In [1]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master
```

In [4]:
```
df.dtypes
```

Out[4]:
```
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 [5]:
```
df.columns
```

Out[5]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [7]:
```
df.describe()
```

Out[7]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [8]:
```
df.dtypes == 'object'
```

Out[8]:
```
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 [9]:
```
df.dtypes[df.dtypes == 'object']
```

Out[9]:
```
Name        object
Sex         object
Ticket      object
Cabin       object
Embarked    object
dtype: object
```

In [11]:
```
df[df.dtypes[df.dtypes == 'object'].index]
```

Out[11]:

	Name	Sex	Ticket	Cabin	Embarked
0	Braund, Mr. Owen Harris	male	A/5 21171	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C85	C
2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	NaN	S
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	C123	S
4	Allen, Mr. William Henry	male	373450	NaN	S
...	...	...	...	...	...
886	Montvila, Rev. Juozas	male	211536	NaN	S
887	Graham, Miss. Margaret Edith	female	112053	B42	S
888	Johnston, Miss. Catherine Helen "Carrie"	female	W./C. 6607	NaN	S
889	Behr, Mr. Karl Howell	male	111369	C148	C
890	Dooley, Mr. Patrick	male	370376	NaN	Q

891 rows × 5 columns

In [12]:
```
df[df.dtypes[df.dtypes == 'object'].index].describe()
```

Out[12]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

In [13]:
```
df.dtypes
```

Out[13]:
```
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 [14]:
```
df.dtypes == 'float'
```

Out[14]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare            True
Cabin          False
Embarked       False
dtype: bool
```

In [15]:
```
df.dtypes == 'int'
```

Out[15]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age            False
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin          False
Embarked       False
dtype: bool
```

In [16]:
```
df.dtypes[0:3]
```

Out[16]:
```
PassengerId    int64
Survived       int64
Pclass         int64
dtype: object
```

In [19]:
```
df[['Survived']][3:10]
```

Out[19]:

	Survived
3	1
4	0
5	0
6	0
7	0
8	1
9	1

In [21]:
```
df[['new_col']] = 0
```

In [22]:
```
df
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
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	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	I
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	N
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N

891 rows × 13 columns

In [25]:
```
df['new_col_1'] = df['PassengerId'] + df['Pclass']
```

In [26]:
```
df
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	I
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	N
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N

891 rows × 14 columns

In [27]:

```
pd.Categorical(df['Pclass'])
```

Out[27]:

```
[3, 1, 3, 1, 3, ..., 2, 1, 3, 1, 3]
Length: 891
Categories (3, int64): [1, 2, 3]
```

In [28]:

```
pd.Categorical(df['Sex'])
```

Out[28]:
```
['male', 'female', 'female', 'female', 'male', ..., 'male', 'female', 'female', 'm
ale', 'male']
Length: 891
Categories (2, object): ['female', 'male']
```

In [31]:
```
df['Embarked'].unique()
```

Out[31]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [33]:
```
df[df['Age']>18]
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
...	...	...	...	...	...	...	...	...	...	...	
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	N
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	E
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N

575 rows × 14 columns

```
In [34]: len(df[df['Age']>18])
```

```
Out[34]: 575
```

```
In [36]: df[df['Fare']==0]
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
179	180	0	3	Leonard, Mr. Lionel	male	36.0	0	0	LINE	0.0	NaN
263	264	0	1	Harrison, Mr. William	male	40.0	0	0	112059	0.0	B94
271	272	1	3	Tornquist, Mr. William Henry	male	25.0	0	0	LINE	0.0	NaN
277	278	0	2	Parkes, Mr. Francis "Frank"	male	NaN	0	0	239853	0.0	NaN
302	303	0	3	Johnson, Mr. William Cahoone Jr	male	19.0	0	0	LINE	0.0	NaN
413	414	0	2	Cunningham, Mr. Alfred Fleming	male	NaN	0	0	239853	0.0	NaN
466	467	0	2	Campbell, Mr. William	male	NaN	0	0	239853	0.0	NaN
481	482	0	2	Frost, Mr. Anthony Wood "Archie"	male	NaN	0	0	239854	0.0	NaN
597	598	0	3	Johnson, Mr. Alfred	male	49.0	0	0	LINE	0.0	NaN
633	634	0	1	Parr, Mr. William Henry Marsh	male	NaN	0	0	112052	0.0	NaN
674	675	0	2	Watson, Mr. Ennis Hastings	male	NaN	0	0	239856	0.0	NaN
732	733	0	2	Knight, Mr. Robert J	male	NaN	0	0	239855	0.0	NaN
806	807	0	1	Andrews, Mr. Thomas Jr	male	39.0	0	0	112050	0.0	A36
815	816	0	1	Fry, Mr. Richard	male	NaN	0	0	112058	0.0	B102
822	823	0	1	Reuchlin, Jonkheer. John George	male	38.0	0	0	19972	0.0	NaN

```
In [39]: df[df['Sex']== 'male']
```

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	N
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	I
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	N
...	...	...	...	...	...	...	...	...	...	...	
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	N
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	N
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N

577 rows × 14 columns

In [40]:
```
df[(df['Sex'] == 'female') & (df['Fare'] > 32)]
```

Out[40]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	146.5208	
43	44	1	2	Laroche, Miss. Simonne Marie Anne Andree	female	3.0	1	2	SC/Paris 2123	41.5792	N
52	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	I
...	...	...	...	...	...	...	...	...	...	...	
853	854	1	1	Lines, Miss. Mary Conover	female	16.0	0	1	PC 17592	39.4000	I
856	857	1	1	Wick, Mrs. George Dennick (Mary Hitchcock)	female	45.0	1	1	36928	164.8667	N
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	N
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	I
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	

104 rows × 14 columns

In [ ]:

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/maste
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [6]:
```
s = df['Name']
```

In [9]:
```
s
```

Out[9]:
```
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
                             ...                        
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

In [7]:
```
type(s)
```

Out[7]:
```
pandas.core.series.Series
```

In [8]:
```
len(s)
```

Out[8]:
```
891
```

In [14]:
```
s = df['Name'][0:10]
```

In [15]:
```
s
```

Out[15]:
```
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
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

