Pandas Optional Assignment

November 19, 2020

Consider the following Python dictionary data and Python list labels:

data = {‘birds’: [‘Cranes’, ‘Cranes’, ‘plovers’, ‘spoonbills’, ‘spoonbills’, ‘Cranes’, ‘plovers’, ‘Cranes’, ‘spoonbills’, ‘spoonbills’], ‘age’: [3.5, 4, 1.5, np.nan, 6, 3, 5.5, np.nan, 8, 4], ‘visits’: [2, 4, 3, 4, 3, 4, 2, 2, 3, 2], ‘priority’: [‘yes’, ‘yes’, ‘no’, ‘yes’, ‘no’, ‘no’, ‘no’, ‘yes’, ‘no’, ‘no’]}

labels = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ‘i’, ‘j’]

1. Create a DataFrame birds from this dictionary data which has the index labels.

[1]:
```
# Importing Numpy and Pandas
import pandas as pd
import numpy as np

# Creating the dataframe 'birds' using labels as the indices
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills',␣
↪'Cranes', 'plovers', 'Cranes', 'spoonbills', 'spoonbills'], 'age': [3.5, 4,␣
↪1.5, np.nan, 6, 3, 5.5, np.nan, 8, 4], 'visits': [2, 4, 3, 4, 3, 4, 2, 2, 3,␣
↪2], 'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no',␣
↪'no']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

birds = pd.DataFrame(data, index=labels)
birds
```

[1]:
```
        birds  age  visits priority
a      Cranes  3.5       2      yes
b      Cranes  4.0       4      yes
c     plovers  1.5       3       no
d  spoonbills  NaN       4      yes
e  spoonbills  6.0       3       no
f      Cranes  3.0       4       no
g     plovers  5.5       2       no
h      Cranes  NaN       2      yes
i  spoonbills  8.0       3       no
j  spoonbills  4.0       2       no
```

2. Display a summary of the basic information about birds DataFrame and its data.

```
[2]: # Using describe to provide a summary of the information about birds DataFrames

     birds.describe
```

```
[2]: <bound method NDFrame.describe of         birds  age  visits priority
     a      Cranes  3.5       2      yes
     b      Cranes  4.0       4      yes
     c     plovers  1.5       3       no
     d  spoonbills  NaN       4      yes
     e  spoonbills  6.0       3       no
     f      Cranes  3.0       4       no
     g     plovers  5.5       2       no
     h      Cranes  NaN       2      yes
     i  spoonbills  8.0       3       no
     j  spoonbills  4.0       2       no>
```

3. Print the first 2 rows of the birds dataframe

```
[3]: # Using iloc since the indices are not the row numbers

     birds.iloc[:2]
```

```
[3]:     birds  age  visits priority
     a  Cranes  3.5       2      yes
     b  Cranes  4.0       4      yes
```

4. Print all the rows with only 'birds' and 'age' columns from the dataframe

```
[4]: # Using column based indexng to print only 'birds' and 'age' columns
     birds[['birds', 'age']]
```

```
[4]:         birds  age
     a      Cranes  3.5
     b      Cranes  4.0
     c     plovers  1.5
     d  spoonbills  NaN
     e  spoonbills  6.0
     f      Cranes  3.0
     g     plovers  5.5
     h      Cranes  NaN
     i  spoonbills  8.0
     j  spoonbills  4.0
```

5. select [2, 3, 7] rows and in columns ['birds', 'age', 'visits']

```
[5]: # r is a list containing the row indices and c is a list containing the column␣
      ↪names. Using iloc to get the rows.
     r = [2,3,7]
```

```
c = ['birds','age','visits']
birds[c].iloc[r]
```

[5]:
```
        birds  age  visits
c     plovers  1.5       3
d  spoonbills  NaN       4
h      Cranes  NaN       2
```

6. select the rows where the number of visits is less than 4

[6]:
```
birds[birds.visits < 4]
```

[6]:
```
        birds  age  visits priority
a      Cranes  3.5       2      yes
c     plovers  1.5       3       no
e  spoonbills  6.0       3       no
g     plovers  5.5       2       no
h      Cranes  NaN       2      yes
i  spoonbills  8.0       3       no
j  spoonbills  4.0       2       no
```

7. select the rows with columns ['birds', 'visits'] where the age is missing i.e NaN

[7]:
```
# Using isnull() to get the rows where the age value is null.
c = ['birds', 'visits']
birds[c][birds.age.isnull()]
```

[7]:
```
        birds  visits
d  spoonbills       4
h      Cranes       2
```

8. Select the rows where the birds is a Cranes and the age is less than 4

[8]:
```
# Using queries since normal accessing is throwing an error about Boolean␣
 ↪Indexing. Using queries eliminated the error.
birds[birds.age < 4].query('birds == "Cranes"')

# Reference: https://stackoverflow.com/questions/41710789/
 ↪boolean-series-key-will-be-reindexed-to-match-dataframe-index
```

[8]:
```
    birds  age  visits priority
a  Cranes  3.5       2      yes
f  Cranes  3.0       4       no
```

