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 [2]:
```
import numpy as np
import pandas as pd
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 [14]:
```
#Reference https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
df=pd.DataFrame(data)
df_birds=pd.DataFrame(data['birds'],index=labels,columns=['birds'])
```

2. Display a summary of the basic information about birds DataFrame and its data.

In [3]:
```
print("Printing basic information about birds DataFrame ")
print(df_birds.describe())
print("----------------")
print("Printing birds data")
print(df_birds)
```

```
Printing basic information about birds DataFrame
             birds
count           10
unique           3
top     spoonbills
freq             4
----------------
Printing birds data
        birds
a      Cranes
b      Cranes
c     plovers
d  spoonbills
e  spoonbills
f      Cranes
g     plovers
h      Cranes
i  spoonbills
j  spoonbills
```

3. Print the first 2 rows of the birds dataframe

In [4]:
```
df_birds.head(2)
```

Out[4]:

	birds
a	Cranes
b	Cranes

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

In [5]:
```
df[['birds','age']]
```

Out[5]:

	birds	age
0	Cranes	3.5
1	Cranes	4.0
2	plovers	1.5
3	spoonbills	NaN
4	spoonbills	6.0
5	Cranes	3.0
6	plovers	5.5
7	Cranes	NaN
8	spoonbills	8.0
9	spoonbills	4.0

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

In [42]:
```
#reference https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
df[['birds','age','visits']].loc[[2,3,7]]
```

Out[42]:

	birds	age	visits
2	plovers	1.5	3
3	spoonbills	NaN	4
7	Cranes	NaN	2

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

In [45]:
```
df[df.visits<4]
```

Out[45]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
2	plovers	1.5	3	no
4	spoonbills	6.0	3	no
6	plovers	5.5	2	no
7	Cranes	NaN	2	yes
8	spoonbills	8.0	3	no
9	spoonbills	4.0	2	no

7. select the rows with columns ['birds', 'visits'] where the age is missing i.e NaN

In [69]:
```
#for missing function - I referred - Stack Overflow
df[['birds','visits']][df.age.isnull()]
#another code
df[['birds','visits']][df.age.isna()]
```

Out[69]:

	birds	visits
3	spoonbills	4
7	Cranes	2

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

In [80]:
```
#reference - https://www.geeksforgeeks.org/selecting-rows-in-pandas-dataframe-based-on-conditions/
df[(df.birds=='Cranes')&(df.age<4)]
```

Out[80]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
5	Cranes	3.0	4	no

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

In [4]:
```
#reference - https://www.geeksforgeeks.org/selecting-rows-in-pandas-dataframe-based-on-conditions/
age_flg=[2,3,4]
df[df.age.isin(age_flg)]
```

Out[4]:

	birds	age	visits	priority
1	Cranes	4.0	4	yes
5	Cranes	3.0	4	no
9	spoonbills	4.0	2	no

10. Find the total number of visits of the bird Cranes

In [13]:
```
df.visits[df.birds=='Cranes'].sum()
```

Out[13]: 12

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

In [34]:
```
g=df.groupby(df.birds)
g

for bird,bird_df in g:
    print("Bird:",bird)
    print("Mean age:",bird_df.age.mean())
```

```
Bird: Cranes
Mean age: 3.5
Bird: plovers
Mean age: 3.5
Bird: spoonbills
Mean age: 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 [110]:
```
#Reference - https://www.activestate.com/resources/quick-reads/how-to-delete-a-column-row-from-a-dataframe/
data=[['Cranes',4,2,'yes']]
new_df=pd.DataFrame(data,columns=['birds','age','visits','priority'])
#new_df.columns=['birds','age','visits','priority']
df=pd.concat([df,new_df],ignore_index=True)
df
df.drop(10,inplace=True)
df
```

Out[110]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
1	Cranes	4.0	4	yes
2	plovers	1.5	3	no
3	spoonbills	NaN	4	yes
4	spoonbills	6.0	3	no
5	Cranes	3.0	4	no
6	plovers	5.5	2	no
7	Cranes	NaN	2	yes
8	spoonbills	8.0	3	no
9	spoonbills	4.0	2	no

13. Find the number of each type of birds in dataframe (Counts)

In [142]:
```
#reference-https://datascienceparichay.com/article/pandas-groupby-count-of-rows-in-each-group/
print(df.groupby(df.birds).size())
```

```
birds
Cranes        4
plovers       2
spoonbills    4
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 [159]:
```
#reference - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html
df.sort_values(by=['age','visits'],ascending=[False,True],inplace=True)
df
```

Out[159]:

	birds	age	visits	priority
8	spoonbills	8.0	3	no
4	spoonbills	6.0	3	no
6	plovers	5.5	2	no
9	spoonbills	4.0	2	no
1	Cranes	4.0	4	yes
0	Cranes	3.5	2	yes
5	Cranes	3.0	4	no
2	plovers	1.5	3	no
7	Cranes	NaN	2	yes
3	spoonbills	NaN	4	yes

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

In [15]:
```
df.priority.replace(['yes','no'],[1,0],inplace=True)
df
```

Out[15]:

	birds	age	visits	priority
0	Cranes	3.5	2	1
1	Cranes	4.0	4	1
2	plovers	1.5	3	0
3	spoonbills	NaN	4	1
4	spoonbills	6.0	3	0
5	Cranes	3.0	4	0
6	plovers	5.5	2	0
7	Cranes	NaN	2	1
8	spoonbills	8.0	3	0
9	spoonbills	4.0	2	0

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

In [16]:
```
df.birds.replace('Cranes','trumpeters',inplace=True)
df
```

Out[16]:

	birds	age	visits	priority
0	trumpeters	3.5	2	1
1	trumpeters	4.0	4	1
2	plovers	1.5	3	0
3	spoonbills	NaN	4	1
4	spoonbills	6.0	3	0
5	trumpeters	3.0	4	0
6	plovers	5.5	2	0
7	trumpeters	NaN	2	1
8	spoonbills	8.0	3	0
9	spoonbills	4.0	2	0