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']

In [41]:
```
import pandas as pd
import numpy as np

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 [42]:
```
d_f = pd.DataFrame(data, index = labels)
d_f
```

Out[42]:

	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 [43]:
```
d_f.describe()
```

Out[43]:

	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

In [44]:
```
d_f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   birds     10 non-null     object
 1   age       8 non-null      float64
 2   visits    10 non-null     int64
 3   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 [45]:
```
d_f.head(2)
```

Out[45]:

	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 [46]:
```
d_f.loc[:,['birds' , 'age']]
```

Out[46]:

	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 [47]:
```
d_f.iloc[[2,3,7],[0,1,2 ]]
```

Out[47]:

	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

In [48]:
```
d_f[d_f['visits'] < 4]
```

Out[48]:

	birds	age	visits	priority
a	Cranes	3.5	2	yes
c	plovers	1.5	3	no

	birds	age	visits	priority
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 [49]:
```
d=d_f[d_f['age'].isnull()]
d.loc[:,['birds','visits']]
```

Out[49]:

	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 [50]:
```
d_f[(d_f['birds'] == 'Cranes') & (d_f['age'] < 4)]
```

Out[50]:

	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 [51]:
```
d_f[(d_f['age']>=2) & (d_f['age'] <= 4)]
```

Out[51]:

	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 [52]:
```
d=d_f[(d_f['birds'] == 'Cranes')]
d['visits'].sum()
```

Out[52]:
```
12
```

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

In [53]:
```
mean_by_bird_group_over_age = d_f.groupby('birds')['age'].mean()
mean_by_bird_group_over_age
```

Out[53]:
```
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 [61]:
```
data1 = {'birds': ['Crane'], 'age': [3.5], 'visits': [2],
        'priority': ['yes']}
d_f1 = pd.DataFrame(data1, index = ['k'])
d_f1

d_f2 = d_f.append(d_f1)
d_f2
```

Out[61]:

	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
k	Crane	3.5	2	yes

In [62]:
```
d_f2 = d_f2[~d_f2.index.str.contains("k")]
d_f2
```

Out[62]:

	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

13. Find the number of each type of birds in dataframe (Counts)

In [57]:
```
count_by_bird_group_over_age = d_f.groupby('birds')['age'].count()
count_by_bird_group_over_age
```

Out[57]:
```
birds
Cranes        3
plovers       2
spoonbills    3
Name: age, 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 [58]:
```
d=d_f.sort_values(by='age',ascending=False)
d.sort_values(by='visits',ascending=True)
```

Out[58]:

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

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

In [59]:
```
d=d_f['priority'].replace('yes',1).replace('no',0)
d_f['priority'] = d
d_f
```

Out[59]:

	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 [60]:
```
d_f.birds.map(lambda change: 'trumpeters' if change=='Cranes' else change)
```

Out[60]:
```
a    trumpeters
b    trumpeters
c       plovers
d    spoonbills
e    spoonbills
f    trumpeters
g       plovers
h    trumpeters
i    spoonbills
j    spoonbills
Name: birds, dtype: object
```