In [2]:

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

In [5]:

```
titanic_train = pd.read_csv("E:/Machine Learning/Machine-Learning-Masters-master/Machine-Le
titanic_train.head(20)
```

4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN

In [10]:

```
titanic_train
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
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
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

891 rows × 12 columns

In [17]:

```
titanic_train.tail(10)
```

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.895
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.516
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.500
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.050
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.125
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.450
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.750

In [19]:

```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [13]:

```
sum(titanic_train.isnull().sum())
```

Out[13]:

866

In [6]:

```
titanic_train.describe() #only of numerical data
```

Out[6]:

	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 [17]:

```
titanic_train['Sex'].value_counts()
```

Out[17]:

```
male      577
female    314
Name: Sex, dtype: int64
```

In [6]:

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

Out[6]:

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

In [15]:

```
titanic_train.dtypes == "object"
```

Out[15]:

```
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 [18]:

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

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

In [18]:

```
a.describe()
```

Out[18]:

```
count          5
unique         1
top       object
freq           5
dtype: object
```

In [101]:

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

Out[101]:

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

In [103]:

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

Out[103]:

	Name	Sex	Ticket	Cabin	Embarked
count	156	156	156	31	155
unique	156	2	145	28	3
top	Williams, Mr. Charles Eugene	male	113803	C123	S
freq	1	100	2	2	110

In [163]:

```
#now for categorical column
titanic_train.dtypes == "object"
a = titanic_train.dtypes[titanic_train.dtypes == "object"].index
a
titanic_train[a].describe()
```

Out[163]:

	Name	Sex	Ticket	Cabin	Embarked
count	156	156	156	31	155
unique	156	2	145	28	3
top	Williams, Mr. Charles Eugene	male	113803	C123	S
freq	1	100	2	2	110

In [4]:

```
titanic_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  156 non-null    int64  
 1   Survived     156 non-null    int64  
 2   Pclass       156 non-null    int64  
 3   Name         156 non-null    object 
 4   Sex          156 non-null    object 
 5   Age          126 non-null    float64
 6   SibSp        156 non-null    int64  
 7   Parch        156 non-null    int64  
 8   Ticket       156 non-null    object 
 9   Fare         156 non-null    float64
 10  Cabin        31 non-null     object 
 11  Embarked     155 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 14.8+ KB
```

In [6]:

```
titanic_train.columns
```

Out[6]:

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

In [110]:

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

Out[110]:

	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
5	Moran, Mr. James	male	NaN
6	McCarthy, Mr. Timothy J	male	54.0
7	Palsson, Master. Gosta Leonard	male	2.0
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0
10	Sandstrom, Miss. Marguerite Rut	female	4.0
11	Bonnell, Miss. Elizabeth	female	58.0
12	Saundercock, Mr. William Henry	male	20.0
13	Andersson, Mr. Anders Johan	male	39.0
14	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0
15	Hewlett, Mrs. (Mary D Kingcome)	female	55.0
16	Rice, Master. Eugene	male	2.0
17	Williams, Mr. Charles Eugene	male	NaN
18	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0
19	Masselmani, Mrs. Fatima	female	NaN
20	Fynney, Mr. Joseph J	male	35.0
21	Beesley, Mr. Lawrence	male	34.0
22	McGowan, Miss. Anna "Annie"	female	15.0
23	Sloper, Mr. William Thompson	male	28.0
24	Palsson, Miss. Torborg Danira	female	8.0
25	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0
26	Emir, Mr. Farred Chehab	male	NaN
27	Fortune, Mr. Charles Alexander	male	19.0
28	O'Dwyer, Miss. Ellen "Nellie"	female	NaN
29	Todoroff, Mr. Lalio	male	NaN
...	...	...	...
126	McMahon, Mr. Martin	male	NaN
127	Madsen, Mr. Fridtjof Arne	male	24.0
128	Peter, Miss. Anna	female	NaN
129	Ekstrom, Mr. Johan	male	45.0

	Name	Sex	Age
130	Drazenoic, Mr. Jozef	male	33.0
131	Coelho, Mr. Domingos Fernandeo	male	20.0
132	Robins, Mrs. Alexander A (Grace Charity Laury)	female	47.0
133	Weisz, Mrs. Leopold (Mathilde Francoise Pede)	female	29.0
134	Sobey, Mr. Samuel James Hayden	male	25.0
135	Richard, Mr. Emile	male	23.0
136	Newsom, Miss. Helen Monypeny	female	19.0
137	Futrelle, Mr. Jacques Heath	male	37.0
138	Osen, Mr. Olaf Elon	male	16.0
139	Giglio, Mr. Victor	male	24.0
140	Boulos, Mrs. Joseph (Sultana)	female	NaN
141	Nysten, Miss. Anna Sofia	female	22.0
142	Hakkarainen, Mrs. Pekka Pietari (Elin Matilda ...	female	24.0
143	Burke, Mr. Jeremiah	male	19.0
144	Andrew, Mr. Edgardo Samuel	male	18.0
145	Nicholls, Mr. Joseph Charles	male	19.0
146	Andersson, Mr. August Edvard ("Wennerstrom")	male	27.0
147	Ford, Miss. Robina Maggie "Ruby"	female	9.0
148	Navratil, Mr. Michel ("Louis M Hoffman")	male	36.5
149	Byles, Rev. Thomas Roussel Davids	male	42.0
150	Bateman, Rev. Robert James	male	51.0
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 [166]:

```
sorted(titanic_train["Name"])[5 :10:2]# Check the first 15 sorted names
```

Out[166]:

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

In [167]:

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

Out[167]:

```
count                              156
unique                             156
top       Williams, Mr. Charles Eugene
freq                                 1
Name: Name, dtype: object
```

In [168]:

```
titanic_train[["Ticket"]][4:9]  #operation on df as [['ticket']]
```

Out[168]:

	Ticket
4	373450
5	330877
6	17463
7	349909
8	347742

In [169]:

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

Out[169]:

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

In [171]:

```
titanic_train["sudh"]="sdffs"
titanic_train.head()
```

Out[171]:

	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 [172]:

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

Out[172]:

```
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 [118]:

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

Out[118]:

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

In [ ]:

```

```

In [18]:

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

Out[18]:

```
0      0
1      1
2      1
3      1
4      0
5      0
6      0
7      0
8      1
9      1
10     1
11     1
12     0
13     0
14     0
15     1
16     0
17     1
18     0
19     1
20     0
21     1
22     1
23     1
24     0
25     1
26     0
27     0
28     1
29     0
      ..
126    0
127    1
128    1
129    0
130    0
131    0
132    0
133    1
134    0
135    0
136    1
137    0
138    0
139    0
140    0
141    1
142    1
143    0
144    0
145    0
146    1
147    0
148    0
149    0
```

```
150    0
151    1
152    0
153    0
154    0
155    0
Name: Survived, Length: 156, dtype: int64
```

In [119]:

```
titanic_train.columns
```

Out[119]:

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

In [120]:

```
titanic_train.describe()
```

Out[120]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	1
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 [179]:

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

Out[179]:

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

In [ ]:

```

```

In [193]:

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

new_Pclass
```

Out[193]:

```
[90, dfsf, 90, dfsf, 90, ..., dfsf, 90, 90, 90, dfsf]
Length: 156
Categories (3, object): [dfsf, 9, 90]
```

In [181]:

```
titanic_train["Cabin"].unique()   # Check unique cabins
```

Out[181]:

```
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 [7]:

```
# to create a new column to 1st leter of cabin
import numpy as np
char_cabin = titanic_train["Cabin"].astype(str) # Convert data to str

# for i in char_cabin:
#   new_cabin.append(i[0])

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

new_Cabin = pd.Categorical(new_Cabin)
#new_Cabin
new_Cabin
```

