Pandas_notes

September 25, 2021

1 Pandas:

- Pandas is a powerful python data analysis Toolkit.
- Open Source

1.1 Data Structure:

- Series *lablled Homogenous*
- DataFrame *Hetrogenous tabular*
- Panel *3d labelled Array*

[1]:
```
import pandas as pd
import numpy as np
```

1.1.1 Series:

[2]:
```
#short method of creating Series:
series_1 = pd.Series([1,2,3,4])
series_1
```

[2]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

[3]:
```
#changing index:
series_2 = pd.Series([1,2,3,4],index = ['a','b','c','d']) #index should be␣
 ↪equal to data values:
series_2
```

[3]:
```
a    1
b    2
c    3
d    4
dtype: int64
```

```
[4]: #changing datatype: (optional)
     series_2 = pd.Series([30, 35, 40], index=['2015 Sales', '2016 Sales', '2017␣
      ↪Sales'], name='Product A')
     series_2
```

```
[4]: 2015 Sales    30
     2016 Sales    35
     2017 Sales    40
     Name: Product A, dtype: int64
```

```
[5]: #creating series with dict:
     people = {'a' : 1,'b':2,'c':3}
     series_3 = pd.Series(people)
     series_3
```

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

access the value of series:

```
[6]: #slicing series:
     series_2[0:2]
```

```
[6]: 2015 Sales    30
     2016 Sales    35
     Name: Product A, dtype: int64
```

aggregate function applicable on Series:

1.1.2 DataFrame:

A DataFrame is a table. It contains an array of individual entries, each of which has a certain value. Each entry corresponds to a row (or record) and a column.

Creating DataFrame:

```
[7]: #creating dataframe:
     pd.DataFrame({'Yes': [50, 21], 'No': [131, 2]})
```

```
[7]:    Yes   No
     0   50  131
     1   21    2
```

```
[8]: pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'], 'Sue': ['Pretty good.',␣
      ↪'Bland.']})
```

```
[8]:              Bob           Sue
     0    I liked it.  Pretty good.
     1  It was awful.        Bland.
```

- Row labels used in a DataFrame is known as an **Index**.
- We can assign values to it by using an index parameter

```
[9]: pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'],
                   'Sue': ['Pretty good.', 'Bland.']},
                  index=['Product A', 'Product B'])
```

```
[9]:                      Bob           Sue
     Product A    I liked it.  Pretty good.
     Product B  It was awful.        Bland.
```

Reading DataFiles:

```
[10]: df = pd.read_csv("D:\\Dataset_ash\\SQL Case Study\\athlete_events.csv")
      df.head()
```

```
[10]:    ID                      Name Sex   Age  Height  Weight            Team  \
      0   1                 A Dijiang   M  24.0   180.0    80.0           China
      1   2                  A Lamusi   M  23.0   170.0    60.0           China
      2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
      3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
      4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

         NOC        Games  Year  Season       City          Sport  \
      0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
      1  CHN  2012 Summer  2012  Summer     London           Judo
      2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
      3  DEN  1900 Summer  1900  Summer      Paris    Tug-Of-War
      4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                                    Event Medal
      0       Basketball Men's Basketball   NaN
      1      Judo Men's Extra-Lightweight   NaN
      2           Football Men's Football   NaN
      3       Tug-Of-War Men's Tug-Of-War  Gold
      4  Speed Skating Women's 500 metres   NaN
```

```
[11]: #in read_csv you can specify your own index:
      pd.read_csv("D:\\Dataset_ash\\SQL Case Study\\athlete_events.csv",index_col =␣
       ↪'ID').tail()
```

```
[11]:                          Name Sex   Age  Height  Weight      Team  NOC  \
      ID
      135569             Andrzej ya   M  29.0   179.0    89.0  Poland-1  POL
      135570               Piotr ya   M  27.0   176.0    59.0    Poland  POL
```

```
135570            Piotr ya   M  27.0   176.0    59.0    Poland  POL
135571  Tomasz Ireneusz ya   M  30.0   185.0    96.0    Poland  POL
135571  Tomasz Ireneusz ya   M  34.0   185.0    96.0    Poland  POL

              Games  Year  Season            City        Sport  \
ID
135569  1976 Winter  1976  Winter       Innsbruck         Luge
135570  2014 Winter  2014  Winter           Sochi  Ski Jumping
135570  2014 Winter  2014  Winter           Sochi  Ski Jumping
135571  1998 Winter  1998  Winter          Nagano    Bobsleigh
135571  2002 Winter  2002  Winter  Salt Lake City    Bobsleigh

                                           Event Medal
ID
135569                Luge Mixed (Men)'s Doubles   NaN
135570  Ski Jumping Men's Large Hill, Individual   NaN
135570        Ski Jumping Men's Large Hill, Team   NaN
135571                      Bobsleigh Men's Four   NaN
135571                      Bobsleigh Men's Four   NaN
```

[12]:
```python
#shape attribute to check the dimension of dataframe:
df.shape
```

[12]:
```
(271116, 15)
```

if you want to see full dataframe:

- pd.set_option('display.max_rows' , 'number_of_rows')
- pd.set_option('display.max_columns', 'number_of_columns')

[13]:
```python
#head() command, which grabs the first five rows:
df.head()
```

[13]:
```
   ID                      Name Sex   Age  Height  Weight            Team  \
0   1                 A Dijiang   M  24.0   180.0    80.0           China
1   2                  A Lamusi   M  23.0   170.0    60.0           China
2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

   NOC        Games  Year  Season       City          Sport  \
0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
1  CHN  2012 Summer  2012  Summer     London           Judo
2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
3  DEN  1900 Summer  1900  Summer      Paris     Tug-Of-War
4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                            Event Medal
```

```
0        Basketball Men's Basketball   NaN
1       Judo Men's Extra-Lightweight   NaN
2            Football Men's Football   NaN
3        Tug-Of-War Men's Tug-Of-War  Gold
4    Speed Skating Women's 500 metres   NaN
```

Native accessors: In Python, we can access the property of an object by accessing it as an attribute.

[14]:
```
#method 1
df.City
```

[14]:
```
0               Barcelona
1                  London
2               Antwerpen
3                   Paris
4                 Calgary
                    …
271111          Innsbruck
271112              Sochi
271113              Sochi
271114             Nagano
271115     Salt Lake City
Name: City, Length: 271116, dtype: object
```

