In [1]:
```
# inspecting,cleansing, transforming,modeling data
# goal discovering useful information
# conclusion
# and support decison making
```

In [2]:
```
# panel data, created by wes mckinney 2008
# cleaning, exploring and manipulating data

```

In [3]:
```
# # data structure
#  series
#     dataframe
#     panel:- 3d data container
```

In [4]:
```
# big data, data scentist, handling missing data, nana, size mutabilit
# high dimensional objecyt
# data set merging and joining, reshapingm and pivoiting
```

In [5]:
```
# series:- 1d array, can srore various data types
```

In [6]:
```
import pandas as pd
```

series

In [7]:
```
a = [2123,24,35,46,7,68,76]

serie = pd.Series(a)
print(serie)
```

```
0    2123
1      24
2      35
3      46
4       7
5      68
6      76
dtype: int64
```

In [8]:
```
print(type(serie))
```

```
<class 'pandas.core.series.Series'>
```

In [11]:
```
serie[3]
```

Out[11]:
```
46
```

In [13]:
```
a = [2123,24,35,46,7,68,76]

serie = pd.Series(a,index = range(1,len(a)+1))
print(serie)
```

```
1    2123
2      24
3      35
4      46
5       7
6      68
7      76
dtype: int64
```

In [14]:
```
a = [2123,24,35,46,7,68,76]

serie = pd.Series(a,index = range(1,len(a)+1),dtype='float')
print(serie)
```

```
1    2123.0
2      24.0
3      35.0
4      46.0
5       7.0
6      68.0
7      76.0
dtype: float64
```

In [15]:
```
a = [2123,24,35,46,7,68,76]

serie = pd.Series(a,index = range(1,len(a)+1),dtype='float',name='worl
print(serie)
```

```
1    2123.0
2      24.0
3      35.0
4      46.0
5       7.0
6      68.0
7      76.0
Name: world, dtype: float64
```

In [16]:
```
abc_dict = {'name':['a','b','c'],'id':[1,2,3]}

var_1 = pd.Series(abc_dict)
var_1
```

Out[16]:
```
name    [a, b, c]
id      [1, 2, 3]
dtype: object
```

In [17]:
```
var_1['name']
```

Out[17]:
```
['a', 'b', 'c']
```

In [18]:
```
a = pd.Series(123)
a
```

Out[18]:
```
0    123
dtype: int64
```

In [19]:
```
a[0]
```

Out[19]:
```
123
```

In [20]:
```
abc = pd.Series(123,index=range(1,6))
abc
```

Out[20]:
```
1    123
2    123
3    123
4    123
5    123
dtype: int64
```

In [28]:
```
a = pd.Series(123,index=range(1,6))
b = pd.Series(123,index=range(1,6))

print(a+b)
```

```
1    246
2    246
3    246
4    246
5    246
dtype: int64
```

In [29]:
```
a = pd.Series(123,index=range(1,6))
b = pd.Series(123,index=range(4,9))

print(a+b)
```

```
1      NaN
2      NaN
3      NaN
4    246.0
5    246.0
6      NaN
7      NaN
8      NaN
dtype: float64
```

DataFrame

In [30]:
```
# 2d data structure
```

In [33]:
```
a = [1,2,3,45,5,6]
abc = pd.DataFrame(a)

print(type(abc))
abc
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[33]:

	0
0	1
1	2
2	3
3	45
4	5
5	6

In [36]:
```
pq = {'A':[12,3,4,5,56,7],
     'B':[23,4,4,345,46,54]}

df = pd.DataFrame(pq)
df
```

Out[36]:

	A	B
0	12	23
1	3	4
2	4	4
3	5	345
4	56	46
5	7	54

In [38]:
```
pq = {'A':[12,3,4,5,56,7],
     'B':[23,4,4,345,46,54]}

df = pd.DataFrame(pq,columns = ['A'])
df
```

Out[38]:

	A
0	12
1	3
2	4
3	5
4	56
5	7

In [40]:
```
pq = {'A':[12,3,4,5,56,7],
     'B':[23,4,4,345,46,54],
     'C':[1,323,24,4,454,6]}

df = pd.DataFrame(pq,columns = ['A','C'])
df
```

Out[40]:
```
    A    C
0  12    1
1   3  323
2   4   24
3   5    4
4  56  454
5   7    6
```

In [41]:
```
pq = {'A':[12,3,4,5,56,7],
     'B':[23,4,4,345,46,54],
     'C':[1,323,24,4,454,6]}

df = pd.DataFrame(pq,columns = ['A','C'],index=['P','Q','R','S','T','U
df
```

Out[41]:
```
    A    C
P  12    1
Q   3  323
R   4   24
S   5    4
T  56  454
U   7    6
```

In [43]:
```
df['A'][2]
```

Out[43]: 4

In [44]:
```
# with list

lst = [[1,2,3,4,5],[6,7,8,9,10]]

var = pd.DataFrame(lst)
var
```

Out[44]:
```
   0  1  2  3   4
0  1  2  3  4   5
1  6  7  8  9  10
```

In [46]:
```
# with series
abc = {'A':pd.Series([3,4,5,6]),
       'B':pd.Series([6,7,8,9])}

df = pd.DataFrame(abc)
df
```

Out[46]:
```
   A  B
0  3  6
1  4  7
2  5  8
3  6  9
```

arithmetic operations in pandas

In [47]:
```
abc = pd.DataFrame({"A":[1,2,3,4],
                    "B":[5,6,7,8]})
abc
```

Out[47]:
```
   A  B
0  1  5
1  2  6
2  3  7
3  4  8
```

In [48]:
```
abc['C'] = abc['A'] + abc['B']
abc
```

Out[48]:
```
   A  B   C
0  1  5   6
1  2  6   8
2  3  7  10
3  4  8  12
```

In [49]:
```
abc['C'] = abc['A'] - abc['B']
abc
```

Out[49]:
```
   A  B   C
0  1  5  -4
1  2  6  -4
2  3  7  -4
3  4  8  -4
```

In [50]:
```
abc['C'] = abc['A'] * abc['B']
abc
```

Out[50]:
```
   A  B   C
0  1  5   5
1  2  6  12
2  3  7  21
3  4  8  32
```

In [51]:
```
abc['C'] = abc['A'] / abc['B']
abc
```

Out[51]:
```
   A  B         C
0  1  5  0.200000
1  2  6  0.333333
2  3  7  0.428571
3  4  8  0.500000
```

In [52]:
```
abc['C'] = abc['A'] ** abc['B']
abc
```

Out[52]:
```
   A  B      C
0  1  5      1
1  2  6     64
2  3  7   2187
3  4  8  65536
```

In [53]:
```
abc['C'] = abc['A'] % abc['B']
abc
```

Out[53]:
```
   A  B  C
0  1  5  1
1  2  6  2
2  3  7  3
3  4  8  4
```

In [54]:
```
abc['C'] = abc['A'] // abc['B']
abc
```

Out[54]:
```
   A  B  C
0  1  5  0
1  2  6  0
2  3  7  0
3  4  8  0
```

In [55]:
```
abc['C'] = abc['A'] & abc['B']
abc
```

Out[55]:
```
   A  B  C
0  1  5  1
1  2  6  2
2  3  7  3
3  4  8  0
```

In [60]:
```
df = pd.DataFrame({'One':[1,2,30,40,30,6,7,8],
                  'Two':[7,6,5,40,30,2,1,6],
                  'Three':[6,56,40,5,70,70,6,5]})
df
```

Out[60]:
```
   One  Two  Three
0    1    7      6
1    2    6     56
2   30    5     40
3   40   40      5
4   30   30     70
5    6    2     70
6    7    1      6
7    8    6      5
```

In [62]:
```
df[df['One']==30]
```

Out[62]:
```
   One  Two  Three
2   30    5     40
4   30   30     70
```

In [64]:

```
df['Above'] = df['One']>20
df["Below"] = df['Two']<50
```

In [65]:

```
df
```

Out[65]:

	One	Two	Three	Above	Below
0	1	7	6	False	True
1	2	6	56	False	True
2	30	5	40	True	True
3	40	40	5	True	True
4	30	30	70	True	True
5	6	2	70	False	True
6	7	1	6	False	True
7	8	6	5	False	True

insert

In [66]:

```
df = pd.DataFrame({'1':[1,2,3,4,556],
                   '2':[3,4,5,6,7]})
df
```

Out[66]:

	1	2
0	1	3
1	2	4
2	3	5
3	4	6
4	556	7

In [67]:

```
# insert column with value
df.insert(1,'C',pd.Series([4,5,6,7,8]))
```

In [68]:

```
df
```

Out[68]:

	1	C	2
0	1	4	3
1	2	5	4
2	3	6	5
3	4	7	6
4	556	8	7

In [69]:

```
df['D'] = df['C'][0:4]
```

In [70]:

```
df
```

Out[70]:

	1	C	2	D
0	1	4	3	4.0
1	2	5	4	5.0
2	3	6	5	6.0
3	4	7	6	7.0
4	556	8	7	NaN

In [71]:

```
# delete, #pop
```

In [72]:

```
df.pop('D')
```

Out[72]:

```
0    4.0
1    5.0
2    6.0
3    7.0
4    NaN
Name: D, dtype: float64
```

In [73]:

```
df
```

Out[73]:

	1	C	2
0	1	4	3
1	2	5	4
2	3	6	5
3	4	7	6
4	556	8	7

In [74]:

```
del df['1']
```

In [75]:

```
df
```

Out[75]:

	C	2
0	4	3
1	5	4
2	6	5
3	7	6
4	8	7

csv

In [76]:

```
# plain text,comma seperated values
# Excel:- binary data
```

In [77]:

```
df
```

Out[77]:

	C	2
0	4	3
1	5	4
2	6	5
3	7	6
4	8	7

In [78]:

```
df.to_csv('df.csv')
```

In [79]:

```
df.to_csv('without_index_df.csv',index=False)
```

In [80]:

```
# change header

df.to_csv('change_header.csv',header=['G','M'])
```

read csv

In [81]:

```
import glob
```

In [98]:

```
glob.glob('*.csv')
```

Out[98]:

```
['change_header.csv', 'df.csv', 'without_index_df.csv']
```

In [101]:

```
df = pd.read_csv('without_index_df.csv',nrows=2)
df
```

Out[101]:

	C	2
0	4	3
1	5	4

In [102]:

```
print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [106]:

```
df = pd.read_csv('AXISBANK.csv',usecols=['Date','Close','Open'])
df
```

Out[106]:

