Pandas notes 1

In [4]:

```
import pandas as pd
```

In [1]:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
titanic_train = pd.read_csv("https://gist.githubusercontent.com/michhar/2dfd2de0d4f8727
                            sep='\t')
```

In [5]:

```
titanic_train.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [6]:

```
1 titanic_train.tail()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
151	152	1	1	Pears, Mrs. Thomas (Edith Wearne)	female	22.0	1	0	113776	66.6000	
152	153	0	3	Meo, Mr. Alfonzo	male	55.5	0	0	A.5. 11206	8.0500	N
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.5000	N
154	155	0	3	Olsen, Mr. Ole Martin	male	NaN	0	0	Fa 265302	7.3125	N
155	156	0	1	Williams, Mr. Charles Duane	male	51.0	0	1	PC 17597	61.3792	N

In [7]:

```
1 titanic_train.dtypes
```

Out[7]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [8]:

```
1 titanic_train.columns
```

Out[8]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [9]:

```
titanic_train.dtypes == "object"
a = titanic_train.dtypes[titanic_train.dtypes == "object"].index
a
titanic_train[a].describe()

```

Out[9]:

	Name	Sex	Ticket	Cabin	Embarked
count	156	156	156	31	155
unique	156	2	145	28	3
top	Emir, Mr. Farred Chehab	male	W./C. 6608	D26	S
freq	1	100	2	2	110

In [10]:

```
titanic_train.describe()     # all numeric data type discription int and float  not tak
# 25% is the 25 percentile of data after arranging in acending order
# same as 50% 75% and so on.

#check for categorical data
# take only categorical data and then apply describe() for that categorical data
```

Out[10]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	156.000000	156.000000	156.000000	126.000000	156.000000	156.000000	156.000000
mean	78.500000	0.346154	2.423077	28.141508	0.615385	0.397436	28.109587
std	45.177428	0.477275	0.795459	14.613880	1.056235	0.870146	39.401047
min	1.000000	0.000000	1.000000	0.830000	0.000000	0.000000	6.750000
25%	39.750000	0.000000	2.000000	19.000000	0.000000	0.000000	8.003150
50%	78.500000	0.000000	3.000000	26.000000	0.000000	0.000000	14.454200
75%	117.250000	1.000000	3.000000	35.000000	1.000000	0.000000	30.371850
max	156.000000	1.000000	3.000000	71.000000	5.000000	5.000000	263.000000

In [11]:

```
titanic_train.dtypes[titanic_train.dtypes == "object"].index   # filter categorical dat
```

Out[11]:

```
Index(['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], dtype='object')
```

In [12]:

```
titanic_train.dtypes =="object"   #filter all the column aof oject
```

Out[12]:

```
PassengerId    False
Survived       False
Pclass         False
Name            True
Sex             True
Age            False
SibSp          False
Parch          False
Ticket          True
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

In [13]:

```
titanic_train.dtypes[titanic_train.dtypes == "object"]


titanic_train.dtypes[titanic_train.dtypes == "object"].index
# index of the series
```

Out[13]:

```
Index(['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], dtype='object')
```

In [17]:

```
s=titanic_train.dtypes
```

In [19]:

```
s     # storr all the series of indexes
```

Out[19]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [23]:

```
titanic_train.dtypes[titanic_train.dtypes == "object"]  # selection operation for ALL t
# and all the columns name are get by index command.
```

Out[23]:

```
Name        object
Sex         object
Ticket      object
Cabin       object
Embarked    object
dtype: object
```

In [24]:

```
titanic_train.dtypes[titanic_train.dtypes == "float"]
```

Out[24]:

```
Age     float64
Fare    float64
dtype: object
```

In [25]:

```
list_col = titanic_train.dtypes[titanic_train.dtypes == "object"].index
titanic_train[list_col].describe()
```

Out[25]:

	Name	Sex	Ticket	Cabin	Embarked
count	156	156	156	31	155
unique	156	2	145	28	3
top	Emir, Mr. Farred Chehab	male	W./C. 6608	D26	S
freq	1	100	2	2	110

In [26]:

```
titanic_train[['Name','Sex','Age']]
```

Out[26]:

	Name	Sex	Age
0	Braund, Mr. Owen Harris	male	22.0
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0
2	Heikkinen, Miss. Laina	female	26.0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0
4	Allen, Mr. William Henry	male	35.0
...	...	...	...
151	Pears, Mrs. Thomas (Edith Wearne)	female	22.0
152	Meo, Mr. Alfonzo	male	55.5
153	van Billiard, Mr. Austin Blyler	male	40.5
154	Olsen, Mr. Ole Martin	male	NaN
155	Williams, Mr. Charles Duane	male	51.0

156 rows × 3 columns

In [36]:

```
s=titanic_train['Name']
s[0:100:5]   # slicing data, step size are also included like for loop, 5 is step size
# not giving data in reverse order because it move in negative direction so we have to
```

Out[36]:

```
0                              Braund, Mr. Owen Harris
5                                    Moran, Mr. James
10                       Sandstrom, Miss. Marguerite Rut
15                      Hewlett, Mrs. (Mary D Kingcome)
20                                  Fynney, Mr. Joseph J
25    Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...
30                             Uruchurtu, Don. Manuel E
35                       Holverson, Mr. Alexander Oskar
40          Ahlin, Mrs. Johan (Johanna Persdotter Larsson)
45                             Rogers, Mr. William John
50                           Panula, Master. Juha Niilo
55                                   Woolner, Mr. Hugh
60                              Sirayanian, Mr. Orsen
65                             Moubarek, Master. Gerios
70                           Jenkin, Mr. Stephen Curnow
75                              Moen, Mr. Sigurd Hansen
80                                 Waelens, Mr. Achille
85    Backstrom, Mrs. Karl Alfred (Maria Mathilda Gu...
90                                 Christmann, Mr. Emil
95                           Shorney, Mr. Charles Joseph
Name: Name, dtype: object
```

In [38]:

```
# reverse order   -3 move the diirection in negative direction.
s[100:0:-3]
```