[15]:
```
#method 2
df['City']
```

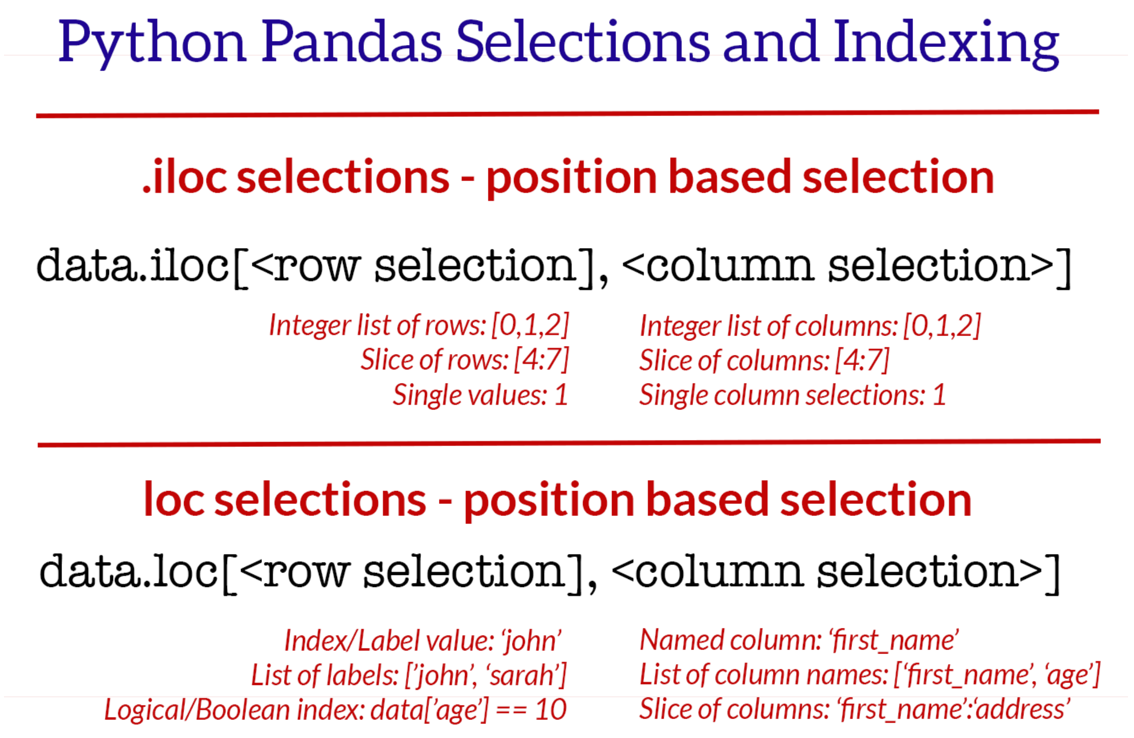
[15]:
```
0               Barcelona
1                  London
2               Antwerpen
3                   Paris
4                 Calgary
                    …
271111          Innsbruck
271112              Sochi
271113              Sochi
271114             Nagano
271115     Salt Lake City
Name: City, Length: 271116, dtype: object
```

Always use second method

Indexing in Pandas: The indexing operator and attribute selection are nice because they work just like they do in the rest of the Python ecosystem. As a novice, this makes them easy to pick up and use. However, pandas has its own accessor operators, loc and iloc.

[16]:
```python
#both loc and iloc is row first and column second:
from IPython import display
display.Image("C:\\Users\\thapa\\Downloads\\Pandas-selections-and-indexing.png")
```

[16]:



Python Pandas Selections and Indexing

.iloc selections - position based selection

data.iloc[<row selection>, <column selection>]

Integer list of rows: [0,1,2]
Slice of rows: [4:7]
Single values: 1

Integer list of columns: [0,1,2]
Slice of columns: [4:7]
Single column selections: 1

loc selections - position based selection

data.loc[<row selection>, <column selection>]

Index/Label value: 'john'
List of labels: ['john', 'sarah']
Logical/Boolean index: data['age'] == 10

Named column: 'first_name'
List of column names: ['first_name', 'age']
Slice of columns: 'first_name':'address'

Index based selection:

The first is index-based selection: selecting data based on its numerical position in the data.

[17]:
```python
#To select the first row of data in a DataFrame:
df.iloc[0]
```

[17]:
```
ID                                1
Name                      A Dijiang
Sex                               M
Age                              24
Height                          180
Weight                           80
Team                          China
NOC                             CHN
Games                   1992 Summer
Year                           1992
Season                       Summer
City                      Barcelona
```

```
Sport                      Basketball
Event     Basketball Men's Basketball
Medal                             NaN
Name: 0, dtype: object
```

[18]:
```
#methods of rows retrieving:
df.iloc[:,0]
```

[18]:
```
0              1
1              2
2              3
3              4
4              5
           …
271111    135569
271112    135570
271113    135570
271114    135571
271115    135571
Name: ID, Length: 271116, dtype: int64
```

[19]:
```
df.iloc[:3,0]
```

[19]:
```
0    1
1    2
2    3
Name: ID, dtype: int64
```

[20]:
```
df.iloc[1:3,0]
```

[20]:
```
1    2
2    3
Name: ID, dtype: int64
```

[21]:
```
df.iloc[[1,2,5],0] #selective rows:
```

[21]:
```
1    2
2    3
5    5
Name: ID, dtype: int64
```

[22]:
```
df.iloc[-5:]
```

[22]:
```
            ID                Name Sex   Age  Height  Weight     Team  NOC  \
271111  135569         Andrzej ya   M  29.0   179.0    89.0  Poland-1  POL
271112  135570           Piotr ya   M  27.0   176.0    59.0    Poland  POL
271113  135570           Piotr ya   M  27.0   176.0    59.0    Poland  POL
271114  135571  Tomasz Ireneusz ya   M  30.0   185.0    96.0    Poland  POL
```

```
271115  135571  Tomasz Ireneusz ya   M  34.0   185.0    96.0    Poland  POL

              Games  Year  Season            City        Sport  \
271111  1976 Winter  1976  Winter       Innsbruck         Luge
271112  2014 Winter  2014  Winter           Sochi  Ski Jumping
271113  2014 Winter  2014  Winter           Sochi  Ski Jumping
271114  1998 Winter  1998  Winter          Nagano    Bobsleigh
271115  2002 Winter  2002  Winter  Salt Lake City    Bobsleigh

                                           Event Medal
271111                Luge Mixed (Men)'s Doubles   NaN
271112  Ski Jumping Men's Large Hill, Individual   NaN
271113        Ski Jumping Men's Large Hill, Team   NaN
271114                      Bobsleigh Men's Four   NaN
271115                      Bobsleigh Men's Four   NaN
```

Label based selection:

In this paradigm, it's the data index value, not its position, which matters.

[23]:
```
df.loc[0,'City']
```

[23]:
```
'Barcelona'
```

[24]:
```
df.loc[:,['Season','City','Sport']].head() #all rows but selective columns,␣
↪it's way to create subtable of data:
```

[24]:
```
   Season       City          Sport
0  Summer  Barcelona     Basketball
1  Summer     London           Judo
2  Summer  Antwerpen       Football
3  Summer      Paris     Tug-Of-War
4  Winter    Calgary  Speed Skating
```

Manipulating the index: Label-based selection derives its power from the labels in the index. Critically, the index we use is not immutable. We can manipulate the index in any way we see fit.

