

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 [12]: 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']}
import pandas as pd
df=pd.DataFrame(data,columns=['birds','age','visits','priority'],index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'])
df
```

Out[12]:

	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 [12]: df.describe()
```

Out[12]:

	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

3. Print the first 2 rows of the birds dataframe

```
In [13]: df.iloc[:2]
```

Out[13]:

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

Out[14]:

	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 [24]: df.iloc[[2,3,7], [0,1,2]]
```

Out[24]:

	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 [39]: df[df.visits < 4]
```

Out[39]:

	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 [38]: import numpy as np
df_missing = df[['birds','visits']][df['age'].isnull()]
df_missing
```

Out[38]:

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

Out[43]:

	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 [44]: df[(df.age <= 4) & (df.age >= 2)]
```

Out[44]:

	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 [49]: df['visits'][df.birds=='Cranes'].sum()
```

Out[49]: 12

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

```
In [64]: (df.groupby('birds')['age'].mean())
```

Out[64]:

```
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 [20]: df.loc['k'] = ['Parrot', 6, 6, 'no']
df.drop('k')
```

Out[20]:

	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 [27]: df.groupby(['birds']).size()
```

Out[27]:

```
birds
Cranes      4
Parrot      1
plovers     2
spoonbills  4
dtype: int64
```

14. Sort dataframe (birds) first by the values in the 'age' in descending order, then by the value in the 'visits' column in ascending order.

```
In [28]: df.sort_values(['age','visits'], ascending=[False,True])
```

Out[28]:

	birds	age	visits	priority
i	spoonbills	8.0	3	no
e	spoonbills	6.0	3	no
k	Parrot	6.0	6	no
g	plovers	5.5	2	no
j	spoonbills	4.0	2	no
b	Cranes	4.0	4	yes
a	Cranes	3.5	2	yes
f	Cranes	3.0	4	no
c	plovers	1.5	3	no
h	Cranes	NaN	2	yes
d	spoonbills	NaN	4	yes

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

```
In [5]: df['priority'].replace('yes', 1,inplace=True)
df['priority'].replace('no', 0,inplace=True)
df
```

Out[5]:

	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

16. In the 'birds' column, change the 'Cranes' entries to 'trumpeters'.

```
In [4]: df['birds'].replace('Cranes', 'trumpeters',inplace=True)
df
```

Out[4]:

	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				
i				
j				