Out[7]:

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

In [8]:

```
titanic_train["Cabin1"] = new_Cabin
titanic_train.head()
```

Out[8]:

	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 [7]:

```
titanic_train["Age"].isnull()  #all index whereage is null
```

Out[7]:

```
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 [8]:

```
titanic_train[titanic_train["Age"].isnull()==True]
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	146.
32	33	1	3	Glynn, Miss. Mary Agatha	female	NaN	0	0	335677	7.
36	37	1	3	Mamee, Mr. Hanna	male	NaN	0	0	2677	7.
42	43	0	3	Kraeff, Mr. Theodor	male	NaN	0	0	349253	7.
45	46	0	3	Rogers, Mr. William John	male	NaN	0	0	S.C./A.4. 23567	8.
46	47	0	3	Lennon, Mr. Denis	male	NaN	1	0	370371	15.
47	48	1	3	O'Driscoll, Miss. Bridget	female	NaN	0	0	14311	7.
48	49	0	3	Samaan, Mr. Youssef	male	NaN	2	0	2662	21.
55	56	1	1	Woolner, Mr. Hugh	male	NaN	0	0	19947	35.
64	65	0	1	Stewart, Mr. Albert A	male	NaN	0	0	PC 17605	27.
65	66	1	3	Moubarek, Master. Gerios	male	NaN	1	1	2661	15.
76	77	0	3	Staneff, Mr. Ivan	male	NaN	0	0	349208	7.
77	78	0	3	Moutal, Mr. Rahamin Haim	male	NaN	0	0	374746	8.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
82	83	1	3	McDermott, Miss. Brigdet Delia	female	NaN	0	0	330932	7.
87	88	0	3	Slocovski, Mr. Selman Francis	male	NaN	0	0	SOTON/OQ 392086	8.
95	96	0	3	Shorney, Mr. Charles Joseph	male	NaN	0	0	374910	8.
101	102	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.
107	108	1	3	Moss, Mr. Albert Johan	male	NaN	0	0	312991	7.
109	110	1	3	Moran, Miss. Bertha	female	NaN	1	0	371110	24.
121	122	0	3	Moore, Mr. Leonard Charles	male	NaN	0	0	A4. 54510	8.
126	127	0	3	McMahon, Mr. Martin	male	NaN	0	0	370372	7.
128	129	1	3	Peter, Miss. Anna	female	NaN	1	1	2668	22.
140	141	0	3	Boulos, Mrs. Joseph (Sultana)	female	NaN	0	2	2678	15.
154	155	0	3	Olsen, Mr. Ole Martin	male	NaN	0	0	Fa 265302	7.

In [ ]:

```
```

In [9]:

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

Out[9]:

```
(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 [19]:

```
max(titanic_train["Fare"])
```

Out[19]:

```
263.0
```

In [20]:

```
titanic_train["Fare"]==max(titanic_train["Fare"])
```

Out[20]:

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

In [ ]:

```

```

In [132]:

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

Out[132]:

```
(array([27, 88]),)
```

In [191]:

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

Out[191]:

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

In [30]:

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

Out[30]:

```
(array([27, 88]),)
```

In [ ]:

```

```

In [31]:

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

Out[31]:

```
30
```

In [194]:

```
titanic_train["Family"] = titanic_train["SibSp"] + titanic_train["Parch"]
titanic_train["Family"]
most_family = np.where(titanic_train["Family"] == max(titanic_train["Family"]))
most_family
```

Out[194]:

```
(array([59, 71]),)
```

In [135]:

```
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[135]:

	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 [196]:

```
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)
pd.Series(my_data, index=labels)
```

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

Out[196]:

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

In [35]:

```
pd.Series(my_data, index=labels)
```

Out[35]:

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

In [199]:

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

```

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

In [198]:

```
print ("\nHolding text labels\n",'-'*20, sep='')
print(pd.Series(labels))
```

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

In [201]:

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

Out[201]:

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

In [203]:

```
print ("\nHolding objects from a dictionary\n",'-'*40, sep='')
print(pd.Series([type, sum, max]))
```

```
Holding objects from a dictionary
----------------------------------------
0             <class 'type'>
1    <built-in function sum>
2    <built-in function max>
dtype: object
```

In [216]:

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

Out[216]:

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

In [217]:

```
ser1[0:3:2]
```

Out[217]:

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

In [42]:

```
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
```

```
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[42]:

```
CA    1
OR    2
CO    3
AZ    4
dtype: int64
```

In [215]:

```
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:
100    1
CO     3
dtype: int64
```

In [3]:

```
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
```

Out[3]:

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

In [45]:

```
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 where in
```

```
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 [46]:

```
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 [4]:

```
from numpy.random import randn as rn
#np.random.seed(101)
matrix_data = rn(5,4)
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[4]:

	W	X	Y	Z
A	-0.964125	1.620393	0.862816	0.877327
B	-1.027721	-0.235968	0.395199	0.105760
C	0.638297	0.191965	-0.910558	-0.262456
D	0.361685	-0.391555	-0.986207	0.231495
E	-0.090619	-1.711412	0.552947	-1.348353

In [5]:

```
df.iloc[2]  #index position
```

Out[5]:

```
W    0.638297
X    0.191965
Y   -0.910558
Z   -0.262456
Name: C, dtype: float64
```

In [8]:

```
df.loc['A']   # label of row
```

Out[8]:

```
W   -0.964125
X    1.620393
Y    0.862816
Z    0.877327
Name: A, dtype: float64
```

In [ ]:

In [6]:

```
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 [7]:

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

```
The 'X' column accessed by DOT method (NOT recommended)
-------------------------------------------------------
A    1.620393
B   -0.235968
C    0.191965
D   -0.391555
E   -1.711412
Name: X, dtype: float64
```

In [9]:

```
print("\nA column is created by assigning it in relation to an existing column\n",'-'*75, s
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 -0.964125  1.620393  0.862816  0.877327  2.497720              2.497720
B -1.027721 -0.235968  0.395199  0.105760 -0.130209             -0.130209
C  0.638297  0.191965 -0.910558 -0.262456 -0.070492             -0.070492
D  0.361685 -0.391555 -0.986207  0.231495 -0.160059             -0.160059
E -0.090619 -1.711412  0.552947 -1.348353 -3.059765             -3.059765
```

In [11]:

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

Out[11]:

```
0.19196476782656316
```

In [14]:

```
df.iloc[1:4 , 2:4]
```

Out[14]:

	Y	Z
B	0.395199	0.105760
C	-0.910558	-0.262456
D	-0.986207	0.231495

In [52]:

```
df
```

Out[52]:

	W	X	Y	Z	New	New (Sum of X and Z)
A	0.150705	-0.446910	1.669238	-1.138071	-1.584981	-1.584981
B	0.462929	0.129871	0.128628	1.865870	1.995741	1.995741
C	0.783021	0.561695	-1.608215	-0.997146	-0.435451	-0.435451
D	1.124639	0.626070	-0.129314	-0.075665	0.550405	0.550405
E	0.133687	1.988570	0.416920	-0.729497	1.259073	1.259073

In [17]:

```
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 to ch
```

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

In [20]:

```
df
```

Out[20]:

	W	X	Y	Z	New	New (Sum of X and Z)
A	-0.964125	1.620393	0.862816	0.877327	2.497720	2.497720
B	-1.027721	-0.235968	0.395199	0.105760	-0.130209	-0.130209
D	0.361685	-0.391555	-0.986207	0.231495	-0.160059	-0.160059
E	-0.090619	-1.711412	0.552947	-1.348353	-3.059765	-3.059765

