PANDAS

Exploratory Data Analysis

```
Modify or Clean(delete) decisive for classification
```

Calling necessary Libraries

In [2]:

```
import pandas as pd
import numpy as np
import warnings
```

In [3]:

```
warnings.filterwarnings('ignore')
```

Dataframe declaration

In [4]:

```
df=pd.read_csv('./Titanic.csv')
```

```
In [4]: df.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

*.reset_index( ) whenever a tabular form(including groupby) also drop=True

Value Counts

```
In [ ]: df.sort_values(by='column_name', ascending=False, inplace=True)
```

```
In [7]: df.nlargest(5,'Age')
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
851	852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.7750	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q

directly use for plot

♡Cross Tab

In [ ]:
```
pd.crosstab(df.Sport, df.Event,margins=True) # mixed value counts
                                              # Sport inwhich less no of events
```

In [ ]:
```
ഒരു SINGLE ഐറ്റം എത്ര തവണ repeat ആയി എന്ന് find  ചെയ്യാൻ
രണ്ട കോളത്തില് ഒരേ ഐറ്റം വന്നാൽ എണ്ണിയെടുക്കാൻ
Southampton nnu Pclass  ലേക്ക് എത്ര പേര് കയറി


```

In [ ]:
```
CT1=pd.crosstab(index=df.Team, columns=df.Year, values=df.Gold, aggfunc='sum')
```

In [ ]:
```
CT1.loc['china'] # index ippo countries aayi
```

In [ ]:
```
easy way to crosstab
Countries without female
Gender_Count=pd.crosstab(df.Sex,df.NOC)
Gender_Count.unstack()
            # o/p NOC  Sex
               #  AFG  Fem  5
               #       Male 121
               #  AHD  Fem  12
               #       Male 121
            again unstack
GCUU=Gender_Count.unstack().unstack()
GCUU.[Gender_Count.Female==0]

```

Progressive Report of Medals by Contry

❓

In [ ]:

```
df.where(df.column
```

In [ ]:

```
 isin ഉണ്ടോ ഇല്ലയോ  എന്ന്  embarked =['S','C']
```

df. # Suggestion

In [12]:

```
df.Age
```

Out[12]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ... 
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

CREATION OF NEW COLUMN

In [13]:

```
df['Age'] # when Groupby outside
          # Spaces between Phrases
          # New Column Create if no quotes
```

Out[13]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ... 
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [ ]:

```
df['recent']=[1 if yr_renovated<=10 else 0 for i in df.yr_renov] # for medal in df.Medal
```

In [ ]:

```
df['House_age']=[2021-i for i in df['yr_built']]
```

In [ ]:

```
df[['Age','Sex']].nunique() # return ie o/p Age=98 Sex -2
                            # axis to count unique values in either columns or rows
    unique lists non null value counts

```

In [ ]:

```
df['Rank']=df.Fare.rank()
```

In [5]:

```
df.Fare.value_counts()
```

Out[5]:

```
8.0500     43
13.0000    42
7.8958     38
7.7500     34
26.0000    31
           ..
35.0000     1
28.5000     1
6.2375      1
14.0000     1
10.5167     1
Name: Fare, Length: 248, dtype: int64
```

Data Cleaning

```
correction => True
```

In [ ]:

```
df.rename(columns={"oldname1":"newname1","oldname2":"newname2"}) #else should use df['col name']
```

In [ ]:

```
Replace
df.Sex.replace({'Male':0,'Female':}, inplace=True)
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [1]:
```
df.Fare.sum()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5016\3344774125.py in <module>
----> 1 df.Fare.sum()

NameError: name 'df' is not defined
```

In [ ]:
```
df['Country'].duplicated.sum() # we will get idea abt indvidual value counts as well but in a different w
```

In [ ]:
```
df.fare.pct_change() # esp financiall data - Numerical
```

In [ ]:
```
df['Age','Fare'].cumsum() # limited application - additive on daily basis eg Profit
```

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

In [ ]:
```
df.sample()
```

df.head loading time

Age column and Cabin column has nullvalue

```
In [7]: df.isna().mean()*100 #20% above then drop
```

```
Out[7]: PassengerId     0.000000
        Survived        0.000000
        Pclass          0.000000
        Name            0.000000
        Sex             0.000000
        Age            19.865320
        SibSp           0.000000
        Parch           0.000000
        Ticket          0.000000
        Fare            0.000000
        Cabin          77.104377
        Embarked        0.224467
        dtype: float64
```

Age has no less than 20percent null value and cabin has more than twenty percent nullvalue- removing the "Cabin"

df.drop(columns='Cabin', inplace=True) # repetetive

In [8]:
```
df.Age.value_counts() # displays NaN also
```

Out[8]:
```
24.00    30
22.00    27
18.00    26
19.00    25
28.00    25
         ..
36.50     1
55.50     1
0.92      1
23.50     1
74.00     1
Name: Age, Length: 88, dtype: int64
```

In [9]:
```
df.Age.unique()
```

Out[9]:
```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [10]:
```
df.Age.nunique() # how to count values in multiple DataFrame using nunique
```