The set_index() method can be used to do the job. Here is what happens when we set_index to the title field:

[25]:
```
df.set_index('City').head()
```

[25]:
```
           ID                      Name Sex   Age  Height  Weight  \
City
Barcelona   1                 A Dijiang   M  24.0   180.0    80.0
London      2                  A Lamusi   M  23.0   170.0    60.0
Antwerpen   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN
Paris       4      Edgar Lindenau Aabye   M  34.0     NaN     NaN
Calgary     5  Christine Jacoba Aaftink   F  21.0   185.0    82.0
```

```
                    Team  NOC        Games  Year  Season         Sport  \
City
Barcelona          China  CHN  1992 Summer  1992  Summer    Basketball
London             China  CHN  2012 Summer  2012  Summer          Judo
Antwerpen        Denmark  DEN  1920 Summer  1920  Summer      Football
Paris     Denmark/Sweden  DEN  1900 Summer  1900  Summer    Tug-Of-War
Calgary      Netherlands  NED  1988 Winter  1988  Winter  Speed Skating

                                  Event Medal
City
Barcelona       Basketball Men's Basketball   NaN
London         Judo Men's Extra-Lightweight   NaN
Antwerpen           Football Men's Football   NaN
Paris               Tug-Of-War Men's Tug-Of-War  Gold
Calgary    Speed Skating Women's 500 metres   NaN
```

Conditional Selection(filter):

[26]:
```
#filter by specific Medal:
filt = (df['Medal'] == 'Gold')
```

[27]:
```
df.loc[filt].head()
```

[27]:
```
    ID                     Name Sex   Age  Height  Weight            Team  \
3    4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
42  17  Paavo Johannes Aaltonen   M  28.0   175.0    64.0         Finland
44  17  Paavo Johannes Aaltonen   M  28.0   175.0    64.0         Finland
48  17  Paavo Johannes Aaltonen   M  28.0   175.0    64.0         Finland
60  20       Kjetil Andr Aamodt   M  20.0   176.0    85.0          Norway

    NOC        Games  Year  Season         City          Sport  \
3   DEN  1900 Summer  1900  Summer        Paris     Tug-Of-War
42  FIN  1948 Summer  1948  Summer       London     Gymnastics
44  FIN  1948 Summer  1948  Summer       London     Gymnastics
48  FIN  1948 Summer  1948  Summer       London     Gymnastics
60  NOR  1992 Winter  1992  Winter  Albertville  Alpine Skiing

                               Event Medal
3          Tug-Of-War Men's Tug-Of-War  Gold
42  Gymnastics Men's Team All-Around  Gold
44      Gymnastics Men's Horse Vault  Gold
48  Gymnastics Men's Pommelled Horse  Gold
60         Alpine Skiing Men's Super G  Gold
```

[28]:
```
#filter by Boolean operators:
filt = (df['Season'] == 'Summer') | (df['City'] == 'London')  # & = and , | =␣
↪or , ~ = negation
```

[29]:
```
df.loc[~filt].head()
```

[29]:
```
   ID                      Name Sex   Age  Height  Weight         Team  NOC  \
4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0  Netherlands  NED
5   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0  Netherlands  NED
6   5  Christine Jacoba Aaftink   F  25.0   185.0    82.0  Netherlands  NED
7   5  Christine Jacoba Aaftink   F  25.0   185.0    82.0  Netherlands  NED
8   5  Christine Jacoba Aaftink   F  27.0   185.0    82.0  Netherlands  NED

         Games  Year  Season         City          Sport  \
4  1988 Winter  1988  Winter      Calgary  Speed Skating
5  1988 Winter  1988  Winter      Calgary  Speed Skating
6  1992 Winter  1992  Winter  Albertville  Speed Skating
7  1992 Winter  1992  Winter  Albertville  Speed Skating
8  1994 Winter  1994  Winter  Lillehammer  Speed Skating

                                Event Medal
4    Speed Skating Women's 500 metres   NaN
5  Speed Skating Women's 1,000 metres   NaN
6    Speed Skating Women's 500 metres   NaN
7  Speed Skating Women's 1,000 metres   NaN
8    Speed Skating Women's 500 metres   NaN
```

[30]:
```
#creating filter by highest fare:
filt = (df['Height'] < df['Height'].median())
```

[31]:
```
df.loc[filt,['Age','Height','Weight','Team']].head()
```

[31]:
```
     Age  Height  Weight         Team
1   23.0   170.0    60.0        China
26  18.0   168.0     NaN  Netherlands
27  18.0   168.0     NaN  Netherlands
31  31.0   172.0    70.0      Finland
32  30.0   159.0    55.5      Finland
```

Pandas comes with a few built-in conditional selectors, two of which we will highlight here. + isin is lets you select data whose value "is in" a list of values.

[32]:
```
#filter by isin():
team_filt = ['China','India','Australia']
filt = df['Team'].isin(team_filt)
```

[33]:
```
df.loc[filt,['Age','Height','Weight','Team']].head()
```

[33]:
```
      Age  Height  Weight       Team
0    24.0   180.0    80.0      China
1    23.0   170.0    60.0      China
274  21.0   184.0    87.0  Australia
```

```
453  30.0   178.0    66.0  Australia
454  34.0   178.0    66.0  Australia
```

- **The second is isnull (and its companion notnull)**

```
[34]: filt = df.Age.notnull()
```

```
[35]: df.loc[filt].head()
```

```
[35]:    ID                      Name Sex   Age  Height  Weight            Team  \
      0   1                 A Dijiang   M  24.0   180.0    80.0           China
      1   2                  A Lamusi   M  23.0   170.0    60.0           China
      2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
      3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
      4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

        NOC        Games  Year  Season       City          Sport  \
      0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
      1  CHN  2012 Summer  2012  Summer     London           Judo
      2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
      3  DEN  1900 Summer  1900  Summer      Paris     Tug-Of-War
      4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                                    Event Medal
      0        Basketball Men's Basketball   NaN
      1       Judo Men's Extra-Lightweight   NaN
      2            Football Men's Football   NaN
      3        Tug-Of-War Men's Tug-Of-War  Gold
      4   Speed Skating Women's 500 metres   NaN
```

```
[36]: #filter by specific string contains:
      filt = df.Event.str.contains("Men's",na=False)
```

```
[37]: df.loc[filt].head()
```

```
[37]:     ID                  Name Sex   Age  Height  Weight           Team  NOC  \
      0    1             A Dijiang   M  24.0   180.0    80.0          China  CHN
      1    2              A Lamusi   M  23.0   170.0    60.0          China  CHN
      2    3   Gunnar Nielsen Aaby   M  24.0     NaN     NaN        Denmark  DEN
      3    4  Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden  DEN
      10   6        Per Knut Aaland   M  31.0   188.0    75.0  United States  USA

                Games  Year  Season         City                Sport  \
      0   1992 Summer  1992  Summer    Barcelona           Basketball
      1   2012 Summer  2012  Summer       London                 Judo
      2   1920 Summer  1920  Summer    Antwerpen             Football
      3   1900 Summer  1900  Summer        Paris           Tug-Of-War
      10  1992 Winter  1992  Winter  Albertville  Cross Country Skiing
```

```
                                       Event Medal
0                 Basketball Men's Basketball   NaN
1                Judo Men's Extra-Lightweight   NaN
2                     Football Men's Football   NaN
3                 Tug-Of-War Men's Tug-Of-War  Gold
10  Cross Country Skiing Men's 10 kilometres   NaN
```