In [18]:
```
l = ['a' ,'b' ,'c', 'd' ,'e' , 'f' , 'g' , 'h' , 'i' , 'j']
```

In [20]:
```
s1 = pd.Series(list(s) , index=l)
```

In [21]:
```
s
```

```
Out[21]: 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
         5                                    Moran, Mr. James
         6                             McCarthy, Mr. Timothy J
         7                      Palsson, Master. Gosta Leonard
         8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
         9                  Nasser, Mrs. Nicholas (Adele Achem)
         Name: Name, dtype: object
```

```
In [22]: s1
```

```
Out[22]: a                             Braund, Mr. Owen Harris
         b    Cumings, Mrs. John Bradley (Florence Briggs Th...
         c                              Heikkinen, Miss. Laina
         d         Futrelle, Mrs. Jacques Heath (Lily May Peel)
         e                            Allen, Mr. William Henry
         f                                    Moran, Mr. James
         g                             McCarthy, Mr. Timothy J
         h                      Palsson, Master. Gosta Leonard
         i    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
         j                  Nasser, Mrs. Nicholas (Adele Achem)
         dtype: object
```

```
In [23]: s[0]
```

```
Out[23]: 'Braund, Mr. Owen Harris'
```

```
In [25]: s1[0]
```

```
Out[25]: 'Braund, Mr. Owen Harris'
```

```
In [27]: s1["a"]
```

```
Out[27]: 'Braund, Mr. Owen Harris'
```

```
In [29]: s2 = s1.append(s)
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_5780\2451741888.py:1: FutureWarning:
The series.append method is deprecated and will be removed from pandas in a future
version. Use pandas.concat instead.
  s2 = s1.append(s)
```

```
In [32]: s2
```

```
Out[32]: a                                  Braund, Mr. Owen Harris
         b     Cumings, Mrs. John Bradley (Florence Briggs Th...
         c                                   Heikkinen, Miss. Laina
         d          Futrelle, Mrs. Jacques Heath (Lily May Peel)
         e                                 Allen, Mr. William Henry
         f                                         Moran, Mr. James
         g                                  McCarthy, Mr. Timothy J
         h                           Palsson, Master. Gosta Leonard
         i     Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
         j                      Nasser, Mrs. Nicholas (Adele Achem)
         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
         5                                         Moran, Mr. James
         6                                  McCarthy, Mr. Timothy J
         7                           Palsson, Master. Gosta Leonard
         8     Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
         9                      Nasser, Mrs. Nicholas (Adele Achem)
         dtype: object
```

```
In [30]: s2[0]
```

```
Out[30]: 'Braund, Mr. Owen Harris'
```

```
In [31]: s2[4]
```

```
Out[31]: 'Allen, Mr. William Henry'
```

```
In [34]: s3 = pd.Series([4,5,9,95,95,985],index=[2,4,466514,641651,85484135,1])
```

```
In [35]: s4 = pd.Series([1,4,5,854,6549,1564,98],index=[15,548,4984,65655,5,4,1])
```

```
In [36]: s4
```

```
Out[36]: 15          1
         548         4
         4984        5
         65655     854
         5        6549
         4        1564
         1          98
         dtype: int64
```

```
In [38]: s6 = s4.append(s3)
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_5780\3526614916.py:1: FutureWarning:
The series.append method is deprecated and will be removed from pandas in a future
version. Use pandas.concat instead.
  s6 = s4.append(s3)
```

```
In [48]: s6
```

```
Out[48]: 15                1
         548               4
         4984              5
         65655           854
         5              6549
         4              1564
         1                98
         2                 4
         4                 5
         466514            9
         641651           95
         85484135         95
         1               985
         dtype: int64
```

```
In [44]: s6[1]
```

```
Out[44]: 1     98
         1    985
         dtype: int64
```

```
In [46]: s3
```

```
Out[46]: 2              4
         4              5
         466514         9
         641651        95
         85484135      95
         1            985
         dtype: int64
```

```
In [47]: s4
```

```
Out[47]: 15          1
         548         4
         4984        5
         65655     854
         5        6549
         4        1564
         1          98
         dtype: int64
```

```
In [45]: s3*s4
```

```
Out[45]: 1           96530.0
         2               NaN
         4            7820.0
         5               NaN
         15              NaN
         548             NaN
         4984            NaN
         65655           NaN
         466514          NaN
         641651          NaN
         85484135        NaN
         dtype: float64
```

```
In [ ]:
```

In [2]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/maste
```

In [4]:
```
df
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	N
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N

891 rows × 12 columns

```
In [5]: df.columns
```

```
Out[5]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
               'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
              dtype='object')
```

```
In [6]: df.drop("PassengerId" , axis=1 , inplace=True)
```

```
In [7]: df
```

Out[7]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 11 columns

In [8]:
```
df.drop(3)
```

Out[8]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
5	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

890 rows × 11 columns

In [9]:
```
df.drop(3,inplace=True)
```

In [10]:
```
df
```

Out[10]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
5	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

890 rows × 11 columns

In [11]:
```
df.set_index("Name",inplace=True)
```

In [12]:
```
df
```

Out[12]:

Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q
...	...	...	...	...	...	...	...	...	...	...
Montvila, Rev. Juozas	0	2	male	27.0	0	0	211536	13.0000	NaN	S
Graham, Miss. Margaret Edith	1	1	female	19.0	0	0	112053	30.0000	B42	S
Johnston, Miss. Catherine Helen "Carrie"	0	3	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
Behr, Mr. Karl Howell	1	1	male	26.0	0	0	111369	30.0000	C148	C
Dooley, Mr. Patrick	0	3	male	32.0	0	0	370376	7.7500	NaN	Q

890 rows × 10 columns

In [13]:
```
df.reset_index(inplace=True)
```

In [14]:
```
df
```

Out[14]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	
...	...	...	...	...	...	...	...	...	...	...	
885	Montvila, Rev. Juozas	0	2	male	27.0	0	0	211536	13.0000	NaN	
886	Graham, Miss. Margaret Edith	1	1	female	19.0	0	0	112053	30.0000	B42	
887	Johnston, Miss. Catherine Helen "Carrie"	0	3	female	NaN	1	2	W./C. 6607	23.4500	NaN	
888	Behr, Mr. Karl Howell	1	1	male	26.0	0	0	111369	30.0000	C148	
889	Dooley, Mr. Patrick	0	3	male	32.0	0	0	370376	7.7500	NaN	

890 rows × 11 columns

In [15]:
```python
d = {'key1': [1,2,3,5,45],
     'key2': [5,6,8,65,55],
     'key3': [5,6,8,5,55]


}
```

In [16]:
```python
pd.DataFrame(d)
```

Out[16]:

	key1	key2	key3
0	1	5	5
1	2	6	6
2	3	8	8
3	5	65	5
4	45	55	55

In [17]:
```
df1 = pd.read_csv('taxonomy.csv.xls')
```

In [18]:
```
df1
```

Out[18]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	NaN	NaN
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

290 rows × 4 columns

In [19]:
```
df1.dropna(inplace=True)
```

In [20]:
```
df1
```

Out[20]:

	taxonomy_id	name	parent_id	parent_name
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
5	101-02-03	Help Pay for Housing	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

279 rows × 4 columns

In [21]:
```
df1.dropna(axis=1)
```

Out[21]:

	taxonomy_id	name	parent_id	parent_name
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
5	101-02-03	Help Pay for Housing	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

279 rows × 4 columns

In [22]:
```
df2 = pd.read_csv('taxonomy.csv.xls')
```

In [23]:
```
df2
```

Out[23]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	NaN	NaN
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

290 rows × 4 columns

In [24]:
```
df2.dropna(axis=1)
```

Out[24]:

	taxonomy_id	name
0	101	Emergency
1	101-01	Disaster Response
2	101-02	Emergency Cash
3	101-02-01	Help Pay for Food
4	101-02-02	Help Pay for Healthcare
...	...	...
285	111-01-07	Workplace Rights
286	111-02	Mediation
287	111-03	Notary
288	111-04	Representation
289	111-05	Translation & Interpretation

290 rows × 2 columns

In [25]:
```
df2.fillna("abhishek")
```

Out[25]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	abhishek	abhishek
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

290 rows × 4 columns

In [26]:
```
df
```

Out[26]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	
...	...	...	...	...	...	...	...	...	...	...	
885	Montvila, Rev. Juozas	0	2	male	27.0	0	0	211536	13.0000	NaN	
886	Graham, Miss. Margaret Edith	1	1	female	19.0	0	0	112053	30.0000	B42	
887	Johnston, Miss. Catherine Helen "Carrie"	0	3	female	NaN	1	2	W./C. 6607	23.4500	NaN	
888	Behr, Mr. Karl Howell	1	1	male	26.0	0	0	111369	30.0000	C148	
889	Dooley, Mr. Patrick	0	3	male	32.0	0	0	370376	7.7500	NaN	

890 rows × 11 columns

In [27]:
```
g = df.groupby('Survived')
```

In [28]:
```
g
```

Out[28]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000012E3AEA3730>
```

In [29]:
```
g.sum()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\1197020669.py:1: FutureWarnin
g: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a f
uture version, numeric_only will default to False. Either specify numeric_only or
select only columns which should be valid for the function.
  g.sum()
```

Out[29]:

Survived	Pclass	Age	SibSp	Parch	Fare
0	1390	12985.50	304	181	12142.7199
1	666	8184.67	161	159	16498.1294

In [30]:
```
g.mean()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\2978112660.py:1: FutureWarnin
g: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only or
select only columns which should be valid for the function.
  g.mean()
```

Out[30]:

Survived	Pclass	Age	SibSp	Parch	Fare
0	2.531876	30.626179	0.553734	0.329690	22.117887
1	1.953079	28.320657	0.472141	0.466276	48.381611

In [31]:
```
p = df.groupby('Pclass')
```

In [32]:
```
p
```

Out[32]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000012E3AEA2A70>
```

