Pandas_and_Alternatives

July 19, 2024

1 Pandas and Alternatives

1. Import pandas aliased as ‘pd’ and numpy aliased as ‘np’

[1]:

```
import pandas as pd
```

2. Create a DataFrame named ‘df’ by reading the file ‘USCG.Search.Rescue.Stats.csv’ using the pandas read_csv method.

[2]:

```
df = pd.read_csv('USCG.Search.Rescue.Stats.csv')
```

3. View the top 5 rows of data using the DataFrame `head()` method.

[3]:

```
df.head()
```

[3]:

```
      Fiscal Year  Cases  Responses  Sorties  Lives Saved  \
1964          NaN  41525        NaN     2932          NaN
1965          NaN  38586        NaN     1984          NaN
1966          NaN  43366        NaN     2629          NaN
1967          NaN  42225        NaN     3028          NaN
1968          NaN  46922        NaN     2434          NaN

      Lives Lost After CG Notification  Lives Lost Before CG Notification  \
1964                               NaN                                NaN
1965                               NaN                                NaN
1966                               NaN                                NaN
1967                               NaN                                NaN
1968                               NaN                                NaN

      Total  Lives Unaccounted For
1964    NaN                    NaN
1965    NaN                    NaN
1966    NaN                    NaN
1967    NaN                    NaN
1968    NaN                    NaN
```

4. View the last 5 rows by using the DataFrame `tail()` method.

```
[4]: df.tail()
```

```
[4]:       Fiscal Year  Cases  Responses  Sorties  Lives Saved  \
     2011      20512.0  43954    21566.0     3793        259.0
     2012      19787.0  43940    21609.0     4037        284.0
     2013      17803.0  38272    19420.0     3753        226.0
     2014      17508.0  38282    19032.0     3443        170.0
     2015      16456.0  37215    18781.0     3536        169.0

           Lives Lost After CG Notification  Lives Lost Before CG Notification  \
     2011                             476.0                              735.0
     2012                             429.0                              713.0
     2013                             425.0                              651.0
     2014                             425.0                              595.0
     2015                             434.0                              603.0

           Total  Lives Unaccounted For
     2011  392.0                    NaN
     2012  440.0                    NaN
     2013  252.0                    NaN
     2014  308.0                    NaN
     2015  330.0                    NaN
```

5. View the values in the ‘Cases’ column using dot syntax, bracket syntax, `loc[]` or `iloc[]`.

```
[11]: df.loc[:,'Cases'].count()
```

```
[11]: 52
```

6. Use `describe()` to view the summary statistics for the DataFrame.

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

```
[15]:         Fiscal Year         Cases      Responses      Sorties  Lives Saved  \
      count     46.000000     52.000000      46.000000    52.000000    46.000000
      mean   46296.608696  58013.769231   67666.586957  4339.230769   670.956522
      std    17438.646933  13480.714228   29300.537271  1334.134847   499.839128
      min    16456.000000  37215.000000   18781.000000  1984.000000   169.000000
      25%    31676.250000  46632.750000   33202.750000  3348.500000   281.750000
      50%    50621.500000  55945.500000   81711.500000  4221.000000   383.500000
      75%    57072.750000  69049.750000   88433.750000  5484.500000  1118.750000
      max    77954.000000  86222.000000  110267.000000  7889.000000  1783.000000

             Lives Lost After CG Notification  Lives Lost Before CG Notification  \
      count                         37.000000                          46.000000
      mean                         508.486486                        1079.956522
      std                          134.761028                         394.869765
      min                          180.000000                         533.000000
```

```
25%                            425.000000                             751.000000
50%                            492.000000                             998.000000
75%                            593.000000                            1440.750000
max                            800.000000                            1821.000000

            Total  Lives Unaccounted For
count   16.000000                    0.0
mean   468.000000                    NaN
std    149.916866                    NaN
min    252.000000                    NaN
25%    336.750000                    NaN
50%    437.500000                    NaN
75%    584.250000                    NaN
max    732.000000                    NaN
```

7. You can filter for particular values by comparing a colum to a value within the square bracket syntax. This creates a mask on the fly. Lets look at all of the rows whose case count is higher than the mean. You can get this number from the summary statistics above.