Sorting:

```
[38]: df.sort_values(by = 'Age',ascending=True).head() #ascending order:
```

```
[38]:             ID                                          Name Sex   Age  \
      142882   71691                            Dimitrios Loundras   M  10.0
      252231  126307                                  Liana Vicens   F  11.0
      101378   51268                                Beatrice Hutiu   F  11.0
      140650   70616                                    Liu Luyang   F  11.0
      237141  118925  Megan Olwen Devenish Taylor (-Mandeville-Ellis)   F  11.0

              Height  Weight                           Team  NOC        Games  Year  \
      142882     NaN     NaN  Ethnikos Gymnastikos Syllogos  GRE  1896 Summer  1896
      252231   158.0    50.0                    Puerto Rico  PUR  1968 Summer  1968
      101378   151.0    38.0                        Romania  ROU  1968 Winter  1968
      140650     NaN     NaN                          China  CHN  1988 Winter  1988
      237141   157.0     NaN                  Great Britain  GBR  1932 Winter  1932

              Season         City          Sport  \
      142882  Summer       Athina     Gymnastics
      252231  Summer  Mexico City       Swimming
      101378  Winter     Grenoble  Figure Skating
      140650  Winter      Calgary  Figure Skating
      237141  Winter  Lake Placid  Figure Skating

                                               Event   Medal
      142882     Gymnastics Men's Parallel Bars, Teams  Bronze
      252231  Swimming Women's 200 metres Breaststroke     NaN
      101378             Figure Skating Women's Singles     NaN
      140650           Figure Skating Mixed Ice Dancing     NaN
      237141             Figure Skating Women's Singles     NaN
```

```
[39]: #descending order sort:
      df.sort_values(by = 'Age' , ascending = False).head()
```

```
[39]:             ID                           Name Sex   Age  Height  Weight  \
      257054  128719        John Quincy Adams Ward   M  97.0     NaN     NaN
      98118    49663                  Winslow Homer   M  96.0     NaN     NaN
      60863    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN
      60861    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN
```

```
60862    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN

                 Team  NOC        Games  Year  Season         City  \
257054  United States  USA  1928 Summer  1928  Summer    Amsterdam
98118   United States  USA  1932 Summer  1932  Summer  Los Angeles
60863   United States  USA  1932 Summer  1932  Summer  Los Angeles
60861   United States  USA  1932 Summer  1932  Summer  Los Angeles
60862   United States  USA  1932 Summer  1932  Summer  Los Angeles

                   Sport                                         Event Medal
257054  Art Competitions     Art Competitions Mixed Sculpturing, Statues   NaN
98118   Art Competitions  Art Competitions Mixed Painting, Unknown Event   NaN
60863   Art Competitions  Art Competitions Mixed Painting, Unknown Event   NaN
60861   Art Competitions  Art Competitions Mixed Painting, Unknown Event   NaN
60862   Art Competitions  Art Competitions Mixed Painting, Unknown Event   NaN
```

[40]:
```
#can sort on multiple column:
df.sort_values(['Height','Weight']).head()
```

[40]:
```
          ID             Name Sex   Age  Height  Weight    Team  NOC  \
29333  15150  Rosario Briones   F  15.0   127.0    42.0  Mexico  MEX
29334  15150  Rosario Briones   F  15.0   127.0    42.0  Mexico  MEX
29335  15150  Rosario Briones   F  15.0   127.0    42.0  Mexico  MEX
29336  15150  Rosario Briones   F  15.0   127.0    42.0  Mexico  MEX
29337  15150  Rosario Briones   F  15.0   127.0    42.0  Mexico  MEX

             Games  Year  Season         City       Sport  \
29333  1968 Summer  1968  Summer  Mexico City  Gymnastics
29334  1968 Summer  1968  Summer  Mexico City  Gymnastics
29335  1968 Summer  1968  Summer  Mexico City  Gymnastics
29336  1968 Summer  1968  Summer  Mexico City  Gymnastics
29337  1968 Summer  1968  Summer  Mexico City  Gymnastics

                                      Event Medal
29333  Gymnastics Women's Individual All-Around   NaN
29334        Gymnastics Women's Team All-Around   NaN
29335         Gymnastics Women's Floor Exercise   NaN
29336            Gymnastics Women's Horse Vault   NaN
29337            Gymnastics Women's Uneven Bars   NaN
```

[41]:
```
#sort asc and desc different by parsing boolean
df.sort_values(by = ['Height','Weight'],ascending=[False,True]).head()
```

[41]:
```
            ID                Name Sex   Age  Height  Weight  \
265040  132627            Yao Ming   M  20.0   226.0   141.0
265041  132627            Yao Ming   M  23.0   226.0   141.0
265042  132627            Yao Ming   M  27.0   226.0   141.0
```

```
32376     16639   Tommy Loren Burleson   M  20.0   223.0   102.0
207373   104059  Arvydas Romas Sabonis   M  23.0   223.0   122.0

                  Team  NOC        Games  Year  Season     City       Sport  \
265040           China  CHN  2000 Summer  2000  Summer   Sydney  Basketball
265041           China  CHN  2004 Summer  2004  Summer   Athina  Basketball
265042           China  CHN  2008 Summer  2008  Summer  Beijing  Basketball
32376    United States  USA  1972 Summer  1972  Summer   Munich  Basketball
207373    Soviet Union  URS  1988 Summer  1988  Summer    Seoul  Basketball

                             Event   Medal
265040  Basketball Men's Basketball     NaN
265041  Basketball Men's Basketball     NaN
265042  Basketball Men's Basketball     NaN
32376   Basketball Men's Basketball  Silver
207373  Basketball Men's Basketball    Gold
```

[42]:
```
#sort_index:
df.sort_index().head()
```

[42]:
```
   ID                      Name Sex   Age  Height  Weight            Team  \
0   1                 A Dijiang   M  24.0   180.0    80.0           China
1   2                  A Lamusi   M  23.0   170.0    60.0           China
2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

   NOC        Games  Year  Season       City          Sport  \
0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
1  CHN  2012 Summer  2012  Summer     London           Judo
2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
3  DEN  1900 Summer  1900  Summer      Paris    Tug-Of-War
4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                             Event Medal
0      Basketball Men's Basketball   NaN
1     Judo Men's Extra-Lightweight   NaN
2          Football Men's Football   NaN
3      Tug-Of-War Men's Tug-Of-War  Gold
4  Speed Skating Women's 500 metres   NaN
```

