt']>400) & (df['Sgpt']<= 420),'TB':'Alkphos']
```

Out[100]:

	TB	DB	Alkphos
43	2.6	1.2	415
90	5.7	2.8	214
91	6.8	3.2	308
92	8.6	4.0	298

Groupby

In [101]:
```
#Grouping on basis of "Gender" and using sum () function for "TB".
df['TB'].groupby([df['Gender']]).head()
```

Out[101]:
```
0      0.7
1     10.9
2      7.3
3      1.0
4      3.9
5      1.8
6      0.9
7      0.9
13     1.1
18     0.9
Name: TB, dtype: float64
```

crosstab

In [102]:
```
#using crosstab () for determining observations of two categorical variables.
df_c = pd.crosstab(df.Gender,df.LiverPatient,margins=True)
df_c
```

Out[102]:

LiverPatient	1	2	All
Gender			
Female	92	50	142
Male	324	117	441
All	416	167	583