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 [399]:
```
import numpy as np
import pandas as pd
```

In [400]:
```
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 [401]:
```
df = pd.DataFrame(data, index = labels)
df
```

Out[401]:

	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 [402]:

```
print(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
None
```

In [403]:
```
print(df.info(verbose=True))
```

```
<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
None
```

3. Print the first 2 rows of the birds dataframe

In [404]:
```
df[:2]
```

Out[404]:

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

or

In [405]:
```
df.head(2)
```

Out[405]:

	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 [406]:
```
df[['birds','age']]
```

Out[406]:

	birds	age
a	Cranes	3.5
b	Cranes	4.0
c	plovers	1.5
d	spoonbills	NaN
e	spoonbills	6.0

	birds	age
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 [407]:

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

Out[407]:

	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 [408]:
```
df[df.visits < 4]
```

Out[408]:

	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 [409]:
```
null_finder = df.isna().any(axis=1)
null_finder
```

Out[409]:
```
a    False
b    False
c    False
d     True
e    False
f    False
```

```
g     False
h      True
i     False
j     False
dtype: bool
```

In [410]:
```
df[['birds', 'visits']][null_finder]
```

Out[410]:

	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 [411]:
```
df[df['birds'] == 'Cranes'][df['age']< 4]
```

```
C:\Users\mnr41\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Use
rWarning: Boolean Series key will be reindexed to match DataFrame inde
x.
  """Entry point for launching an IPython kernel.
```

Out[411]:

	birds	age	visits	priority
a	Cranes	3.5	2	yes

	birds	age	visits	priority
f	Cranes	3.0	4	no

9. Select the rows the age is between 2 and 4(inclusive)

In [412]:
```
df[df['age'].between(2,4)]
```

Out[412]:

	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 [413]:
```
df['visits'][df['birds']=="Cranes"].sum()
```

Out[413]:
```
12
```

11. Calculate the mean age for each different birds in

dataframe.

In [414]:
```
df['age'].mean()
```

Out[414]:
```
4.4375
```

In [415]:
```
df
```

Out[415]:

	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

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 [416]:
```
df.loc['k'] = ['spoonbills', 7, 4 , 'no']
```

In [417]:
```
df
```

Out[417]:

	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

	birds	age	visits	priority
j	spoonbills	4.0	2	no
k	spoonbills	7.0	4	no

drop row k

In [418]:

```
df = df.drop('k')
df
```

Out[418]:

	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

	birds	age	visits	priority
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 [419]:
```
bird_group = df.groupby('birds')
```

In [420]:
```
for i,j in bird_group:
    print("Threre are ",str(len(j)),i, " birds.")
```

```
Threre are  4 Cranes  birds.
Threre are  2 plovers  birds.
Threre are  4 spoonbills  birds.
```

using value_counts method:

For our case, value_counts method is more useful. This method will return the number of unique values for a particular column

In [421]:
```
df['birds'].value_counts()
```

Out[421]:
```
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 [422]:

```
sorted_by_age = df.sort_values(by='age',ascending=False)
sorted_by_age
```

Out[422]:

	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

In [423]:
```python
sorted_by_visits = df.sort_values(by='visits')
sorted_by_visits
```

Out[423]:

	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

In [424]:
```python
df
```

Out[424]:

	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

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

In [425]:
```
for_yes_mask = df['priority'] == 'yes'
column_name = 'priority'
for_no_mask = df['priority'] == 'no'
```

```
In [426]: # replace 'yes' with 1
          df.loc[for_yes_mask, column_name] = 1
```

```
In [427]: # replace 'yes' with 0
          df.loc[for_no_mask, column_name] = 0
```

```
In [428]: df
```

Out[428]:

	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 [429]:
```
crane_mask = df['birds'] == 'Cranes'
bird_column = 'birds'
```

In [430]:
```
# replace  the 'Cranes' entries to 'trumpeters'.
df.loc[crane_mask, bird_column] = 'trumpeters'
```

In [431]:
```
df
```

Out[431]:

	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

	birds	age	visits	priority
i	spoonbills	8.0	3	0
j	spoonbills	4.0	2	0

```
In [ ]:
```

```
In [ ]:
```