[43]:
```
#nsmallest value
df.nsmallest(5,'Age')
```

[43]:
```
            ID                                 Name Sex   Age  Height  \
142882   71691                   Dimitrios Loundras   M  10.0     NaN
43468    22411           Magdalena Cecilia Colledge   F  11.0   152.0
```

```
73461   37333           Carlos Bienvenido Front Barrera   M  11.0     NaN
79024   40129                        Luigina Giavotti   F  11.0     NaN
94058   47618  Sonja Henie (-Topping, -Gardiner, -Onstad)   F  11.0   155.0

        Weight                        Team  NOC        Games  Year  Season  \
142882     NaN  Ethnikos Gymnastikos Syllogos  GRE  1896 Summer  1896  Summer
43468      NaN                 Great Britain  GBR  1932 Winter  1932  Winter
73461      NaN                         Spain  ESP  1992 Summer  1992  Summer
79024      NaN                         Italy  ITA  1928 Summer  1928  Summer
94058     45.0                        Norway  NOR  1924 Winter  1924  Winter

              City           Sport                                 Event  \
142882      Athina      Gymnastics  Gymnastics Men's Parallel Bars, Teams
43468   Lake Placid  Figure Skating         Figure Skating Women's Singles
73461     Barcelona          Rowing              Rowing Men's Coxed Eights
79024     Amsterdam      Gymnastics     Gymnastics Women's Team All-Around
94058      Chamonix  Figure Skating         Figure Skating Women's Singles

         Medal
142882  Bronze
43468      NaN
73461      NaN
79024   Silver
94058      NaN
```

```
[44]: #nlargest value:
      df.nlargest(7,'Age')
```

```
[44]:            ID                       Name Sex   Age  Height  Weight  \
257054  128719       John Quincy Adams Ward   M  97.0     NaN     NaN
98118    49663                Winslow Homer   M  96.0     NaN     NaN
60861    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN
60862    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN
60863    31173  Thomas Cowperthwait Eakins   M  88.0     NaN     NaN
9371      5146        George Denholm Armour   M  84.0     NaN     NaN
154855   77710          Robert Tait McKenzie   M  81.0     NaN     NaN

                 Team  NOC        Games  Year  Season         City  \
257054  United States  USA  1928 Summer  1928  Summer    Amsterdam
98118   United States  USA  1932 Summer  1932  Summer  Los Angeles
60861   United States  USA  1932 Summer  1932  Summer  Los Angeles
60862   United States  USA  1932 Summer  1932  Summer  Los Angeles
60863   United States  USA  1932 Summer  1932  Summer  Los Angeles
9371    Great Britain  GBR  1948 Summer  1948  Summer       London
154855         Canada  CAN  1948 Summer  1948  Summer       London

                Sport                                     Event  \
257054  Art Competitions        Art Competitions Mixed Sculpturing, Statues
98118   Art Competitions     Art Competitions Mixed Painting, Unknown Event
60861   Art Competitions     Art Competitions Mixed Painting, Unknown Event
60862   Art Competitions     Art Competitions Mixed Painting, Unknown Event
60863   Art Competitions     Art Competitions Mixed Painting, Unknown Event
9371    Art Competitions     Art Competitions Mixed Painting, Unknown Event
154855  Art Competitions  Art Competitions Mixed Sculpturing, Unknown Event

       Medal
257054   NaN
98118    NaN
60861    NaN
60862    NaN
60863    NaN
9371     NaN
154855   NaN
```

Summary and Aggregate Function: Pandas provides many simple "summary functions" which restructure the data in some useful way.

[45]:
```
df.describe()
```

[45]:
```
                  ID            Age         Height         Weight  \
count  271116.000000  261642.000000  210945.000000  208241.000000
mean    68248.954396      25.556898     175.338970      70.702393
std     39022.286345       6.393561      10.518462      14.348020
min         1.000000      10.000000     127.000000      25.000000
25%     34643.000000      21.000000     168.000000      60.000000
50%     68205.000000      24.000000     175.000000      70.000000
75%    102097.250000      28.000000     183.000000      79.000000
max    135571.000000      97.000000     226.000000     214.000000

                Year
count  271116.000000
mean     1978.378480
std        29.877632
min      1896.000000
25%      1960.000000
50%      1988.000000
75%      2002.000000
max      2016.000000
```

[46]:
```
df['Age'].describe()
```

[46]:
```
count    261642.000000
mean         25.556898
std           6.393561
```

```
min         10.000000
25%         21.000000
50%         24.000000
75%         28.000000
max         97.000000
Name: Age, dtype: float64
```

```
[47]: df.count()
```

```
[47]: ID        271116
      Name      271116
      Sex       271116
      Age       261642
      Height    210945
      Weight    208241
      Team      271116
      NOC       271116
      Games     271116
      Year      271116
      Season    271116
      City      271116
      Sport     271116
      Event     271116
      Medal      39783
      dtype: int64
```

```
[48]: df['Age'].count()
```

```
[48]: 261642
```

```
[49]: df['NOC'].value_counts()
```

```
[49]: USA    18853
      FRA    12758
      GBR    12256
      ITA    10715
      GER     9830
             …
      YMD        5
      SSD        3
      NBO        2
      UNK        2
      NFL        1
      Name: NOC, Length: 230, dtype: int64
```

```
[50]: df['NOC'].value_counts(normalize=True)  #it gives percentage of data
```

[50]:
```
USA    0.069539
FRA    0.047057
GBR    0.045206
ITA    0.039522
GER    0.036258
         …
YMD    0.000018
SSD    0.000011
NBO    0.000007
UNK    0.000007
NFL    0.000004
Name: NOC, Length: 230, dtype: float64
```

[51]:
```
df['NOC'].unique()
```

[51]:
```
array(['CHN', 'DEN', 'NED', 'USA', 'FIN', 'NOR', 'ROU', 'EST', 'FRA',
       'MAR', 'ESP', 'EGY', 'IRI', 'BUL', 'ITA', 'CHA', 'AZE', 'SUD',
       'RUS', 'ARG', 'CUB', 'BLR', 'GRE', 'CMR', 'TUR', 'CHI', 'MEX',
       'URS', 'NCA', 'HUN', 'NGR', 'ALG', 'KUW', 'BRN', 'PAK', 'IRQ',
       'UAR', 'LIB', 'QAT', 'MAS', 'GER', 'CAN', 'IRL', 'AUS', 'RSA',
       'ERI', 'TAN', 'JOR', 'TUN', 'LBA', 'BEL', 'DJI', 'PLE', 'COM',
       'KAZ', 'BRU', 'IND', 'KSA', 'SYR', 'MDV', 'ETH', 'UAE', 'YAR',
       'INA', 'PHI', 'SGP', 'UZB', 'KGZ', 'TJK', 'EUN', 'JPN', 'CGO',
       'SUI', 'BRA', 'FRG', 'GDR', 'MON', 'ISR', 'URU', 'SWE', 'ISV',
       'SRI', 'ARM', 'CIV', 'KEN', 'BEN', 'UKR', 'GBR', 'GHA', 'SOM',
       'LAT', 'NIG', 'MLI', 'AFG', 'POL', 'CRC', 'PAN', 'GEO', 'SLO',
       'CRO', 'GUY', 'NZL', 'POR', 'PAR', 'ANG', 'VEN', 'COL', 'BAN',
       'PER', 'ESA', 'PUR', 'UGA', 'HON', 'ECU', 'TKM', 'MRI', 'SEY',
       'TCH', 'LUX', 'MTN', 'CZE', 'SKN', 'TTO', 'DOM', 'VIN', 'JAM',
       'LBR', 'SUR', 'NEP', 'MGL', 'AUT', 'PLW', 'LTU', 'TOG', 'NAM',
       'AHO', 'ISL', 'ASA', 'SAM', 'RWA', 'DMA', 'HAI', 'MLT', 'CYP',
       'GUI', 'BIZ', 'YMD', 'KOR', 'THA', 'BER', 'ANZ', 'SCG', 'SLE',
       'PNG', 'YEM', 'IOA', 'OMA', 'FIJ', 'VAN', 'MDA', 'YUG', 'BAH',
       'GUA', 'SRB', 'IVB', 'MOZ', 'CAF', 'MAD', 'MAL', 'BIH', 'GUM',
       'CAY', 'SVK', 'BAR', 'GBS', 'TLS', 'COD', 'GAB', 'SMR', 'LAO',
       'BOT', 'ROT', 'CAM', 'PRK', 'SOL', 'SEN', 'CPV', 'CRT', 'GEQ',
       'BOL', 'SAA', 'AND', 'ANT', 'ZIM', 'GRN', 'HKG', 'LCA', 'FSM',
       'MYA', 'MAW', 'ZAM', 'RHO', 'TPE', 'STP', 'MKD', 'BOH', 'TGA',
       'LIE', 'MNE', 'GAM', 'COK', 'ALB', 'WIF', 'SWZ', 'BUR', 'NBO',
       'BDI', 'ARU', 'NRU', 'VNM', 'VIE', 'BHU', 'MHL', 'KIR', 'UNK',
       'TUV', 'NFL', 'KOS', 'SSD', 'LES'], dtype=object)
```