	Date	Open	Close
0	2000-01-03	26.7	26.70
1	2000-01-04	27.0	26.85
2	2000-01-05	26.0	26.30
3	2000-01-06	25.8	25.95
4	2000-01-07	25.0	24.80
...	...	...	...
5301	2021-04-26	694.0	700.45
5302	2021-04-27	691.1	699.55
5303	2021-04-28	708.0	708.15
5304	2021-04-29	712.0	719.40
5305	2021-04-30	705.0	714.90

5306 rows × 3 columns

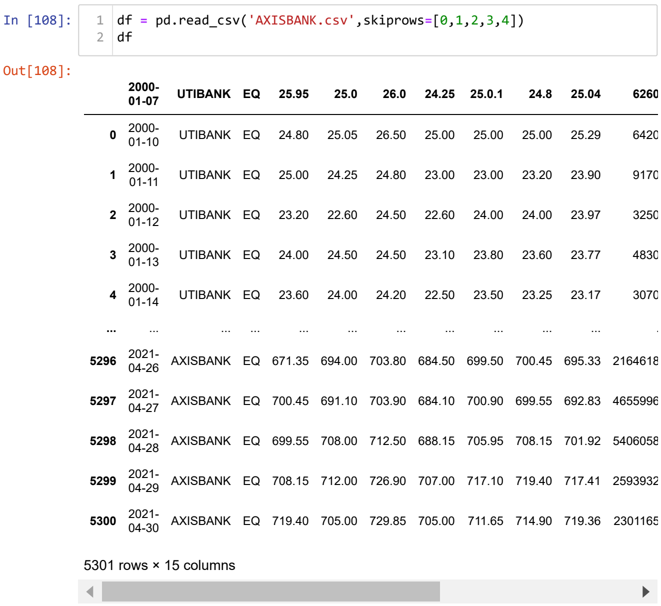
In [107]:

```
df = pd.read_csv('AXISBANK.csv',usecols=[0,3,6])
df
```

Out[107]:

	Date	Prev Close	Low
0	2000-01-03	24.70	26.70
1	2000-01-04	26.70	26.50
2	2000-01-05	26.85	25.50
3	2000-01-06	26.30	25.80
4	2000-01-07	25.95	24.25
...	...	...	...
5301	2021-04-26	671.35	684.50
5302	2021-04-27	700.45	684.10
5303	2021-04-28	699.55	688.15
5304	2021-04-29	708.15	707.00
5305	2021-04-30	719.40	705.00

5306 rows × 3 columns



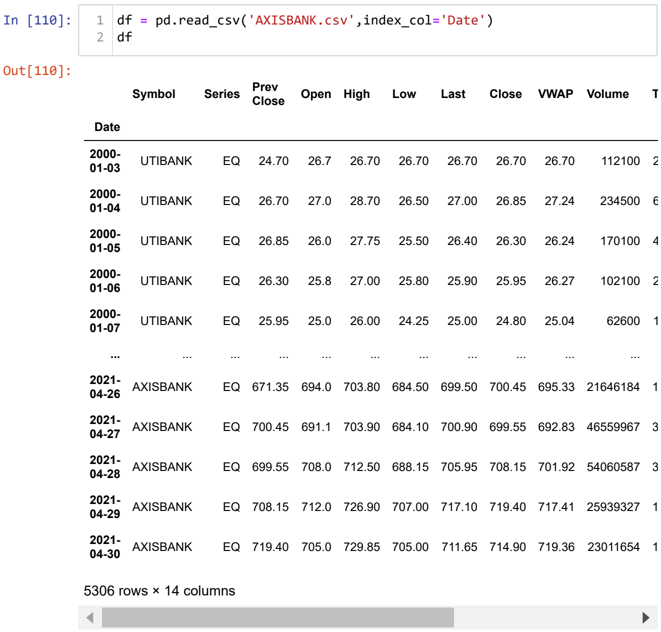
In [108]:

```python
df = pd.read_csv('AXISBANK.csv',skiprows=[0,1,2,3,4])
df
```

Out[108]:

	2000-01-07	UTIBANK	EQ	25.95	25.0	26.0	24.25	25.0.1	24.8	25.04	6260
0	2000-01-10	UTIBANK	EQ	24.80	25.05	26.50	25.00	25.00	25.00	25.29	6420
1	2000-01-11	UTIBANK	EQ	25.00	24.25	24.80	23.00	23.00	23.20	23.90	9170
2	2000-01-12	UTIBANK	EQ	23.20	22.60	24.50	22.60	24.00	24.00	23.97	3250
3	2000-01-13	UTIBANK	EQ	24.00	24.50	24.50	23.10	23.80	23.60	23.77	4830
4	2000-01-14	UTIBANK	EQ	23.60	24.00	24.20	22.50	23.50	23.25	23.17	3070
...	...	...	...	...	...	...	...	...	...	...	.
5296	2021-04-26	AXISBANK	EQ	671.35	694.00	703.80	684.50	699.50	700.45	695.33	2164618
5297	2021-04-27	AXISBANK	EQ	700.45	691.10	703.90	684.10	700.90	699.55	692.83	4655996
5298	2021-04-28	AXISBANK	EQ	699.55	708.00	712.50	688.15	705.95	708.15	701.92	5406058
5299	2021-04-29	AXISBANK	EQ	708.15	712.00	726.90	707.00	717.10	719.40	717.41	2593932
5300	2021-04-30	AXISBANK	EQ	719.40	705.00	729.85	705.00	711.65	714.90	719.36	2301165

5301 rows × 15 columns



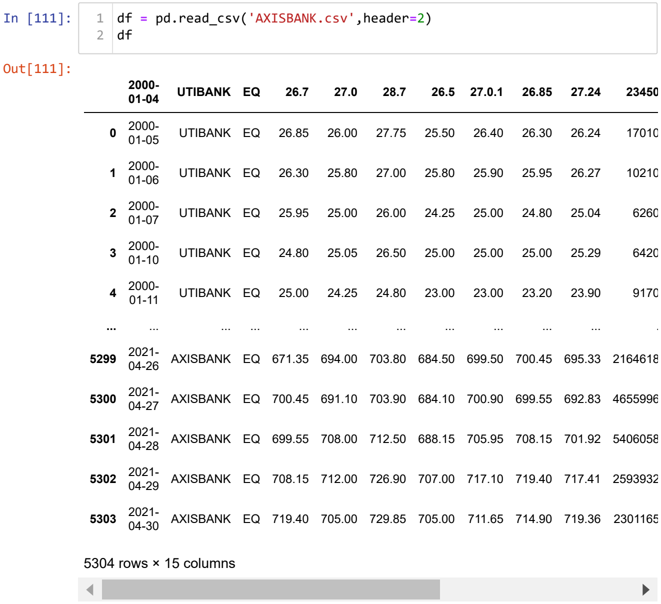
In [110]:

```python
df = pd.read_csv('AXISBANK.csv',index_col='Date')
df
```

Out[110]:

	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	T
Date											
2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112100	2
2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234500	6
2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170100	4
2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102100	2
2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62600	1
...	...	...	...	...	...	...	...	...	...	...	
2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646184	1
2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559967	3
2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060587	3
2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939327	1
2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011654	1

5306 rows × 14 columns



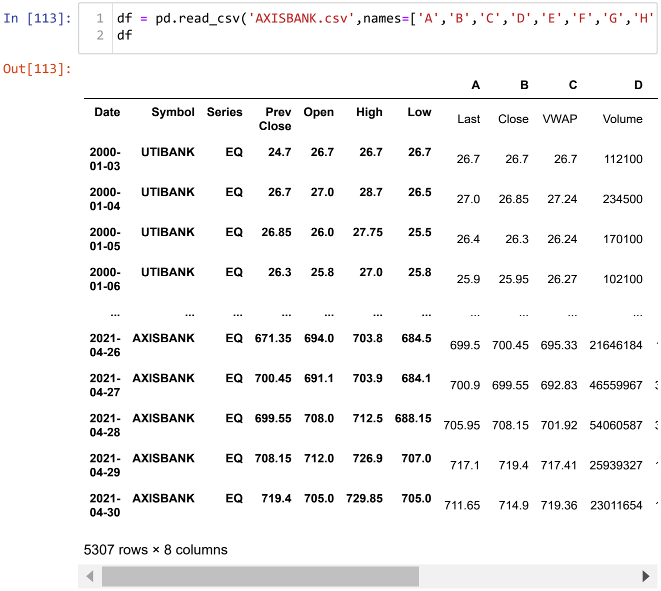
In [111]:

```python
df = pd.read_csv('AXISBANK.csv',header=2)
df
```

Out[111]:

	2000-01-04	UTIBANK	EQ	26.7	27.0	28.7	26.5	27.0.1	26.85	27.24	23450
0	2000-01-05	UTIBANK	EQ	26.85	26.00	27.75	25.50	26.40	26.30	26.24	17010
1	2000-01-06	UTIBANK	EQ	26.30	25.80	27.00	25.80	25.90	25.95	26.27	10210
2	2000-01-07	UTIBANK	EQ	25.95	25.00	26.00	24.25	25.00	24.80	25.04	6260
3	2000-01-10	UTIBANK	EQ	24.80	25.05	26.50	25.00	25.00	25.00	25.29	6420
4	2000-01-11	UTIBANK	EQ	25.00	24.25	24.80	23.00	23.00	23.20	23.90	9170
...	...	...	...	...	...	...	...	...	...	...	.
5299	2021-04-26	AXISBANK	EQ	671.35	694.00	703.80	684.50	699.50	700.45	695.33	2164618
5300	2021-04-27	AXISBANK	EQ	700.45	691.10	703.90	684.10	700.90	699.55	692.83	4655996
5301	2021-04-28	AXISBANK	EQ	699.55	708.00	712.50	688.15	705.95	708.15	701.92	5406058
5302	2021-04-29	AXISBANK	EQ	708.15	712.00	726.90	707.00	717.10	719.40	717.41	2593932
5303	2021-04-30	AXISBANK	EQ	719.40	705.00	729.85	705.00	711.65	714.90	719.36	2301165

5304 rows × 15 columns



In [113]:

```python
df = pd.read_csv('AXISBANK.csv',names=['A','B','C','D','E','F','G','H'
df
```

Out[113]:

							A	B	C	D
Date	**Symbol**	**Series**	**Prev Close**	**Open**	**High**	**Low**	Last	Close	VWAP	Volume
2000-01-03	**UTIBANK**	**EQ**	**24.7**	**26.7**	**26.7**	**26.7**	26.7	26.7	26.7	112100
2000-01-04	**UTIBANK**	**EQ**	**26.7**	**27.0**	**28.7**	**26.5**	27.0	26.85	27.24	234500
2000-01-05	**UTIBANK**	**EQ**	**26.85**	**26.0**	**27.75**	**25.5**	26.4	26.3	26.24	170100
2000-01-06	**UTIBANK**	**EQ**	**26.3**	**25.8**	**27.0**	**25.8**	25.9	25.95	26.27	102100
...	**...**	**...**	**...**	**...**	**...**	**...**	...	...	...	...
2021-04-26	**AXISBANK**	**EQ**	**671.35**	**694.0**	**703.8**	**684.5**	699.5	700.45	695.33	21646184
2021-04-27	**AXISBANK**	**EQ**	**700.45**	**691.1**	**703.9**	**684.1**	700.9	699.55	692.83	46559967
2021-04-28	**AXISBANK**	**EQ**	**699.55**	**708.0**	**712.5**	**688.15**	705.95	708.15	701.92	54060587
2021-04-29	**AXISBANK**	**EQ**	**708.15**	**712.0**	**726.9**	**707.0**	717.1	719.4	717.41	25939327
2021-04-30	**AXISBANK**	**EQ**	**719.4**	**705.0**	**729.85**	**705.0**	711.65	714.9	719.36	23011654

5307 rows × 8 columns

In [114]:

```
df = pd.read_csv('AXISBANK.csv',header = None)
df
```

Out[114]:

	0	1	2	3	4	5	6	7	8	9	
0	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
1	2000-01-03	UTIBANK	EQ	24.7	26.7	26.7	26.7	26.7	26.7	26.7	112
2	2000-01-04	UTIBANK	EQ	26.7	27.0	28.7	26.5	27.0	26.85	27.24	234
3	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.5	26.4	26.3	26.24	170
4	2000-01-06	UTIBANK	EQ	26.3	25.8	27.0	25.8	25.9	25.95	26.27	102
...	...	...	...	...	...	...	...	...	...	...	
5302	2021-04-26	AXISBANK	EQ	671.35	694.0	703.8	684.5	699.5	700.45	695.33	21646
5303	2021-04-27	AXISBANK	EQ	700.45	691.1	703.9	684.1	700.9	699.55	692.83	46559
5304	2021-04-28	AXISBANK	EQ	699.55	708.0	712.5	688.15	705.95	708.15	701.92	54060
5305	2021-04-29	AXISBANK	EQ	708.15	712.0	726.9	707.0	717.1	719.4	717.41	25939
5306	2021-04-30	AXISBANK	EQ	719.4	705.0	729.85	705.0	711.65	714.9	719.36	23011

5307 rows × 15 columns

In [115]:

```
df = pd.read_csv('AXISBANK.csv',header = None,prefix='A')
df
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\4153800144.py:1: Futur
eWarning: The prefix argument has been deprecated and will be removed in
a future version. Use a list comprehension on the column names in the fut
ure.


  df = pd.read_csv('AXISBANK.csv',header = None,prefix='A')
```

Out[115]:

	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A
0	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
1	2000-01-03	UTIBANK	EQ	24.7	26.7	26.7	26.7	26.7	26.7	26.7	112
2	2000-01-04	UTIBANK	EQ	26.7	27.0	28.7	26.5	27.0	26.85	27.24	234
3	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.5	26.4	26.3	26.24	170
4	2000-01-06	UTIBANK	EQ	26.3	25.8	27.0	25.8	25.9	25.95	26.27	102
...	...	...	...	...	...	...	...	...	...	...	
5302	2021-04-26	AXISBANK	EQ	671.35	694.0	703.8	684.5	699.5	700.45	695.33	21646
5303	2021-04-27	AXISBANK	EQ	700.45	691.1	703.9	684.1	700.9	699.55	692.83	46559
5304	2021-04-28	AXISBANK	EQ	699.55	708.0	712.5	688.15	705.95	708.15	701.92	54060
5305	2021-04-29	AXISBANK	EQ	708.15	712.0	726.9	707.0	717.1	719.4	717.41	25939
5306	2021-04-30	AXISBANK	EQ	719.4	705.0	729.85	705.0	711.65	714.9	719.36	23011

5307 rows × 15 columns

In [124]:

```
df = pd.read_csv('AXISBANK.csv',dtype={"Volume":'float64'})
df
```

Out[124]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Vo
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

Pandas Function

In [125]:

```
df = pd.read_csv('AXISBANK.csv')
df
```

Out[125]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [126]:

```
df.index
```

Out[126]:

```
RangeIndex(start=0, stop=5306, step=1)
```

In [127]:

```
print(dir(df))
```

```
['Close', 'Date', 'High', 'Last', 'Low', 'Open', 'Series', 'Symbol', 'T',
'Trades', 'Turnover', 'VWAP', 'Volume', '_AXIS_LEN', '_AXIS_ORDERS', '_AX
IS_TO_AXIS_NUMBER', '_HANDLED_TYPES', '__abs__', '__add__', '__and__', '_
_annotations__', '__array__', '__array_priority__', '__array_ufunc__', '_
_array_wrap__', '__bool__', '__class__', '__contains__', '__copy__', '__d
ataframe__', '__deepcopy__', '__delattr__', '__delitem__', '__dict__', '_
_dir__', '__divmod__', '__doc__', '__eq__', '__finalize__', '__floordiv_
_', '__format__', '__ge__', '__getattr__', '__getattribute__', '__getitem
__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__iand__', '__ifl
oordiv__', '__imod__', '__imul__', '__init__', '__init_subclass__', '__in
vert__', '__ior__', '__ipow__', '__isub__', '__iter__', '__itruediv__',
'__ixor__', '__le__', '__len__', '__lt__', '__matmul__', '__mod__', '__mo
dule__', '__mul__', '__ne__', '__neg__', '__new__', '__nonzero__', '__or_
_', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduc
e__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rmatmul__', '__rmo
d__', '__rmul__', '__ror__', '__round__', '__rpow__', '__rsub__', '__rtru
ediv__', '__rxor__', '__setattr__', '__setitem__', '__setstate__', '__siz
eof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__weakr
ef__', '__xor__', '_accessors', '_accum_func', '_add_numeric_operations',
'_agg_by_level', '_agg_examples_doc', '_agg_summary_and_see_also_doc', '_
align_frame', '_align_series', '_append', '_arith_method', '_as_manager',
'_attrs', '_box_col_values', '_can_fast_transpose', '_check_inplace_and_a
llows_duplicate_labels', '_check_inplace_setting', '_check_is_chained_ass
ignment_possible', '_check_label_or_level_ambiguity', '_check_setitem_cop
y', '_clear_item_cache', '_clip_with_one_bound', '_clip_with_scalar', '_c
mp_method', '_combine_frame', '_consolidate', '_consolidate_inplace', '_c
onstruct_axes_dict', '_construct_axes_from_arguments', '_construct_resul
t', '_constructor', '_constructor_sliced', '_convert', '_count_level', '_
data', '_dir_additions', '_dir_deletions', '_dispatch_frame_op', '_drop_a
xis', '_drop_labels_or_levels', '_ensure_valid_index', '_find_valid_inde
x', '_flags', '_from_arrays', '_get_agg_axis', '_get_axis', '_get_axis_na
me', '_get_axis_number', '_get_axis_resolvers', '_get_block_manager_axi
s', '_get_bool_data', '_get_cleaned_column_resolvers', '_get_column_arra
y', '_get_index_resolvers', '_get_item_cache', '_get_label_or_level_value
s', '_get_numeric_data', '_get_value', '_getitem_bool_array', '_getitem_m
ultilevel', '_gotitem', '_hidden_attrs', '_indexed_same', '_info_axis',
'_info_axis_name', '_info_axis_number', '_info_repr', '_init_mgr', '_inpl
ace_method', '_internal_names', '_internal_names_set', '_is_copy', '_is_h
omogeneous_type', '_is_label_or_level_reference', '_is_label_reference',
'_is_level_reference', '_is_mixed_type', '_is_view', '_iset_item', '_iset
_item_mgr', '_iset_not_inplace', '_item_cache', '_iter_column_arrays', '_
ixs', '_join_compat', '_logical_func', '_logical_method', '_maybe_cache_c
hanged', '_maybe_update_cacher', '_metadata', '_mgr', '_min_count_stat_fu
nction', '_needs_reindex_multi', '_protect_consolidate', '_reduce', '_red
uce_axis1', '_reindex_axes', '_reindex_columns', '_reindex_index', '_rein
dex_multi', '_reindex_with_indexers', '_rename', '_replace_columnwise',
'_repr_data_resource_', '_repr_fits_horizontal_', '_repr_fits_vertical_',
'_repr_html_', '_repr_latex_', '_reset_cache', '_reset_cacher', '_sanitiz
e_column', '_series', '_set_axis', '_set_axis_name', '_set_axis_nocheck',
'_set_is_copy', '_set_item', '_set_item_frame_value', '_set_item_mgr', '_
set_value', '_setitem_array', '_setitem_frame', '_setitem_slice', '_slic
e', '_stat_axis', '_stat_axis_name', '_stat_axis_number', '_stat_functio
n', '_stat_function_ddof', '_take', '_take_with_is_copy', '_to_dict_of_bl
ocks', '_typ', '_update_inplace', '_validate_dtype', '_values', '_where',
'abs', 'add', 'add_prefix', 'add_suffix', 'agg', 'aggregate', 'align', 'a
ll', 'any', 'append', 'apply', 'applymap', 'asfreq', 'asof', 'assign', 'a
stype', 'at', 'at_time', 'attrs', 'axes', 'backfill', 'between_time', 'bf
ill', 'bool', 'boxplot', 'clip', 'columns', 'combine', 'combine_first',
'compare', 'convert_dtypes', 'copy', 'corr', 'corrwith', 'count', 'cov',
'cummax', 'cummin', 'cumprod', 'cumsum', 'describe', 'diff', 'div', 'divi
de', 'dot', 'drop', 'drop_duplicates', 'droplevel', 'dropna', 'dtypes',
'duplicated', 'empty', 'eq', 'equals', 'eval', 'ewm', 'expanding', 'explo
de', 'ffill', 'fillna', 'filter', 'first', 'first_valid_index', 'flags',
'floordiv', 'from_dict', 'from_records', 'ge', 'get', 'groupby', 'gt', 'h
ead', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc', 'index', 'infer_object
s', 'info', 'insert', 'interpolate', 'isetitem', 'isin', 'isna', 'isnul
l', 'items', 'iteritems', 'iterrows', 'itertuples', 'join', 'keys', 'kur
t', 'kurtosis', 'last', 'last_valid_index', 'le', 'loc', 'lookup', 'lt',
'mad', 'mask', 'max', 'mean', 'median', 'melt', 'memory_usage', 'merge',
'min', 'mod', 'mode', 'mul', 'multiply', 'ndim', 'ne', 'nlargest', 'notn
a', 'notnull', 'nsmallest', 'nunique', 'pad', 'pct_change', 'pipe', 'pivo
t', 'pivot_table', 'plot', 'pop', 'pow', 'prod', 'product', 'quantile',
'query', 'radd', 'rank', 'rdiv', 'reindex', 'reindex_like', 'rename', 're
name_axis', 'reorder_levels', 'replace', 'resample', 'reset_index', 'rflo
ordiv', 'rmod', 'rmul', 'rolling', 'round', 'rpow', 'rsub', 'rtruediv',
'sample', 'select_dtypes', 'sem', 'set_axis', 'set_flags', 'set_index',
'shape', 'shift', 'size', 'skew', 'slice_shift', 'sort_index', 'sort_valu
es', 'squeeze', 'stack', 'std', 'style', 'sub', 'subtract', 'sum', 'swapa
xes', 'swaplevel', 'tail', 'take', 'to_clipboard', 'to_csv', 'to_dict',
'to_excel', 'to_feather', 'to_gbq', 'to_hdf', 'to_html', 'to_json', 'to_l
atex', 'to_markdown', 'to_numpy', 'to_orc', 'to_parquet', 'to_period', 't
o_pickle', 'to_records', 'to_sql', 'to_stata', 'to_string', 'to_timestam
p', 'to_xarray', 'to_xml', 'transform', 'transpose', 'truediv', 'truncat
e', 'tz_convert', 'tz_localize', 'unstack', 'update', 'value_counts', 'va
lues', 'var', 'where', 'xs']
```

In [128]:

```
df.columns
```

Out[128]:

```
Index(['Date', 'Symbol', 'Series', 'Prev Close', 'Open', 'High', 'Low',
'Last',
       'Close', 'VWAP', 'Volume', 'Turnover', 'Trades', 'Deliverable Volu
me',
       '%Deliverble'],
      dtype='object')
```

In [130]:

```
df.describe()
```

Out[130]:

	Prev Close	Open	High	Low	Last	Close	
count	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.000000	5306.0
mean	585.763852	586.507388	596.476187	575.571598	585.897399	585.893931	586.0
std	436.714128	436.602194	443.044833	430.108921	436.609147	436.649765	436.6
min	22.150000	21.000000	23.700000	21.000000	22.150000	22.150000	22.1
25%	230.950000	232.000000	235.125000	227.075000	230.550000	230.975000	231.1
50%	519.450000	520.100000	528.400000	512.025000	519.425000	519.500000	519.5
75%	877.312500	880.075000	897.987500	852.762500	877.275000	877.312500	875.8
max	2023.350000	2034.400000	2043.050000	2002.600000	2022.550000	2023.350000	2020.3

In [131]:

```
df.head()
```

Out[131]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Tu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.7	26.70	26.70	112100	2.99307
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.0	26.85	27.24	234500	6.38727
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.4	26.30	26.24	170100	4.46298
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.9	25.95	26.27	102100	2.68173
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.0	24.80	25.04	62600	1.56722

In [132]:

```
df.head(4)
```

Out[132]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turn
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.7	26.7	26.70	26.70	112100	2.993070
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.5	27.0	26.85	27.24	234500	6.387275
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.5	26.4	26.30	26.24	170100	4.462980
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.8	25.9	25.95	26.27	102100	2.681730

In [135]:

```
df.head(1)
```

Out[135]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turn
0	2000-01-03	UTIBANK	EQ	24.7	26.7	26.7	26.7	26.7	26.7	26.7	112100	2.993070

In [136]:
```python
df.tail()
```
Out[136]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

In [137]:
```python
df.tail(2)
```
Out[137]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volum
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.0	717.10	719.4	717.41	2593932
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.0	711.65	714.9	719.36	2301165

In [147]:
```python
df[0:2]
```
Out[147]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turn
0	2000-01-03	UTIBANK	EQ	24.7	26.7	26.7	26.7	26.7	26.70	26.70	112100	2.993070
1	2000-01-04	UTIBANK	EQ	26.7	27.0	28.7	26.5	27.0	26.85	27.24	234500	6.387275

In [148]:
```python
df[0:6]
```
Out[148]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Tu
0	2000-01-03	UTIBANK	EQ	24.70	26.70	26.70	26.70	26.7	26.70	26.70	112100	2.99307
1	2000-01-04	UTIBANK	EQ	26.70	27.00	28.70	26.50	27.0	26.85	27.24	234500	6.38727
2	2000-01-05	UTIBANK	EQ	26.85	26.00	27.75	25.50	26.4	26.30	26.24	170100	4.46298
3	2000-01-06	UTIBANK	EQ	26.30	25.80	27.00	25.80	25.9	25.95	26.27	102100	2.68173
4	2000-01-07	UTIBANK	EQ	25.95	25.00	26.00	24.25	25.0	24.80	25.04	62600	1.56722
5	2000-01-10	UTIBANK	EQ	24.80	25.05	26.50	25.00	25.0	25.00	25.29	64200	1.62334

In [149]:
```python
df.index
```
Out[149]:
```
RangeIndex(start=0, stop=5306, step=1)
```

In [150]:
```python
df.index.array
```
Out[150]:
```
<PandasArray>
[   0,    1,    2,    3,    4,    5,    6,    7,    8,    9,
 ...
 5296, 5297, 5298, 5299, 5300, 5301, 5302, 5303, 5304, 5305]
Length: 5306, dtype: int64
```

In [152]:
```python
df.to_numpy()
```
Out[152]:
```
array([['2000-01-03', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ['2000-01-04', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ['2000-01-05', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ...,
       ['2021-04-28', 'AXISBANK', 'EQ', ..., 507747.0, 17851331.0,
        0.3302],
       ['2021-04-29', 'AXISBANK', 'EQ', ..., 312079.0, 7357520.0, 0.283
6],
       ['2021-04-30', 'AXISBANK', 'EQ', ..., 232879.0, 6786072.0, 0.294
9]],
      dtype=object)
```

In [153]:
```python
import numpy as np
```

In [154]:
```python
df2 = np.asarray(df)
```

In [155]:
```python
df2
```
Out[155]:
```
array([['2000-01-03', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ['2000-01-04', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ['2000-01-05', 'UTIBANK', 'EQ', ..., nan, nan, nan],
       ...,
       ['2021-04-28', 'AXISBANK', 'EQ', ..., 507747.0, 17851331.0,
        0.3302],
       ['2021-04-29', 'AXISBANK', 'EQ', ..., 312079.0, 7357520.0, 0.283
6],
       ['2021-04-30', 'AXISBANK', 'EQ', ..., 232879.0, 6786072.0, 0.294
9]],
      dtype=object)
```

In [157]:
```python
df.sort_index(axis=0,ascending=False)
```
Out[157]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
...	...	...	...	...	...	...	...	...	...	...	
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112

5306 rows × 15 columns

In [160]:
```python
df['Symbol'][3] = 'TechVidya'
```
```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\3944233658.py:1: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df['Symbol'][3] = 'TechVidya'
```

In [161]:
```python
df
```
Out[161]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	TechVidya	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [164]:
```python
df.loc[2,'Symbol'] = "Hello"
```

In [165]:

```
df
```

Out[165]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	Hello	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	TechVidya	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [168]:

```
df.loc[2:3,['Open','Close']]
```

Out[168]:

	Open	Close
2	26.0	26.30
3	25.8	25.95

In [169]:

```
df.loc[:,['Open','Close']]
```

Out[169]:

	Open	Close
0	26.7	26.70
1	27.0	26.85
2	26.0	26.30
3	25.8	25.95
4	25.0	24.80
...	...	...
5301	694.0	700.45
5302	691.1	699.55
5303	708.0	708.15
5304	712.0	719.40
5305	705.0	714.90

5306 rows × 2 columns

In [170]:

```
df.loc[:,:]
```

Out[170]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	Hello	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	TechVidya	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [172]:

```
df.iloc[0:2]
```

Out[172]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turn
0	2000-01-03	UTIBANK	EQ	24.7	26.7	26.7	26.7	26.7	26.70	26.70	112100	2.993070
1	2000-01-04	UTIBANK	EQ	26.7	27.0	28.7	26.5	27.0	26.85	27.24	234500	6.387275

In [175]:

```
df.iloc[0,10]
```

Out[175]:

```
112100
```

In [176]:

```
df.drop('Trades',axis=1)
```

Out[176]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	Hello	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	TechVidya	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 14 columns

In [177]:

```
df.drop(2,axis=0)
```

Out[177]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Vo
0	2000-01-03	UTIBANK	EQ	24.70	26.70	26.70	26.70	26.70	26.70	26.70	11
1	2000-01-04	UTIBANK	EQ	26.70	27.00	28.70	26.50	27.00	26.85	27.24	23
3	2000-01-06	TechVidya	EQ	26.30	25.80	27.00	25.80	25.90	25.95	26.27	10
4	2000-01-07	UTIBANK	EQ	25.95	25.00	26.00	24.25	25.00	24.80	25.04	6
5	2000-01-10	UTIBANK	EQ	24.80	25.05	26.50	25.00	25.00	25.00	25.29	6
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.00	703.80	684.50	699.50	700.45	695.33	2164
5302	2021-04-27	AXISBANK	EQ	700.45	691.10	703.90	684.10	700.90	699.55	692.83	4655
5303	2021-04-28	AXISBANK	EQ	699.55	708.00	712.50	688.15	705.95	708.15	701.92	5406
5304	2021-04-29	AXISBANK	EQ	708.15	712.00	726.90	707.00	717.10	719.40	717.41	2593
5305	2021-04-30	AXISBANK	EQ	719.40	705.00	729.85	705.00	711.65	714.90	719.36	2301

5305 rows × 15 columns

In [ ]:

In [ ]:

In [178]:

```
# Handling Missing Values
```

In [180]:

```
abc = {'A':[1,2,3,45,np.nan,67,78],
       'B':[3,4,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
```

In [181]:
```
df
```
Out[181]:
```
      A     B
0   1.0   3.0
1   2.0   4.0
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

In [183]:
```
df.dropna()
```
Out[183]:
```
      A     B
0   1.0   3.0
1   2.0   4.0
2   3.0  56.0
3  45.0   7.0
5  67.0  34.0
6  78.0   1.0
```

In [189]:
```
df['C'] = [1,2,35,6,7,8,9]
```

In [191]:
```
df.drop('B',axis=1)
```
Out[191]:
```
      A   C
0   1.0   1
1   2.0   2
2   3.0  35
3  45.0   6
4   NaN   7
5  67.0   8
6  78.0   9
```

In [192]:
```
df
```
Out[192]:
```
      A     B   C
0   1.0   3.0   1
1   2.0   4.0   2
2   3.0  56.0  35
3  45.0   7.0   6
4   NaN   NaN   7
5  67.0  34.0   8
6  78.0   1.0   9
```

In [196]:
```
df.dropna(how='any')
```
Out[196]:
```
      A     B   C
0   1.0   3.0   1
1   2.0   4.0   2
2   3.0  56.0  35
3  45.0   7.0   6
5  67.0  34.0   8
6  78.0   1.0   9
```

In [197]:
```
df.dropna(how='all')
```
Out[197]:
```
      A     B   C
0   1.0   3.0   1
1   2.0   4.0   2
2   3.0  56.0  35
3  45.0   7.0   6
4   NaN   NaN   7
5  67.0  34.0   8
6  78.0   1.0   9
```

In [200]:
```
df.dropna(subset=['B'],inplace=True)  # only drop null value along a g
```

In [201]:
```
df
```
Out[201]:
```
      A     B   C
0   1.0   3.0   1
1   2.0   4.0   2
2   3.0  56.0  35
3  45.0   7.0   6
5  67.0  34.0   8
6  78.0   1.0   9
```

In [203]:
```
df.dropna(thresh=2)  # threshold value of na
```
Out[203]:
```
      A     B   C
0   1.0   3.0   1
1   2.0   4.0   2
2   3.0  56.0  35
3  45.0   7.0   6
5  67.0  34.0   8
6  78.0   1.0   9
```

In [204]:
```
# fillna
```

In [207]:
```
abc = {'A':[1,2,3,45,np.nan,67,78],
       'B':[3,np.nan,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
```

In [208]:
```
df
```
Out[208]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

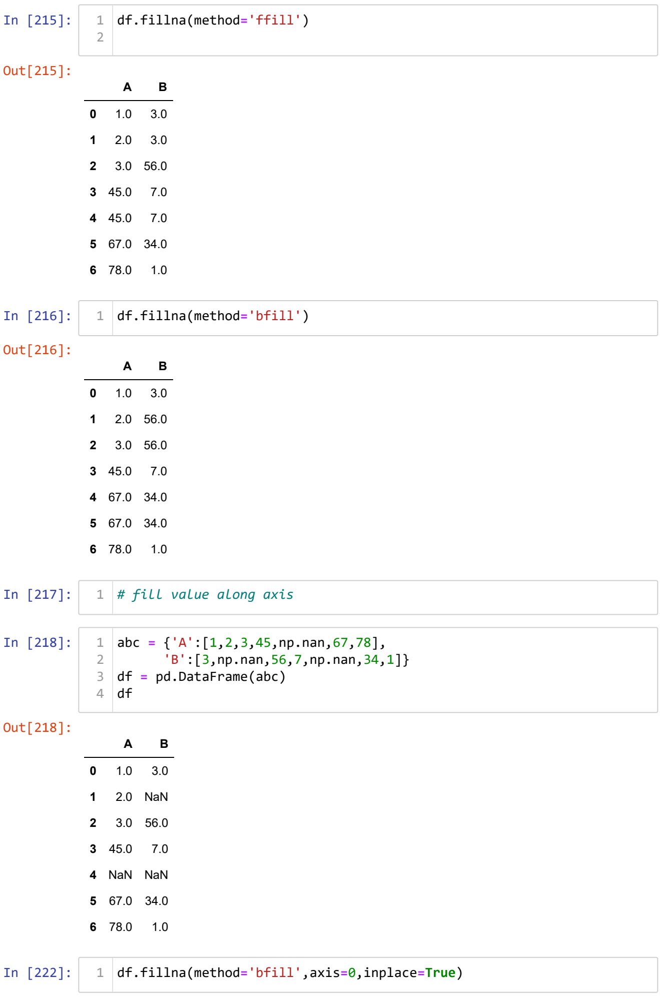
In [209]:
```
df.fillna('techvidya')
```
Out[209]:
```
           A          B
0        1.0        3.0
1        2.0  techvidya
2        3.0       56.0
3       45.0        7.0
4  techvidya  techvidya
5       67.0       34.0
6       78.0        1.0
```

In [210]:
```
# col particular data fill na
```

In [211]:
```
df.fillna({"A":'Hello','B':'World'})
```
Out[211]:
```
       A      B
0    1.0    3.0
1    2.0  World
2    3.0   56.0
3   45.0    7.0
4  Hello  World
5   67.0   34.0
6   78.0    1.0
```

In [212]:
```
# fill na with forward or backward data
```

In [213]:
```
abc = {'A':[1,2,3,45,np.nan,67,78],
       'B':[3,np.nan,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
df
```
Out[213]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```



In [215]:
```
df.fillna(method='ffill')

```
Out[215]:

	A	B
0	1.0	3.0
1	2.0	3.0
2	3.0	56.0
3	45.0	7.0
4	45.0	7.0
5	67.0	34.0
6	78.0	1.0

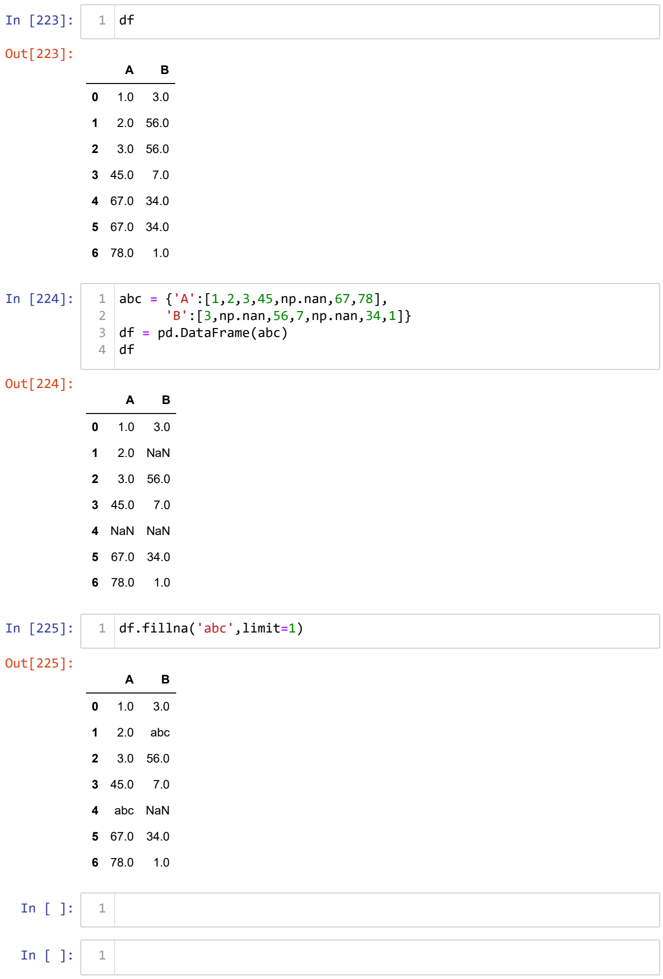
In [216]:
```
df.fillna(method='bfill')
```
Out[216]:

	A	B
0	1.0	3.0
1	2.0	56.0
2	3.0	56.0
3	45.0	7.0
4	67.0	34.0
5	67.0	34.0
6	78.0	1.0

In [217]:
```
# fill value along axis
```
In [218]:
```
abc = {'A':[1,2,3,45,np.nan,67,78],
      'B':[3,np.nan,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
df
```
Out[218]:

	A	B
0	1.0	3.0
1	2.0	NaN
2	3.0	56.0
3	45.0	7.0
4	NaN	NaN
5	67.0	34.0
6	78.0	1.0

In [222]:
```
df.fillna(method='bfill',axis=0,inplace=True)
```



In [223]:
```
df
```
Out[223]:

	A	B
0	1.0	3.0
1	2.0	56.0
2	3.0	56.0
3	45.0	7.0
4	67.0	34.0
5	67.0	34.0
6	78.0	1.0

In [224]:
```
abc = {'A':[1,2,3,45,np.nan,67,78],
      'B':[3,np.nan,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
df
```
Out[224]:

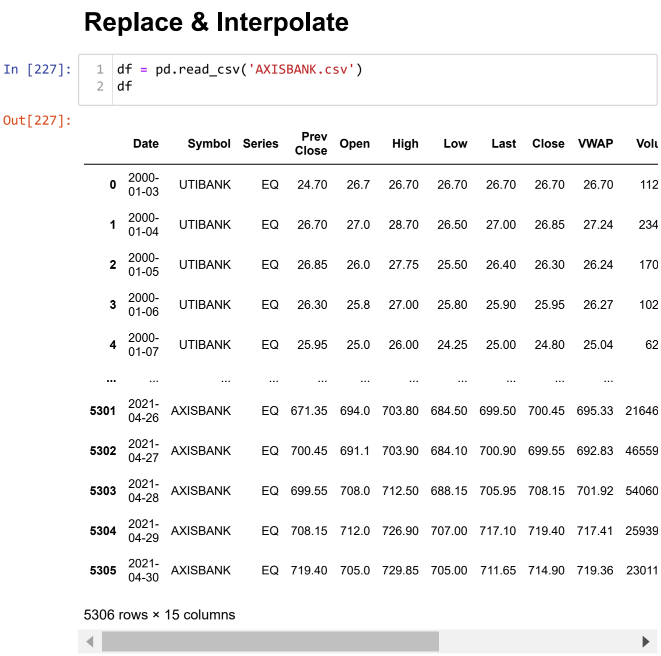
	A	B
0	1.0	3.0
1	2.0	NaN
2	3.0	56.0
3	45.0	7.0
4	NaN	NaN
5	67.0	34.0
6	78.0	1.0

In [225]:
```
df.fillna('abc',limit=1)
```
Out[225]:

	A	B
0	1.0	3.0
1	2.0	abc
2	3.0	56.0
3	45.0	7.0
4	abc	NaN
5	67.0	34.0
6	78.0	1.0

In [ ]:

In [ ]:

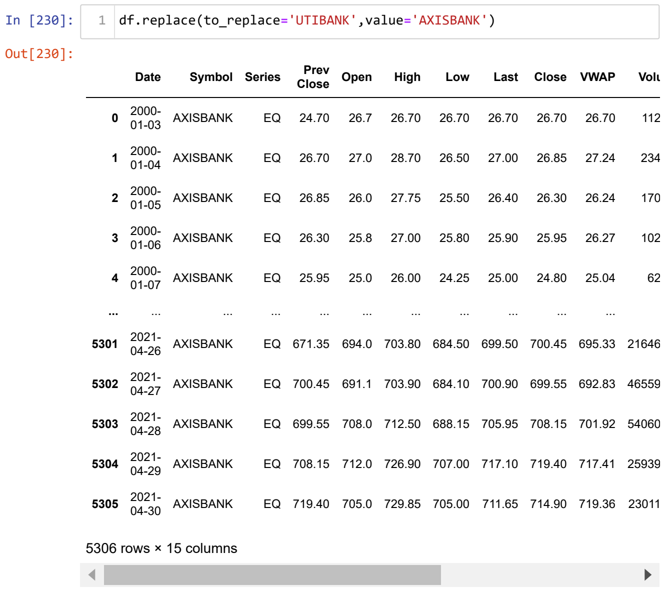


Replace & Interpolate

In [227]:
```
df = pd.read_csv('AXISBANK.csv')
df
```
Out[227]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	...
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns



In [230]:
```
df.replace(to_replace='UTIBANK',value='AXISBANK')
```
Out[230]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	AXISBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	AXISBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	AXISBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	AXISBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	AXISBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	...
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [238]:

```
df.replace(to_replace="NaN",value=0)
```

Out[238]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [241]:

```
df.fillna(method='bfill',axis=0,inplace=True)
```

In [242]:

```
df
```

Out[242]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	UTIBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	UTIBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	UTIBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	UTIBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	UTIBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [246]:

```
df.replace({'Symbol':r"UTI"},'AXIS',regex=True,inplace=True)
```

In [247]:

```
df
```

Out[247]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	AXISBANK	EQ	24.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	AXISBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	AXISBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	AXISBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	AXISBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [249]:

```
df.replace(24.70,method='bfill')
```

Out[249]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volu
0	2000-01-03	AXISBANK	EQ	26.70	26.7	26.70	26.70	26.70	26.70	26.70	112
1	2000-01-04	AXISBANK	EQ	26.70	27.0	28.70	26.50	27.00	26.85	27.24	234
2	2000-01-05	AXISBANK	EQ	26.85	26.0	27.75	25.50	26.40	26.30	26.24	170
3	2000-01-06	AXISBANK	EQ	26.30	25.8	27.00	25.80	25.90	25.95	26.27	102
4	2000-01-07	AXISBANK	EQ	25.95	25.0	26.00	24.25	25.00	24.80	25.04	62
...	...	...	...	...	...	...	...	...	...	...	
5301	2021-04-26	AXISBANK	EQ	671.35	694.0	703.80	684.50	699.50	700.45	695.33	21646
5302	2021-04-27	AXISBANK	EQ	700.45	691.1	703.90	684.10	700.90	699.55	692.83	46559
5303	2021-04-28	AXISBANK	EQ	699.55	708.0	712.50	688.15	705.95	708.15	701.92	54060
5304	2021-04-29	AXISBANK	EQ	708.15	712.0	726.90	707.00	717.10	719.40	717.41	25939
5305	2021-04-30	AXISBANK	EQ	719.40	705.0	729.85	705.00	711.65	714.90	719.36	23011

5306 rows × 15 columns

In [250]:
```
df.replace(708.0,method='ffill')
```
Out[250]:
```
      Date   Symbol     Series  Prev    Open   High    Low     Last    Close   VWAP    Volu
                                Close
0     2000-  AXISBANK   EQ      24.70   26.7   26.70   26.70   26.70   26.70   26.70   112
      01-03
1     2000-  AXISBANK   EQ      26.70   27.0   28.70   26.50   27.00   26.85   27.24   234
      01-04
2     2000-  AXISBANK   EQ      26.85   26.0   27.75   25.50   26.40   26.30   26.24   170
      01-05
3     2000-  AXISBANK   EQ      26.30   25.8   27.00   25.80   25.90   25.95   26.27   102
      01-06
4     2000-  AXISBANK   EQ      25.95   25.0   26.00   24.25   25.00   24.80   25.04   62
      01-07
...   ...    ...        ...     ...     ...    ...     ...     ...     ...     ...
5301  2021-  AXISBANK   EQ      671.35  694.0  703.80  684.50  699.50  700.45  695.33  21646
      04-26
5302  2021-  AXISBANK   EQ      700.45  691.1  703.90  684.10  700.90  699.55  692.83  46559
      04-27
5303  2021-  AXISBANK   EQ      699.55  691.1  712.50  688.15  705.95  708.15  701.92  54060
      04-28
5304  2021-  AXISBANK   EQ      708.15  712.0  726.90  707.00  717.10  719.40  717.41  25939
      04-29
5305  2021-  AXISBANK   EQ      719.40  705.0  729.85  705.00  711.65  714.90  719.36  23011
      04-30

5306 rows × 15 columns
```

In [253]:
```
df.replace(to_replace='EQ',method='ffill',limit=5)
```
Out[253]:
```
      Date   Symbol     Series  Prev    Open   High    Low     Last    Close   VWAP    Volu
                                Close
0     2000-  AXISBANK   EQ      24.70   26.7   26.70   26.70   26.70   26.70   26.70   112
      01-03
1     2000-  AXISBANK   EQ      26.70   27.0   28.70   26.50   27.00   26.85   27.24   234
      01-04
2     2000-  AXISBANK   EQ      26.85   26.0   27.75   25.50   26.40   26.30   26.24   170
      01-05
3     2000-  AXISBANK   EQ      26.30   25.8   27.00   25.80   25.90   25.95   26.27   102
      01-06
4     2000-  AXISBANK   EQ      25.95   25.0   26.00   24.25   25.00   24.80   25.04   62
      01-07
...   ...    ...        ...     ...     ...    ...     ...     ...     ...     ...
5301  2021-  AXISBANK   EQ      671.35  694.0  703.80  684.50  699.50  700.45  695.33  21646
      04-26
5302  2021-  AXISBANK   EQ      700.45  691.1  703.90  684.10  700.90  699.55  692.83  46559
      04-27
5303  2021-  AXISBANK   EQ      699.55  708.0  712.50  688.15  705.95  708.15  701.92  54060
      04-28
5304  2021-  AXISBANK   EQ      708.15  712.0  726.90  707.00  717.10  719.40  717.41  25939
      04-29
5305  2021-  AXISBANK   EQ      719.40  705.0  729.85  705.00  711.65  714.90  719.36  23011
      04-30

5306 rows × 15 columns
```

Interpolate

In [254]:
```
# to fill data automatic
```

In [255]:
```
abc = {'A':[1,2,3,45,np.nan,67,78],
      'B':[3,np.nan,56,7,np.nan,34,1]}
df = pd.DataFrame(abc)
df
```
Out[255]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

In [256]:
```
df
```
Out[256]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

In [258]:
```
df.interpolate()  # it claculate value automatic and fill, work only w
```
Out[258]:
```
      A     B
0   1.0   3.0
1   2.0  29.5
2   3.0  56.0
3  45.0   7.0
4  56.0  20.5
5  67.0  34.0
6  78.0   1.0
```

In [261]:
```
# fill data linear
df
```
Out[261]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

In [264]:
```
df.interpolate(method='polynomial',order=2,axis=0) # by col
```
Out[264]:
```
           A          B
0   1.000000   3.000000
1   2.000000  60.316239
2   3.000000  56.000000
3  45.000000   7.000000
4  63.212963  15.008547
5  67.000000  34.000000
6  78.000000   1.000000
```

In [278]:
```
df.interpolate(axis=0,limit_direction='forward') # by col
```
Out[278]:
```
      A     B
0   1.0   3.0
1   2.0  29.5
2   3.0  56.0
3  45.0   7.0
4  56.0  20.5
5  67.0  34.0
6  78.0   1.0
```

In [279]:
```
df.interpolate(limit_area='inside')
```
Out[279]:
```
      A     B
0   1.0   3.0
1   2.0  29.5
2   3.0  56.0
3  45.0   7.0
4  56.0  20.5
5  67.0  34.0
6  78.0   1.0
```

In [280]:
```
df.interpolate(limit_area='outside')
```

Out[280]:
```
      A     B
0   1.0   3.0
1   2.0   NaN
2   3.0  56.0
3  45.0   7.0
4   NaN   NaN
5  67.0  34.0
6  78.0   1.0
```

In [281]:
```
df.interpolate(limit_area='inside',inplace=True)
```

In [282]:
```
df
```

Out[282]:
```
      A     B
0   1.0   3.0
1   2.0  29.5
2   3.0  56.0
3  45.0   7.0
4  56.0  20.5
5  67.0  34.0
6  78.0   1.0
```

In [ ]:
```

```

In [ ]:
```

```

Merging and concat

Merge

In [284]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df1,df2,on='A')
df
```

Out[284]:
```
   A  B_x  B_y
0  1    3   13
1  2    4   14
2  3    5   15
3  4    6   16
```

In [285]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df2,df1,on='A')
df
```

Out[285]:
```
   A  B_x  B_y
0  1   13    3
1  2   14    4
2  3   15    5
3  4   16    6
```

In [286]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,40],'B':[13,14,15,16]})

df = pd.merge(df1,df2,on='A')
df
```

Out[286]:
```
   A  B_x  B_y
0  1    3   13
1  2    4   14
2  3    5   15
```

In [291]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

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

Out[291]:
```
   A  B_x  B_y
0  1    3   13
1  2    4   14
2  3    5   15
3  4    6   16
```

In [294]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

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

Out[294]:
```
   A_x  B  A_y
0    1  3  NaN
1    2  4  NaN
2    3  5  NaN
3    4  6  NaN
```

In [295]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df1,df2,how ='right',on='B')
df
```

Out[295]:
```
   A_x   B  A_y
0  NaN  13    1
1  NaN  14    2
2  NaN  15    3
3  NaN  16    4
```

In [298]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df1,df2,how ='left',on='A',indicator=True)
df
```

Out[298]:
```
   A  B_x  B_y  _merge
0  1    3   13    both
1  2    4   14    both
2  3    5   15    both
3  4    6   16    both
```

In [300]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df1,df2)
df
```

Out[300]:
```
  A  B
```

In [301]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,4],'B':[13,14,15,16]})

df = pd.merge(df1,df2,left_index=True,right_index=True)
df
```

Out[301]:
```
   A_x  B_x  A_y  B_y
0    1    3    1   13
1    2    4    2   14
2    3    5    3   15
3    4    6    4   16
```

In [303]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,40],'B':[13,14,15,16]})

df = pd.merge(df1,df2,left_index=True,right_index=True,suffixes=['Tech
df
```

Out[303]:
```
   ATech  BTech  AVidya  BVidya
0      1      3       1      13
1      2      4       2      14
2      3      5       3      15
3      4      6      40      16
```

concat

In [305]:
```
a = pd.Series([1,2,3,4,5])
b = pd.Series([10,20,30,40,50])

c = pd.concat([a,b])
c
```

Out[305]:
```
0     1
1     2
2     3
3     4
4     5
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

In [306]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[3,4,5,6]})
df2 = pd.DataFrame({'A':[1,2,3,40],'B':[13,14,15,16]})
df1
```

Out[306]:

	A	B
0	1	3
1	2	4
2	3	5
3	4	6

In [307]:
```
df2
```

Out[307]:

	A	B
0	1	13
1	2	14
2	3	15
3	40	16

In [308]:
```
pd.concat([df1,df2])
```

Out[308]:

	A	B
0	1	3
1	2	4
2	3	5
3	4	6
0	1	13
1	2	14
2	3	15
3	40	16

In [321]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[21,35,56,76]})
df2 = pd.DataFrame({'C':[10,20],'D':[11,12]})

df = pd.concat([df1,df2],axis=1,join='outer')
df
```

Out[321]:

	A	B	C	D
0	1	21	10.0	11.0
1	2	35	20.0	12.0
2	3	56	NaN	NaN
3	4	76	NaN	NaN

In [322]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[21,35,56,76]})
df2 = pd.DataFrame({'C':[10,20],'D':[11,12]})

df = pd.concat([df1,df2],axis=1,join='inner')
df
```

Out[322]:

	A	B	C	D
0	1	21	10	11
1	2	35	20	12

In [324]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[21,35,56,76]})
df2 = pd.DataFrame({'C':[1,2,3,4],'D':[11,12,13,14]})

df = pd.concat([df1,df2],axis=1,keys=['P','Q'])
df
```

Out[324]:

	P		Q	
	A	B	C	D
0	1	21	1	11
1	2	35	2	12
2	3	56	3	13
3	4	76	4	14

In [325]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],'B':[21,35,56,76]})
df2 = pd.DataFrame({'C':[1,2,3,4],'D':[11,12,13,14]})

df = pd.concat([df1,df2],axis=0,keys=['P','Q'])
df
```

Out[325]:

		A	B	C	D
P	**0**	1.0	21.0	NaN	NaN
	1	2.0	35.0	NaN	NaN
	2	3.0	56.0	NaN	NaN
	3	4.0	76.0	NaN	NaN
Q	**0**	NaN	NaN	1.0	11.0
	1	NaN	NaN	2.0	12.0
	2	NaN	NaN	3.0	13.0
	3	NaN	NaN	4.0	14.0

In [327]:
```
df1 = pd.DataFrame({'A':[1,2,3,4]})
df2 = pd.DataFrame({'C':[1,2,3,4],'D':[11,12,13,14]})

df = pd.concat([df1,df2])
df
```

Out[327]:

	A	C	D
0	1.0	NaN	NaN
1	2.0	NaN	NaN
2	3.0	NaN	NaN
3	4.0	NaN	NaN
0	NaN	1.0	11.0
1	NaN	2.0	12.0
2	NaN	3.0	13.0
3	NaN	4.0	14.0

join and append

In [330]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]})
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2)
```

Out[330]:

	A	B	C	D
0	1	14	10.0	12.0
1	2	34	30.0	13.0
2	3	56	40.0	14.0
3	4	67	NaN	NaN

In [331]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2)
```

Out[331]:

	A	B	C	D
A	1	14	NaN	NaN
B	2	34	NaN	NaN
C	3	56	NaN	NaN
D	4	67	NaN	NaN

In [333]:
```
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2,how='left')
```

Out[333]:

	A	B	C	D
A	1	14	NaN	NaN
B	2	34	NaN	NaN
C	3	56	NaN	NaN
D	4	67	NaN	NaN

In [334]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2,how='right')
```

Out[334]:

	A	B	C	D
0	NaN	NaN	10	12
1	NaN	NaN	30	13
2	NaN	NaN	40	14

In [335]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2,how='inner')
```

Out[335]:

A	B	C	D

In [336]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'C':[10,30,40],'D':[12,13,14]})

df1.join(df2,how='outer')
```

Out[336]:

	A	B	C	D
A	1.0	14.0	NaN	NaN
B	2.0	34.0	NaN	NaN
C	3.0	56.0	NaN	NaN
D	4.0	67.0	NaN	NaN
0	NaN	NaN	10.0	12.0
1	NaN	NaN	30.0	13.0
2	NaN	NaN	40.0	14.0

In [340]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'A':[10,30,40],'B':[12,13,14]})

df1.join(df2,how='outer',lsuffix='One',rsuffix='Two')
```

Out[340]:

	AOne	BOne	ATwo	BTwo
A	1.0	14.0	NaN	NaN
B	2.0	34.0	NaN	NaN
C	3.0	56.0	NaN	NaN
D	4.0	67.0	NaN	NaN
0	NaN	NaN	10.0	12.0
1	NaN	NaN	30.0	13.0
2	NaN	NaN	40.0	14.0

In [ ]:

```python

```

Append

In [345]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'D':[10,30,40],'B':[12,13,14]})

df1.append(df2)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\965237366.py:4: Future
Warning: The frame.append method is deprecated and will be removed from p
andas in a future version. Use pandas.concat instead.
  df1.append(df2)
```

Out[345]:

	A	B	D
A	1.0	14	NaN
B	2.0	34	NaN
C	3.0	56	NaN
D	4.0	67	NaN
0	NaN	12	10.0
1	NaN	13	30.0
2	NaN	14	40.0

In [344]:

```python
df1 = pd.DataFrame({'A':[1,2,3,4],"B":[14,34,56,67]},index=['A','B','C
df2 = pd.DataFrame({'D':[10,30,40],'B':[12,13,14]})

df1.append(df2,ignore_index=True)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\153391876.py:4: Future
Warning: The frame.append method is deprecated and will be removed from p
andas in a future version. Use pandas.concat instead.
  df1.append(df2,ignore_index=True)
```

Out[344]:

	A	B	D
0	1.0	14	NaN
1	2.0	34	NaN
2	3.0	56	NaN
3	4.0	67	NaN
4	NaN	12	10.0
5	NaN	13	30.0
6	NaN	14	40.0

group by

In [347]:

```python
a = pd.DataFrame({'Name':['karan','madan','mohan','karan',
                          'madan','madan','ravi','madan','ravi'],
                 'Marks':[23,45,34,56,78,65,34,78,56],
                 'Marks2':[33,55,44,76,78,35,74,88,65]})
a
```

Out[347]:

	Name	Marks	Marks2
0	karan	23	33
1	madan	45	55
2	mohan	34	44
3	karan	56	76
4	madan	78	78
5	madan	65	35
6	ravi	34	74
7	madan	78	88
8	ravi	56	65

In [348]:

```python
name = a.groupby('Name')
```

In [357]:

```python
for i in name:
    print(i)
    print()
```

```
('karan',     Name  Marks  Marks2
0  karan     23      33
3  karan     56      76)

('madan',     Name  Marks  Marks2
1  madan     45      55
4  madan     78      78
5  madan     65      35
7  madan     78      88)

('mohan',     Name  Marks  Marks2
2  mohan     34      44)

('ravi',    Name  Marks  Marks2
6  ravi     34      74
8  ravi     56      65)
```

In [358]:

```python
name.get_group('mohan')
```

Out[358]:

	Name	Marks	Marks2
2	mohan	34	44

In [359]:

```python
name.get_group('madan')
```

Out[359]:

	Name	Marks	Marks2
1	madan	45	55
4	madan	78	78
5	madan	65	35
7	madan	78	88

In [360]:

```python
name.min()
```

Out[360]:

	Marks	Marks2
Name		
karan	23	33
madan	45	35
mohan	34	44
ravi	34	65

In [361]:

```python
name.max()
```

Out[361]:

	Marks	Marks2
Name		
karan	56	76
madan	78	88
mohan	34	44
ravi	56	74

In [362]:

```python
name.mean()
```

Out[362]:

	Marks	Marks2
Name		
karan	39.5	54.5
madan	66.5	64.0
mohan	34.0	44.0
ravi	45.0	69.5

In [364]:

```
list(name)
```

Out[364]:

```
[('karan',
     Name  Marks  Marks2
  0  karan     23      33
  3  karan     56      76),
 ('madan',
     Name  Marks  Marks2
  1  madan     45      55
  4  madan     78      78
  5  madan     65      35
  7  madan     78      88),
 ('mohan',
     Name  Marks  Marks2
  2  mohan     34      44),
 ('ravi',
    Name  Marks  Marks2
  6  ravi     34      74
  8  ravi     56      65)]
```

In [ ]:

In [ ]:

pivot table and melt

melt

In [366]:

```
data = pd.DataFrame({'days':[1,2,3,4,5,6],
                    'eng':[12,23,15,16,17,11],
                    'maths':[17,13,15,18,20,18]})
data
```

Out[366]:

	days	eng	maths
0	1	12	17
1	2	23	13
2	3	15	15
3	4	16	18
4	5	17	20
5	6	11	18

In [367]:

```
pd.melt(data)
```

Out[367]:

	variable	value
0	days	1
1	days	2
2	days	3
3	days	4
4	days	5
5	days	6
6	eng	12
7	eng	23
8	eng	15
9	eng	16
10	eng	17
11	eng	11
12	maths	17
13	maths	13
14	maths	15
15	maths	18
16	maths	20
17	maths	18

In [368]:

```
pd.melt(data,id_vars=['eng'])
```

Out[368]:

	eng	variable	value
0	12	days	1
1	23	days	2
2	15	days	3
3	16	days	4
4	17	days	5
5	11	days	6
6	12	maths	17
7	23	maths	13
8	15	maths	15
9	16	maths	18
10	17	maths	20
11	11	maths	18

In [369]:

```
pd.melt(data,id_vars=['days'])
```

Out[369]:

	days	variable	value
0	1	eng	12
1	2	eng	23
2	3	eng	15
3	4	eng	16
4	5	eng	17
5	6	eng	11
6	1	maths	17
7	2	maths	13
8	3	maths	15
9	4	maths	18
10	5	maths	20
11	6	maths	18

