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 [79]:
```
import numpy as np
import pandas as pd
```

In [80]:
```
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills', 'Cranes', 'plovers', 'Crane
s', '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 [81]:
```
data=pd.DataFrame(data)
label=pd.DataFrame(labels)
data['labels'] = labels
data= data.set_index('labels')
data
```

Out[81]:

labels	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 [82]:
```
data.info()
```

```
<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
```

OBSERVATION :

- In the age row there is 8 value so it means Nan value is not consider

3. Print the first 2 rows of the birds dataframe

In [83]:
```
df= data[['birds','age','visits','priority']].head(2)
df
```

Out[83]:

labels	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 [84]:
```
df1= data[['birds','age']]
df1
```

Out[84]:

labels	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 [85]:
```
print(data['birds'].iloc[2],data['age'].iloc[2],data['visits'].iloc[2])
print(data['birds'].iloc[3],data['age'].iloc[3],data['visits'].iloc[3])
print(data['birds'].iloc[7],data['age'].iloc[7],data['visits'].iloc[7])
```

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

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

In [86]:
```
data[data['visits'] < 4]
```

Out[86]:

labels	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 order to check null values in Pandas DataFrame, we use isnull() function
- this function return dataframe of Boolean values which are True for NaN values

In [87]:
```
data[data['age'].isnull()]
```

Out[87]:

labels	birds	age	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 [88]:
```
data[(data['birds'] == 'Cranes') & (data['age'] < 4)]
```

Out[88]:

labels	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)

In [89]:
```
data[(data['age'] <= 4) & (data['age'] >= 2)]
```

Out[89]:

labels	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 [90]:
```
data[(data['birds'] == 'Cranes') & (data['visits'] > 0)].sum()
```

Out[90]:
```
birds       CranesCranesCranesCranes
age                             10.5
visits                            12
priority                 yesyesnoyes
dtype: object
```

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

In [91]:
```
mean_age=data.groupby(['birds']).mean()['age']
print(mean_age)
```

```
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.

- Add dictionary as a row to dataframe.
- Add Series as a row in the dataframe.
- Add multiple rows to pandas dataframe.
- Add row from one dataframe to another dataframe.
- Add list as a row to pandas dataframe using loc[]
- Add a row in the dataframe at index position using iloc[]

In [92]:
```
data.loc['k'] = ['sparrow', 3.5, 3, 'yes']
data
```

Out[92]:

labels	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	sparrow	3.5	3	yes

In [93]:
```
data = data.drop('k')
data
```

Out[93]:

labels	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)

In [94]:
```
data["birds"].value_counts
```

Out[94]:
```
<bound method IndexOpsMixin.value_counts of labels
a        Cranes
b        Cranes
c       plovers
d    spoonbills
e    spoonbills
f        Cranes
g       plovers
h        Cranes
i    spoonbills
j    spoonbills
Name: birds, dtype: object>
```

In [95]:
```
data["birds"].value_counts()
```

Out[95]:
```
Cranes        4
spoonbills    4
plovers       2
Name: birds, dtype: int64
```

14.

In [96]:
```
data.sort_values(["age"], axis=0,ascending=[False], inplace=True)
display(data)
```

labels	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 [97]:
```
data.sort_values(["visits"], axis=0,ascending=[True], inplace=True)
display(data)
```

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

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

In [98]:
```
data["priority"].replace({"yes":1,"no":0}, inplace=True)
display(data)
```

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

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

In [102]:
```
data['birds'].replace({'Cranes':'trumpeters'},inplace=True)
display(data)
```

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