In [152]:
```
import pandas as pd
import numpy as np
```

In [153]:
```
##**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 [154]:
```
df = pd.DataFrame(data , index=labels)
df
```

Out[154]:

	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 [155]:
```
df.info()
```

```
<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 [156]: df.head(2)
```

Out[156]:

	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 [157]: df[['birds', 'age']]
```

Out[157]:

	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 [158]: df.iloc[[2,3,7], [1,2,3]]
```

Out[158]:

	age	visits	priority
c	1.5	3	no
d	NaN	4	yes
h	NaN	2	yes

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

```
In [159]: df[df['visits'] < 4]
```

Out[159]:

	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 [160]: df.loc[df['age'].isnull(), ['birds', 'visits']]
```

Out[160]:

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

Out[161]:

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

Out[162]:

	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 [163]: df.loc[(df['birds'] == 'Cranes'), ['visits']].sum()
```

```
Out[163]: visits    12
          dtype: int64
```

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

```
In [164]: df['age'].mean() # mean for all birds.
```

```
Out[164]: 4.4375
```

```
In [171]: df.groupby('birds')['age'].mean() # mean for different birds.
```

```
Out[171]: birds
          plovers       3.5
          spoonbills    6.0
          trumpeters    3.5
          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 [165]: df.loc['k'] = ['Blue Bird', 1.2, 10, 'no']
          df
```

Out[165]:

	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	Blue Bird	1.2	10	no

In [166]:
```
df = df.drop('k')
df
```

Out[166]:

	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 [167]:
```
df.groupby(['birds']).count()
```

Out[167]:

	age	visits	priority
birds			
Cranes	3	4	4
plovers	2	2	2
spoonbills	3	4	4

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 [168]:
```
print(df.sort_values('age', ascending=False))
print(" ")
print(df.sort_values('visits', ascending=True))
```

```
        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

        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 [169]:
```
df['priority'] = df['priority'].map(dict(yes=1, no=0)) # It will update origin
al data frame.
# df.priority = df.priority.map({'yes': 1, 'no': 0}) # Same as above It will u
pdate original data frame.
#df.priority.map({'yes': 1, 'no': 0}) # It will update at run time only.
df
```

Out[169]:

	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 [170]: # df['birds'] = df['birds'].map(lambda b: 'trumpeters' if b=='Cranes' else b)
           # one other method to replace
          df = df.replace('Cranes', 'trumpeters')
          df
```

```
Out[170]:
```

	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