In [33]:
```
p.sum()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\943096461.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sel
ect only columns which should be valid for the function.
  p.sum()
```

Out[33]:

Pclass	Survived	Age	SibSp	Parch	Fare
1	135	7076.42	89	77	18124.3125
2	87	5168.83	74	70	3801.8417
3	119	8924.92	302	193	6714.6951

In [34]:
```
p.mean()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\288629575.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a fut
ure version, numeric_only will default to False. Either specify numeric_only or se
lect only columns which should be valid for the function.
  p.mean()
```

Out[34]:

	Survived	Age	SibSp	Parch	Fare
Pclass					
1	0.627907	38.250919	0.413953	0.358140	84.299128
2	0.472826	29.877630	0.402174	0.380435	20.662183
3	0.242363	25.140620	0.615071	0.393075	13.675550

In [35]:
```
p.median()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\2867037694.py:1: FutureWarnin
g: The default value of numeric_only in DataFrameGroupBy.median is deprecated. In
a future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  p.median()
```

Out[35]:

	Survived	Age	SibSp	Parch	Fare
Pclass					
1	1.0	37.0	0.0	0.0	61.175
2	0.0	29.0	0.0	0.0	14.250
3	0.0	24.0	0.0	0.0	8.050

In [36]:
```
p.max()['Fare']
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\287658238.py:1: FutureWarning:
Dropping invalid columns in DataFrameGroupBy.max is deprecated. In a future versio
n, a TypeError will be raised. Before calling .max, select only columns which shou
ld be valid for the function.
  p.max()['Fare']
```

Out[36]:
```
Pclass
1    512.3292
2     73.5000
3     69.5500
Name: Fare, dtype: float64
```

In [37]:
```
p.max()
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\2743088561.py:1: FutureWarnin
g: Dropping invalid columns in DataFrameGroupBy.max is deprecated. In a future ver
sion, a TypeError will be raised. Before calling .max, select only columns which s
hould be valid for the function.
  p.max()
```

Out[37]:

	Name	Survived	Sex	Age	SibSp	Parch	Ticket	Fare
Pclass								
1	Young, Miss. Marie Grice	1	male	80.0	3	4	WE/P 5735	512.3292
2	del Carlo, Mr. Sebastiano	1	male	70.0	3	3	W/C 14208	73.5000
3	van Melkebeke, Mr. Philemon	1	male	74.0	8	6	W./C. 6609	69.5500

In [38]:
```
p.max().T
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\1665691157.py:1: FutureWarnin
g: Dropping invalid columns in DataFrameGroupBy.max is deprecated. In a future ver
sion, a TypeError will be raised. Before calling .max, select only columns which s
hould be valid for the function.
  p.max().T
```

Out[38]:

Pclass	1	2	3
Name	Young, Miss. Marie Grice	del Carlo, Mr. Sebastiano	van Melkebeke, Mr. Philemon
Survived	1	1	1
Sex	male	male	male
Age	80.0	70.0	74.0
SibSp	3	3	8
Parch	4	3	6
Ticket	WE/P 5735	W/C 14208	W./C. 6609
Fare	512.3292	73.5	69.55

In [39]:
```
p.max().transpose
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_24028\4065314366.py:1: FutureWarnin
g: Dropping invalid columns in DataFrameGroupBy.max is deprecated. In a future ver
sion, a TypeError will be raised. Before calling .max, select only columns which s
hould be valid for the function.
  p.max().transpose
```

Out[39]:
```
<bound method DataFrame.transpose of                                 Name  Survived
Sex   Age  SibSp  Parch  \
Pclass
1          Young, Miss. Marie Grice         1  male  80.0      3      4
2         del Carlo, Mr. Sebastiano         1  male  70.0      3      3
3       van Melkebeke, Mr. Philemon         1  male  74.0      8      6

            Ticket      Fare
Pclass
1        WE/P 5735  512.3292
2        W/C 14208   73.5000
3       W./C. 6609   69.5500  >
```

In [40]:
```
df
```

Out[40]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	
...	...	...	...	...	...	...	...	...	...	...	
885	Montvila, Rev. Juozas	0	2	male	27.0	0	0	211536	13.0000	NaN	
886	Graham, Miss. Margaret Edith	1	1	female	19.0	0	0	112053	30.0000	B42	
887	Johnston, Miss. Catherine Helen "Carrie"	0	3	female	NaN	1	2	W./C. 6607	23.4500	NaN	
888	Behr, Mr. Karl Howell	1	1	male	26.0	0	0	111369	30.0000	C148	
889	Dooley, Mr. Patrick	0	3	male	32.0	0	0	370376	7.7500	NaN	

890 rows × 11 columns

In [41]:
```
df5 = df[['Survived' ,'Pclass' ,'Sex']][0:5]
```

In [42]:
```
df6 = df[['Survived' ,'Pclass' ,'Sex']][5:10]
```

In [43]:
```
df5
```

Out[43]:

	Survived	Pclass	Sex
0	0	3	male
1	1	1	female
2	1	3	female
3	0	3	male
4	0	3	male

In [44]:
```
df6
```

Out[44]:

	Survived	Pclass	Sex
5	0	1	male
6	0	3	male
7	1	3	female
8	1	2	female
9	1	3	female

In [45]:
```
pd.concat([df5,df6])
```

Out[45]:

	Survived	Pclass	Sex
0	0	3	male
1	1	1	female
2	1	3	female
3	0	3	male
4	0	3	male
5	0	1	male
6	0	3	male
7	1	3	female
8	1	2	female
9	1	3	female

In [46]:
```
df7 = pd.concat([df5,df6],axis=1)
```

In [47]:
```
df7.fillna("abhi")
```

Out[47]:

	Survived	Pclass	Sex	Survived	Pclass	Sex
0	0.0	3.0	male	abhi	abhi	abhi
1	1.0	1.0	female	abhi	abhi	abhi
2	1.0	3.0	female	abhi	abhi	abhi
3	0.0	3.0	male	abhi	abhi	abhi
4	0.0	3.0	male	abhi	abhi	abhi
5	abhi	abhi	abhi	0.0	1.0	male
6	abhi	abhi	abhi	0.0	3.0	male
7	abhi	abhi	abhi	1.0	3.0	female
8	abhi	abhi	abhi	1.0	2.0	female
9	abhi	abhi	abhi	1.0	3.0	female

In [48]:
```
data1 = pd.DataFrame({'key1': [1,2,4,5,6],
                      'key2': [4,5,6,7,8],
                      'key3': [3,4,5,6,6]
}
)
```

In [49]:
```
data1
```

Out[49]:

	key1	key2	key3
0	1	4	3
1	2	5	4
2	4	6	5
3	5	7	6
4	6	8	6

In [50]:
```
data2 = pd.DataFrame({'key1': [1,2,45,6,67],
                      'key4': [56,5,6,7,8],
                      'key5': [3,56,5,6,6]
}
)
```

In [51]:
```
data2
```

Out[51]:

	key1	key4	key5
0	1	56	3
1	2	5	56
2	45	6	5
3	6	7	6
4	67	8	6

In [52]:
```
pd.merge(data1,data2)
```

Out[52]:

	key1	key2	key3	key4	key5
0	1	4	3	56	3
1	2	5	4	5	56
2	6	8	6	7	6

In [53]:
```
pd.merge(data1,data2 , how='left')
```

Out[53]:

	key1	key2	key3	key4	key5
0	1	4	3	56.0	3.0
1	2	5	4	5.0	56.0
2	4	6	5	NaN	NaN
3	5	7	6	NaN	NaN
4	6	8	6	7.0	6.0

In [54]:
```
pd.merge(data1,data2 , how='right')
```

Out[54]:

	key1	key2	key3	key4	key5
0	1	4.0	3.0	56	3
1	2	5.0	4.0	5	56
2	45	NaN	NaN	6	5
3	6	8.0	6.0	7	6
4	67	NaN	NaN	8	6

