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.

In [3]:
```
import numpy as np
import  pandas as pd
```

In [4]:
```
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbi
lls', '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']}
```

In [5]:
```
birds=pd.DataFrame(data,index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'
, 'i', 'j'])
```

In [6]:
```
birds
```

Out[6]:

	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

	birds	age	visits	priority
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.

In [6]:
```
birds.info() #birds.describe()
```

```
<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
```

3. Print the first 2 rows of the birds dataframe

In [7]:
```
birds.iloc[0:2]
```

Out[7]:

	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

In [8]:
```
birds[['birds','age']]
```

Out[8]:

	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']

In [13]:
```
birds.iloc[[2,3,7]][['age','birds','visits']]
```

Out[13]:

	age	birds	visits
c	1.5	plovers	3
d	NaN	spoonbills	4
h	NaN	Cranes	2

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

In [14]:
```
birds[birds['visits']<4]
```

Out[14]:

	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

In [15]:
```
birds[birds['age'].isnull()][['birds','visits']]
```

Out[15]:

	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

In [16]:
```
birds[(birds['birds']=='Cranes') & (birds['age']<4)]
```

Out[16]:

	birds	age	visits	priority

	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)

In [17]:
```
birds[(birds['age']>=2) & (birds['age']<=4)]
```

Out[17]:

	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

In [18]:
```
birds[birds['birds']=='Cranes']['visits'].sum()
```

Out[18]:
```
12
```

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

In [20]:
```
birds.groupby('birds')['age'].mean()
```

Out[20]:
```
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.

In [22]:
```
birds.loc['x']=('peacock',3,3,'yes') #appending a row (x)
birds
```

Out[22]:

	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
x	peacock	3.0	3	yes

In [23]:
```
birds.drop('x',axis=0,inplace=True) # drop the appended row (x)
```

In [24]:
```
birds
```

Out[24]:

	birds	age	visits	priority
a	Cranes	3.5	2	yes
b	Cranes	4.0	4	yes

	birds	age	visits	priority
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)

In [25]:
```
birds['birds'].value_counts()
```

Out[25]:
```
Cranes        4
spoonbills    4
plovers       2
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.

In [26]:
```
birds.sort_values('age',ascending=False)
```

Out[26]:

	birds	age	visits	priority
i	spoonbills	8.0	3	no
e	spoonbills	6.0	3	no

	birds	age	visits	priority
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

In [27]:
```
birds.sort_values('visits',ascending=True)
```

Out[27]:

	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

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

In [7]:
```
def boo(x):
    if x== 'yes':
        return 1
    else:
        return 0
```

In [8]:
```
birds["priority"]=birds["priority"].apply(boo)
```

In [9]:
```
birds
```

Out[9]:

	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'.

In [10]:
```
birds['birds']=birds['birds'].apply(lambda x: 'trumpeters' if x=='Crane
s' else x)
```

In [11]:
```
birds
```

Out[11]:

	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