9. Select the rows the age is between 2 and 4(inclusive)

[9]:
```
# Using query to get the rows with 2 <= age <= 4
birds.query("(age>=2) and (age<=4)")
```

```
[9]:         birds  age  visits priority
     a      Cranes  3.5       2      yes
     b      Cranes  4.0       4      yes
     f      Cranes  3.0       4       no
     j  spoonbills  4.0       2       no
```

10. Find the total number of visits of the bird Cranes

```
[10]: # crane_visits is a list containing the visits of each crane. Using␣
       ↪crane_visits.sum() to print the sum.
      crane_visits = birds[birds.birds == 'Cranes']['visits']
      print("Number of Crane visits: ", crane_visits.sum())
```

```
Number of Crane visits:  12
```

11. Calculate the mean age for each different birds in dataframe.

```
[11]: # Using grouping to group birds by their name and using pd.mean() to print␣
       ↪their mean values by looping over them.
      group_birds = birds.groupby('birds')
      for bird, bird_info in group_birds:
        print("The mean age of", bird.title(), "is", (bird_info['age'].mean()))
```

```
The mean age of Cranes is 3.5
The mean age of Plovers is 3.5
The mean age of Spoonbills is 6.0
```

12. Append a new row 'k' to dataframe with your choice of values for each column. Then delete that row to return the original DataFrame.

```
[12]: # Using randint to get 10 random values between 0 and 10. Then appending the␣
       ↪new column to the dataframe.
      k_vals = ['Crow', 4.2, 3, 'yes']
      birds.loc['k'] = k_vals
      birds
```

```
[12]:         birds  age  visits priority
      a      Cranes  3.5       2      yes
      b      Cranes  4.0       4      yes
      c     plovers  1.5       3       no
      d  spoonbills  NaN       4      yes
      e  spoonbills  6.0       3       no
      f      Cranes  3.0       4       no
      g     plovers  5.5       2       no
      h      Cranes  NaN       2      yes
      i  spoonbills  8.0       3       no
      j  spoonbills  4.0       2       no
      k        Crow  4.2       3      yes
```

[13]:
```
# Dropping the column using .drop() function with axis=1 denoting column and␣
↪inplace=True modifying the original DataFrame.
birds.drop('k', inplace=True)
birds
```

[13]:
```
        birds  age  visits priority
a      Cranes  3.5       2      yes
b      Cranes  4.0       4      yes
c     plovers  1.5       3       no
d  spoonbills  NaN       4      yes
e  spoonbills  6.0       3       no
f      Cranes  3.0       4       no
g     plovers  5.5       2       no
h      Cranes  NaN       2      yes
i  spoonbills  8.0       3       no
j  spoonbills  4.0       2       no
```

13. Find the number of each type of birds in dataframe (Counts)

[14]:
```
# Using unique() to get names of each unique bird as a list and then printing␣
↪the length of that list using len() function.
print("Number of types of birds in the given dataframe is", len(pd.
↪unique(birds['birds'])))

#https://www.geeksforgeeks.org/
↪how-to-count-distinct-values-of-a-pandas-dataframe-column/
```

```
Number of types of birds in the given dataframe is 3
```

14. Sort dataframe (birds) first by the values in the 'age' in decending order, then by the value in the 'visits' column in ascending order.

[15]:
```
# Using sort_values() function to sort the values.
birds.sort_values(by='age', ascending=False).sort_values(by='visits')
```

[15]:
```
        birds  age  visits priority
g     plovers  5.5       2       no
j  spoonbills  4.0       2       no
a      Cranes  3.5       2      yes
h      Cranes  NaN       2      yes
i  spoonbills  8.0       3       no
e  spoonbills  6.0       3       no
c     plovers  1.5       3       no
b      Cranes  4.0       4      yes
f      Cranes  3.0       4       no
d  spoonbills  NaN       4      yes
```

15. Replace the priority column values with'yes' should be 1 and 'no' should be 0

[16]:
```
# Looping over the values and changing the values according to the condition
for i in birds.index:
  birds.loc[i, 'priority'] = 1 if birds.loc[i, 'priority'] == 'yes' else 0

birds
```

[16]:
```
        birds  age  visits priority
a      Cranes  3.5       2        1
b      Cranes  4.0       4        1
c     plovers  1.5       3        0
d  spoonbills  NaN       4        1
e  spoonbills  6.0       3        0
f      Cranes  3.0       4        0
g     plovers  5.5       2        0
h      Cranes  NaN       2        1
i  spoonbills  8.0       3        0
j  spoonbills  4.0       2        0
```

16. In the 'birds' column, change the 'Cranes' entries to 'trumpeters'.

[17]:
```
# Looping over the values and changing the values according to the condition
for i in birds.index:
  if birds.loc[i, 'birds'] == "Cranes":
    birds.loc[i, 'birds'] = "trumpeters"

birds
```

[17]:
```
        birds  age  visits priority
a  trumpeters  3.5       2        1
b  trumpeters  4.0       4        1
c     plovers  1.5       3        0
d  spoonbills  NaN       4        1
e  spoonbills  6.0       3        0
f  trumpeters  3.0       4        0
g     plovers  5.5       2        0
h  trumpeters  NaN       2        1
i  spoonbills  8.0       3        0
j  spoonbills  4.0       2        0
```