In [55]:
```
pd.merge(data1,data2 , how='outer')
```

Out[55]:

	key1	key2	key3	key4	key5
0	1	4.0	3.0	56.0	3.0
1	2	5.0	4.0	5.0	56.0
2	4	6.0	5.0	NaN	NaN
3	5	7.0	6.0	NaN	NaN
4	6	8.0	6.0	7.0	6.0
5	45	NaN	NaN	6.0	5.0
6	67	NaN	NaN	8.0	6.0

In [56]:
```
pd.merge(data1,data2 , how='cross')
```

Out[56]:

	key1_x	key2	key3	key1_y	key4	key5
0	1	4	3	1	56	3
1	1	4	3	2	5	56
2	1	4	3	45	6	5
3	1	4	3	6	7	6
4	1	4	3	67	8	6
5	2	5	4	1	56	3
6	2	5	4	2	5	56
7	2	5	4	45	6	5
8	2	5	4	6	7	6
9	2	5	4	67	8	6
10	4	6	5	1	56	3
11	4	6	5	2	5	56
12	4	6	5	45	6	5
13	4	6	5	6	7	6
14	4	6	5	67	8	6
15	5	7	6	1	56	3
16	5	7	6	2	5	56
17	5	7	6	45	6	5
18	5	7	6	6	7	6
19	5	7	6	67	8	6
20	6	8	6	1	56	3
21	6	8	6	2	5	56
22	6	8	6	45	6	5
23	6	8	6	6	7	6
24	6	8	6	67	8	6

In [57]:
```
pd.concat([data1,data2],axis=1)
```

Out[57]:

	key1	key2	key3	key1	key4	key5
0	1	4	3	1	56	3
1	2	5	4	2	5	56
2	4	6	5	45	6	5
3	5	7	6	6	7	6
4	6	8	6	67	8	6

In [58]:
```
pd.merge(data1,data2 , how='outer',on='key1')
```

Out[58]:

	key1	key2	key3	key4	key5
0	1	4.0	3.0	56.0	3.0
1	2	5.0	4.0	5.0	56.0
2	4	6.0	5.0	NaN	NaN
3	5	7.0	6.0	NaN	NaN
4	6	8.0	6.0	7.0	6.0
5	45	NaN	NaN	6.0	5.0
6	67	NaN	NaN	8.0	6.0

In [59]:
```
data1 = pd.DataFrame({'key1': [1,2,4,5,6],
                      'key2': [4,5,6,7,8],
                      'key3': [3,4,5,6,6]},
                      index = ['a','b','c','d','e']
)
```

In [60]:
```
data2 = pd.DataFrame({'key11': [1,2,45,6,67],
                      'key22' :[6,5,6,7,8],
                      'key33':[56,5,6,6,8]},
                     index = ['a','b','h','j','k']
)
```

In [61]:
```
data1
```

Out[61]:

	key1	key2	key3
a	1	4	3
b	2	5	4
c	4	6	5
d	5	7	6
e	6	8	6

In [62]:
```
data2
```

Out[62]:

	key11	key22	key33
a	1	6	56
b	2	5	5
h	45	6	6
j	6	7	6
k	67	8	8

In [68]:
```
data1.join(data2).fillna('abhi')
```

Out[68]:

	key1	key2	key3	key11	key22	key33
a	1	4	3	1.0	6.0	56.0
b	2	5	4	2.0	5.0	5.0
c	4	6	5	abhi	abhi	abhi
d	5	7	6	abhi	abhi	abhi
e	6	8	6	abhi	abhi	abhi

In [64]:
```
data1.join(data2 , how='right')
```

Out[64]:

	key1	key2	key3	key11	key22	key33
a	1.0	4.0	3.0	1	6	56
b	2.0	5.0	4.0	2	5	5
h	NaN	NaN	NaN	45	6	6
j	NaN	NaN	NaN	6	7	6
k	NaN	NaN	NaN	67	8	8

In [65]:
```
data1.join(data2 , how='inner')
```

Out[65]:

	key1	key2	key3	key11	key22	key33
a	1	4	3	1	6	56
b	2	5	4	2	5	5

In [66]:
```
data1.join(data2 , how='outer')
```

Out[66]:

	key1	key2	key3	key11	key22	key33
a	1.0	4.0	3.0	1.0	6.0	56.0
b	2.0	5.0	4.0	2.0	5.0	5.0
c	4.0	6.0	5.0	NaN	NaN	NaN
d	5.0	7.0	6.0	NaN	NaN	NaN
e	6.0	8.0	6.0	NaN	NaN	NaN
h	NaN	NaN	NaN	45.0	6.0	6.0
j	NaN	NaN	NaN	6.0	7.0	6.0
k	NaN	NaN	NaN	67.0	8.0	8.0

In [71]:
```
df['Fare_INR'] = df['Fare'].apply(lambda x : x*80)
```

In [73]:
```
df.head()
```

Out[73]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [75]:
```
def euro_inr(x):
    return x*80
df['Fare_INR'] = df['Fare'].apply(euro_inr)
```

In [79]:
```
df['Fare_INR']
```

Out[79]:
```
0       580.000
1      5702.664
2       634.000
3       644.000
4       676.664
         ...   
885    1040.000
886    2400.000
887    1876.000
888    2400.000
889     620.000
Name: Fare_INR, Length: 890, dtype: float64
```

In [80]:
```
df['name_length'] = df['Name'].apply(len)
```

In [81]:
```
df.head()
```

Out[81]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [82]:
```
def cat_fare(x):
    if x<10:
        return "cheap"
    elif x>=10 and x<20:
        return 'mid'
    else:
        return 'high'
```

In [83]:
```
df['cat_fare'] = df['Fare'].apply(cat_fare)
```

In [84]:
```
df['cat_fare']
```

Out[84]:
```
0      cheap
1       high
2      cheap
3      cheap
4      cheap
       ...
885      mid
886     high
887     high
888     high
889    cheap
Name: cat_fare, Length: 890, dtype: object
```

In [86]:
```
df.head()
```

Out[86]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [ ]:

In [1]:
```
import pandas as pd
```

In [2]:
```
data = {"a":[1,2,5,6,6],
        "b":[2,3,5,6,5],
        "c":["Abhi","Amit","Aniket","Mayank","Ravi"]
        }
```

In [3]:
```
df = pd.DataFrame(data)
```

In [4]:
```
df
```

Out[4]:

	a	b	c
0	1	2	Abhi
1	2	3	Amit
2	5	5	Aniket
3	6	6	Mayank
4	6	5	Ravi

In [5]:
```
df.set_index("a" , inplace=True)
```

In [6]:
```
df
```

Out[6]:

	b	c
a		
1	2	Abhi
2	3	Amit
5	5	Aniket
6	6	Mayank
6	5	Ravi

In [7]:
```
df = df.reset_index()
```

In [8]:
```
df
```

Out[8]:

	a	b	c
0	1	2	Abhi
1	2	3	Amit
2	5	5	Aniket
3	6	6	Mayank
4	6	5	Ravi

In [9]:
```
data = {"a":[1,2,5,6,6],
        "b":[2,3,5,6,5],
        "c":["Abhi","Amit","Aniket","Mayank","Ravi"]
        }
df1 = pd.DataFrame(data ,index=["a","b","c","d","e"])
```

In [10]:
```
df1
```

Out[10]:

	a	b	c
a	1	2	Abhi
b	2	3	Amit
c	5	5	Aniket
d	6	6	Mayank
e	6	5	Ravi

In [11]:
```
df1.reindex(["b","d","a","c"])
```

Out[11]:

	a	b	c
b	2	3	Amit
d	6	6	Mayank
a	1	2	Abhi
c	5	5	Aniket

In [12]:
```
df1
```

Out[12]:

	a	b	c
a	1	2	Abhi
b	2	3	Amit
c	5	5	Aniket
d	6	6	Mayank
e	6	5	Ravi

In [13]:
```
for i in df1.iterrows():
    print(i)
```

```
('a', a       1
b       2
c    Abhi
Name: a, dtype: object)
('b', a       2
b       3
c    Amit
Name: b, dtype: object)
('c', a         5
b         5
c    Aniket
Name: c, dtype: object)
('d', a         6
b         6
c    Mayank
Name: d, dtype: object)
('e', a       6
b       5
c    Ravi
Name: e, dtype: object)
```

