pandas_basics_practice

November 3, 2021

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

df = pd.DataFrame({'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills',␣
 ↪'spoonbills', 'Cranes',        #creating dataframe from dictionary
                                   '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']},
                               index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',␣
 ↪'j'])
df
```

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

```
             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

[3]:
```
print("First 2 rows :\n\n",df.head(2))
```

```
First 2 rows :

     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]:
```
print("All the rows with birds and age columns:\n\n",df[['birds','age']])
```

```
All the rows with birds and age columns:

         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]:
```
df.iloc[[2,3,7],[0,1,2]]        # bird_data.loc[['c','d','h'],['birds', 'age',␣
 ↪'visits']] - another method
```

[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]: print("Rows which have number of visits < 4 :\n\n",df[df['visits']<4])
```

```
Rows which have number of visits < 4 :

         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]: df1 = df[df['age'].isna()]     # selects rows with NaN
     print("birds and visits columns for which age is missing:
      ↪\n\n",df1[['birds','visits']]) # selects birds and visits columns of NaN rows
```

```
birds and visits columns for which age is missing:

         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]: df2 = df[df['age']<4]                  # Selects the rows with age < 4
     print("Cranes with age < 4:\n\n",df2[df2['birds'] == 'Cranes'])         #␣
      ↪selects only cranes with age < 4
```

```
Cranes with age < 4:

     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]: print("All the rows where age is between 2 and 4:\n\n",df[df['age'].
      ↪between(2,4)])
```

```
All the rows where age is between 2 and 4:

         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]:
```
df3 = df[df['birds'] == 'Cranes']
print("Total number of visits of the bird Crane :",sum(df3['visits']))
```

```
Total number of visits of the bird Crane : 12
```

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

[11]:
```
g = df.groupby('birds')
print("Mean age for the birds:\n\n",g['age'].mean())
```

```
Mean age for the birds:

 birds
Cranes        3.5
plovers       3.5
spoonbills    6.0
Name: age, dtype: float64
```

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]:
```
new_row = pd.Series({'birds':'Peacock', 'age':4, 'visits':10, 'priority':
 ↪'yes'}, name='k')
df4 = df.append(new_row)
print("DataFrame with new row k:\n\n",df4)
df4 = df4.drop('k')
print("\nDataFrame with deleted row k:\n\n",df4)
```

```
DataFrame with new row k:

         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  4.0      10      yes

DataFrame with deleted row k:

         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)

```
[13]: print("Count of different type of birds:\n\n",g['birds'].count())
```

```
Count of different type of birds:

 birds
Cranes        4
plovers       2
spoonbills    4
Name: birds, dtype: int64
```

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.

```
[14]: age_des = df.sort_values('age',ascending = False)
      vis_asc = df.sort_values('visits',ascending = True)
      age_vis = df.sort_values(['age','visits'],ascending = [False,True])
      print("Dataframe sorted by age in descending order:\n\n",age_des)
      print("\nDataframe sorted by visits in ascending order:\n\n",vis_asc)
      print("\nDataframe sorted by age in descending and visits in ascending order:
       ↪\n\n",age_vis)
```

```
Dataframe sorted by age in descending order:

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

Dataframe sorted by visits in ascending order:

         birds  age  visits priority
a      Cranes  3.5       2      yes
g     plovers  5.5       2       no
h      Cranes  NaN       2      yes
```

```
j  spoonbills  4.0       2       no
c     plovers  1.5       3       no
e  spoonbills  6.0       3       no
i  spoonbills  8.0       3       no
b      Cranes  4.0       4      yes
d  spoonbills  NaN       4      yes
f      Cranes  3.0       4       no

Dataframe sorted by age in descending and visits in ascending order:

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

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

[15]:
```
df5 = df.replace(to_replace = ["yes","no"], value=[1,0])
print("Dataframe with prioroty as 1 and 0:\n\n",df5)
```

```
Dataframe with prioroty as 1 and 0:

        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'.

[16]:
```
df6 = df.replace(to_replace = "Cranes", value = "trumpeters")
print("original Dataframe:\n\n",df)
print("\nDataframe after replacing Cranes by trumpeters:\n\n",df6)
```

```
original Dataframe:

        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

Dataframe after replacing Cranes by trumpeters:

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