Updating rows and columns:

[52]:
```
#for this we will use this code snippet:
people = { 'first name' : ['Keshav','Ashish','Nadeem'],
           'last name' : ['Choudhary','Thapa','Khan'],
           'email' : ['choudharykeshav@gmail.com','thapa.ashishkumar3@gmail.
 ↪com','khanNadeem@gmail.com']
}
```

```
[53]: df_2 = pd.DataFrame(people)
      df_2
```

```
[53]:   first name  last name                         email
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2     Nadeem       Khan          khanNadeem@gmail.com
```

Update columns:

```
[54]: df_2.columns = [x.upper() for x in df_2.columns]
```

```
[55]: df_2
```

```
[55]:   FIRST NAME  LAST NAME                         EMAIL
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2     Nadeem       Khan          khanNadeem@gmail.com
```

```
[56]: df_2.columns = df_2.columns.str.replace(' ', '_')
      df_2
```

```
[56]:   FIRST_NAME  LAST_NAME                         EMAIL
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2     Nadeem       Khan          khanNadeem@gmail.com
```

Renaming

- it lets you change index names and/or column names.
- lets you rename index or column values by specifying a index or column keyword parameter, respectively. It supports a variety of input formats, but usually a Python dictionary is the most convenient.

```
[57]: df_2.rename(columns={'FIRST_NAME':'first_name','LAST_NAME':'last_name','EMAIL':
       ↪'email'},inplace=True)
      df_2
```

```
[57]:   first_name  last_name                         email
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2     Nadeem       Khan          khanNadeem@gmail.com
```

```
[58]: df_2.rename(index={0: 'firstEntry', 1: 'secondEntry',2:'thirdEntry'})
```

```
[58]:             first_name  last_name                        email
      firstEntry      Keshav  Choudhary     choudharykeshav@gmail.com
      secondEntry     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      thirdEntry      Nadeem       Khan          khanNadeem@gmail.com
```

```
[59]: #this is rare: (optional)
      df_2.rename_axis("Entry", axis='rows').rename_axis("Col", axis='columns')
```

```
[59]: Col   first_name  last_name                        email
      Entry
      0         Keshav  Choudhary     choudharykeshav@gmail.com
      1         Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2         Nadeem       Khan          khanNadeem@gmail.com
```

```
[60]: #whole column update:
      df_2['email'] = df_2['email'].str.upper()
      df_2
```

```
[60]:   first_name  last_name                        email
      0     Keshav  Choudhary     CHOUDHARYKESHAV@GMAIL.COM
      1     Ashish      Thapa  THAPA.ASHISHKUMAR3@GMAIL.COM
      2     Nadeem       Khan          KHANNADEEM@GMAIL.COM
```

```
[61]: df_2['email'] = df_2['email'].str.lower()
      df_2
```

```
[61]:   first_name  last_name                        email
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2     Nadeem       Khan          khannadeem@gmail.com
```

Update rows

```
[62]: #updating whole row
      df_2.loc[2] = ['John','Smith','johnsmith@gmail.com']
      df_2
```

```
[62]:   first_name  last_name                        email
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
      2       John      Smith           johnsmith@gmail.com
```

```
[63]: #updating specific data point
      df_2.iloc[2 , 0] = 'Aakash'
      df_2
```

```
[63]:   first_name  last_name                        email
      0     Keshav  Choudhary     choudharykeshav@gmail.com
      1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
```

```
2     Aakash      Smith           johnsmith@gmail.com
```

[64]:
```
df_2.loc[2,['last_name','email']] = ['thapa','thapa.aakashkumar3@gmail.com']
df_2
```

[64]:
```
  first_name  last_name                         email
0     Keshav  Choudhary     choudharykeshav@gmail.com
1     Ashish      Thapa  thapa.ashishkumar3@gmail.com
2     Aakash      thapa  thapa.aakashkumar3@gmail.com
```

The four method to update rows + apply + map + applymap + replace

[65]:
```
#apply can be applicable to both series and dataframes: it help to update and␣
↪apply function.
df_2['email'].apply(len)
```

[65]:
```
0    25
1    28
2    28
Name: email, dtype: int64
```

[66]:
```
df_2['first_name'].apply(lambda x: x.lower())
```

[66]:
```
0    keshav
1    ashish
2    aakash
Name: first_name, dtype: object
```

[67]:
```
#applymap is only applicable to dataframe.
df_2.applymap(str.lower)
```

[67]:
```
  first_name  last_name                         email
0     keshav  choudhary     choudharykeshav@gmail.com
1     ashish      thapa  thapa.ashishkumar3@gmail.com
2     aakash      thapa  thapa.aakashkumar3@gmail.com
```

[68]:
```
#map only works in series:
df_2['first_name'].map({'Ashish': 'nadeem' ,'Aakash' :'kashif' })
```

[68]:
```
0       NaN
1    nadeem
2    kashif
Name: first_name, dtype: object
```

[69]:
```
#use replace if you don't want nan:
df_2['first_name'].replace({'Ashish': 'nadeem' ,'Aakash' :'kashif' })
```

[69]:
```
0    Keshav
1    nadeem
2    kashif
Name: first_name, dtype: object
```

Add/Remove Rows and Columns:

[70]:
```
#add a column:
df_2['full_name'] = df_2['first_name'] + ' ' + df_2['last_name']
df_2
```

[70]:
```
  first_name  last_name                          email          full_name
0     Keshav  Choudhary     choudharykeshav@gmail.com   Keshav Choudhary
1     Ashish      Thapa  thapa.ashishkumar3@gmail.com       Ashish Thapa
2     Aakash      thapa  thapa.aakashkumar3@gmail.com       Aakash thapa
```

[71]:
```
#remove columns:
df_2.drop(columns = ['first_name','last_name'],inplace = True)
```

[72]:
```
df_2[['first','last']] = df_2['full_name'].str.split(' ',expand =True)
df_2
```

[72]:
```
                          email          full_name   first       last
0     choudharykeshav@gmail.com   Keshav Choudhary  Keshav  Choudhary
1  thapa.ashishkumar3@gmail.com       Ashish Thapa  Ashish      Thapa
2  thapa.aakashkumar3@gmail.com       Aakash thapa  Aakash      thapa
```

[73]:
```
df_2 = df_2.loc[:,['first','last','full_name','email']]
```

[74]:
```
#add a row:
x = {'first' : 'Teshav', 'last': 'Ahoudhary','full_name' : 'Teshav_Ahoudhary' ,␣
 ↪'email' : ['xyz@gmail.com'] }
```

[75]:
```
df_2 = df_2.append(x,ignore_index=True)
df_2
```

[75]:
```
    first       last         full_name                         email
0  Keshav  Choudhary  Keshav Choudhary     choudharykeshav@gmail.com
1  Ashish      Thapa      Ashish Thapa  thapa.ashishkumar3@gmail.com
2  Aakash      thapa      Aakash thapa  thapa.aakashkumar3@gmail.com
3  Teshav  Ahoudhary  Teshav_Ahoudhary               [xyz@gmail.com]
```

[76]:
```
#remove a row:
df_2.drop(index = 3)
```

[76]:
```
    first       last         full_name                         email
0  Keshav  Choudhary  Keshav Choudhary     choudharykeshav@gmail.com
1  Ashish      Thapa      Ashish Thapa  thapa.ashishkumar3@gmail.com
2  Aakash      thapa      Aakash thapa  thapa.aakashkumar3@gmail.com
```

```
[77]: #dropping with condition:
      drop_filt = df_2['last'] == 'thapa'
```

```
[78]: df_2.drop(index= df_2[drop_filt].index)
```

```
[78]:     first       last         full_name                          email
      0  Keshav  Choudhary  Keshav Choudhary     choudharykeshav@gmail.com
      1  Ashish      Thapa      Ashish Thapa  thapa.ashishkumar3@gmail.com
      3  Teshav  Ahoudhary  Teshav_Ahoudhary                [xyz@gmail.com]
```

Group by:

```
[79]: embarked_grp = df.groupby(['Embarked'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-79-8a5541432bdc> in <module>
----> 1 embarked_grp = df.groupby(['Embarked'])

~\anaconda3\lib\site-packages\pandas\core\frame.py in groupby(self, by, axis,␣
 ↪level, as_index, sort, group_keys, squeeze, observed, dropna)
   6509         axis = self._get_axis_number(axis)
   6510
-> 6511         return DataFrameGroupBy(

   6512             obj=self,
   6513             keys=by,

~\anaconda3\lib\site-packages\pandas\core\groupby\groupby.py in __init__(self,␣
 ↪obj, keys, axis, level, grouper, exclusions, selection, as_index, sort,␣
 ↪group_keys, squeeze, observed, mutated, dropna)
    523             from pandas.core.groupby.grouper import get_grouper
    524
--> 525             grouper, exclusions, obj = get_grouper(

    526                 obj,
    527                 keys,

~\anaconda3\lib\site-packages\pandas\core\groupby\grouper.py in get_grouper(obj ␣
 ↪key, axis, level, sort, observed, mutated, validate, dropna)
    779                 in_axis, name, level, gpr = False, None, gpr, None
    780             else:
--> 781                 raise KeyError(gpr)
    782         elif isinstance(gpr, Grouper) and gpr.key is not None:
    783             # Add key to exclusions

KeyError: 'Embarked'
```

```
[ ]: embarked_grp['Embarked'].count()
```

```
[ ]: embarked_grp.get_group('C') #only c embarked
```

Above method can also be done by filter

```
[ ]: #group by embarked and in specific selective columns value_counts()
     embarked_grp['Cabin'].value_counts()
```

```
[ ]: embarked_grp['Cabin'].value_counts(normalize = True).loc['S'].head()  #specific␣
     ↪group data
```

```
[ ]: #group by aggregate:
     embarked_grp['Fare'].mean()
```

```
[ ]: #applying more aggregate method
     embarked_grp['Fare'].agg(['mean','median'])
```

Advance group by method:

In group by df i am selecting specific column and applying a function which gives me sum of specific string within group:

country_uses_python = country_grp['LanguageWorkedWith'].apply(lambda x : x.str.contains('Python').sum())

country_uses_python

multi-index: - A multi-index differs from a regular index in that it has multiple levels.

survey_public.groupby(['Country','SocialMedia']).Employment.agg([len])

Handling missing Data:

```
[ ]: #code snippet:
     ppl = {
         'first' : ['corey','jane','john','chris',np.nan,None,'NA'],
         'last' : ['schafer','doe','doe','schafer',np.nan,np.nan,'Missing'],
         'email' : ['coreymscha@gmail.com','JaneDoe@email.com','Johndoe@email.
     ↪com',None,np.nan,'anonympus@gmail.com','NA'],
         'age' : ['33','55','63','36',None,None,'Missing']
     }
```

```
[ ]: datafram = pd.DataFrame(ppl)

     datafram.replace(['NA','Missing'],np.nan,inplace= True)
     datafram
```

```
[ ]: #dropping all na values:
     datafram.dropna()
```

[ ]:
```
#any deletes all the row which having Nan values.
#all deletes those rows only which have all rows missing value.
#axis = 'index' always us all.
#axis  = 'columns' use all
datafram.dropna(axis = 'index' , how = 'all')
```

[ ]:
```
#dropping for specific column:
datafram.dropna(how = 'any', subset = ['email'])
```

[ ]:
```
datafram.isnull()
```

casting:

[ ]:
```
datafram['age'].dtype
```

[ ]:
```
#nan values can't be converted into int:
#nan is float

datafram['age'] = datafram['age'].astype('float')
datafram['age'].dtype
```

Fill na:

[ ]:
```
#filling with scaler data:
datafram.fillna(0)
```

[ ]:
```
#filling with specific column using dict:
datafram.fillna({'first':'Missing','age':0})
```

[ ]:
```
#filling with forward value:
datafram.fillna(method = 'ffill')
```

[ ]:
```
#filling with backward fill:
datafram.fillna(method = 'bfill')
```

Interpolate

Pandas interpolate() function is basically used to fill nan values in dataframes or series. + very powerful function + uses various technique.