In [14]:
```
for i in df1.iteritems():
    print(i)
```

```
('a', a    1
b    2
c    5
d    6
e    6
Name: a, dtype: int64)
('b', a    2
b    3
c    5
d    6
e    5
Name: b, dtype: int64)
('c', a      Abhi
b      Amit
c    Aniket
d    Mayank
e      Ravi
Name: c, dtype: object)
```

```
C:\Users\Mr Abhi\AppData\Local\Temp\ipykernel_20552\2757294222.py:1: FutureWarnin
g: iteritems is deprecated and will be removed in a future version. Use .items ins
tead.
  for i in df1.iteritems():
```

In [15]:
```
list(df['a'])
```

Out[15]:
```
[1, 2, 5, 6, 6]
```

In [16]:
```
[i for i in df['a']]
```

Out[16]:
```
[1, 2, 5, 6, 6]
```

In [17]:
```
df1
```

Out[17]:

	a	b	c
a	1	2	Abhi
b	2	3	Amit
c	5	5	Aniket
d	6	6	Mayank
e	6	5	Ravi

In [18]:
```
def test(x):
    return x.sum()
df1.apply(test)
```

Out[18]:
```
a                       20
b                       21
c     AbhiAmitAniketMayankRavi
dtype: object
```

In [19]:
```
df2 = df1[['a','b']]
```

In [20]:
```
df2
```

Out[20]:

	a	b
a	1	2
b	2	3
c	5	5
d	6	6
e	6	5

In [21]:
```
df2.applymap(lambda x : x**2)
```

Out[21]:

	a	b
a	1	4
b	4	9
c	25	25
d	36	36
e	36	25

In [22]:
```
df
```

Out[22]:

	a	b	c
0	1	2	Abhi
1	2	3	Amit
2	5	5	Aniket
3	6	6	Mayank
4	6	5	Ravi

```
In [23]: df.sort_values('c' )
```

```
Out[23]:
   a  b       c
0  1  2    Abhi
1  2  3    Amit
2  5  5  Aniket
3  6  6  Mayank
4  6  5    Ravi
```

```
In [24]: df.sort_values('c' , ascending= False)
```

```
Out[24]:
   a  b       c
4  6  5    Ravi
3  6  6  Mayank
2  5  5  Aniket
1  2  3    Amit
0  1  2    Abhi
```

```
In [25]: df.sort_index(ascending=False)
```

```
Out[25]:
   a  b       c
4  6  5    Ravi
3  6  6  Mayank
2  5  5  Aniket
1  2  3    Amit
0  1  2    Abhi
```

```
In [34]: pd.set_option("display.max_colwidth" , 1000)
         df4 = pd.DataFrame({"Desc" : ["Data Science Masters course is highly curated and u
```

```
In [35]: df4
```

```
Out[35]:
                                                                                                                       Desc
0  Data Science Masters course is highly curated and uniquely designed according to the latest industry
   standards. This program instills students the skills essential to knowledge discovery efforts to identify
   standard, novel, and truly differentiated solutions and decision-making, including skills in managing,
   querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending
   program provides students with the statistical, mathematical and computational skills needed to meet
   the large-scale data science challenges of today's professional world. You will learn all the stack
   required to work in data science industry including cloud infrastructure and real-time industry projects.
   This course will be taught in Hindi language.
```

```
In [36]: pd.set_option("display.max_colwidth" , 1000)
         df4 = pd.DataFrame({"Desc" : ["Data Science Masters course is highly curated and u
```

```
In [37]: df4
```

Out[37]:

	Desc
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.
1	My name is abhishek mishra
2	I am studying in Pwskills

In [38]:
```
df4['len'] = df4['Desc'].apply(len)
```

In [39]:
```
df4
```

Out[39]:

	Desc	len
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.	765
1	My name is abhishek mishra	27
2	I am studying in Pwskills	25

In [42]:
```
t = "My name is abhishek mishra"
t.split()
```

Out[42]:
```
['My', 'name', 'is', 'abhishek', 'mishra']
```

In [43]:
```
t = "My name is abhishek mishra"
len(t.split())
```

Out[43]:
```
5
```

In [40]:
```
df4["word_count"] = df4['Desc'].apply(lambda x :len(x.split()))
```

In [44]:
```
df4
```

Out[44]:

	Desc	len	word_count
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.	765	104
1	My name is abhishek mishra	27	5
2	I am studying in Pwskills	25	5

In [45]:
```
df
```

Out[45]:

	a	b	c
0	1	2	Abhi
1	2	3	Amit
2	5	5	Aniket
3	6	6	Mayank
4	6	5	Ravi

In [47]:
```
df['a'].mean()
```

Out[47]:
```
4.0
```

In [48]:
```
df['a'].median()
```

Out[48]:
```
5.0
```

In [49]:
```
df['a'].mode()
```

Out[49]:
```
0    6
Name: a, dtype: int64
```

In [50]:
```
df['a'].std()
```

Out[50]:
```
2.345207879911715
```

In [51]:
```
df['a'].var()
```

Out[51]:
```
5.5
```

In [53]:
```
df5 = pd.DataFrame({'a' : [1,2,3,6,5,98,5,5]})
```

In [55]:
```
df5
```

Out[55]:

	a
0	1
1	2
2	3
3	6
4	5
5	98
6	5
7	5

In [64]:
```
# what is windowing function in python pandas
df5['a'].rolling(window=1).mean()
```

Out[64]:
```
0     1.0
1     2.0
2     3.0
3     6.0
4     5.0
5    98.0
6     5.0
7     5.0
Name: a, dtype: float64
```

In [65]:
```
df5['a'].rolling(window=2).mean()
```

Out[65]:
```
0     NaN
1     1.5
2     2.5
3     4.5
4     5.5
5    51.5
6    51.5
7     5.0
Name: a, dtype: float64
```

In [66]:
```
df5['a'].rolling(window=3).mean()
```

Out[66]:
```
0          NaN
1          NaN
2     2.000000
3     3.666667
4     4.666667
5    36.333333
6    36.000000
7    36.000000
Name: a, dtype: float64
```

In [67]:
```
df5['a'].rolling(window=3).sum()
```

Out[67]:
```
0      NaN
1      NaN
2      6.0
3     11.0
4     14.0
5    109.0
6    108.0
7    108.0
Name: a, dtype: float64
```

In [68]:
```
df5['a'].rolling(window=3).max()
```

Out[68]:
```
0     NaN
1     NaN
2     3.0
3     6.0
4     6.0
5    98.0
6    98.0
7    98.0
Name: a, dtype: float64
```

In [69]:
```
df5['a'].rolling(window=3).min()
```

Out[69]:
```
0    NaN
1    NaN
2    1.0
3    2.0
4    3.0
5    5.0
6    5.0
7    5.0
Name: a, dtype: float64
```

In [70]:
```
df5['a'].cumsum()
```

Out[70]:
```
0      1
1      3
2      6
3     12
4     17
5    115
6    120
7    125
Name: a, dtype: int64
```

In [75]:
```
# What is Date functionality in pyton pandas
date = pd.date_range(start="2023-03-23" ,end="2023-05-23")
```

In [76]:
```
date
```

Out[76]:
```
DatetimeIndex(['2023-03-23', '2023-03-24', '2023-03-25', '2023-03-26',
               '2023-03-27', '2023-03-28', '2023-03-29', '2023-03-30',
               '2023-03-31', '2023-04-01', '2023-04-02', '2023-04-03',
               '2023-04-04', '2023-04-05', '2023-04-06', '2023-04-07',
               '2023-04-08', '2023-04-09', '2023-04-10', '2023-04-11',
               '2023-04-12', '2023-04-13', '2023-04-14', '2023-04-15',
               '2023-04-16', '2023-04-17', '2023-04-18', '2023-04-19',
               '2023-04-20', '2023-04-21', '2023-04-22', '2023-04-23',
               '2023-04-24', '2023-04-25', '2023-04-26', '2023-04-27',
               '2023-04-28', '2023-04-29', '2023-04-30', '2023-05-01',
               '2023-05-02', '2023-05-03', '2023-05-04', '2023-05-05',
               '2023-05-06', '2023-05-07', '2023-05-08', '2023-05-09',
               '2023-05-10', '2023-05-11', '2023-05-12', '2023-05-13',
               '2023-05-14', '2023-05-15', '2023-05-16', '2023-05-17',
               '2023-05-18', '2023-05-19', '2023-05-20', '2023-05-21',
               '2023-05-22', '2023-05-23'],
              dtype='datetime64[ns]', freq='D')
```

In [78]:
```
df_date = pd.DataFrame({'date':date})
```

In [86]:
```
df_date.dtypes
```

Out[86]:
```
date    datetime64[ns]
dtype: object
```

In [79]:
```
df_date
```

Out[79]:

	date
0	2023-03-23
1	2023-03-24
2	2023-03-25
3	2023-03-26
4	2023-03-27
...	...
57	2023-05-19
58	2023-05-20
59	2023-05-21
60	2023-05-22
61	2023-05-23

62 rows × 1 columns

In [95]:
```
df7 = pd.DataFrame({"date" : ["2023-05-22" , '2023-05-21' , "2023-05-20"]})
```

In [96]:
```
df7
```

Out[96]:

	date
0	2023-05-22
1	2023-05-21
2	2023-05-20

In [97]:
```
df7.dtypes
```
Out[97]:
```
date     object
dtype: object
```

In [98]:
```
df7['update_date'] = pd.to_datetime(df7['date'])
```

In [99]:
```
df7
```
Out[99]:

	date	update_date
0	2023-05-22	2023-05-22
1	2023-05-21	2023-05-21
2	2023-05-20	2023-05-20

In [100…
```
df7.dtypes
```
Out[100]:
```
date                      object
update_date     datetime64[ns]
dtype: object
```

In [104…
```
df7['month'] = df7["update_date"].dt.month
```

In [105…
```
df7
```
Out[105]:

	date	update_date	month
0	2023-05-22	2023-05-22	5
1	2023-05-21	2023-05-21	5
2	2023-05-20	2023-05-20	5

In [106…
```
df7['year'] = df7["update_date"].dt.year
```

In [107…
```
df7
```
Out[107]:

	date	update_date	month	year
0	2023-05-22	2023-05-22	5	2023
1	2023-05-21	2023-05-21	5	2023
2	2023-05-20	2023-05-20	5	2023

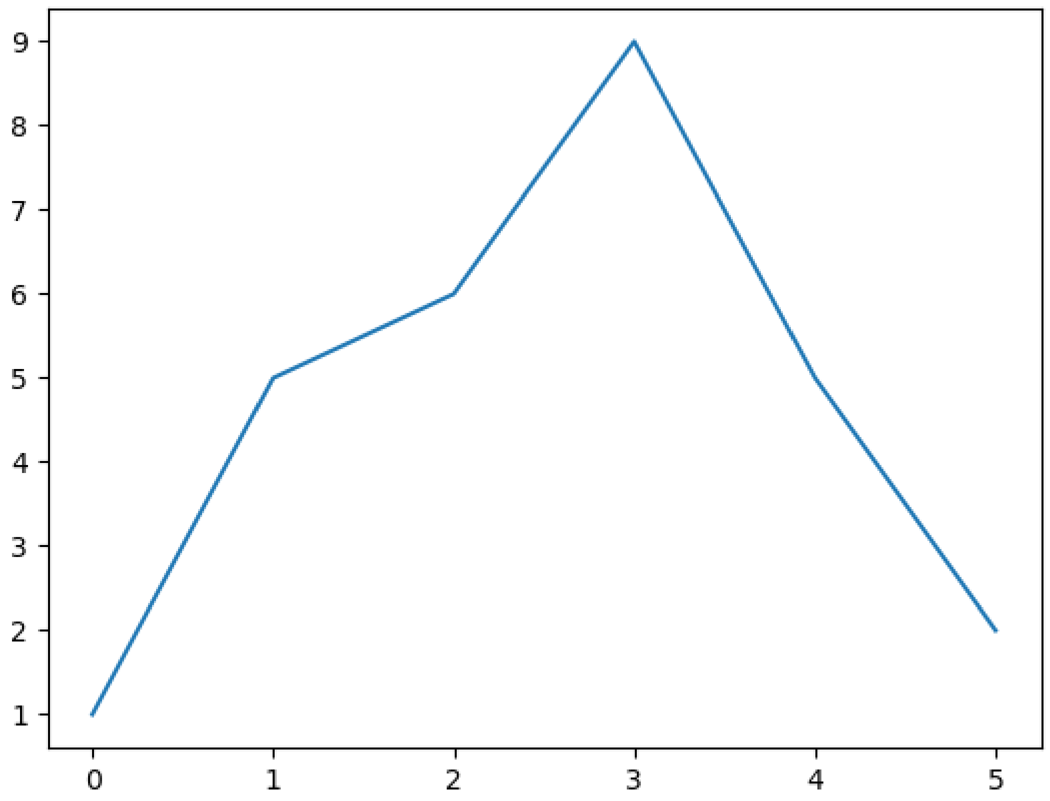
In [108…
```
df7['day'] = df7["update_date"].dt.day
```

In [109…
```
df7
```
Out[109]:

	date	update_date	month	year	day
0	2023-05-22	2023-05-22	5	2023	22
1	2023-05-21	2023-05-21	5	2023	21
2	2023-05-20	2023-05-20	5	2023	20

In [112…
```
# What is Time Delta in pandas python
pd.Timedelta(days = 1 , hours=5 , minutes=45 , seconds=36)
```

```
Out[112]: Timedelta('1 days 05:45:36')

In [113…  dt = pd.to_datetime("2023-06-23")

In [114…  td = pd.Timedelta(days = 1)

In [116…  dt+td

Out[116]: Timestamp('2023-06-24 00:00:00')

In [119…  # python pandas = categorical Data

          data = ["Abhi","Mayank" , "Amit","Aniket" , "Abhi","Abhi","Abhi"]

In [121…  cat = pd.Categorical(data)

In [122…  cat

Out[122]: ['Abhi', 'Mayank', 'Amit', 'Aniket', 'Abhi', 'Abhi', 'Abhi']
          Categories (4, object): ['Abhi', 'Amit', 'Aniket', 'Mayank']

In [123…  cat.value_counts()

Out[123]: Abhi      4
          Amit      1
          Aniket    1
          Mayank    1
          dtype: int64

In [124…  #python pandas = Visulalzation

In [128…  d = pd.Series([1,5,6,9,5,2])

In [129…  d

Out[129]: 0    1
          1    5
          2    6
          3    9
          4    5
          5    2
          dtype: int64

In [130…  d.plot()

Out[130]: <Axes: >
```



In [132…
```
df = pd.DataFrame({"a":[1,2,3,6,5,65],
                  "b":[2,6,5,9,6,36]})
```