In [22]:

```
df1=df.drop('X',axis =1)   #1 =coloum , 0 = row
#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         Y         Z       New  New (Sum of X and Z)
A -0.964125  0.862816  0.877327  2.497720              2.497720
B -1.027721  0.395199  0.105760 -0.130209             -0.130209
D  0.361685 -0.986207  0.231495 -0.160059             -0.160059
E -0.090619  0.552947 -1.348353 -3.059765             -3.059765
```

In [56]:

```
print("\nAn in-place change can be done by making inplace=True in the drop method\n",'-'*75
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         Y         Z       New
A  0.150705 -0.446910  1.669238 -1.138071 -1.584981
B  0.462929  0.129871  0.128628  1.865870  1.995741
D  1.124639  0.626070 -0.129314 -0.075665  0.550405
E  0.133687  1.988570  0.416920 -0.729497  1.259073
```

In [11]:

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

Out[11]:

	W	X	Y	Z
A	0.463014	2.351775	0.355781	0.127966
B	0.783331	1.794840	-0.892002	-0.596769
C	0.119252	-0.020044	-2.072349	2.130845
D	0.128286	0.206469	-1.227162	1.602765
E	-0.197037	1.957458	0.238325	1.542864

In [58]:

```
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      0.133687
X      1.988570
Y      0.416920
Z     -0.729497
New    1.259073
Name: E, dtype: float64
```

In [59]:

```
print("\nMultiple rows\n")
print(df.loc[['B','C']])          #index first and last row
```

```
Multiple rows

          W         X         Y        Z       New
B  0.462929  0.129871  0.128628  1.86587  1.995741
C       NaN       NaN       NaN      NaN       NaN

/Users/sudhanshukumar/anaconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:2: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-rei
ndex-listlike (https://pandas.pydata.org/pandas-docs/stable/indexing.html#de
precate-loc-reindex-listlike)
```

In [60]:

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

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

Single row

W      1.124639
X      0.626070
Y     -0.129314
Z     -0.075665
New    0.550405
Name: D, dtype: float64
```

In [61]:

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

```
Multiple rows

          W         X         Y         Z       New
B  0.462929  0.129871  0.128628  1.865870  1.995741
D  1.124639  0.626070 -0.129314 -0.075665  0.550405
```

In [62]:

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

In [63]:

```
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 [64]:

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

```
The DatFrame
--------------------------------------------
          W         X         Y         Z
A -0.320697  1.586737 -0.962361  0.984044
B -0.236481 -0.520044 -0.137304 -1.719868
C -0.415743 -0.280577  1.293790 -0.495861
D -0.564725 -1.331318 -1.098884  0.385147
E  1.079266  1.101438  1.097766  0.452282
```

In [65]:

```
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 -0.137304 -0.236481
C  1.293790 -0.415743
```

In [66]:

```
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[66]:

	W	X
B	-0.236481	-0.520044
C	-0.415743	-0.280577
D	-0.564725	-1.331318

In [67]:

```
print(df.loc[['B','C'],])
```

```
          W         X         Y         Z
B -0.236481 -0.520044 -0.137304 -1.719868
C -0.415743 -0.280577  1.293790 -0.495861
```

In [ ]:

In [68]:

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

```
The DataFrame
---------------------------------------------
          W         X         Y         Z
A -0.320697  1.586737 -0.962361  0.984044
B -0.236481 -0.520044 -0.137304 -1.719868
C -0.415743 -0.280577  1.293790 -0.495861
D -0.564725 -1.331318 -1.098884  0.385147
E  1.079266  1.101438  1.097766  0.452282
```

In [23]:

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

```
Boolean DataFrame(s) where we are checking if the values are greater than 0
-------------------------------------------------------------------------
       W      X      Y      Z    New  New (Sum of X and Z)
A  False   True   True   True   True                 True
B  False  False   True   True  False                False
D   True  False  False   True  False                False
E  False  False   True  False  False                False
```

In [70]:

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

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

In [71]:

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