Out[10]:
```
88
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

Removed the Cabin

In [10]:

```python
df.drop(columns=['Name','Ticket', 'PassengerId', 'SibSp', 'Parch'])
```

Out[10]:

```
     Survived  Pclass     Sex   Age     Fare  Embarked
  0         0       3    male  22.0   7.2500         S
  1         1       1  female  38.0  71.2833         C
  2         1       3  female  26.0   7.9250         S
  3         1       1  female  35.0  53.1000         S
  4         0       3    male  35.0   8.0500         S
...       ...     ...     ...   ...      ...       ...
886         0       2    male  27.0  13.0000         S
887         1       1  female  19.0  30.0000         S
888         0       3  female   NaN  23.4500         S
889         1       1    male  26.0  30.0000         C
890         0       3    male  32.0   7.7500         Q

891 rows × 6 columns
```

In [11]:

```python
df.Age.fillna(df.Age.median(),inplace=True) # replaced with median
```

In [7]:

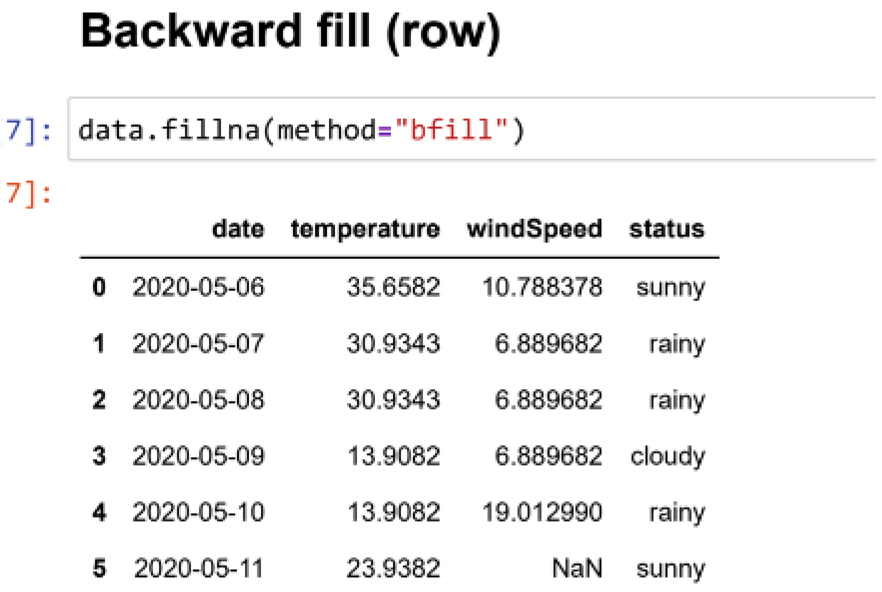
```python
df.fillna({'Pclass':3,'Embarked':'S'},inplace=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5924\2103977946.py in <module>
----> 1 df.filna({'Temperature'})

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5900         ):
   5901             return self[name]
-> 5902         return object.__getattribute__(self, name)
   5903 
   5904     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'filna'
```

date temperature windSpeed status



Backward fill (row)

```
7]: data.fillna(method="bfill")
```

7]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	30.9343	6.889682	rainy
2	2020-05-08	30.9343	6.889682	rainy
3	2020-05-09	13.9082	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	NaN	sunny

Forward fill (column)

In [8]:
```
data.fillna(method="ffill", axis="columns")
```

Out[8]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	2020-05-07 00:00:00	2020-05-07 00:00:00	2020-05-07 00:00:00
2	2020-05-08	30.9343	30.9343	rainy
3	2020-05-09	2020-05-09 00:00:00	6.889682	cloudy
4	2020-05-10	13.9082	19.01299	rainy
5	2020-05-11	23.9382	23.9382	sunny

Backward fill (column)

In [9]:
```
data.fillna(method="bfill", axis="columns")
```

Out[9]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	NaT	NaT	NaT
2	2020-05-08	30.9343	rainy	rainy
3	2020-05-09	6.889682	6.889682	cloudy
4	2020-05-10	13.9082	19.01299	rainy
5	2020-05-11	23.9382	sunny	sunny

Limiting the forward/backward fill

We can limit the number of rows or columns getting filled.