[ ]:
```
#only in numeric value:
datafram.interpolate(method = 'linear')
```

[ ]:
```
#fill with nearest
datafram.interpolate(method = 'nearest')
```

fill with time

method == time:

if column time is not in data time format it will throw an error.

datafram.interpolate(method= ‘time’)

fill with polynomial:

datafram.interpolate(method = ‘polynomial’,order = 2)

```
[ ]: datafram.interpolate(limit_direction = 'both')
```

1.1.3 Merge, Concat and Join

Merge

Pandas merge connects columns or index in dataframe based on one or more keys.

```
[80]: df2 = pd.read_csv("D:\\Dataset_ash\\SQL Case Study\\noc_regions.csv")
      df2.head()
```

```
[80]:   NOC       region                 notes
      0 AFG  Afghanistan                   NaN
      1 AHO      Curacao  Netherlands Antilles
      2 ALB      Albania                   NaN
      3 ALG      Algeria                   NaN
      4 AND      Andorra                   NaN
```

```
[82]: # 'on' should be same on both the dataframe.
      # if NOC is not common then only upto same part it gonna print.
      pd.merge(df,df2,on = 'NOC',how = 'inner').head()   #left join, right join,␣
       ↪inner and outer.
```

```
[82]:     ID           Name Sex   Age  Height  Weight   Team  NOC        Games  \
      0     1       A Dijiang   M  24.0   180.0    80.0  China  CHN  1992 Summer
      1     2        A Lamusi   M  23.0   170.0    60.0  China  CHN  2012 Summer
      2   602  Abudoureheman   M  22.0   182.0    75.0  China  CHN  2000 Summer
      3  1463       Ai Linuer   M  25.0   160.0    62.0  China  CHN  2004 Summer
      4  1464       Ai Yanhan   F  14.0   168.0    54.0  China  CHN  2016 Summer

         Year  Season            City       Sport  \
      0  1992  Summer       Barcelona  Basketball
      1  2012  Summer          London        Judo
      2  2000  Summer          Sydney      Boxing
      3  2004  Summer          Athina   Wrestling
      4  2016  Summer  Rio de Janeiro    Swimming

                                         Event Medal region notes
      0             Basketball Men's Basketball   NaN  China   NaN
      1            Judo Men's Extra-Lightweight   NaN  China   NaN
      2               Boxing Men's Middleweight   NaN  China   NaN
      3  Wrestling Men's Lightweight, Greco-Roman   NaN  China   NaN
      4     Swimming Women's 200 metres Freestyle   NaN  China   NaN
```

CONCAT

Pandas provides various facilities for easily combining together series, DataFrame and Panel objects

```
[88]: pd.concat([df,df2],axis =1,join = 'inner').head()
```

```
[88]:    ID                      Name Sex   Age  Height  Weight            Team  \
      0   1                 A Dijiang   M  24.0   180.0    80.0           China
      1   2                  A Lamusi   M  23.0   170.0    60.0           China
      2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
      3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
      4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands

         NOC        Games  Year  Season       City          Sport  \
      0  CHN  1992 Summer  1992  Summer  Barcelona     Basketball
      1  CHN  2012 Summer  2012  Summer     London           Judo
      2  DEN  1920 Summer  1920  Summer  Antwerpen       Football
      3  DEN  1900 Summer  1900  Summer      Paris     Tug-Of-War
      4  NED  1988 Winter  1988  Winter    Calgary  Speed Skating

                                    Event Medal  NOC       region  \
      0       Basketball Men's Basketball   NaN  AFG  Afghanistan
      1      Judo Men's Extra-Lightweight   NaN  AHO      Curacao
      2           Football Men's Football   NaN  ALB      Albania
      3       Tug-Of-War Men's Tug-Of-War  Gold  ALG      Algeria
      4  Speed Skating Women's 500 metres   NaN  AND      Andorra

                        notes
      0                   NaN
      1  Netherlands Antilles
      2                   NaN
      3                   NaN
      4                   NaN
```

JOIN

DataFrame join is a convenient method for combining the columns of two potentially differently indexed.

```
[94]: #if dataframe has same column then we have to provide a suffix.
      df.join(df2,lsuffix= '-1').head()
```

```
[94]:    ID                      Name Sex   Age  Height  Weight            Team  \
      0   1                 A Dijiang   M  24.0   180.0    80.0           China
      1   2                  A Lamusi   M  23.0   170.0    60.0           China
      2   3       Gunnar Nielsen Aaby   M  24.0     NaN     NaN         Denmark
      3   4      Edgar Lindenau Aabye   M  34.0     NaN     NaN  Denmark/Sweden
      4   5  Christine Jacoba Aaftink   F  21.0   185.0    82.0     Netherlands
```

```
  NOC-1         Games  Year  Season       City          Sport  \
0   CHN   1992 Summer  1992  Summer  Barcelona     Basketball
1   CHN   2012 Summer  2012  Summer     London           Judo
2   DEN   1920 Summer  1920  Summer  Antwerpen       Football
3   DEN   1900 Summer  1900  Summer      Paris    Tug-Of-War
4   NED   1988 Winter  1988  Winter    Calgary  Speed Skating

                             Event Medal  NOC       region  \
0       Basketball Men's Basketball   NaN  AFG  Afghanistan
1      Judo Men's Extra-Lightweight   NaN  AHO      Curacao
2           Football Men's Football   NaN  ALB      Albania
3       Tug-Of-War Men's Tug-Of-War  Gold  ALG      Algeria
4  Speed Skating Women's 500 metres   NaN  AND      Andorra

                  notes
0                   NaN
1  Netherlands Antilles
2                   NaN
3                   NaN
4                   NaN
```

Pandas Pivot Table: The levels in the pivot table will be stored in multiindex objects(hierarchical indexes) on the index and columns of the result DataFrame

[97]:
```
df.pivot_table(index = 'NOC').head()
```

[97]:
```
           Age      Height            ID     Weight         Year
NOC
AFG  23.538462  170.592593  70778.373016  65.901639  1966.031746
AHO  26.589744  177.294118  56438.518987  76.176471  1980.911392
ALB  25.342857  173.000000  76332.300000  71.491803  2002.257143
ALG  24.370642  174.702869  49394.666062  68.693252  1997.851180
AND  23.065089  173.703704  64885.218935  70.644444  1997.029586
```

[100]:
```
df.pivot_table(index = 'NOC',columns = 'Year',aggfunc='count',fill_value = 0).
 ↪head()
```

[100]:
```
      Age                                            … Weight          \
Year 1896 1900 1904 1906 1908 1912 1920 1924 1928 1932  …   1998 2000 2002
NOC                                                     …
AFG     0    0    0    0    0    0    0    0    0    0  …      0    0    0
AHO     0    0    0    0    0    0    0    0    0    0  …      0    8    0
ALB     0    0    0    0    0    0    0    0    0    0  …      0    5    0
ALG     0    0    0    0    0    0    0    0    0    0  …      0   51    0
AND     0    0    0    0    0    0    0    0    0    0  …      7    5    5
```

```
Year  2004  2006  2008  2010  2012  2014  2016
NOC
AFG      4     0     4     0     6     0     3
AHO      3     0     4     0     0     0     0
ALB      7     3    12     2    10     2     6
ALG     69     3    56     1    35     0    74
AND      6    10     5    20     5    12     4

[5 rows x 455 columns]
```

unpivot: A dataframe turn it from a wide format (many columns) to a long format (few columns but many rows)

Melt function Pandas melt function is used to transform or reshape data.

[102]:
```
pd.melt(df, id_vars = ['Medal'])
```

[102]:
```
        Medal variable                                       value
0         NaN       ID                                           1
1         NaN       ID                                           2
2         NaN       ID                                           3
3        Gold       ID                                           4
4         NaN       ID                                           5
…         …       …                                            …
3795619   NaN    Event                 Luge Mixed (Men)'s Doubles
3795620   NaN    Event  Ski Jumping Men's Large Hill, Individual
3795621   NaN    Event        Ski Jumping Men's Large Hill, Team
3795622   NaN    Event                       Bobsleigh Men's Four
3795623   NaN    Event                       Bobsleigh Men's Four

[3795624 rows x 3 columns]
```

[ ]: