Data Science Africa Challenge: Solutions

Solution to the Data Science Africa Challenge problem set by:
Name: Ismail, Dawud Ibrahim
Email: ismaildawud96@gmail.com
Mobile: +233546742189
Date: 9th Oct, 2018

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

In [2]:
```
data = {
        'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat',
 'dog', 'dog'],

        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],

        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no',
 'no']
      }

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

1. Creating a DataFrame df from this dictionary data which has the index labels.

In [3]:
```
df = pd.DataFrame(data=data, index=labels)
df
```

Out[3]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	2.0	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
animal      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: 280.0+ bytes
```

3. Return the first 3 rows of the DataFrame df.

In [5]:
```
df[ :3]
```

Out[5]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no

4. Select just the 'animal' and 'age' columns from the DataFrame df.

In [6]:
```
df[ ['animal','age'] ]
```

Out[6]:

	animal	age
a	cat	2.5
b	cat	3.0
c	snake	0.5
d	dog	NaN
e	dog	5.0
f	cat	2.0
g	snake	4.5
h	cat	NaN
i	dog	7.0
j	dog	3.0

5. Selecting the data in rows [3, 4, 8] and in columns ['animal', 'age'].

In [7]:
```
df[ ['animal','age'] ].iloc[ [3, 4, 8] ]
```

Out[7]:

	animal	age
d	dog	NaN
e	dog	5.0
i	dog	7.0

6. Selecting only the rows where the number of visits is greater than 3.

In [8]:
```
df[ df['visits'] > 3 ]
```

Out[8]:

	animal	age	visits	priority

7. Selecting the rows where the age is missing, i.e. is NaN

In [9]:
```
df[ df['age'].isna() ]
```

Out[9]:

	animal	age	visits	priority
d	dog	NaN	3	yes
h	cat	NaN	1	yes

8. Selecting the rows where the animal is a cat and the age is less than 3.

In [10]:
```
df[ (df['animal'] == 'cat') & (df['age'] < 3) ]
```

Out[10]:

	animal	age	visits	priority
a	cat	2.5	1	yes
f	cat	2.0	3	no

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

In [11]:
```
df[ (df['age'] >= 2) & (df['age'] <=4)]
```

Out[11]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
f	cat	2.0	3	no
j	dog	3.0	1	no

10. Changing the age in row 'f' to 1.5.

In [15]:
```
df['age'].loc['f'] = 1.5
```

11. Calculating the sum of all visits (the total number of visits).

In [16]:
```
sum( df['visits'])
```

Out[16]:
```
19
```

12. Calculating the mean age for each different animal in df.

In [17]:
```
df.groupby(['animal'])['age'].mean()
```

Out[17]:
```
animal
cat      2.333333
dog      5.000000
snake    2.500000
Name: age, dtype: float64
```

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

In [18]:
```
k = pd.DataFrame(data={ 'animal': 'rat', 'age': 2.5, 'visits': 4, 'priority':
'yes' }, index=['k'] )
df = df.append(k)
df
```

Out[18]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no
k	rat	2.5	4	yes

appending a new row k to df

In [19]:
```
df = df.drop(['k'])
df
```

Out[19]:

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

droping the row k

14. Counting the number of each type of animal in df

In [20]:
```
pd.value_counts(df['animal'].values)
```

Out[20]:
```
cat      4
dog      4
snake    2
dtype: int64
```

15. Sort df first by the values in the 'age' in descending order, then by the value in the 'visit' column in ascending order.

In [21]:
```
df.sort_values(['age'],ascending=False)
```

Out[21]:

	animal	age	visits	priority
i	dog	7.0	2	no
e	dog	5.0	2	no
g	snake	4.5	1	no
b	cat	3.0	3	yes
j	dog	3.0	1	no
a	cat	2.5	1	yes
f	cat	1.5	3	no
c	snake	0.5	2	no
d	dog	NaN	3	yes
h	cat	NaN	1	yes

Above is sort by age only in descending order

In [22]:
```
df.sort_values(['visits'],ascending=True)
```

Out[22]:

	animal	age	visits	priority
a	cat	2.5	1	yes
g	snake	4.5	1	no
h	cat	NaN	1	yes
j	dog	3.0	1	no
c	snake	0.5	2	no
e	dog	5.0	2	no
i	dog	7.0	2	no
b	cat	3.0	3	yes
d	dog	NaN	3	yes
f	cat	1.5	3	no

Above is sort by visits only in ascending order

In [23]:
```
df.sort_values(['age'],ascending=False).sort_values(['visits'],ascending=True)
```

Out[23]:

	animal	age	visits	priority
g	snake	4.5	1	no
j	dog	3.0	1	no
a	cat	2.5	1	yes
h	cat	NaN	1	yes
i	dog	7.0	2	no
e	dog	5.0	2	no
c	snake	0.5	2	no
b	cat	3.0	3	yes
f	cat	1.5	3	no
d	dog	NaN	3	yes

Above is sort by age only in descending order which is then sorted by visits in ascending order

16. The 'priority' column contains the values 'yes' and 'no'. Replace this column with a column of boolean values: 'yes' should be True and 'no' should be False.

In [24]:
```
df['priority'].replace(['yes','no'], [True,False], inplace=True)
df
```

Out[24]:

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	snake	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	snake	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

17. In the 'animal' column, change the 'snake' entries to 'python'.

In [25]:
```
df['animal'].replace('snake', 'python', inplace=True)
df
```

Out[25]:

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	python	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	python	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

18. For each animal type and each number of visits, find the mean age. In other words, each row is an animal, each column is a number of visits and the values are the mean ages (hint: use a pivot table).

In [26]:
```
df.groupby(['animal','visits'])['age'].mean()
```

Out[26]:
```
animal  visits
cat     1         2.50
        3         2.25
dog     1         3.00
        2         6.00
        3          NaN
python  1         4.50
        2         0.50
Name: age, dtype: float64
```

In [ ]:

In [ ]: