In [73]:
```
import numpy as np
import pandas as pd
from numpy import nan as NA
```

In [74]:
```
data = pd.DataFrame([[2., 4, 3.], [1., 5, NA], [NA, 2, NA], [NA, 6.5, 3.]
data
```

Out[74]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 2.0 | 4.0 | 3.0 |
| **1** | 1.0 | 5.0 | NaN |
| **2** | NaN | 2.0 | NaN |
| **3** | NaN | 6.5 | 3.0 |

In [75]:
```
#1. identify missing values and give boolean output

data.isnull()
```

Out[75]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | False | False | False |
| **1** | False | False | True |
| **2** | True | False | True |
| **3** | True | False | False |

In [76]:
```
#2. Replace the missing values

data.fillna(1)
```

Out[76]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 2.0 | 4.0 | 3.0 |
| **1** | 1.0 | 5.0 | 1.0 |
| **2** | 1.0 | 2.0 | 1.0 |
| **3** | 1.0 | 6.5 | 3.0 |

In [77]:
```
#3. Replace missing values with 0

data.fillna(0)
```

Out[77]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 2.0 | 4.0 | 3.0 |
| **1** | 1.0 | 5.0 | 0.0 |
| **2** | 0.0 | 2.0 | 0.0 |
| **3** | 0.0 | 6.5 | 3.0 |

In [78]:
```
#4. Replace missing values with the mean of each column

data.fillna(data.mean())
```

Out[78]:

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 2.0 | 4.0 | 3.0 |
| **1** | 1.0 | 5.0 | 3.0 |
| **2** | 1.5 | 2.0 | 3.0 |
| **3** | 1.5 | 6.5 | 3.0 |

In [79]:
```
#✓ Create DataFrame with two columns as “Column1’ and ‘Column2”. Use nump
#check if there are any duplicated values. If any remove the same

#df1 = pd.DataFrame(np.arange(10.).reshape((5,2)), columns = list('Column1

df1 = {'column1':pd.Series(['E', 'F', 'G', 'H', 'E']),'column2':pd.Series
df1
```

Out[79]:
```
{'column1': 0    E
 1    F
 2    G
 3    H
 4    E
 dtype: object,
 'column2': 0    3.0
 1    4.0
 2    5.0
 3    6.0
 4    2.0
 dtype: float64}
```

In [80]:
```
#df1.duplicated()
```

In [81]:
```
#df1.drop_duplicates()
```

In [82]:
```
# Create data = pd.Series([1., -888., 2., -999., -1000., 3.]). Replace all

data = pd.Series([1., -888., 2., -999., -1000., 3.])
data
```

Out[82]:
```
0       1.0
1    -888.0
2       2.0
3    -999.0
4   -1000.0
5       3.0
dtype: float64
```

In [83]:
```
data = data.replace(1., 8.0)

data = data.replace(-888., 2.0)

data = data.replace(2., 4.0)

data = data.replace(-999., 8.0)

data = data.replace(-1000., 2.0)

data = data.replace(3., 4.0)



data
```

Out[83]:
```
0    8.0
1    4.0
2    4.0
3    8.0
4    2.0
5    4.0
dtype: float64
```

In [84]:
```
#df1 = pd.DataFrame({'key': ['g', 'b', 'a', 'c', 'a', 'b', 'b'],'data1': r
#df2 = pd.DataFrame({'key': ['a', 'b', 'd'],'data2': range(3)})
#Merge above two dataset and do Inner Join, Outer Join, Left Join and Left

df1 = pd.DataFrame({'key': ['g', 'b', 'a', 'c', 'a', 'b', 'b'],'data1': ra
df1
```

Out[84]:

| | key | data1 |
|---|---|---|
| **0** | g | 0 |
| **1** | b | 1 |
| **2** | a | 2 |
| **3** | c | 3 |
| **4** | a | 4 |
| **5** | b | 5 |
| **6** | b | 6 |

In [85]:
```
df2 = pd.DataFrame({'key': ['a', 'b', 'd'],'data2': range(3)})
df2
```

Out[85]:

| | key | data2 |
|---|---|---|
| **0** | a | 0 |
| **1** | b | 1 |
| **2** | d | 2 |

In [86]:
```
#Merge above two dataset and do Inner Join, Outer Join, Left Join and Left

pd.merge(df1, df2)
```

Out[86]:

| | key | data1 | data2 |
|---|---|---|---|
| **0** | b | 1 | 1 |
| **1** | b | 5 | 1 |
| **2** | b | 6 | 1 |
| **3** | a | 2 | 0 |
| **4** | a | 4 | 0 |

In [87]:
```
#Outer join

pd.merge(df1, df2, how='outer')
```

Out[87]:

| | key | data1 | data2 |
|---|---|---|---|
| **0** | g | 0.0 | NaN |
| **1** | b | 1.0 | 1.0 |
| **2** | b | 5.0 | 1.0 |
| **3** | b | 6.0 | 1.0 |
| **4** | a | 2.0 | 0.0 |
| **5** | a | 4.0 | 0.0 |
| **6** | c | 3.0 | NaN |
| **7** | d | NaN | 2.0 |

In [88]:
```
#Left join

pd.merge(df1, df2, on = 'key', how = 'left')
```

Out[88]:

| | key | data1 | data2 |
|---|---|---|---|
| **0** | g | 0 | NaN |
| **1** | b | 1 | 1.0 |
| **2** | a | 2 | 0.0 |
| **3** | c | 3 | NaN |
| **4** | a | 4 | 0.0 |
| **5** | b | 5 | 1.0 |
| **6** | b | 6 | 1.0 |

In [92]:
```
#Use data = pd.DataFrame(np.arange(6).reshape((2, 3)), index=pd.Index(['K

data = pd.DataFrame(np.arange(6).reshape((2, 3)),
index=pd.Index(['Kolkata', 'Chennai'], name='state'), columns=pd.Index(['
'three'], name='number'))
```

In [96]:
```
#Stack

result = data.stack()
result
```

Out[96]:
```
state    number
Kolkata  one       0
         two       1
         three     2
Chennai  one       3
         two       4
         three     5
dtype: int32
```

In [97]:
```
#Unstackresult

result.unstack()
```

Out[97]:

| number | one | two | three |
|---|---|---|---|
| state | | | |
| **Kolkata** | 0 | 1 | 2 |
| **Chennai** | 3 | 4 | 5 |