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']

Imports

```
import pandas as pd
import numpy as np
```

1. Create a DataFrame birds from this dictionary data which has the index labels.

```
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills', 'Cranes', 'plover
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
birds = pd.DataFrame(data=data, index=labels)
birds
```

	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.

```
birds.info() #summary of basic innformation
```

```
<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
```

```
birds.describe() #describes the statistical data about the numerical columns
```

	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 *

```
birds[0:2] #we can use indexing or .head() function
```

	birds	age	visits	priority
a	Cranes	3.5	2	yes
b	Cranes	4.0	4	yes

```
birds.head(2)
```

	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

```
birds[['birds', 'age']] #sending the array of required columns to the dataframe
```

	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
i	spoonbills	8.0

5. select [2, 3, 7] rows and in columns ['birds', 'age', 'visits']

```
birds.iloc[[2,3,7]][['birds', 'age', 'visits']] #2,3,7 are guessed as 2nd row (i.e. with index=
```

	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

```
birds[birds['visits'] < 4] #Form the mask and filter it
```

	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

```
birds[birds['age'].isna()][['birds', 'visits']] #use .isna() function to get the the NaN values
```

birds **visits**

8. Select the rows where the birds is a Cranes and the age is less than 4

b Cranes 2

```
birds[(birds['birds'] == "Cranes") & (birds['age'] < 4)] #use brackets and &-symbol for mutiple
```

	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)

```
birds[(birds['age'] >= 2) & (birds['age'] <= 4)] #same reason as above
```

	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

```
birds[birds['birds'] == 'Cranes']['visits'].sum() #Filter and fetch the requird columns. Then c
```

```
12
```

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

```
birds.groupby('birds')['age'].mean() #Group by the birds. Then choose 'age' to find the 'mean'
```

```
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.

```
birds.loc['k'] = ["Cranes", 4, 3, "no"] #adding a new data in the index location 'k'
print(birds)
birds = birds.drop('k') #dropping that index data
birds
```

```
        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      Cranes  4.0       3       no
```

	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)

```
birds.groupby('birds')['birds'].count() #group by 'birds' and get the 'birds' data to get the
```

```
birds
Cranes        4
plovers       2
spoonbills    4
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.

```
birds = birds.sort_values(by=['age', 'visits'], ascending=True) #send the columns as array to
```

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

```
birds['priority'] = birds['priority'].replace(['yes', 'no'],[1, 0])
birds
```

	birds	age	visits	priority
c	plovers	1.5	3	0
f	Cranes	3.0	4	0
a	Cranes	3.5	2	1
j	spoonbills	4.0	2	0
b	Cranes	4.0	4	1
g	plovers	5.5	2	0
e	spoonbills	6.0	3	0
i	spoonbills	8.0	3	0
h	Cranes	NaN	2	1
d	spoonbills	NaN	4	1

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

```
birds['birds'] = birds['birds'].replace(['Cranes'], ['trumpeters'])
birds
```

	birds	age	visits	priority
c	plovers	1.5	3	0
f	trumpeters	3.0	4	0
a	trumpeters	3.5	2	1
j	spoonbills	4.0	2	0
b	trumpeters	4.0	4	1
g	plovers	5.5	2	0
e	spoonbills	6.0	3	0
i	spoonbills	8.0	3	0
h	trumpeters	NaN	2	1
d	spoonbills	NaN	4	1

✓ 0s completed at 7:20 PM ● ✕