```
DataFrame indexed by boolean dataframe
---------------------------------------------
          W         X         Y         Z
A       NaN  1.586737       NaN  0.984044
B       NaN       NaN       NaN       NaN
C       NaN       NaN  1.293790       NaN
D       NaN       NaN       NaN  0.385147
E  1.079266  1.101438  1.097766  0.452282
```

In [72]:

```
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[72]:

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

In [73]:

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

```
A new 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 [74]:

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

Out[74]:

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

In [ ]:

In [75]:

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

```
Rows with Height > 65 inch
-----------------------------------
   Age  Height  Weight
A   22      66     140
B   42      70     148
D   35      68     160
```

In [76]:

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

Out[76]:

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

In [77]:

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

In [78]:

```
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 [79]:

```
print("\nDataFrame with only Age and Weight columns whose Height > 65 inch\n",'-'*68, sep='
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 [80]:

```
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 [81]:

```
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 [82]:

```
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 [83]:

```
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
```

Out[83]:

```
['Student', 'Teacher', 'Engineer', 'Doctor', 'Nurse']
```

In [84]:

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

```
Adding a new column 'Profession'
--------------------------------------------
   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 [85]:

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

```
Setting 'Profession' column as index
--------------------------------------------
            Age  Height  Weight
Profession
Student      22      66     140
Teacher      42      70     148
Engineer     30      62     125
Doctor       35      68     160
Nurse        25      62     152
```

In [14]:

```
#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 [15]:

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

```
Tuple pairs after the zip and list command
--------------------------------------------
[('G1', 1), ('G1', 2), ('G1', 3), ('G2', 1), ('G2', 2), ('G2', 3)]
```

In [24]:

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

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

In [17]:

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

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

In [18]:

```
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'])
print(df1)
```

```
Creating DataFrame with multi-index
-------------------------------------
        A    B    C
G1 1 -1.0 -0.0  1.0
   2  2.0  1.0  1.0
   3  1.0 -2.0  0.0
G2 1 -0.0 -0.0 -1.0
   2  1.0 -0.0 -0.0
   3  1.0 -1.0  1.0
```

In [91]:

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

In [92]:

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

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

In [19]:

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

Out[19]:

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

In [22]:

```
df1.loc['G1'].loc[3,['B','C']]
```

Out[22]:

```
B   -2.0
C    0.0
Name: 3, dtype: float64
```

In [26]:

```
#do it in 5
l1 = ['a','a','a','b','b','b','c','c','c']
l2 = [1,2,3,1,2,3,1,2,3]
l3 = [1,2,3,4,5,6,7,8,9]
l = list(zip(l1,l2,l3))
g = pd.MultiIndex.from_tuples(l)
g
```

Out[26]:

```
MultiIndex([('a', 1, 1),
            ('a', 2, 2),
            ('a', 3, 3),
            ('b', 1, 4),
            ('b', 2, 5),
            ('b', 3, 6),
            ('c', 1, 7),
            ('c', 2, 8),
            ('c', 3, 9)],
           )
```

In [27]:

```
df= pd.DataFrame(data=np.round(rn(9,4)),index=g)
df
```

Out[27]:

			0	1	2	3
a	**1**	**1**	-1.0	0.0	-0.0	1.0
	2	**2**	-1.0	3.0	2.0	1.0
	3	**3**	0.0	-0.0	0.0	-0.0
b	**1**	**4**	1.0	-1.0	-2.0	-0.0
	2	**5**	1.0	-0.0	1.0	-1.0
	3	**6**	-1.0	1.0	0.0	1.0
c	**1**	**7**	-0.0	1.0	1.0	-1.0
	2	**8**	1.0	-1.0	-0.0	-0.0
	3	**9**	0.0	-0.0	1.0	0.0

In [ ]:

```

```