```
0]: data.fillna(method="ffill", limit=1)
```

0]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	35.6582	10.788378	sunny
2	2020-05-08	30.9343	NaN	rainy
3	2020-05-09	30.9343	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	19.012990	sunny

Filling with Pandas objects

There are many Pandas objects like df.sum(), df.max(), etc. we can fill the missing values with these too.

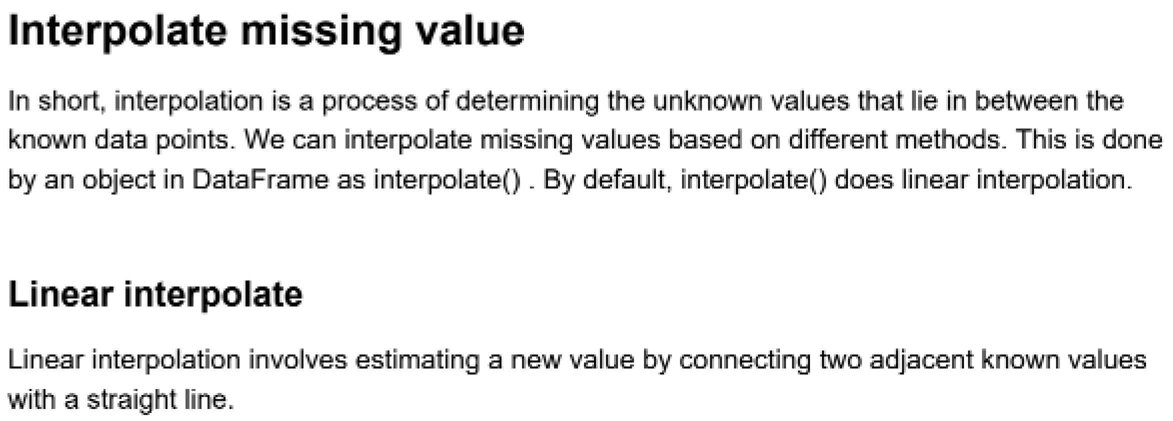
```
1]: data.fillna(data.mean())
```

1]:

	date	temperature	windSpeed	status
0	2020-05-06	35.658200	10.788378	sunny
1	2020-05-07	26.109725	12.230350	NaN
2	2020-05-08	30.934300	12.230350	rainy
3	2020-05-09	26.109725	6.889682	cloudy
4	2020-05-10	13.908200	19.012990	rainy
5	2020-05-11	23.938200	12.230350	sunny

limiting the fillna

only one NaN (not as a row but in individual column or rows , no consecutive NaN filling)



Interpolate missing value

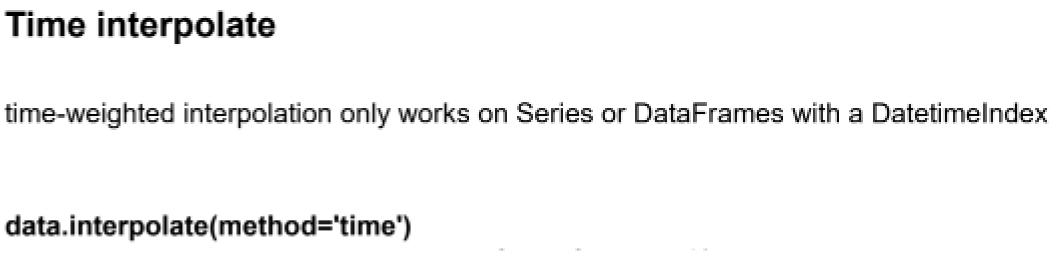
In short, interpolation is a process of determining the unknown values that lie in between the known data points. We can interpolate missing values based on different methods. This is done by an object in DataFrame as interpolate() . By default, interpolate() does linear interpolation.

Linear interpolate

Linear interpolation involves estimating a new value by connecting two adjacent known values with a straight line.

```
In [13]: data.temperature.interpolate()
```

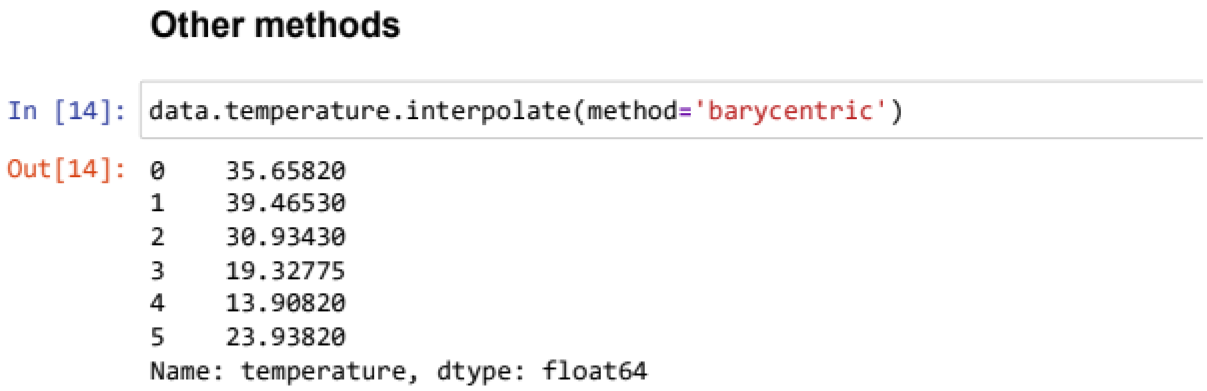
```
Out[13]: 0    35.65820
         1    33.29625
         2    30.93430
         3    22.42125
         4    13.90820
         5    23.93820
         Name: temperature, dtype: float64
```



Time interpolate

time-weighted interpolation only works on Series or DataFrames with a DatetimeIndex

data.interpolate(method='time')



Other methods

```
In [14]: data.temperature.interpolate(method='barycentric')

Out[14]: 0    35.65820
         1    39.46530
         2    30.93430
         3    19.32775
         4    13.90820
         5    23.93820
         Name: temperature, dtype: float64
```

In [12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

In [ ]:

```

```

In [ ]:

```
New Column Creation or transform
```

Value Counts

In [49]:

```
df['Survived'].value_counts(normalize=True) #parameters YT
```

Out[49]:

```
0    0.616162
1    0.383838
Name: Survived, dtype: float64
```

In [11]:

```
df.Age.value_counts().max() # Most occuring item in it
                            # Top number of players
```

Out[11]:

```
30
```

In [ ]:

```

```

df.dropduplicates(inplace=True) Check for matching rows

In [6]:

```
df.duplicated().sum()
```

Out[6]:

```
0
```

In [16]:

```
df['Embarked'].duplicated().sum()
```

Out[16]:

```
887
```

In [50]:

```
# Proportion
```

boolean MASK

In [ ]:

```
filter df based on conditions esp: rows ie Quantify sub-collection in a collection.
when you want manipulated data in collection based on some criteria True or False
df[BM3&BM4]
```

In [ ]:

```
BM=df.listed_in=='comedies,'
df.[BM].country
combine two masks using
                        logical or not and where() logical and()
    df[BM3 & BM4] #dataframe is displayed
    df[BM3 & BM4].country  # that column is displayed

    # This method will work only if df has only one value and that value must be either True or False
    df.Age.[BM1& BM2] #BM1  is  Pclass=3 BM2 is Age<20
```

"groupby"

In [ ]:

```

```

In [13]:

```
df.groupby(['Pclass'])['Survived'].sum()
```

Out[13]:

```
Pclass
1    136
2     87
3    119
Name: Survived, dtype: int64
```

In [14]:

```
df.groupby(['Pclass','Sex'])['Survived'].sum()
```

Out[14]:

```
Pclass  Sex
1       female    91
        male      45
2       female    70
        male      17
3       female    72
        male      47
Name: Survived, dtype: int64
```

In [15]:

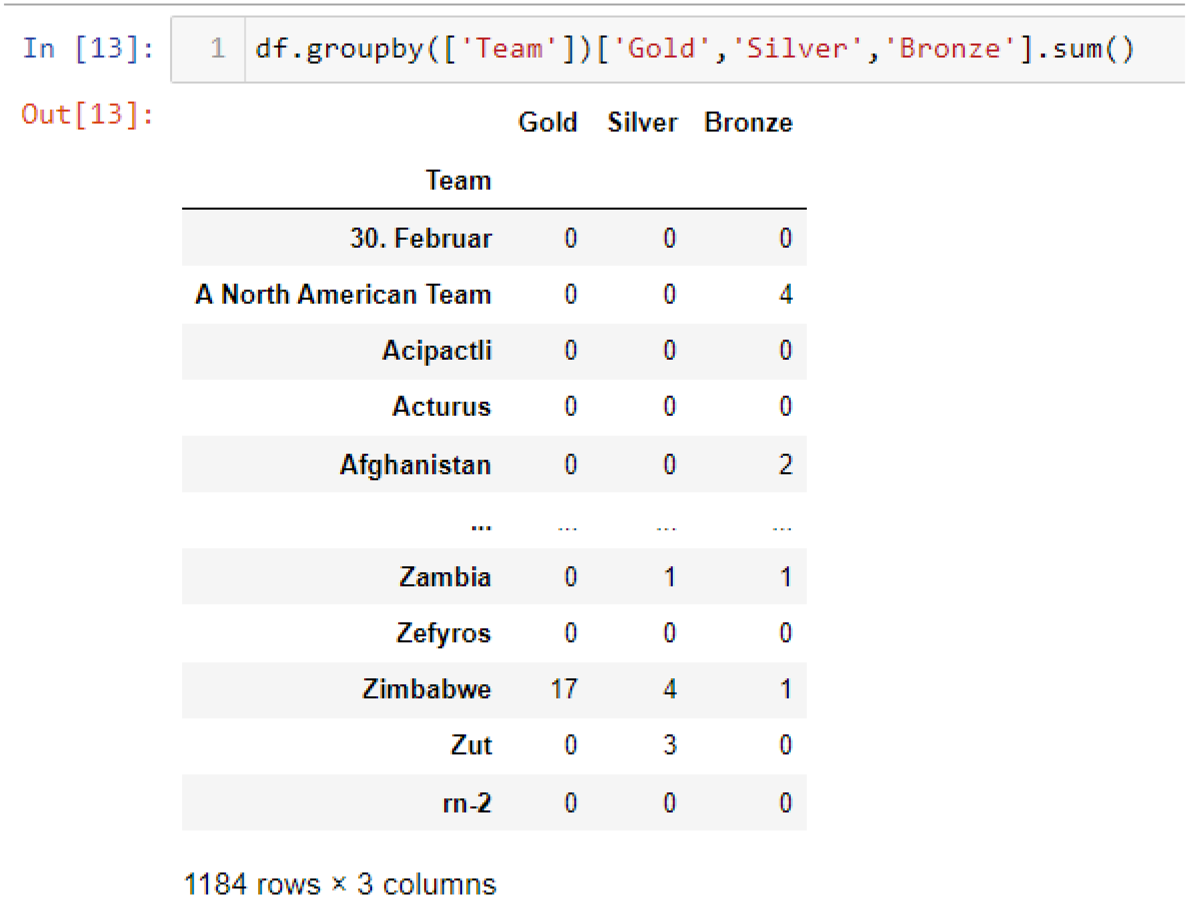
```
df.groupby(['Sex', 'Pclass'] )['Survived'].count()
```

Out[15]:

```
Sex     Pclass
female  1          94
        2          76
        3         144
male    1         122
        2         108
        3         347
Name: Survived, dtype: int64
```

```
In [16]: df.groupby(['Sex', 'Pclass'] )['Survived'].sum()
```

```
Out[16]: Sex     Pclass
         female  1         91
                 2         70
                 3         72
         male    1         45
                 2         17
                 3         47
         Name: Survived, dtype: int64
```



```
In [13]: df.groupby(['Team'])['Gold','Silver','Bronze'].sum()
```

Out[13]:

Team	Gold	Silver	Bronze
30. Februar	0	0	0
A North American Team	0	0	4
Acipactli	0	0	0
Acturus	0	0	0
Afghanistan	0	0	2
...	...	...	...
Zambia	0	1	1
Zefyros	0	0	0
Zimbabwe	17	4	1
Zut	0	3	0
rn-2	0	0	0

1184 rows × 3 columns

In [ ]:

```
List_No_Gold=list(No_Gold.index)
```

In [ ]:

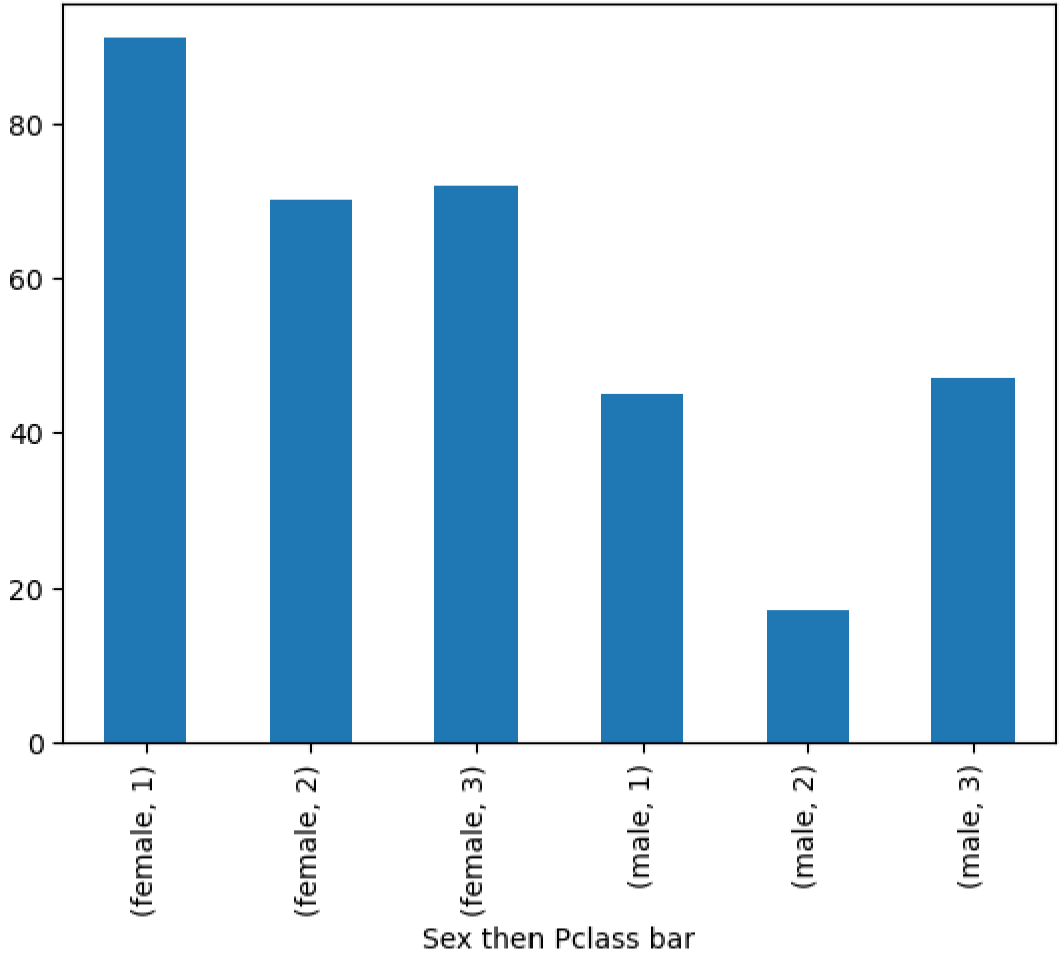
```
lem(List_No_Gold)=942 # That means there are 942 Players without gold
```

In [17]:

```
df.groupby(['Sex', 'Pclass'] )['Survived'].sum().plot.bar(xlabel="Sex then Pclass bar")
```

Out[17]:

```
<AxesSubplot:xlabel='Sex then Pclass bar'>
```



In [18]:
```
df.groupby(['Sex', 'Survived'] )['Survived'].count().unstack('Sex')
```

Out[18]:

Sex	female	male
Survived		
0	81	468
1	233	109

unstacking

In [19]:
```
Sex_Survived_Sum=df.groupby(['Sex'])['Survived']
```

In [20]:
```
Sex_Survived_Sum
```

Out[20]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000002589E89B250>
```

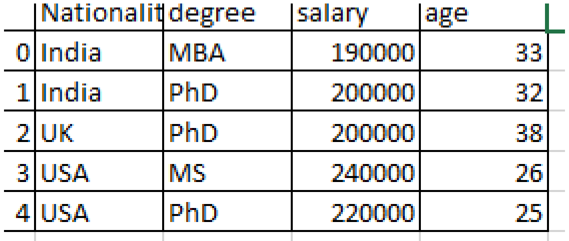
In [21]:
```
# the groupby is an object creation
```

In [22]:
```
Sex_Survived_Sum_Avg=Sex_Survived_Sum.mean()*100
```

In [23]:
```
Sex_Survived_Sum_Avg
```

Out[23]:
```
Sex
female    74.203822
male      18.890815
Name: Survived, dtype: float64
```

groupby on Multiple Columns and mulltiple fn's on single column



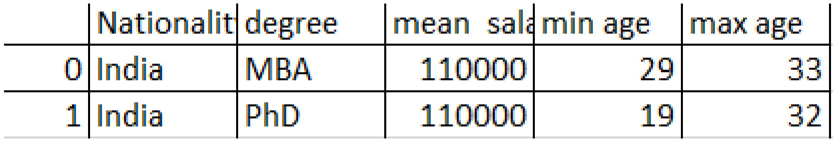
	Nationalit	degree	salary	age
0	India	MBA	190000	33
1	India	PhD	200000	32
2	UK	PhD	200000	38
3	USA	MS	240000	26
4	USA	PhD	220000	25

In [ ]:

```
ie for indians with MBA the max age is 33.
```

In [ ]:

```
df.groupby(['nationality','degree'])
                .agg(
                (mean_salary=('salary','mean')
                (min_age=('age','min')
                (max_age=('age','max')
                ).reset_index()
```



	Nationalit	degree	mean sala	min age	max age
0	India	MBA	110000	29	33
1	India	PhD	110000	19	32

In [ ]:

```
rename columns name in Groupby python pandas automatically
automate cumbersome manually renaming

Pclass_Surv.columns=Pclass_Surv.columns.droplevel(level=0)
Pclass_Surv.columns=[' '+ looping]
```

Operations on Groups

In [ ]:
```
df.groupby(['nationality','degree'])

select group: View Groups
    get_group()
    for eg: degree has 3 groups MS MBA and PhD lets select
        select the degree values present in the MS group
df.groupby()
```

🥇New DataFrame via filter or query or pd.Grp

```
### Permute and combine Groupby@last
### Then only analyse
```

Filter ie Equation Form

In [ ]:
```
df1=df[0:100]
```

In [ ]:
```
df[df.Price!=0]
```

In [ ]:
```
Summer=df[df["Season"=="Summer"]] # len(summer)
```

In [ ]:
```
df[df.Price==0]
```

In [ ]:
```
df_nonzero=df[df.yr_renov!=0] # table corresponding to non zero column values of that table
```

In [ ]:
```
df[df.yr_renov!=0][yr_renov].min()
```

In [ ]:

```
df_nonzero=df[df.yr_renov>10]
```

In [ ]:

```
df_nonzero.['yr_renov'].min()
```

In [5]:

```
df1=df[['Age','Pclass']] # CREATE NEW DF
```

In [ ]:

```
Country with most medals and less number of players

TMC['Total']=TMC['Bronze']+TMC['Silver']+TMC[''] #len(TMC)

```

In [ ]:

```
Team_and_Medal_count=pd.crosstab(df.Team,df.Medal) #<df.Team,df.Medal>
```

In [6]:

```
df1
```

Out[6]:

```
      Age  Pclass
  0  22.0       3
  1  38.0       1
  2  26.0       3
  3  35.0       1
  4  35.0       3
...   ...     ...
886  27.0       2
887  19.0       1
888   NaN       3
889  26.0       1
890  32.0       3

891 rows × 2 columns
```

```
In [17]: df2=df[df.Fare!=0] # Non Zero Values in column
```

```
In [ ]: df['Gold']=[1 if medal=='Gold'][1 if medal =='Gold' else 0 ]
```

```
In [ ]: df[df.yr_renovated!=0][yr_renovated].min()
```

```
In [ ]: 
```

```
In [18]: df2
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

876 rows × 12 columns

☼Is the renovation recent or not

df['was renovated']=[1 if yr_renovated!= 0 else 0 for **yr_renovated in df.yr_renovated**] # housing data

what all after FOR is treated as a block

Query

```
can't find sum
```

In [14]:

```
Sex_Age_20 = df.query("Sex=='male' & Age == 20.0")
```

#query - display the row in full so that we get other rdb column value

In [ ]:

```
Board_Surv=df.query("Boarded=='S' & Age=22 & survived ==1.0")
```

In [ ]:

```
Gold Medalist in 1992 basketball event =df.query("Year==1992 & Sport =='Basketball' & Medal=='Gold'  & Se
```

In [25]:
```
Sex_Age_20
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	S
91	92	0	3	Andreasson, Mr. Paul Edvin	male	20.0	0	0	347466	7.8542	S
131	132	0	3	Coelho, Mr. Domingos Fernandeo	male	20.0	0	0	SOTON/O.Q. 3101307	7.0500	S
378	379	0	3	Betros, Mr. Tannous	male	20.0	0	0	2648	4.0125	C
441	442	0	3	Hampe, Mr. Leon	male	20.0	0	0	345769	9.5000	S
622	623	1	3	Nakid, Mr. Sahid	male	20.0	1	1	2653	15.7417	C
640	641	0	3	Jensen, Mr. Hans Peder	male	20.0	0	0	350050	7.8542	S
664	665	1	3	Lindqvist, Mr. Eino William	male	20.0	1	0	STON/O 2. 3101285	7.9250	S
682	683	0	3	Olsvigen, Mr. Thor Anderson	male	20.0	0	0	6563	9.2250	S
725	726	0	3	Oreskovic, Mr. Luka	male	20.0	0	0	315094	8.6625	S
762	763	1	3	Barah, Mr. Hanna Assi	male	20.0	0	0	2663	7.2292	C
840	841	0	3	Alhomaki, Mr. Ilmari Rudolf	male	20.0	0	0	SOTON/O2 3101287	7.9250	S
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	S

In [26]:
```
Sex_Age_20_SibSp_1=df.query("Sex=='male' & Age==22.0 & SibSp == 1")
```

In [27]:
```
Sex_Age_20_SibSp_1
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	S

PIVOT TABLE

In [28]:
```
pivot01=df.pivot_table(values='Fare',index='Sex',columns='Pclass',aggfunc='sum')
```

In [29]:

```
pivot01
```

Out[29]:

```
Pclass          1          2          3
   Sex
female  9975.8250  1669.7292  2321.1086
  male  8201.5875  2132.1125  4393.5865
```

In [30]:

```
pivot02=df.pivot_table(values='Fare',index='Embarked',columns='Pclass',aggfunc='sum')
```

In [31]:

```
pivot02
```

Out[31]:

```
  Pclass          1          2          3
Embarked
       C  8901.0750   431.0917   740.1295
       Q   180.0000    37.0500   805.2043
       S  8936.3375  3333.7000  5169.3613
```

Insert

this will insert a new column in the desired place. For this purpose let's create a new column first using np.random

In [32]:

```
column = np.random.randint(0,100, size=len(df))
```

column was created with random numbers 0 -100 length of column as the length of df

df.info()

In [35]:

```
df.insert(4,'column',column)
```

In [36]:

```
df.head()
```

Out[36]:

	PassengerId	Survived	Pclass	Name	column	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	95	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	86	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	92	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	23	male	35.0	0	0	373450	8.0500	S

In [37]:

```
df[['Fare','Age']].cumsum()
```

Out[37]:

	Fare	Age
0	7.2500	22.00
1	78.5333	60.00
2	86.4583	86.00
3	139.5583	121.00
4	147.6083	156.00
...	...	...
886	28602.7493	26056.17
887	28632.7493	26075.17
888	28656.1993	26103.17
889	28686.1993	26129.17
890	28693.9493	26161.17

891 rows × 2 columns

In [38]:

```
df.where(df.column>50)
```

Out[38]:

	PassengerId	Survived	Pclass	Name	column	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1.0	0.0	3.0	Braund, Mr. Owen Harris	95.0	male	22.0	1.0	0.0	A/5 21171	7.250	S
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	3.0	1.0	3.0	Heikkinen, Miss. Laina	86.0	female	26.0	0.0	0.0	STON/O2. 3101282	7.925	S
3	4.0	1.0	1.0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	92.0	female	35.0	1.0	0.0	113803	53.100	S
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
886	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
887	888.0	1.0	1.0	Graham, Miss. Margaret Edith	89.0	female	19.0	0.0	0.0	112053	30.000	S
888	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
890	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

891 rows × 12 columns

In [39]:

```
df.Pclass.unique()
```

Out[39]:

```
array([3, 1, 2], dtype=int64)
```

pd.cut

In [41]:

```
cutoff = [0,18,50,85]
LABELS=["Child","Adult","Old"]
df['AGE_TYPE']=pd.cut(df.Age, bins=cutoff, labels=LABELS)
```

In [42]:

```
df
```

Out[42]:

	PassengerId	Survived	Pclass	Name	column	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	AGE_TYPE
0	1	0	3	Braund, Mr. Owen Harris	95	male	22.0	1	0	A/5 21171	7.2500	S	Adult
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38	female	38.0	1	0	PC 17599	71.2833	C	Adult
2	3	1	3	Heikkinen, Miss. Laina	86	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Adult
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	92	female	35.0	1	0	113803	53.1000	S	Adult
4	5	0	3	Allen, Mr. William Henry	23	male	35.0	0	0	373450	8.0500	S	Adult
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	7	male	27.0	0	0	211536	13.0000	S	Adult
887	888	1	1	Graham, Miss. Margaret Edith	89	female	19.0	0	0	112053	30.0000	S	Adult
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	37	female	28.0	1	2	W./C. 6607	23.4500	S	Adult
889	890	1	1	Behr, Mr. Karl Howell	1	male	26.0	0	0	111369	30.0000	C	Adult
890	891	0	3	Dooley, Mr. Patrick	44	male	32.0	0	0	370376	7.7500	Q	Adult

891 rows × 13 columns

In [43]:

```

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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   column       891 non-null    int32   
 5   Sex          891 non-null    object  
 6   Age          891 non-null    float64 
 7   SibSp        891 non-null    int64   
 8   Parch        891 non-null    int64   
 9   Ticket       891 non-null    object  
 10  Fare         891 non-null    float64 
 11  Embarked     889 non-null    object  
 12  AGE_TYPE     891 non-null    category
dtypes: category(1), float64(2), int32(1), int64(5), object(4)
memory usage: 81.2+ KB
```

In [44]:

```
df.AGE_TYPE
```

Out[44]:

```
0      Adult
1      Adult
2      Adult
3      Adult
4      Adult
       ...  
886    Adult
887    Adult
888    Adult
889    Adult
890    Adult
Name: AGE_TYPE, Length: 891, dtype: category
Categories (3, object): ['Child' < 'Adult' < 'Old']
```

nlargest & nsmallest : count also

In [46]:

```
df.nlargest(5,'Age') # five athletes with highest age
```

Out[46]:

	PassengerId	Survived	Pclass	Name	column	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	AGE_TYPE
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	1	male	80.0	0	0	27042	30.0000	S	Old
851	852	0	3	Svensson, Mr. Johan	44	male	74.0	0	0	347060	7.7750	S	Old
96	97	0	1	Goldschmidt, Mr. George B	86	male	71.0	0	0	PC 17754	34.6542	C	Old
493	494	0	1	Artagaveytia, Mr. Ramon	91	male	71.0	0	0	PC 17609	49.5042	C	Old
116	117	0	3	Connors, Mr. Patrick	28	male	70.5	0	0	370369	7.7500	Q	Old

In [47]:

```
df.nsmallest(5,'Age')
```

Out[47]:

	PassengerId	Survived	Pclass	Name	column	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	AGE_TYPE
803	804	1	3	Thomas, Master. Assad Alexander	95	male	0.42	0	1	2625	8.5167	C	Child
755	756	1	2	Hamalainen, Master. Viljo	12	male	0.67	1	1	250649	14.5000	S	Child
469	470	1	3	Baclini, Miss. Helene Barbara	87	female	0.75	2	1	2666	19.2583	C	Child
644	645	1	3	Baclini, Miss. Eugenie	27	female	0.75	2	1	2666	19.2583	C	Child
78	79	1	2	Caldwell, Master. Alden Gates	68	male	0.83	0	2	248738	29.0000	S	Child

In [51]:

```
"""We often want to break down the rows by more than one category. Remember that the Titanic passengers a

This is the role of the Pandas crosstab function. It is a Pandas function because it is function inside t

The first argument to pd.crosstab is the category we want to see in the rows; the second argument is the

Here is a cross-tabulation of gender (in the rows) by survived (in the columns):"""
```

Out[51]:

```
'We often want to break down the rows by more than one category. Remember that the Titanic passengers and cr
ew tended to give preference to women and children, when loading the lifeboats. So, we may want to see the c
ounts of passengers who survived, broken down by gender.\n\nThis is the role of the Pandas crosstab functio
n. It is a Pandas function because it is function inside the Pandas module; we can get this function with p
d.crosstab (assuming we have done the usual import pandas as pd).\n\nThe first argument to pd.crosstab is th
e category we want to see in the rows; the second argument is the category we want to see in the columns.\n
\nHere is a cross-tabulation of gender (in the rows) by survived (in the columns):'
```

In [53]:

```
# Cross-tabulation of counts for 'gender' (rows) by 'survived' (columns).
pd.crosstab(df['Sex'], df['Survived'])
```

Out[53]:

Survived	0	1
Sex		
female	81	233
male	468	109

In [54]:

```
"""We will often want to see these values as proportions rather than counts. For example, we may be inter
```

Out[54]:

```
'We will often want to see these values as proportions rather than counts. For example, we may be interested
in the proportion of women and men that survived. As for value_counts above, we use the normalize keyword to
ask for proportions. This time we have to specify the direction that Pandas should use for the proportion. W
e could be interested in the proportion across the column (proportions of male and female passengers within
the yes “survived” category, likewise for the no category). More likely, in this case, we will be interested
in proportions across the row (proportion who survived within male category, proportion who survived within
female category). We give Pandas this information with the value for the normalize keyword argument. Pandas
uses the term index to refer to the rows. Remember, Pandas also uses the term index for the row labels'
```

In [ ]:

```

```

In [ ]:

```
RESET IND grp graph
```