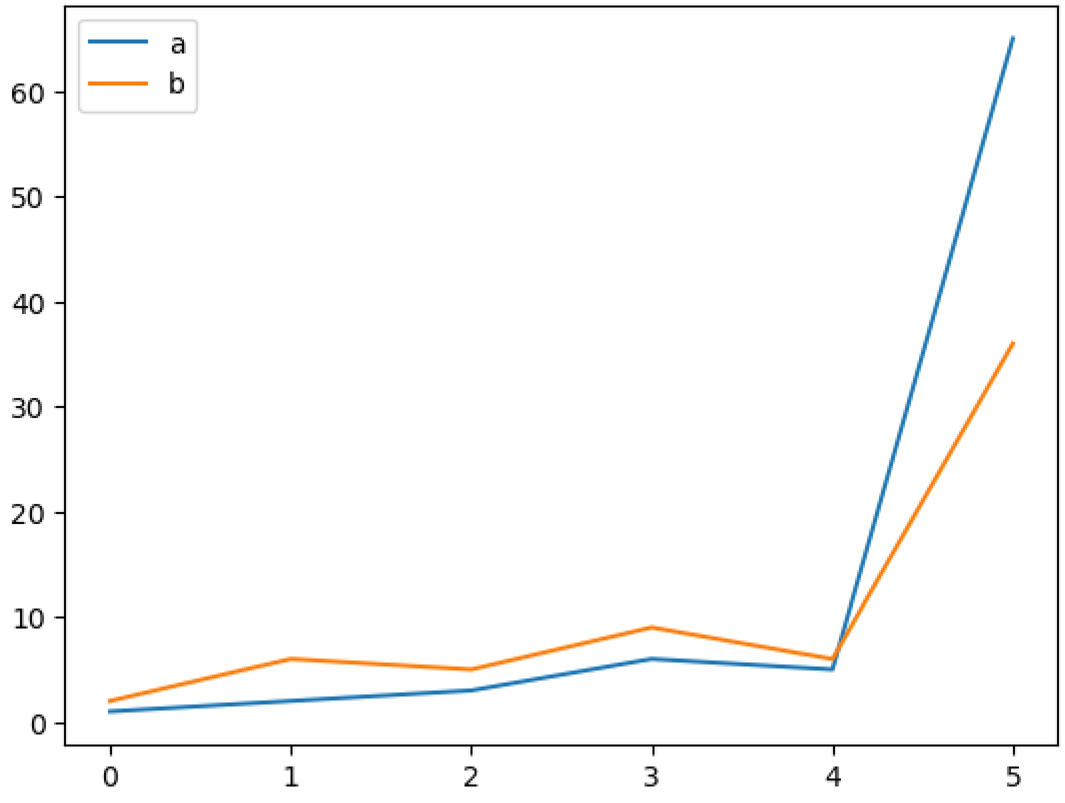
In [133…
```
df
```

Out[133]:

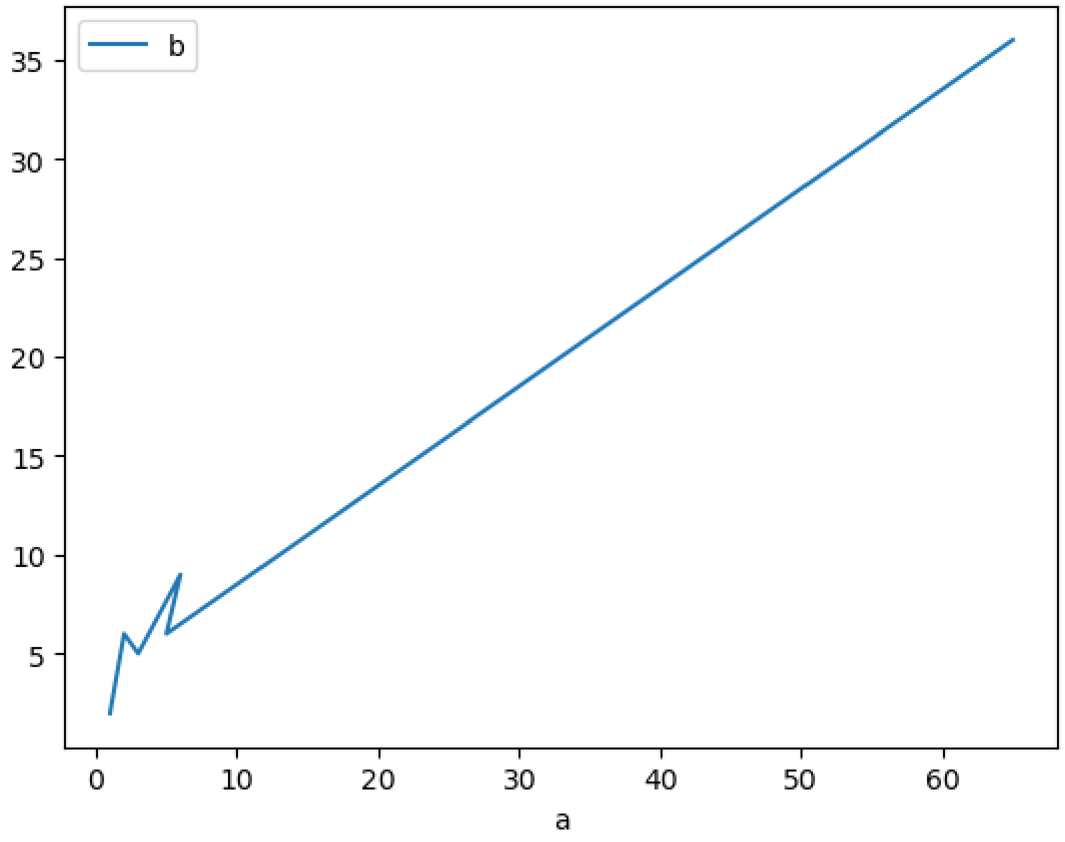
	a	b
0	1	2
1	2	6
2	3	5
3	6	9
4	5	6
5	65	36

In [136…
```
df.plot()
```

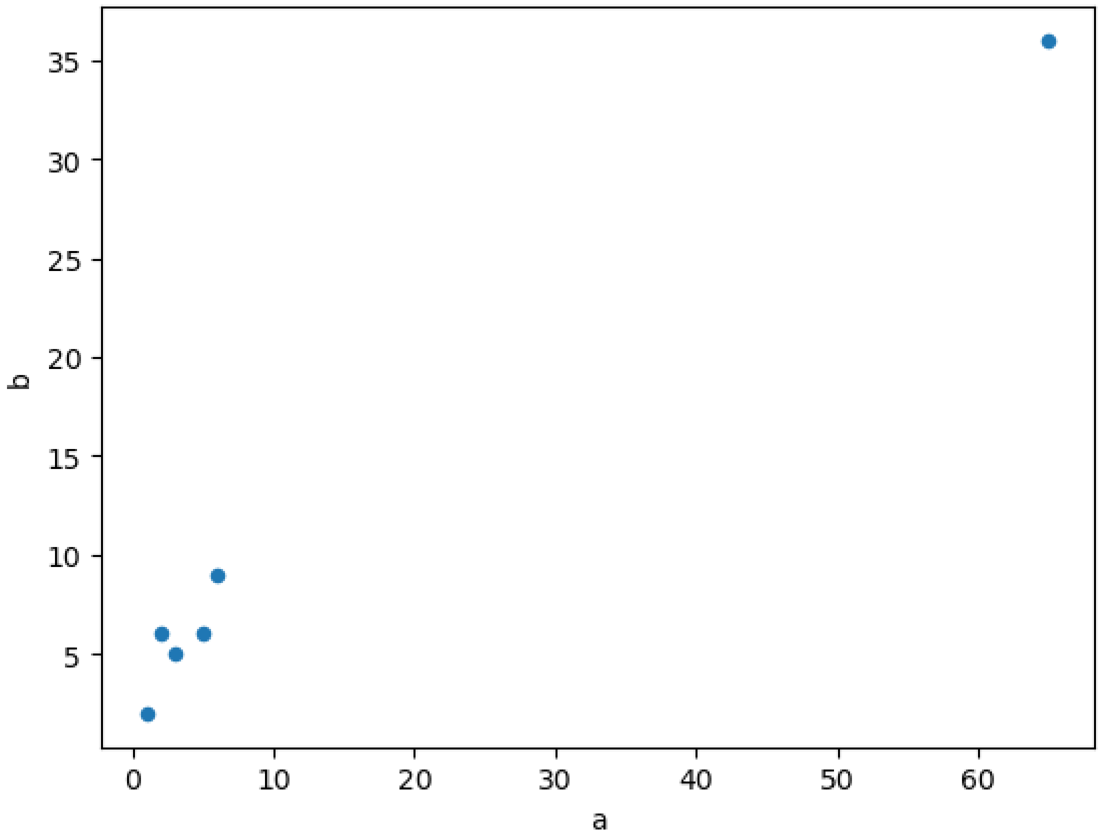
Out[136]:
```
<Axes: >
```



```
In [139…  df.plot(x='a',y='b')
Out[139]: <Axes: xlabel='a'>
```

