parth.pandey13103447@gmail.com2

September 6, 2019

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.

[4]:
```
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']
# creating dataframe birds from thhe given data from above
# importing the pandas library
# importing the numpy library to use NaN
import numpy as np
import pandas as pd
birds = pd.DataFrame(data,index=labels)
```

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

[5]:
```
# Showing the info about the dataframe
print('DataFrame info ')
print(birds.info())
# Showing the info about the data
print('Data info')
print(birds.describe())
```

```
DataFrame info
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
birds       10 non-null object
age         8 non-null float64
visits      10 non-null int64
```

```
priority    10 non-null object
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
None
Data info
            age     visits
count  8.000000  10.000000
mean   4.437500   2.900000
std    2.007797   0.875595
min    1.500000   2.000000
25%    3.375000   2.000000
50%    4.000000   3.000000
75%    5.625000   3.750000
max    8.000000   4.000000
```

3. Print the first 2 rows of the birds dataframe

[6]:
```
# Showing the dataframe first 2 rows
birds.head(2)
```

[6]:

	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

[7]:
```
# Showing the data for first
birds[['birds','age']]
```

[7]:

	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']

[8]:
```
# The 2,3,7 can be mapped to b,c,g
birds[['birds','age','visits']].loc[['b','c','g']]
```

[8]:

	birds	age	visits
b	Cranes	4.0	4
c	plovers	1.5	3

```
g  plovers  5.5       2
```

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

[9]:
```
birds[birds['visits'] < 4]
```

[9]:
```
         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

[10]:
```
# filtering the data for NaN
birds[np.isnan(birds['age'])]
```

[10]:
```
         birds  age  visits priority
d  spoonbills  NaN       4      yes
h       Cranes  NaN       2      yes
```

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

[11]:
```
birds[(birds['birds'] == 'Cranes') & (birds['age'] < 4) ]
```

[11]:
```
     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)

[12]:
```
birds[(birds['age'] <= 4 ) & (birds['age'] >2  )]
```

[12]:
```
         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

[13]:
```
birds[birds['birds'] == 'Cranes']['visits'].sum()
```

[13]:
```
12
```

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

[14]:
```
for bird_type , bird_df in birds.groupby('birds'):
    print(bird_type,' = ',bird_df['age'].mean())
```

```
Cranes  =  3.5
plovers  =  3.5
spoonbills  =  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.

[15]:
```
birds2 = pd.DataFrame({'birds':['peacock'] , 'age':[10] , 'visits':[2] ,␣
 ↪'priority':['yes'] } , index = ['k'])
# birds2
birds = pd.concat([birds ,birds2])
print('The new birds dataframe with new row added')
print(birds)
print('The original dataframe after deleting the new row')
birds.drop('k',inplace = True)
print(birds)
```

```
The new birds dataframe with new row added
        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     peacock  10.0       2      yes
The original dataframe after deleting the new row
        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)

[16]:
```
print(birds['birds'].unique().tolist())
```

```
['Cranes', 'plovers', 'spoonbills']
```

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.

[17]:
```
birds.sort_values('age',ascending=False).sort_values('visits',inplace=True)
print(birds)
```

```
        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
```

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

[18]:
```
birds['priority'] = birds['priority'].map({'yes':1,'no':0})
print(birds)
```

```
        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'.

[24]:
```
birds['birds'] = birds['birds'].apply(lambda x : x if x != 'Cranes' else␣
 ↪'trumpeters')
print(birds)
```

```
        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
```