Out[38]:

```
100                              Petranec, Miss. Matilda
97                         Greenfield, Mr. William Bertram
94                                     Coxon, Mr. Daniel
91                               Andreasson, Mr. Paul Edvin
88                               Fortune, Miss. Mabel Helen
85     Backstrom, Mrs. Karl Alfred (Maria Mathilda Gu...
82                           McDermott, Miss. Brigdet Delia
79                                 Dowdell, Miss. Elizabeth
76                                        Staneff, Mr. Ivan
73                             Chronopoulos, Mr. Apostolos
70                              Jenkin, Mr. Stephen Curnow
67                                Crease, Mr. Ernest James
64                                   Stewart, Mr. Albert A
61                                     Icard, Miss. Amelie
58                               West, Miss. Constance Mirium
55                                        Woolner, Mr. Hugh
52               Harper, Mrs. Henry Sleeper (Myna Haxtun)
49          Arnold-Franchi, Mrs. Josef (Josefine Franchi)
46                                        Lennon, Mr. Denis
43               Laroche, Miss. Simonne Marie Anne Andree
40         Ahlin, Mrs. Johan (Johanna Persdotter Larsson)
37                                Cann, Mr. Ernest Charles
34                                 Meyer, Mr. Edgar Joseph
31         Spencer, Mrs. William Augustus (Marie Eugenie)
28                           O'Dwyer, Miss. Ellen "Nellie"
25     Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...
22                               McGowan, Miss. Anna "Annie"
19                                   Masselmani, Mrs. Fatima
16                                      Rice, Master. Eugene
13                               Andersson, Mr. Anders Johan
10                           Sandstrom, Miss. Marguerite Rut
7                             Palsson, Master. Gosta Leonard
4                                    Allen, Mr. William Henry
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
Name: Name, dtype: object
```

In [44]:

```
sorted(titanic_train["Name"])[5:10:2]  # sorted data of name from 5 to 10
```

Out[44]:

```
['Andersson, Mr. August Edvard ("Wennerstrom")',
 'Andrew, Mr. Edgardo Samuel',
 'Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)']
```

In [57]:

```
titanic_train["Name"].describe()
```

Out[57]:

```
count                          156
unique                         156
top       Emir, Mr. Farred Chehab
freq                             1
Name: Name, dtype: object
```

In [58]:

```
titanic_train[["Ticket"]][4:9]
```

Out[58]:

	Ticket
4	373450
5	330877
6	17463
7	349909
8	347742

In [59]:

```
titanic_train["Ticket"].describe()

```

Out[59]:

```
count            156
unique           145
top       W./C. 6608
freq               2
Name: Ticket, dtype: object
```

In [60]:

```
titanic_train.columns
```

Out[60]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'sudh'],
      dtype='object')
```

In [61]:

```
titanic_train["sudh"]="sdffs"  # created new column
titanic_train.head()
```

Out[61]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [63]:

```
#all the name starting with s
a=titanic_train["Name"]
for i in a:
    if i[0][0]=='S':
        print(i)
```

```
Sandstrom, Miss. Marguerite Rut
Saundercock, Mr. William Henry
Sloper, Mr. William Thompson
Spencer, Mrs. William Augustus (Marie Eugenie)
Samaan, Mr. Youssef
Sirayanian, Mr. Orsen
Skoog, Master. Harald
Stewart, Mr. Albert A
Staneff, Mr. Ivan
Sheerlinck, Mr. Jan Baptist
Slocovski, Mr. Selman Francis
Shorney, Mr. Charles Joseph
Salkjelsvik, Miss. Anna Kristine
Sobey, Mr. Samuel James Hayden
```

In [ ]:

```

```

In [64]:

```
1 titanic_train["Cabin"][0:15]       # Check the first 15 tickets
```

Out[64]:

```
0      NaN
1      C85
2      NaN
3     C123
4      NaN
5      NaN
6      E46
7      NaN
8      NaN
9      NaN
10      G6
11    C103
12     NaN
13     NaN
14     NaN
Name: Cabin, dtype: object
```

In [65]:

```
1 titanic_train.dtypes# Check number of unique cabins
```

Out[65]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
sudh            object
dtype: object
```

In [ ]:

```
1
```

In [66]:

```
1 titanic_train["Survived"]
```

Out[66]:

```
0      0
1      1
2      1
3      1
4      0
      ..
151    1
152    0
153    0
154    0
155    0
Name: Survived, Length: 156, dtype: int64
```

In [67]:

```
1 titanic_train.columns
```

Out[67]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'sudh'],
      dtype='object')
```

In [68]:

```
1 titanic_train.describe()
```

Out[68]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	156.000000	156.000000	156.000000	126.000000	156.000000	156.000000	156.000000
mean	78.500000	0.346154	2.423077	28.141508	0.615385	0.397436	28.109587
std	45.177428	0.477275	0.795459	14.613880	1.056235	0.870146	39.401047
min	1.000000	0.000000	1.000000	0.830000	0.000000	0.000000	6.750000
25%	39.750000	0.000000	2.000000	19.000000	0.000000	0.000000	8.003150
50%	78.500000	0.000000	3.000000	26.000000	0.000000	0.000000	14.454200
75%	117.250000	1.000000	3.000000	35.000000	1.000000	0.000000	30.371850
max	156.000000	1.000000	3.000000	71.000000	5.000000	5.000000	263.000000

In [70]:

```
titanic_train["Pclass"]
```

Out[70]:

```
0      3
1      1
2      3
3      1
4      3
      ..
151    1
152    3
153    3
154    3
155    1
Name: Pclass, Length: 156, dtype: int64
```

In [75]:

```
# Categorical give categories and there value, founf distict categories same as unique
```

In [76]:

```
new_Pclass = pd.Categorical(titanic_train["Pclass"])

new_Pclass
```

Out[76]:

```
[3, 1, 3, 1, 3, ..., 1, 3, 3, 3, 1]
Length: 156
Categories (3, int64): [1, 2, 3]
```

In [77]:

```
new_Pclass = pd.Categorical(titanic_train["Sex"])
new_Pclass
```

Out[77]:

```
[male, female, female, female, male, ..., female, male, male, male, male]
Length: 156
Categories (2, object): [female, male]
```

In [80]:

```
titanic_train["Cabin"].unique()   # Check unique cabins
# give total number of unique data

```

Out[80]:

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2'], dtype=object)
```

In [85]:

```
titanic_train["Cabin"].unique()
```

Out[85]:

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2'], dtype=object)
```

In [87]:

```
type(titanic_train["Pclass"])
```

Out[87]:

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

In [90]:

```
# extract first alphabet of all the Cabin by using list comprehensions
import numpy as np
char_cabin = titanic_train["Cabin"].astype(str) # Convert data to str

new_Cabin = [cabin[0] for cabin in char_cabin] # Take/extract first letter

new_Cabin = pd.Categorical(new_Cabin)
#new_Cabin
new_Cabin

```

Out[90]:

```
[n, C, n, C, n, ..., C, n, n, n, n]
Length: 156
Categories (8, object): [A, B, C, D, E, F, G, n]
```

In [91]:

```
1 titanic_train.head()
```

Out[91]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [186]:

```
titanic_train["Cabin"] = new_Cabin
titanic_train.head()
```

Out[186]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [92]:

```
titanic_train['Name']
```

Out[92]:

```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
                           ...
151                 Pears, Mrs. Thomas (Edith Wearne)
152                                   Meo, Mr. Alfonzo
153                    van Billiard, Mr. Austin Blyler
154                             Olsen, Mr. Ole Martin
155                         Williams, Mr. Charles Duane
Name: Name, Length: 156, dtype: object
```

In [94]:

```
[i[0] for i in titanic_train['Name']]  # first alphabet of all the name
```

Out[94]:

```
['B',
 'C',
 'H',
 'F',
 'A',
 'M',
 'M',
 'P',
 'J',
 'N',
 'S',
 'B',
 'S',
 'A',
 'V',
 'H',
 'R',
 'W',
 'V',
 'M',
 'F',
 'B',
 'M',
 'S',
 'P',
 'A',
 'E',
 'F',
 'O',
 'T',
 'U',
 'S',
 'G',
 'W',
 'M',
 'H',
 'M',
 'C',
 'V',
 'N',
 'A',
 'T',
 'K',
 'L',
 'D',
 'R',
 'L',
 'O',
 'S',
 'A',
 'P',
 'N',
 'H',
 'F',
 'O',
```

```
    'W',
    'R',
    'N',
    'W',
    'G',
    'S',
    'I',
    'H',
    'S',
    'S',
    'M',
    'N',
    'C',
    'A',
    'K',
    'J',
    'G',
    'H',
    'C',
    'B',
    'M',
    'S',
    'M',
    'C',
    'D',
    'W',
    'S',
    'M',
    'C',
    'I',
    'B',
    'F',
    'S',
    'F',
    'C',
    'C',
    'A',
    'C',
    'D',
    'C',
    'S',
    'G',
    'G',
    'D',
    'K',
    'P',
    'P',
    'W',
    'J',
    'G',
    'M',
    'S',
    'M',
    'R',
    'M',
    'P',
    'Z',
    'B',
    'J',
    'A',
    'P',
```

```
 'C',
 'T',
 'B',
 'A',
 'H',
 'M',
 'N',
 'W',
 'W',
 'N',
 'M',
 'M',
 'P',
 'E',
 'D',
 'C',
 'R',
 'W',
 'S',
 'R',
 'N',
 'F',
 'O',
 'G',
 'B',
 'N',
 'H',
 'B',
 'A',
 'N',
 'A',
 'F',
 'N',
 'B',
 'B',
 'P',
 'M',
 'v',
 'O',
 'W']
```

In [96]:

```
1 pd.Series([i[0] for i in titanic_train['Name']]).unique()
```

Out[96]:

```
array(['B', 'C', 'H', 'F', 'A', 'M', 'P', 'J', 'N', 'S', 'V', 'R', 'W',
       'E', 'O', 'T', 'U', 'G', 'K', 'L', 'D', 'I', 'Z', 'v'],
      dtype=object)
```

In [98]:

```
1 pd.Categorical([i[0] for i in titanic_train['Name']])
```

Out[98]:

```
[B, C, H, F, A, ..., P, M, v, O, W]
Length: 156
Categories (24, object): [A, B, C, D, ..., V, W, Z, v]
```

In [100]:

```
for i in titanic_train['Name']:
    print(i[0])

# l=[]
# for i in titanic_train['Name']:
#     l.append(i[0])
```

```
B
C
H
F
A
M
M
P
J
N
S
B
S
A
V
H
R
W
V
M
F
B
M
S
P
A
E
F
O
T
U
S
G
W
M
H
M
C
V
N
A
T
K
L
D
R
L
O
S
A
P
N
```

H
F
O
W
R
N
W
G
S
I
H
S
S
M
N
C
A
K
J
G
H
C
B
M
S
M
C
D
W
S
M
C
I
B
F
S
F
C
C
A
C
D
C
S
G
G
D
K
P
P
W
J
G
M
S
M
R
M
P
Z
B

J
A
P
C
T
B
A
H
M
N
W
W
N
M
M
P
E
D
C
R
W
S
R
N
F
O
G
B
N
H
B
A
N
A
F
N
B
B
P
M
v
O
W

In [102]:

```
titanic_train["Age"].isnull() # how many have null value, give true if having null valu
```

Out[102]:

```
0      False
1      False
2      False
3      False
4      False
       ...
151    False
152    False
153    False
154     True
155    False
Name: Age, Length: 156, dtype: bool
```

In [106]:

```
missing = np.where(titanic_train["Age"].isnull() == True)   # give rows number havng nu
missing
```

Out[106]:

```
(array([  5,  17,  19,  26,  28,  29,  31,  32,  36,  42,  45,  46,  47,
         48,  55,  64,  65,  76,  77,  82,  87,  95, 101, 107, 109, 121,
        126, 128, 140, 154], dtype=int64),)
```

In [107]:

```
np.where(titanic_train["Fare"]==max(titanic_train["Fare"]))
```

Out[107]:

```
(array([27, 88], dtype=int64),)
```

In [110]:

```
# get datasets by using indexes
# iloc are used
```

In [115]:

```
titanic_train.iloc[np.where(titanic_train["Fare"]==max(titanic_train["Fare"]))]
```

Out[115]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0	C23 C25 C27
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0	C23 C25 C27

In [116]:

```
index=np.where(titanic_train["Age"].isnull() == True)
titanic_train.iloc[index]
```

Out[116]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	1
32	33	1	3	Glynn, Miss. Mary Agatha	female	NaN	0	0	335677	
36	37	1	3	Mamee, Mr. Hanna	male	NaN	0	0	2677	
42	43	0	3	Kraeff, Mr. Theodor	male	NaN	0	0	349253	
45	46	0	3	Rogers, Mr. William John	male	NaN	0	0	S.C./A.4. 23567	
46	47	0	3	Lennon, Mr. Denis	male	NaN	1	0	370371	
47	48	1	3	O'Driscoll, Miss. Bridget	female	NaN	0	0	14311	
48	49	0	3	Samaan, Mr. Youssef	male	NaN	2	0	2662	
55	56	1	1	Woolner, Mr. Hugh	male	NaN	0	0	19947	
64	65	0	1	Stewart, Mr. Albert A	male	NaN	0	0	PC 17605	
65	66	1	3	Moubarek, Master. Gerios	male	NaN	1	1	2661	
76	77	0	3	Staneff, Mr. Ivan	male	NaN	0	0	349208	

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
77	78	0	3	Moutal, Mr. Rahamin Haim	male	NaN	0	0	374746
82	83	1	3	McDermott, Miss. Brigdet Delia	female	NaN	0	0	330932
87	88	0	3	Slocovski, Mr. Selman Francis	male	NaN	0	0	SOTON/OQ 392086
95	96	0	3	Shorney, Mr. Charles Joseph	male	NaN	0	0	374910
101	102	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215
107	108	1	3	Moss, Mr. Albert Johan	male	NaN	0	0	312991
109	110	1	3	Moran, Miss. Bertha	female	NaN	1	0	371110
121	122	0	3	Moore, Mr. Leonard Charles	male	NaN	0	0	A4. 54510
126	127	0	3	McMahon, Mr. Martin	male	NaN	0	0	370372
128	129	1	3	Peter, Miss. Anna	female	NaN	1	1	2668
140	141	0	3	Boulos, Mrs. Joseph (Sultana)	female	NaN	0	2	2678
154	155	0	3	Olsen, Mr. Ole Martin	male	NaN	0	0	Fa 265302

In [117]:

```
1 np.where(titanic_train["Fare"]==max(titanic_train["Fare"]))
```

Out[117]:

```
(array([27, 88], dtype=int64),)
```

In [ ]:

```
1
```

In [118]:

```
1 len(missing[0])
```

Out[118]:

```
30
```

In [119]:

```
1 titanic_train.head()
```

Out[119]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [121]:

```
1 # combine two columns
2 titanic_train["Family"] = titanic_train["SibSp"] + titanic_train["Parch"]
3 titanic_train["Family"]
4 most_family = np.where(titanic_train["Family"] == max(titanic_train["Family"]))
5 most_family
```

Out[121]:

```
(array([59, 71], dtype=int64),)
```

In [122]:

```
titanic_train["Family"] = titanic_train["SibSp"] + titanic_train["Parch"]

most_family = np.where(titanic_train["Family"] == max(titanic_train["Family"]))
most_family
titanic_train.iloc[most_family]
```

Out[122]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
59	60	0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.9	NaN
71	72	0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.9	NaN

In [ ]:

```
# two question
```

In [129]:

```
# first - find details of person having min fare and min age
titanic_train.iloc[np.where((titanic_train["Age"]==min(titanic_train["Age"])) & (titani
```

Out[129]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark

In [130]:

```
# second - find details of person having min fare and >0 age
titanic_train.iloc[np.where(titanic_train['Age']>0) and np.where(titanic_train['Fare']=
```

Out[130]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
143	144	0	3	Burke, Mr. Jeremiah	male	19.0	0	0	365222	6.75	NaN

In [154]:

```
import numpy as np
import pandas as pd

labels = ['a','b','c']
my_data = [10,20,30]
arr = np.array(my_data)
d = {'a':10,'b':20,'c':30}

print ("Labels:", labels)
print("My data:", my_data)
print("Dictionary:", d)

```

```
Labels: ['a', 'b', 'c']
My data: [10, 20, 30]
Dictionary: {'a': 10, 'b': 20, 'c': 30}
```

In [155]:

```
pd.Series(my_data, index=labels) # create own index as array
```

Out[155]:

```
a    10
b    20
c    30
dtype: int64
```

In [156]:

```
print ("\nHolding numerical data\n",'-'*25, sep='')
print(pd.Series(arr)[1])
# we are creating series of array and accessing first element

```

```
Holding numerical data
-------------------------
20
```

In [157]:

```
print ("\nHolding numerical data\n",'-'*25, sep='')
print(pd.Series(my_data)[1])

```

```
Holding numerical data
-------------------------
20
```

In [158]:

```
print ("\nHolding text labels\n",'-'*20, sep='')
print(pd.Series(labels))
# default index are used, it is either numeric , categorical data
# we also access data by their index name

```

```
Holding text labels
--------------------
0    a
1    b
2    c
dtype: object
```

In [160]:

```
# dictionary in series
# pd.Series(d, index=['sd', 'fgh', 'hhh']) --->  this in not working
pd.Series(d) # so give the original index from dictionary
```

Out[160]:

```
a    10
b    20
c    30
dtype: int64
```

In [ ]:

```

```

In [165]:

```

d = {'a':"kjhk",'b':20,'c':30}
d.items
pd.DataFrame(d,index = ['a','b','c'])
```

Out[165]:

	a	b	c
a	kjhk	20	30
b	kjhk	20	30
c	kjhk	20	30

In [173]:

```
1 pd.DataFrame(d)    # give error so label is necessary
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-173-b671fb4c1ade> in <module>
----> 1 pd.DataFrame(d)    # give error so label is necessary

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\frame.py in __init__(self, data, index, columns, dtype, copy)
    433                 )
    434         elif isinstance(data, dict):
--> 435             mgr = init_dict(data, index, columns, dtype=dtype)
    436         elif isinstance(data, ma.MaskedArray):
    437             import numpy.ma.mrecords as mrecords

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\internals\construction.py in init_dict(data, index, columns, dtype)
    252             arr if not is_datetime64tz_dtype(arr) else arr.copy() fo
r arr in arrays
    253         ]
--> 254     return arrays_to_mgr(arrays, data_names, index, columns, dtype=d
type)
    255
    256

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\internals\construction.py in arrays_to_mgr(arrays, arr_names, index,
columns, dtype)
     62     # figure out the index, if necessary
     63     if index is None:
---> 64         index = extract_index(arrays)
     65     else:
     66         index = ensure_index(index)

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\internals\construction.py in extract_index(data)
    353
    354         if not indexes and not raw_lengths:
--> 355             raise ValueError("If using all scalar values, you must p
ass an index")
    356
    357         if have_series:

ValueError: If using all scalar values, you must pass an index
```

In [177]:

```
print ("\nHolding objects from a dictionary\n",'-'*40, sep='')
pd.Series([type, sum, max])
# some built in function the it identify automatically
```

```
Holding objects from a dictionary
----------------------------------------
```

Out[177]:

```
0             <class 'type'>
1    <built-in function sum>
2    <built-in function max>
dtype: object
```

In [184]:

```
ser1 = pd.Series([1,2,3,4],index = [2,4,6,8])
ser2 = pd.Series([1,2,5,4],['CA', 'OR', 'NV', 'AZ'])    # seconf array taken as index
ser1
```

Out[184]:

```
2    1
4    2
6    3
8    4
dtype: int64
```

In [185]:

```
ser2
```

Out[185]:

```
CA    1
OR    2
NV    5
AZ    4
dtype: int64
```

In [193]:

```
ser1[0:3:2]

```

Out[193]:

```
2    1
6    3
dtype: int64
```

In [194]:

```
ser1[1:3]
```

Out[194]:

```
4    2
6    3
dtype: int64
```

In [204]:

```
print ("\nIndexing by number (positional value in the series)\n",'-'*52, sep='')
print("Value for CA in ser1:", ser2[0])
print("Value for AZ in ser1:", ser2[3])
print("Value for NV in ser2:", ser2[2])
ser2

```

```
Indexing by number (positional value in the series)
----------------------------------------------------
Value for CA in ser1: 1
Value for AZ in ser1: 4
Value for NV in ser2: 5
```

Out[204]:

```
CA    1
OR    2
NV    5
AZ    4
dtype: int64
```

In [205]:

```
print ("\nIndexing by number (positional value in the series)\n",'-'*52, sep='')
print("Value for CA in ser1:", ser1[0])
print("Value for AZ in ser1:", ser1[3])
print("Value for NV in ser2:", ser2[2])
ser1
# it should work, working for categorical index
```

```
Indexing by number (positional value in the series)
----------------------------------------------------

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-205-53bab654f04e> in <module>
      1 print ("\nIndexing by number (positional value in the series)\n",'-'
*52, sep='')
----> 2 print("Value for CA in ser1:", ser1[0])
      3 print("Value for AZ in ser1:", ser1[3])
      4 print("Value for NV in ser2:", ser2[2])
      5 ser1

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\series.py in __getitem__(self, key)
    869         key = com.apply_if_callable(key, self)
    870         try:
--> 871             result = self.index.get_value(self, key)
    872
    873             if not is_scalar(result):

c:\users\jaypr\appdata\local\programs\python\python38\lib\site-packages\pand
as\core\indexes\base.py in get_value(self, series, key)
   4402         k = self._convert_scalar_indexer(k, kind="getitem")
   4403         try:
-> 4404             return self._engine.get_value(s, k, tz=getattr(series.dt
ype, "tz", None))
   4405         except KeyError as e1:
   4406             if len(self) > 0 and (self.holds_integer() or self.is_bo
olean()):

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashT
able.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashT
able.get_item()

KeyError: 0
```

In [206]:

```
print ("\nIndexing by a range\n",'-'*25, sep='')
print ("Value for OR, CO, and AZ in ser1:\n", ser1[0:3:2], sep='')
```

```
Indexing by a range
-------------------------
Value for OR, CO, and AZ in ser1:
2    1
6    3
dtype: int64
```

In [210]:

```
ser1 = pd.Series([1,2,3,4],['CA', 'OR', 'CO', 'CA'])
ser2 = pd.Series([1,2,5,4],['CA', 'NV', 'AZ','OR'])
ser3 = ser1+ser2
ser3
# same index are going to added, index wise summation, taken all combinatiuon of CA
```

Out[210]:

```
AZ    NaN
CA    2.0
CA    5.0
CO    NaN
NV    NaN
OR    6.0
dtype: float64
```

In [211]:

```
print ("\nAfter adding the two series, the result looks like this...\n",'-'*59, sep='')
print(ser3)
print("\nPython tries to add values where it finds common index name, and puts NaN wher

```

```
After adding the two series, the result looks like this...
-----------------------------------------------------------
AZ    NaN
CA    2.0
CA    5.0
CO    NaN
NV    NaN
OR    6.0
dtype: float64

Python tries to add values where it finds common index name, and puts NaN wh
ere indices are missing
```

In [212]:

```
print ("\nThe idea works even for multiplication...\n",'-'*43, sep='')
print (ser1*ser2)

```

```
The idea works even for multiplication...
-------------------------------------------
AZ    NaN
CA    1.0
CA    4.0
CO    NaN
NV    NaN
OR    8.0
dtype: float64
```

In [215]:

```
ser1/ser2
```

Out[215]:

```
AZ    NaN
CA    1.0
CA    4.0
CO    NaN
NV    NaN
OR    0.5
dtype: float64
```

In [228]:

```
from numpy.random import randn as rn
#np.random.seed(101)
matrix_data = rn(5,4)   # 5 rows and 4 column
row_labels = ['A','B','C','D','E']
column_headings = ['W','X','Y','Z']

df = pd.DataFrame(matrix_data,row_labels,column_headings)
#print("\nThe data frame looks like\n",'-'*45, sep='')
df
```

Out[228]:

	W	X	Y	Z
A	0.597418	-1.192192	0.940539	-0.581566
B	1.393623	-0.662949	0.371339	0.541969
C	0.574835	-1.965033	0.102969	2.133068
D	-0.636327	-0.795718	-1.379543	0.223155
E	-0.895104	-1.604224	1.162133	-0.374080

In [229]:

```
1 rn(4,6)   # 4 rows and 6 column
```

Out[229]:

```
array([[-0.26087212, -1.2327159 ,  0.15458115,  0.45000366, -0.11004691,
        -1.09547358],
       [-0.40756229,  0.45717073,  2.50879017,  0.31478148, -1.13898969,
        -0.50459658],
       [-0.7933102 , -1.36319285,  0.68384137, -0.08982813, -0.65246033,
         0.70466569],
       [ 0.25583847, -0.09057124,  1.16802992, -0.48052859,  0.71079378,
        -0.93914932]])
```

In [230]:

```
1 pd.DataFrame(rn(4,3), index=['a','b','c','d'], columns=['w','x','z'])
```

Out[230]:

	w	x	z
a	-0.113053	1.696779	-0.471853
b	1.466606	-0.672540	-1.837277
c	-0.075398	-1.013407	-1.637271
d	1.111014	0.611250	-0.349529

In [232]:

```
1 df
```

Out[232]:

	W	X	Y	Z
A	0.597418	-1.192192	0.940539	-0.581566
B	1.393623	-0.662949	0.371339	0.541969
C	0.574835	-1.965033	0.102969	2.133068
D	-0.636327	-0.795718	-1.379543	0.223155
E	-0.895104	-1.604224	1.162133	-0.374080

In [243]:

```
1 df.iloc[0]   # get 0 row, also iloc take integer data not categorical, take default inc
```

Out[243]:

```
W    0.597418
X   -1.192192
Y    0.940539
Z   -0.581566
Name: A, dtype: float64
```

In [240]:

```
df.iloc[2]
```

Out[240]:

```
W    0.574835
X   -1.965033
Y    0.102969
Z    2.133068
Name: C, dtype: float64
```

In [242]:

```
df.loc['A']   # loc take categorical value mean our given index
```

Out[242]:

```
W    0.597418
X   -1.192192
Y    0.940539
Z   -0.581566
Name: A, dtype: float64
```

In [244]:

```
df
```

Out[244]:

	W	X	Y	Z
A	0.597418	-1.192192	0.940539	-0.581566
B	1.393623	-0.662949	0.371339	0.541969
C	0.574835	-1.965033	0.102969	2.133068
D	-0.636327	-0.795718	-1.379543	0.223155
E	-0.895104	-1.604224	1.162133	-0.374080

In [245]:

```
print("\nType of the pair of columns: ", type(df[['X','Z']]), sep='')
print ("\nSo, for more than one column, the object turns into a DataFrame")
```

```
Type of the pair of columns: <class 'pandas.core.frame.DataFrame'>

So, for more than one column, the object turns into a DataFrame
```

In [247]:

```
print("\nThe 'X' column accessed by DOT method (NOT recommended)\n",'-'*55, sep='')
df["X"]
```

```
The 'X' column accessed by DOT method (NOT recommended)
-------------------------------------------------------
```

Out[247]:

```
A   -1.192192
B   -0.662949
C   -1.965033
D   -0.795718
E   -1.604224
Name: X, dtype: float64
```

In [248]:

```
print("\nA column is created by assigning it in relation to an existing column\n",'-'*7
df['New'] = df['X']+df['Z']
df['New (Sum of X and Z)'] = df['X']+df['Z']
print(df)

```

```
A column is created by assigning it in relation to an existing column
------------------------------------------------------------------------
          W         X         Y         Z       New  New (Sum of X and Z)
A  0.597418 -1.192192  0.940539 -0.581566 -1.773758             -1.773758
B  1.393623 -0.662949  0.371339  0.541969 -0.120980             -0.120980
C  0.574835 -1.965033  0.102969  2.133068  0.168034              0.168034
D -0.636327 -0.795718 -1.379543  0.223155 -0.572563             -0.572563
E -0.895104 -1.604224  1.162133 -0.374080 -1.978304             -1.978304
```

In [249]:

```
df
```

Out[249]:

	W	X	Y	Z	New	New (Sum of X and Z)
A	0.597418	-1.192192	0.940539	-0.581566	-1.773758	-1.773758
B	1.393623	-0.662949	0.371339	0.541969	-0.120980	-0.120980
C	0.574835	-1.965033	0.102969	2.133068	0.168034	0.168034
D	-0.636327	-0.795718	-1.379543	0.223155	-0.572563	-0.572563
E	-0.895104	-1.604224	1.162133	-0.374080	-1.978304	-1.978304

In [250]:

```
print("\nA column is dropped by using df.drop() method\n",'-'*55, sep=' ')
df.drop("C",inplace=True) # Notice the axis=1 option, axis = 0 is default, so one has t
# inplace change into dataset

```

```
A column is dropped by using df.drop() method
 -----------------------------------------------------
```

In [251]:

```
df
```

Out[251]:

	W	X	Y	Z	New	New (Sum of X and Z)
A	0.597418	-1.192192	0.940539	-0.581566	-1.773758	-1.773758
B	1.393623	-0.662949	0.371339	0.541969	-0.120980	-0.120980
D	-0.636327	-0.795718	-1.379543	0.223155	-0.572563	-0.572563
E	-0.895104	-1.604224	1.162133	-0.374080	-1.978304	-1.978304

In [252]:

```
df.drop("Y", axis=1, inplace=True)
```

In [253]:

```
df
```

Out[253]:

	W	X	Z	New	New (Sum of X and Z)
A	0.597418	-1.192192	-0.581566	-1.773758	-1.773758
B	1.393623	-0.662949	0.541969	-0.120980	-0.120980
D	-0.636327	-0.795718	0.223155	-0.572563	-0.572563
E	-0.895104	-1.604224	-0.374080	-1.978304	-1.978304

In [254]:

```
df1=df.drop('A')
print("\nA row (index) is dropped by using df.drop() method and axis=0\n",'-'*65, sep='
print(df1)

```

```
A row (index) is dropped by using df.drop() method and axis=0
-----------------------------------------------------------------
          W         X         Z       New  New (Sum of X and Z)
B  1.393623 -0.662949  0.541969 -0.120980             -0.120980
D -0.636327 -0.795718  0.223155 -0.572563             -0.572563
E -0.895104 -1.604224 -0.374080 -1.978304             -1.978304
```

In [255]:

```
print("\nAn in-place change can be done by making inplace=True in the drop method\n",'
df.drop('New (Sum of X and Z)', axis=1, inplace=True)
print(df)
```

```
An in-place change can be done by making inplace=True in the drop method
--------------------------------------------------------------------------
          W         X         Z       New
A  0.597418 -1.192192 -0.581566 -1.773758
B  1.393623 -0.662949  0.541969 -0.120980
D -0.636327 -0.795718  0.223155 -0.572563
E -0.895104 -1.604224 -0.374080 -1.978304
```

In [256]:

```
### Selecting/indexing Rows
#* Label-based 'loc' method
#* Index (numeric) 'iloc' method
df
```

Out[256]:

	W	X	Z	New
A	0.597418	-1.192192	-0.581566	-1.773758
B	1.393623	-0.662949	0.541969	-0.120980
D	-0.636327	-0.795718	0.223155	-0.572563
E	-0.895104	-1.604224	-0.374080	-1.978304

In [257]:

```
print("\nLabel-based 'loc' method can be used for selecting row(s)\n",'-'*60, sep='')
print("\nSingle row\n")
print(df.iloc[3])

```

```
Label-based 'loc' method can be used for selecting row(s)
------------------------------------------------------------

Single row

W     -0.895104
X     -1.604224
Z     -0.374080
New   -1.978304
Name: E, dtype: float64
```

In [263]:

```
print("\nMultiple rows\n")
df.loc[['B','D']]
```

```
Multiple rows
```

Out[263]:

	W	X	Z	New
B	1.393623	-0.662949	0.541969	-0.120980
D	-0.636327	-0.795718	0.223155	-0.572563

In [260]:

```
print("\nIndex position based 'iloc' method can be used for selecting row(s)\n",'-'*70,
print("\nSingle row\n")
print(df.iloc[2])

```

```
Index position based 'iloc' method can be used for selecting row(s)
----------------------------------------------------------------------

Single row

W     -0.636327
X     -0.795718
Z      0.223155
New   -0.572563
Name: D, dtype: float64
```

In [264]:

```
# intersection
df.loc[['B', 'D']]['Z']
```

Out[264]:

```
B    0.541969
D    0.223155
Name: Z, dtype: float64
```

In [267]:

```
df.loc[['B', 'D'],'Z'] # both approach is right
```

Out[267]:

```
B    0.541969
D    0.223155
Name: Z, dtype: float64
```

In [269]:

```
df
```

Out[269]:

	W	X	Z	New
A	0.597418	-1.192192	-0.581566	-1.773758
B	1.393623	-0.662949	0.541969	-0.120980
D	-0.636327	-0.795718	0.223155	-0.572563
E	-0.895104	-1.604224	-0.374080	-1.978304

In [270]:

```
print("\nMultiple rows\n")
print(df.iloc[[1,2]])
```

```
Multiple rows

          W         X         Z       New
B  1.393623 -0.662949  0.541969 -0.120980
D -0.636327 -0.795718  0.223155 -0.572563
```

In [271]:

```
#### Subsetting DataFrame
```

In [272]:

```


matrix_data = rn(5,4)
row_labels = ['A','B','C','D','E']
column_headings = ['W','X','Y','Z']
df = pd.DataFrame(data=matrix_data, index=row_labels, columns=column_headings)


```

In [273]:

```
print("\nThe DatFrame\n",'-'*45, sep='')
print(df)

```

```
The DatFrame
---------------------------------------------
          W         X         Y         Z
A -0.935137 -0.065725 -0.891921 -0.670637
B  1.485753  0.973594  0.206299 -0.392597
C -0.494930  0.784308  2.937391  0.566050
D -2.632020  0.046309  0.198821 -0.876563
E  0.782996  0.691180  1.679765  1.623690
```

In [274]:

```
print("\nElement at row 'B' and column 'Y' is\n")
print(df.loc[['B','C'],['Y','W']])

```

```
Element at row 'B' and column 'Y' is

          Y         W
B  0.206299  1.485753
C  2.937391 -0.494930
```

In [275]:

```
print("\nSubset comprising of rows B and D, and columns W and Y, is\n")
df.iloc[[1,2,3],[0,1]]
```

```
Subset comprising of rows B and D, and columns W and Y, is
```

Out[275]:

	W	X
B	1.485753	0.973594
C	-0.494930	0.784308
D	-2.632020	0.046309

In [280]:

```
df.loc[['B','C'],]   # taken all columns
```

Out[280]:

	W	X	Y	Z
B	1.485753	0.973594	0.206299	-0.392597
C	-0.494930	0.784308	2.937391	0.566050

In [281]:

```
df.loc[:,['W','X']]    # taken all rows
```

Out[281]:

	W	X
A	-0.935137	-0.065725
B	1.485753	0.973594
C	-0.494930	0.784308
D	-2.632020	0.046309
E	0.782996	0.691180

In [ ]:

```
1
```

In [282]:

```
1 print("\nThe DataFrame\n",'-'*45, sep='')
2 df
3
```

```
The DataFrame
---------------------------------------------
```

Out[282]:

	W	X	Y	Z
A	-0.935137	-0.065725	-0.891921	-0.670637
B	1.485753	0.973594	0.206299	-0.392597
C	-0.494930	0.784308	2.937391	0.566050
D	-2.632020	0.046309	0.198821	-0.876563
E	0.782996	0.691180	1.679765	1.623690

In [283]:

```
1 print("\nBoolean DataFrame(s) where we are checking if the values are greater than 0\n'
2 df>0
3
```

```
Boolean DataFrame(s) where we are checking if the values are greater than 0
--------------------------------------------------------------------------------
```

Out[283]:

	W	X	Y	Z
A	False	False	False	False
B	True	True	True	False
C	False	True	True	True
D	False	True	True	False
E	True	True	True	True

In [289]:

```
print("\n")
df.loc[['A','B','C']]>0
# df[df.loc[['A','B','C']]>0]
```

Out[289]:

	W	X	Y	Z
A	False	False	False	False
B	True	True	True	False
C	False	True	True	True

In [290]:

```
df[df>0]
```

Out[290]:

	W	X	Y	Z
A	NaN	NaN	NaN	NaN
B	1.485753	0.973594	0.206299	NaN
C	NaN	0.784308	2.937391	0.56605
D	NaN	0.046309	0.198821	NaN
E	0.782996	0.691180	1.679765	1.62369

In [291]:

```
booldf = df>0
print("\nDataFrame indexed by boolean dataframe\n",'-'*45, sep='')
df[booldf]
```

```
DataFrame indexed by boolean dataframe
---------------------------------------------
```

Out[291]:

	W	X	Y	Z
A	NaN	NaN	NaN	NaN
B	1.485753	0.973594	0.206299	NaN
C	NaN	0.784308	2.937391	0.56605
D	NaN	0.046309	0.198821	NaN
E	0.782996	0.691180	1.679765	1.62369

In [293]:

```
import pandas as pd
import numpy as np
matrix_data = np.matrix('22,66,140;42,70,148;30,62,125;35,68,160;25,62,152')
row_labels = ['A','B','C','D','E']
column_headings = ['Age', 'Height', 'Weight']
matrix_data
```

Out[293]:

```
matrix([[ 22,  66, 140],
        [ 42,  70, 148],
        [ 30,  62, 125],
        [ 35,  68, 160],
        [ 25,  62, 152]])
```

In [294]:

```
df = pd.DataFrame(data=matrix_data, index=row_labels, columns=column_headings)
print("\nA new DataFrame\n",'-'*25, sep='')
(df)
```

```
A new DataFrame
-------------------------
```

Out[294]:

	Age	Height	Weight
A	22	66	140
B	42	70	148
C	30	62	125
D	35	68	160
E	25	62	152

In [299]:

```
df[df['Height']>65]
```

Out[299]:

	Age	Height	Weight
A	22	66	140
B	42	70	148
D	35	68	160

In [304]:

```
print("\nRows with Height > 65 inch\n",'-'*35, sep='')
df[df['Height']>65]

```

```
Rows with Height > 65 inch
-----------------------------------
```

Out[304]:

	Age	Height	Weight
A	22	66	140
B	42	70	148
D	35	68	160

In [305]:

```
df['Height']>65
```

Out[305]:

```
A     True
B     True
C    False
D     True
E    False
Name: Height, dtype: bool
```

In [306]:

```
booldf1 = df['Height']>65
booldf2 = df['Weight']>145

```

In [307]:

```
print("\nRows with Height > 65 inch and Weight >145 lbs\n",'-'*55, sep='')
print(df[(booldf1) & (booldf2)])

```

```
Rows with Height > 65 inch and Weight >145 lbs
-------------------------------------------------------
   Age  Height  Weight
B   42      70     148
D   35      68     160
```

In [308]:

```
df[booldf1]
```

Out[308]:

	Age	Height	Weight
A	22	66	140
B	42	70	148
D	35	68	160

In [309]:

```
print("\nDataFrame with only Age and Weight columns whose Height > 65 inch\n",'-'*68, s
print(df[booldf1][['Age','Weight']])
```

```
DataFrame with only Age and Weight columns whose Height > 65 inch
--------------------------------------------------------------------
   Age  Weight
A   22     140
B   42     148
D   35     160
```

In [310]:

```
matrix_data = np.matrix('22,66,140;42,70,148;30,62,125;35,68,160;25,62,152')
row_labels = ['A','B','C','D','E']
column_headings = ['Age', 'Height', 'Weight']

```

In [311]:

```
df = pd.DataFrame(data=matrix_data, index=row_labels, columns=column_headings)
print("\nThe DataFrame\n",'-'*25, sep='')
print(df)

```

```
The DataFrame
-------------------------
   Age  Height  Weight
A   22      66     140
B   42      70     148
C   30      62     125
D   35      68     160
E   25      62     152
```

In [315]:

```
print("\nAfter resetting index\n",'-'*35, sep='')
print(df.reset_index())

```

```
After resetting index
-----------------------------------
  index  Age  Height  Weight
0     A   22      66     140
1     B   42      70     148
2     C   30      62     125
3     D   35      68     160
4     E   25      62     152
```

In [317]:

```
print("\nAfter resetting index with 'drop' option TRUE\n",'-'*45, sep='')
print(df.reset_index(drop=True))
# "Student Teacher Engineer Doctor Nurse".split()
```

```
After resetting index with 'drop' option TRUE
---------------------------------------------
   Age  Height  Weight
0   22      66     140
1   42      70     148
2   30      62     125
3   35      68     160
4   25      62     152
```

In [319]:

```
print("\nAdding a new column 'Profession'\n",'-'*45, sep='')
df['Profession'] = "Student Teacher Engineer Doctor Nurse".split()
df

```

```
Adding a new column 'Profession'
---------------------------------------------
```

Out[319]:

	Age	Height	Weight	Profession
A	22	66	140	Student
B	42	70	148	Teacher
C	30	62	125	Engineer
D	35	68	160	Doctor
E	25	62	152	Nurse

In [320]:

```
print("\nSetting 'Profession' column as index\n",'-'*45, sep='')
df.set_index('Profession')
```

```
Setting 'Profession' column as index
--------------------------------------------
```

Out[320]:

Profession	Age	Height	Weight
Student	22	66	140
Teacher	42	70	148
Engineer	30	62	125
Doctor	35	68	160
Nurse	25	62	152

In [325]:

```
#multi-indexing
# Index Levels
outside = ['G1','G1','G1','G2','G2','G2']
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))

```

In [326]:

```
print("\nTuple pairs after the zip and list command\n",'-'*45, sep='')
(hier_index)

```

```
Tuple pairs after the zip and list command
--------------------------------------------
```

Out[326]:

```
[('G1', 1), ('G1', 2), ('G1', 3), ('G2', 1), ('G2', 2), ('G2', 3)]
```

In [327]:

```
hier_index = pd.MultiIndex.from_tuples(hier_index)
print("\nIndex hierarchy\n",'-'*25, sep='')
(hier_index)
```

```
Index hierarchy
-------------------------
```

Out[327]:

```
MultiIndex([('G1', 1),
            ('G1', 2),
            ('G1', 3),
            ('G2', 1),
            ('G2', 2),
            ('G2', 3)],
           )
```

In [330]:

```
print("\nIndex hierarchy type\n",'-'*25, sep='')
print(type(hier_index))
```

```
Index hierarchy type
-------------------------
<class 'pandas.core.indexes.multi.MultiIndex'>
```

In [332]:

```
hier_index[0]  # tuples
```

Out[332]:

```
('G1', 1)
```

In [340]:

```
np.round(rn(6,3))
```

Out[340]:

```
array([[ 0.,  2.,  2.],
       [ 0., -0., -2.],
       [ 1.,  2., -1.],
       [ 1., -0.,  1.],
       [ 0., -1.,  0.],
       [-1.,  1.,  0.]])
```

In [342]:

```
print("\nCreating DataFrame with multi-index\n",'-'*37, sep='')
#np.random.seed(101)
df1 = pd.DataFrame(data=np.round(rn(6,3)), index= hier_index, columns= ['A','B','C'])
(df1)

```

```
Creating DataFrame with multi-index
-----------------------------------
```

Out[342]:

		A	B	C
G1	1	-1.0	-3.0	1.0
	2	-1.0	1.0	0.0
	3	1.0	2.0	-1.0
G2	1	1.0	-0.0	0.0
	2	0.0	1.0	2.0
	3	-0.0	0.0	2.0

In [337]:

```
#cross tabulation like pivot table
```

In [344]:

```
print("\nGrabbing a cross-section from outer level\n",'-'*45, sep='')
print(df1.xs('G1'))
# select only content of G1,

```

```
Grabbing a cross-section from outer level
---------------------------------------------
     A    B    C
1 -1.0 -3.0  1.0
2 -1.0  1.0  0.0
3  1.0  2.0 -1.0
```

In [346]:

```
df1.loc['G1']
```

Out[346]:

	A	B	C
1	-1.0	-3.0	1.0
2	-1.0	1.0	0.0
3	1.0	2.0	-1.0

In [349]:

```
1 df1.loc['G1'].loc[3,['B','C']]   # row inside a row
```

Out[349]:

```
B    2.0
C   -1.0
Name: 3, dtype: float64
```

In [ ]:

```
1
```