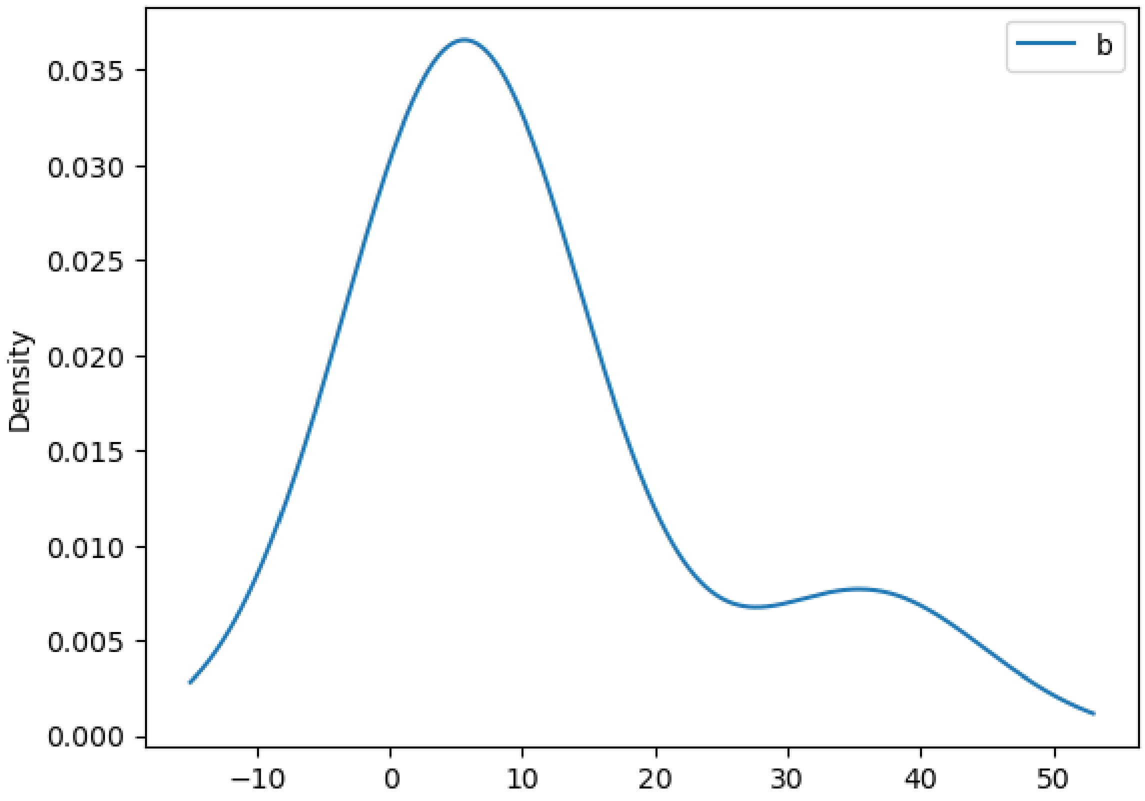


```
In [140…  df.plot.scatter(x='a',y='b')
Out[140]: <Axes: xlabel='a', ylabel='b'>
```



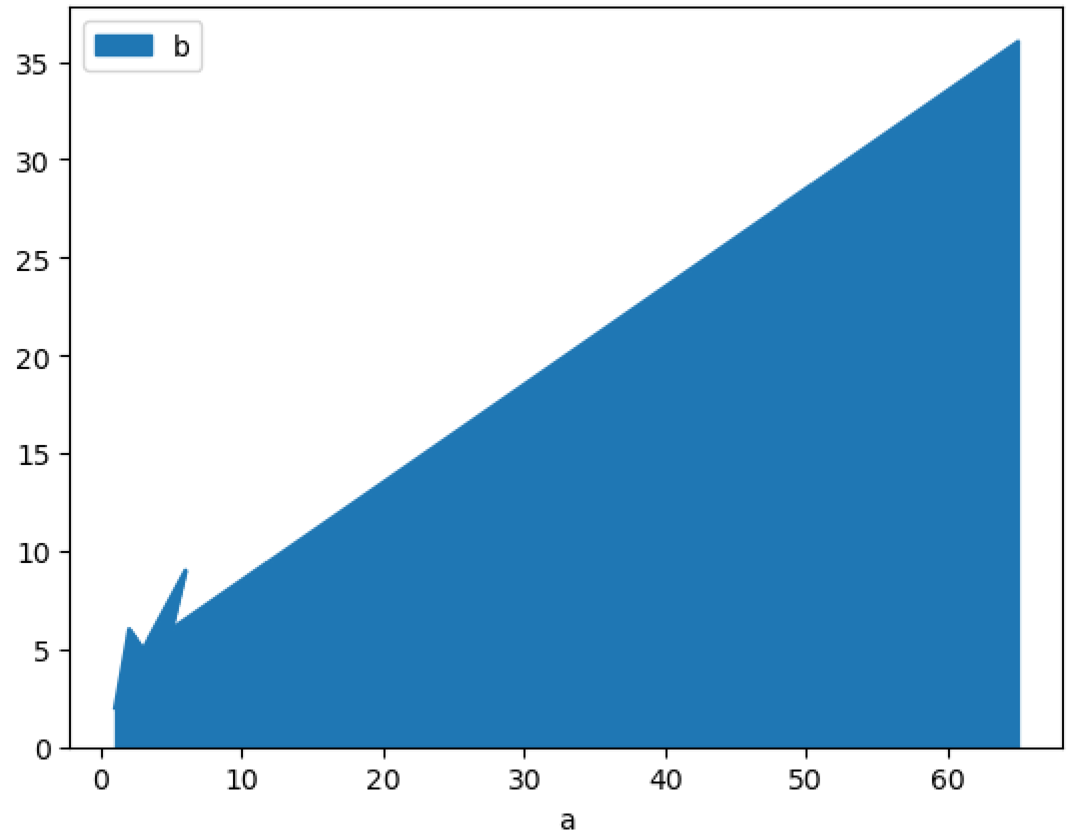
In [141…
```
df.plot.density(x='a',y='b')
```

Out[141]:
```
<Axes: ylabel='Density'>
```



In [142…
```
df.plot.area(x='a',y='b')
```

Out[142]:
```
<Axes: xlabel='a'>
```



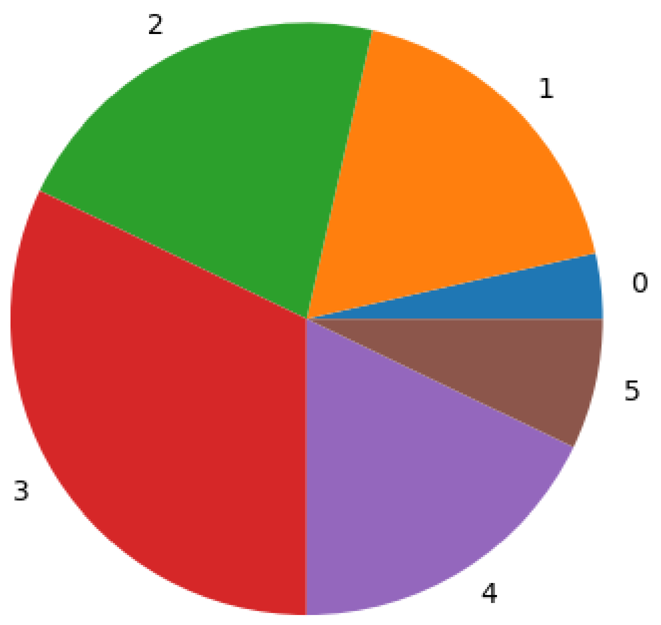
In [144…
```
d = pd.Series([1,5,6,9,5,2])
```

In [146…
```
d
```

Out[146]:
```
0    1
1    5
2    6
3    9
4    5
5    2
dtype: int64
```

In [145…
```
d.plot.pie()
```

Out[145]:
```
<Axes: >
```



```
In [ ]:
```