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 [128]:

```
# Importing packages
import pandas as pd
import numpy as np

# creating data
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']}

# Creating Dataframe with index value as labels
df = pd.DataFrame(data, index =  ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'])
print("Dataframe is :\n", df)
```

```
Dataframe is :
    age       birds priority  visits
a  3.5      Cranes      yes       2
b  4.0      Cranes      yes       4
c  1.5     plovers       no       3
d  NaN  spoonbills      yes       4
e  6.0  spoonbills       no       3
f  3.0      Cranes       no       4
```

```
g  5.5     plovers       no       2
h  NaN      Cranes      yes       2
i  8.0  spoonbills       no       3
j  4.0  spoonbills       no       2
```

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

In [130]:

```
# Printing basic information for all the columns.
print("Basic information about birds DataFrame are : \n", df.describe(i
nclude = "all"))
```

```
Basic information about birds DataFrame are :
              age       birds priority     visits
count   8.000000          10       10  10.000000
unique       NaN           3        2        NaN
top          NaN  spoonbills       no        NaN
freq         NaN           4        6        NaN
mean    4.437500         NaN      NaN   2.900000
std     2.007797         NaN      NaN   0.875595
min     1.500000         NaN      NaN   2.000000
25%     3.375000         NaN      NaN   2.000000
50%     4.000000         NaN      NaN   3.000000
75%     5.625000         NaN      NaN   3.750000
max     8.000000         NaN      NaN   4.000000
```

3. Print the first 2 rows of the birds dataframe

In [131]:

```
# printing top 2 rows by applying slicing.
print("Top 2 rows of birds DataFrame are: \n", df[0:2])
```

```
Top 2 rows of birds DataFrame are:
    age   birds priority  visits
a  3.5  Cranes      yes       2
b  4.0  Cranes      yes       4
```

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

In [132]:
```
# printing only birds and age columns.
print(df[['birds', 'age']])
```

```
        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 [133]:
```
# printing by providing location of rows and columns
print(df.iloc[[2,3,7], [0,1,3]])
```

```
   age       birds  visits
c  1.5     plovers       3
d  NaN  spoonbills       4
h  NaN      Cranes       2
```

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

In [134]:
```
visits_filter = (df['visits'] < 4)           # Creating filter data
print("Rows with visits less tan 4 are : \n", df[visits_filter])
            # Printing filter data
```

```
Rows with visits less tan 4 are :
    age       birds priority  visits
a  3.5      Cranes      yes       2
c  1.5     plovers       no       3
e  6.0  spoonbills       no       3
g  5.5     plovers       no       2
h  NaN      Cranes      yes       2
```

```
i   8.0  spoonbills        no       3
j   4.0  spoonbills        no       2
```

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

In [135]:

```
age_missing = (df['age'].isnull())
print(df[age_missing])
```

```
   age       birds priority  visits
d  NaN  spoonbills      yes       4
h  NaN      Cranes      yes       2
```

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

In [136]:

```
filter_data = ((df['birds'] == "Cranes") & (visits_filter))
print(df[filter_data])
```

```
   age   birds priority  visits
a  3.5  Cranes      yes       2
h  NaN  Cranes      yes       2
```

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

In [137]:

```
filter_age = df['age'].between(2, 4, inclusive=True)
print(df[filter_age])
```

```
   age       birds priority  visits
a  3.5      Cranes      yes       2
b  4.0      Cranes      yes       4
f  3.0      Cranes       no       4
j  4.0  spoonbills       no       2
```

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

In [138]:

```
grp_by_birds = df.groupby('birds')                          # Creating Birds gr
oup
```

```
cranes_grp = grp_by_birds.get_group("Cranes")        # Getting specific
 Bird group
visits_sum = cranes_grp['visits'].sum()              # Calculating Sum

print("Total number of visits of the bird Cranes are: ", visits_sum)
```

```
Total number of visits of the bird Cranes are:  12
```

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

In [139]:
```
birds_mean = grp_by_birds.mean()
birds_age_mean = birds_mean.loc[:,['age']]            # selecting all ro
ws for column age.

print("The mean age for each different birds in dataframe are : \n\n",
birds_age_mean)
```

```
The mean age for each different birds in dataframe are :

             age
birds
Cranes      3.5
plovers     3.5
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 [140]:
```
print("Original DataFrame: \n", df)

new_row_data = {'birds': ['Parrot'], 'age' : [7], 'visits': [2], 'prior
ity': ['yes']}
new_row = pd.DataFrame(new_row_data, ['k'])
df = df.append(new_row)
print("\n DataFrame after appending row 'k': \n", df)

df = df.drop('k')
print("\n DataFrame after deleting row 'k' \n", df)
```

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

 DataFrame after appending row 'k':
    age       birds priority  visits
a  3.5      Cranes      yes        2
b  4.0      Cranes      yes        4
c  1.5     plovers       no        3
d  NaN  spoonbills      yes        4
e  6.0  spoonbills       no        3
f  3.0      Cranes       no        4
g  5.5     plovers       no        2
h  NaN      Cranes      yes        2
i  8.0  spoonbills       no        3
j  4.0  spoonbills       no        2
k  7.0      Parrot      yes        2

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

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

In [141]:
```
print("Number of each type of birds in dataframe are: \n", grp_by_birds
['birds'].count())
```

```
Number of each type of birds in dataframe are:
 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.

In [142]:
```
df_sort_by_des_age = df.sort_values(by='age', ascending=False)
print("Sorting data by age in Descneding Order: \n\n", df_sort_by_des_a
ge)
print("\nSorting data by visits in Ascending Order: \n\n", df_sort_by_d
es_age.sort_values(by='visits'))
```

```
Sorting data by age in Descneding Order:

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

Sorting data by visits in Ascending Order:
    age       birds priority  visits
g  5.5     plovers       no       2
j  4.0  spoonbills       no       2
a  3.5      Cranes      yes       2
h  NaN      Cranes      yes       2
i  8.0  spoonbills       no       3
e  6.0  spoonbills       no       3
c  1.5     plovers       no       3
b  4.0      Cranes      yes       4
f  3.0      Cranes       no       4
d  NaN  spoonbills      yes       4
```

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

In [143]:
```
df = df.replace(['yes'], 1)
df = df.replace(['no'], 0)
print("Replaced Dataframe: \n", df)
```

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

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

In [144]:
```
df = df.replace(['Cranes'], 'trumpeters')
print("Replaced DataFrame :\n", df)
```

```
Replaced DataFrame :
    age       birds  priority  visits
```

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