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 [1]:

```
#importing required packages
import numpy as np
import pandas as pd
''' Decalring the given data'''
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills', 'Crane

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 [2]:

```
df = pd.DataFrame(data, index=labels)
df
```

Out[2]:

	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 [3]:

```
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
```

In [4]:

```
''' (optional) Extra inforamation can be accessed by: df.describe() '''
df.describe()
```

Out[4]:

	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 [5]:

```
df['birds'][:2]
```

Out[5]:

```
a    Cranes
b    Cranes
Name: birds, dtype: object
```

4. Print all the rows with only 'birds' and 'age' columns from the dataframe

In [6]:

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

Out[6]:

	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 [7]:

```
''' other approach: df.loc[['c', 'd', 'h'],['birds', 'age', 'visits']] '''

df.iloc[[2, 3, 7]][['birds', 'age', 'visits']]
```

Out[7]:

	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 [8]:

```
''' another approach: df.query('visits <4 ') '''
df[df['visits']<4]
```

Out[8]:

	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 [9]:

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

Out[9]:

	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 [10]:

```
''' another approach: df[(df['birds']=='Cranes') & (df['age']<4)] '''
df.query( 'birds == "Cranes" & age<4 ')
```

Out[10]:

	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 [11]:

```
''' another approach: df.query('age >= 2 & age<=4 ') '''
df[(df['age']>=2) & (df['age']<=4)]
```

Out[11]:

	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 [12]:

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

Out[12]:

12

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

In [13]:

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

Out[13]:

	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 [14]:

```
''' Appending k row to dataframe'''
new_df = pd.DataFrame({'birds': ['pikachu'],'visits':5,'priority':'yes'}, index=['k
df = df.append(new_df,sort=False)
df
```

Out[14]:

	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	pikachu	NaN	5	yes

In [15]:

```
#Deleting appended row from dataframe
df.drop('k',inplace=True)
df
```

Out[15]:

	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 [16]:

```
df['birds'].value_counts()
```

Out[16]:

```
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 [17]:

```
'''
For only birds & age columns : df.sort_values(by=['birds', 'age'], ascending=[False
'''

#For Dataframe
df.sort_values(by=['birds', 'age'], ascending=[False, True])
```

Out[17]:

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

In [18]:

```
#For birds dataframe
df.sort_values(by=['birds', 'age'], ascending=[False, True])['birds']
```

Out[18]:

```
j    spoonbills
e    spoonbills
i    spoonbills
d    spoonbills
c       plovers
g       plovers
f        Cranes
a        Cranes
b        Cranes
h        Cranes
Name: birds, dtype: object
```

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

In [19]:

```
df.replace({'yes':1, 'no':0},inplace=True)
df
```

Out[19]:

	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 [20]:

```
df.replace('Cranes','trumpeters',inplace=True)
df
```

Out[20]:

	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