[20]:

```
df[df.Cases > df.Cases.mean()]
```

[20]:

```
      Fiscal Year  Cases  Responses  Sorties  Lives Saved  \
1972      51539.0  60328    72306.0     2633       1389.0
1973      55107.0  64182    77209.0     2918       1474.0
1974      59335.0  67692    79950.0     2751       1509.0
1975      62334.0  70551    81561.0     3024       1254.0
1976      67179.0  75069    87807.0     2995       1112.0
1977      74637.0  82601    96021.0     4121       1458.0
1978      77954.0  86222   100262.0     4386       1556.0
1979      72517.0  79858    92117.0     5747        949.0
1980      73345.0  81476    93726.0     6868       1235.0
1981      71781.0  78951    91432.0     6339       1080.0
1982      68552.0  75717    87715.0     5675       1359.0
1983      63980.0  72585    85796.0     5946       1121.0
1984      57431.0  66073    80698.0     5645       1148.0
1985      60775.0  70237    88449.0     6497       1076.0
1986      51765.0  68805    89318.0     4307        475.0
1987      55998.0  66656    87211.0     5785       1015.0
1988      54199.0  63446    83616.0     4307        583.0
1989      52776.0  64027    81862.0     3981        461.0
1990      53097.0  64971    84033.0     4407        463.0
1991      52782.0  66409    84872.0     5465        368.0
1992      53294.0  69856    88388.0     5543        399.0
1993      53026.0  69784    88147.0     5826        415.0
1994      53899.0  70337   108758.0     7889        338.0
1995      49704.0  63679   110267.0     4453        304.0
2004      32418.0  59995    33460.0     5557        281.0
```

```
      Lives Lost After CG Notification  Lives Lost Before CG Notification  \
1972                               NaN                             1389.0   
1973                               NaN                             1474.0   
1974                               NaN                             1509.0   
1975                               NaN                             1254.0   
1976                               NaN                             1112.0   
1977                               NaN                             1458.0   
1978                               NaN                             1556.0   
1979                             672.0                             1621.0   
1980                             586.0                             1821.0   
1981                             637.0                             1717.0   
1982                             446.0                             1805.0   
1983                             640.0                             1761.0   
1984                             319.0                             1467.0   
1985                             259.0                             1335.0   
1986                             180.0                              655.0   
1987                             576.0                             1591.0   
1988                             449.0                             1032.0   
1989                             646.0                             1107.0   
1990                             622.0                             1085.0   
1991                             748.0                             1116.0   
1992                             540.0                              939.0   
1993                             800.0                             1215.0   
1994                             593.0                              931.0   
1995                             468.0                              772.0   
2004                             502.0                              783.0   

      Total  Lives Unaccounted For  
1972    NaN                    NaN  
1973    NaN                    NaN  
1974    NaN                    NaN  
1975    NaN                    NaN  
1976    NaN                    NaN  
1977    NaN                    NaN  
1978    NaN                    NaN  
1979    NaN                    NaN  
1980    NaN                    NaN  
1981    NaN                    NaN  
1982    NaN                    NaN  
1983    NaN                    NaN  
1984    NaN                    NaN  
1985    NaN                    NaN  
1986    NaN                    NaN  
1987    NaN                    NaN  
1988    NaN                    NaN  
1989    NaN                    NaN  
```

```
1990    NaN                      NaN
1991    NaN                      NaN
1992    NaN                      NaN
1993    NaN                      NaN
1994    NaN                      NaN
1995    NaN                      NaN
2004  691.0                      NaN
```

9. Now lets create a NumPy array with the same data. Pandas DataFrames have a `to_numpy()` method. Use this method to create an array named ‘np_array’.

```
[29]: import numpy as np
      np_array = df.to_numpy()
```

10. Call the shape attribute on the array.

```
[32]: np_array.shape
```

```
[32]: (52, 9)
```

11. Use the array `reshape()` method to return a 4 x 13 x 9 array (the arguments to the method will be these numbers) .

```
[35]: np_array = np_array.reshape(4,13,9)
      np_array.shape
```

```
[35]: (4, 13, 9)
```

12. Import the dask.dataframe module aliased as ‘dd’

```
[37]: import dask.dataframe as dd
```

13. the `dask.dataframe` module has a `read_csv()` method which works in a similar fasion to the Pandas one. Use this method to read the file ‘USCG.Search.Rescue.Stats.csv’ into a dask DataFrame named ‘ddf’

```
[39]: ddf = dd.read_csv('USCG.Search.Rescue.Stats.csv')
```

14. Call the DataFrames `std()` method.

```
[40]: ddf.std()
```

```
[40]: Dask Series Structure:
      npartitions=1
      Cases     float64
      Total           …
      dtype: float64
      Dask Name: dataframe-std, 9 tasks
```

15. Notice that this did not calculate the standard deviation due to dask's use of lazy evaluation. add a `.compute()` after the `std()` to compute the result.

```
[41]: ddf.std().compute()
```

```
[41]: Fiscal Year                          17438.646933
      Cases                                13480.714228
      Responses                            29300.537271
      Sorties                               1334.134847
      Lives Saved                            499.839128
      Lives Lost After CG Notification       134.761028
      Lives Lost Before CG Notification      394.869765
      Total                                  149.916866
      Lives Unaccounted For                         NaN
      dtype: float64
```

```
[ ]:
```