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 [7]:

```
#DataFrame of Birds with Index Labels
import pandas as pd
import numpy as np
birds=[('Cranes',3.5,2,'yes'),
       ('Cranes',4,4,'yes'),
       ('plovers',1.5,3,'no'),
       ('spoonbills',np.nan,4,'yes'),
       ('spoonbills',6,3,'no'),
       ('Cranes',3,4,'no'),
       ('plovers',5.5,2,'no'),
       ('Cranes',np.nan,2,'yes'),
       ('spoonbills',8,3,'no'),
       ('spoonbills',4,2,'no')] #List of Tuples

df=pd.DataFrame(birds, columns=['birds','age','visits','priority'], index=['a', 'b', 'c', 'd', 'e',
'f', 'g', 'h', 'i', 'j'])
df
```

Out[7]:

	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.

In [9]:

```
# Basic Information of Birds Dataframe


a11=df.describe() # Description of age
print(' total count, mean, std_dev, and percentile values are: \n', a11)

print('\n information of Bird DataFrame :\n')
a12=df.info()
```

```
 total count, mean, std_dev, and percentile values are:
             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
```

```
50%     4.000000    3.000000
75%     5.625000    3.750000
max     8.000000    4.000000

 information of Bird DataFrame :

<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 [10]:

```
r1=df[0:2] # first two rows using slicing
print('\n first two rows of birds DataFrame are:\n',r1)
```

```
 first two rows of birds DataFrame are:
     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 [11]:

```
r2=df[['birds', 'age']]
print('\n birds and their respective age:\n', r2)
```

```
 birds and their respective age:
         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 [12]:

```
df.loc[['b','c','g'], ['birds', 'age', 'visits']] #[2, 3, 7] rows and columns ['birds', 'age',
'visits']
```

Out[12]:

	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

In [13]:

```
r3=df[df['visits']<4 ]#print particular column another way of accessing column
print('\n visit of birds include:\n',r3)
```

```
 visit of birds include:
         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 [14]:

```
df[df['age'].isnull()]
```

Out[14]:

	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

In [15]:

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

Out[15]:

	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 [16]:

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

Out[16]:

	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 [17]:

```
t1=df.visits[df['birds']=='Cranes'] #
print(t1)
print('\ntotal number of bird cranes visits are:', t1.sum())
```

```
a    2
b    4
```

```
f    4
h    2
Name: visits, dtype: int64

total number of bird cranes visits are: 12
```

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

In [8]:

```
df.groupby('birds', as_index=False)['age'].mean()
```

Out[8]:

	birds	age
0	Cranes	3.5
1	plovers	3.5
2	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.

In [10]:

```
df1=df.loc['k']=['eagle', 4, 2, 'yes'] #add new row with index 'k'
print('with new row k: \n',df)

df.drop(df.index[10])
```

```
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        eagle  4.0       2      yes
```

Out[10]:

	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)

In [6]:

```
df.birds.value_counts()
```

Out[6]:

```
spoonbills    4
Cranes        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 [20]:

```
dd=df.sort_values(by=['age'], ascending=False)
print('DataFrame birds in decending order based on age:  \n',dd)

da=df.sort_values(by=['visits'])
print('\nDataFrame birds in ascending order based on visits: \n',da)
```

```
DataFrame birds in decending order based on age:
         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
k       eagle  4.0       2      yes
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 birds in ascending order based on visits:
         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
k       eagle  4.0       2      yes
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]:

```
df.replace({

    'yes':1,
    'no' :0
})
```

Out[7]:

	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

	birds	age	visits	priority
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 [8]:

```
dr=df.replace({

    'Cranes':'trumpeters'
})
dr
```

Out[8]:

	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

In [ ]:

```

```