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, 1.5, 6, 3, 5.5, 5.5, 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 [5]:
```
import pandas as pd
```

In [6]:
```
import numpy as np
```

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

In [31]:
```
bird_data = [ ["Cranes",3.5,2,"yes"],
            ["Cranes",4,4,"yes"],
             ["plovers",1.5,3,"no"],
             ["spoonbills",np.NaN,4,"yes"],
             ["spoonbills",6,3,"no"],
             ["Cranes",3,4,"no"],
             ["plovers",5.5,2,"no"],
             ["Cranes",np.NaN,2,"yes"],
             ["spoonbills",8,3,"no"],
             ["spoonbills",4,2,"no"]

             ]
df=pd.DataFrame(bird_data,index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'], columns=['birds','ages'
,'visits','priority'] )
print(df)
```

```
        birds  ages  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 [8]:
```
df.describe()
```

Out[8]:

	ages	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

In [9]:
```
df.fillna(0)
```

Out[9]:

	birds	ages	visits	priority
a	Cranes	3.5	2	yes
b	Cranes	4.0	4	yes
c	plovers	1.5	3	no
d	spoonbills	0.0	4	yes
e	spoonbills	6.0	3	no
f	Cranes	3.0	4	no
g	plovers	5.5	2	no
h	Cranes	0.0	2	yes
i	spoonbills	8.0	3	no
j	spoonbills	4.0	2	no

3. Print the first 2 rows of the birds dataframe

In [10]:
```
df.head(2)
```

Out[10]:

	birds	ages	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 [11]:
```
df[['birds', 'ages']]
```

Out[11]:

	birds	ages
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 [12]:
```
print(df['birds'].iloc[2], df['ages'].iloc[2],df['visits'].iloc[2])
print(df['birds'].iloc[3], df['ages'].iloc[3],df['visits'].iloc[3])
print(df['birds'].iloc[7], df['ages'].iloc[7],df['visits'].iloc[7])

#reference from google
```

```
plovers 1.5 3
spoonbills nan 4
Cranes nan 2
```

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

In [13]:
```
df.loc[(df['visits']<4)]
```

Out[13]:

	birds	ages	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 [14]:
```
df[df['ages'].isnull()]
```

Out[14]:

	birds	ages	visits	priority
d	spoonbills	NaN	4	yes
h	Cranes	NaN	2	yes

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

In [15]:
```
df.loc[(df['birds'] == 'Cranes') & (df['ages']!=4)]
```

Out[15]:

	birds	ages	visits	priority
a	Cranes	3.5	2	yes
f	Cranes	3.0	4	no
h	Cranes	NaN	2	yes

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

In [16]:
```
print(df[(df['ages'] >= 2) & (df['ages'] <= 4)])
```

```
        birds  ages  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 [36]:
```
df[(df['birds'] == 'Cranes') & (df['visits'] > 0)].sum()
```

Out[36]:
```
birds       CranesCranesCranesCranes
ages                            10.5
visits                            12
priority                  yesyesnoyes
dtype: object
```

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

In [34]:
```
df.groupby('birds')['ages'].mean()
```

Out[34]:
```
birds
Cranes        3.5
plovers       3.5
spoonbills    6.0
Name: ages, 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.

In [20]:
```
#adding the new row
df.loc['k'] = ['eagle', 8,5, 'yes']

#printing the new table
df
```

Out[20]:

	birds	ages	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	eagle	8.0	5	yes

In [21]:
```
#deleting the newly added row
df = df.drop('k')


#printing the table
df
```

Out[21]:

	birds	ages	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 [22]:
```
df.count()
```

Out[22]:
```
birds       10
ages         8
visits      10
priority    10
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 [23]:
```
df.sort_values(by=['birds', 'visits'], ascending=[False, True])
```

Out[23]:

	birds	ages	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
g	plovers	5.5	2	no
c	plovers	1.5	3	no
a	Cranes	3.5	2	yes
h	Cranes	NaN	2	yes
b	Cranes	4.0	4	yes
f	Cranes	3.0	4	no

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

In [24]:
```
#code for replaceing the yes with 1
df['priority'] = df['priority'].replace('yes', 1)


# replacing the no with 0
df['priority'] = df['priority'].replace('no', 0)


#printing the table
df
```

Out[24]:

	birds	ages	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 [25]:
```
#changing the column name from Cranes to Trumpeters

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

df
```

Out[25]:

	birds	ages	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