In [370]:

```
pd.melt(data,id_vars=['eng'],var_name='techvidya')
```

Out[370]:

	eng	techvidya	value
0	12	days	1
1	23	days	2
2	15	days	3
3	16	days	4
4	17	days	5
5	11	days	6
6	12	maths	17
7	23	maths	13
8	15	maths	15
9	16	maths	18
10	17	maths	20
11	11	maths	18

In [371]:

```
pd.melt(data,id_vars=['eng'],var_name='techvidya',value_name='tech')
```

Out[371]:

	eng	techvidya	tech
0	12	days	1
1	23	days	2
2	15	days	3
3	16	days	4
4	17	days	5
5	11	days	6
6	12	maths	17
7	23	maths	13
8	15	maths	15
9	16	maths	18
10	17	maths	20
11	11	maths	18

pivot

In [373]:

```
data = pd.DataFrame({'days':[1,2,3,4,5,6],
                     'Name':['A','B','C','A',"C","B"],
                    'eng':[12,23,15,16,17,11],
                    'maths':[17,13,15,18,20,18]})
data
```

Out[373]:

	days	Name	eng	maths
0	1	A	12	17
1	2	B	23	13
2	3	C	15	15
3	4	A	16	18
4	5	C	17	20
5	6	B	11	18

In [374]:

```python
data.pivot(index='days',columns='Name')
```

Out[374]:

```
        eng                maths
Name    A     B     C      A     B     C
days
   1    12.0  NaN   NaN    17.0  NaN   NaN
   2    NaN   23.0  NaN    NaN   13.0  NaN
   3    NaN   NaN   15.0   NaN   NaN   15.0
   4    16.0  NaN   NaN    18.0  NaN   NaN
   5    NaN   NaN   17.0   NaN   NaN   20.0
   6    NaN   11.0  NaN    NaN   18.0  NaN
```

In [375]:

```python
data = pd.DataFrame({'days':[1,2,3,4,5,6],
                     'Name':['A','B']*3,
                    'eng':[12,23,15,16,17,11],
                    'maths':[17,13,15,18,20,18]})
data
```

Out[375]:

```
   days  Name  eng  maths
0     1     A   12     17
1     2     B   23     13
2     3     A   15     15
3     4     B   16     18
4     5     A   17     20
5     6     B   11     18
```

In [377]:

```python
data.pivot(index='days',columns='Name')
```

Out[377]:

```
        eng          maths
Name    A     B      A     B
days
   1    12.0  NaN    17.0  NaN
   2    NaN   23.0   NaN   13.0
   3    15.0  NaN    15.0  NaN
   4    NaN   16.0   NaN   18.0
   5    17.0  NaN    20.0  NaN
   6    NaN   11.0   NaN   18.0
```

In [378]:

```python
data.pivot(index='days',columns='Name',values='eng')
```

Out[378]:

```
Name    A     B
days
   1    12.0  NaN
   2    NaN   23.0
   3    15.0  NaN
   4    NaN   16.0
   5    17.0  NaN
   6    NaN   11.0
```

In [380]:

```python
data.pivot_table(index='Name',columns='days',aggfunc='mean')
```

Out[380]:

```
       eng                                  maths
days   1     2     3     4     5     6     1     2     3     4     5     6
Name
   A   12.0  NaN   15.0  NaN   17.0  NaN   17.0  NaN   15.0  NaN   20.0  NaN
   B   NaN   23.0  NaN   16.0  NaN   11.0  NaN   13.0  NaN   18.0  NaN   18.0
```

In [381]:

```python
data.pivot_table(index='Name',columns='days',aggfunc='sum')
```

Out[381]:

```
       eng                                  maths
days   1     2     3     4     5     6     1     2     3     4     5     6
Name
   A   12.0  NaN   15.0  NaN   17.0  NaN   17.0  NaN   15.0  NaN   20.0  NaN
   B   NaN   23.0  NaN   16.0  NaN   11.0  NaN   13.0  NaN   18.0  NaN   18.0
```

In [382]:

```python
data.pivot_table(index='Name',columns='days',aggfunc='count')
```

Out[382]:

```
       eng                                  maths
days   1     2     3     4     5     6     1     2     3     4     5     6
Name
   A    1.0  NaN    1.0  NaN    1.0  NaN    1.0  NaN    1.0  NaN    1.0  NaN
   B   NaN    1.0  NaN    1.0  NaN    1.0  NaN    1.0  NaN    1.0  NaN    1.0
```

In [383]:

```python
data.pivot_table(index='Name',columns='days',aggfunc='mean',margins=Tr
```

Out[383]:

```
       eng                                             maths
days   1     2     3     4     5     6     All         1     2     3     4     5     6     All
Name
   A   12.0  NaN   15.0  NaN   17.0  NaN   14.666667   17.0  NaN   15.0  NaN   20.0  NaN   17.3
   B   NaN   23.0  NaN   16.0  NaN   11.0  16.666667   NaN   13.0  NaN   18.0  NaN   18.0  16.3
 All   12.0  23.0  15.0  16.0  17.0  11.0  15.666667   17.0  13.0  15.0  18.0  20.0  18.0  16.8
```

In [384]:

```python
df = pd.read_csv('Bengaluru_House_Data.csv')
df
```

Out[384]:

```
       area_type  availability            location      size  society  total_sqft  bath  balcon
           Super
       built-up        19-Dec    Electronic City     2 BHK   Coomee        1056   2.0     1.(
    0       Area                        Phase II
           
                     Ready To                           4
    1  Plot Area         Move   Chikka Tirupathi  Bedroom  Theanmp        2600   5.0     3.(

       Built-up      Ready To
    2       Area         Move        Uttarahalli     3 BHK      NaN        1440   2.0     3.(

           Super
       built-up      Ready To
    3       Area         Move  Lingadheeranahalli    3 BHK   Soiewre        1521   3.0     1.(

           Super
       built-up      Ready To
    4       Area         Move           Kothanur     2 BHK      NaN        1200   2.0     1.(

  ...        ...          ...                ...       ...      ...         ...   ...     ..

       Built-up      Ready To                           5
13315       Area         Move         Whitefield  Bedroom   ArsiaEx        3453   4.0     0.(

           Super
       built-up      Ready To
13316       Area         Move      Richards Town     4 BHK      NaN        3600   5.0     Nal

       Built-up      Ready To   Raja Rajeshwari
13317       Area         Move              Nagar     2 BHK   Mahla T        1141   2.0     1.(

           Super
       built-up
13318       Area       18-Jun   Padmanabhanagar     4 BHK   SollyCl        4689   4.0     1.(

           Super
       built-up      Ready To
13319       Area         Move       Doddathoguru     1 BHK      NaN         550   1.0     1.(

13320 rows × 9 columns
```

In [396]:

```python
df.groupby(['size']).count()
```

Out[396]:

```
            area_type  availability  location  society  total_sqft  bath  balcony  price
size
     1 BHK        538           538       538      361         538   531      530    538
 1 Bedroom        105           105       105        0         105   105      105    105
      1 RK         13            13        13       10          13    13       13     13
    10 BHK          2             2         2        0           2     2        0      2
10 Bedroom         12            12        12        1          12    12        3     12
    11 BHK          2             2         2        1           2     2        1      2
11 Bedroom          2             2         2        0           2     2        2      2
12 Bedroom          1             1         1        0           1     1        1      1
    13 BHK          1             1         1        0           1     1        1      1
    14 BHK          1             1         1        0           1     1        1      1
    16 BHK          1             1         1        0           1     1        0      1
18 Bedroom          1             1         1        1           1     1        0      1
    19 BHK          1             1         1        0           1     1        0      1
     2 BHK       5199          5199      5199     3439        5199  5198     5152   5199
 2 Bedroom        329           329       329       16         329   329      328    329
    27 BHK          1             1         1        0           1     1        1      1
     3 BHK       4310          4310      4309     3154        4310  4287     4129   4310
 3 Bedroom        547           547       547      128         547   546      527    547
     4 BHK        591           591       591      416         591   577      489    591
 4 Bedroom        826           826       826      219         826   818      749    826
43 Bedroom          1             1         1        0           1     1        1      1
     5 BHK         59            59        59       23          59    57       36     59
 5 Bedroom        297           297       297       22         297   296      263    297
     6 BHK         30            30        30        5          30    30       23     30
 6 Bedroom        191           191       191        4         191   191      169    191
     7 BHK         17            17        17        1          17    17       16     17
 7 Bedroom         83            83        83        0          83    83       69     83
     8 BHK          5             5         5        0           5     5        3      5
 8 Bedroom         84            84        84        1          84    84       65     84
     9 BHK          8             8         8        1           8     8        5      8
 9 Bedroom         46            46        46        2          46    46       29     46
```

In [399]:
```
df.groupby('size').count()
```

Out[399]:

size	area_type	availability	location	society	total_sqft	bath	balcony	price
1 BHK	538	538	538	361	538	531	530	538
1 Bedroom	105	105	105	0	105	105	105	105
1 RK	13	13	13	10	13	13	13	13
10 BHK	2	2	2	0	2	2	0	2
10 Bedroom	12	12	12	1	12	12	3	12
11 BHK	2	2	2	1	2	2	1	2
11 Bedroom	2	2	2	0	2	2	2	2
12 Bedroom	1	1	1	0	1	1	1	1
13 BHK	1	1	1	0	1	1	1	1
14 BHK	1	1	1	0	1	1	1	1
16 BHK	1	1	1	0	1	1	0	1
18 Bedroom	1	1	1	1	1	1	0	1
19 BHK	1	1	1	0	1	1	0	1
2 BHK	5199	5199	5199	3439	5199	5198	5152	5199
2 Bedroom	329	329	329	16	329	329	328	329
27 BHK	1	1	1	0	1	1	1	1
3 BHK	4310	4310	4309	3154	4310	4287	4129	4310
3 Bedroom	547	547	547	128	547	546	527	547
4 BHK	591	591	591	416	591	577	489	591
4 Bedroom	826	826	826	219	826	818	749	826
43 Bedroom	1	1	1	0	1	1	1	1
5 BHK	59	59	59	23	59	57	36	59
5 Bedroom	297	297	297	22	297	296	263	297
6 BHK	30	30	30	5	30	30	23	30
6 Bedroom	191	191	191	4	191	191	169	191
7 BHK	17	17	17	1	17	17	16	17
7 Bedroom	83	83	83	0	83	83	69	83
8 BHK	5	5	5	0	5	5	3	5
8 Bedroom	84	84	84	1	84	84	65	84
9 BHK	8	8	8	1	8	8	5	8
9 Bedroom	46	46	46	2	46	46	29	46

In [400]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

In [402]:
```
df.nunique()
```

Out[402]:
```
area_type          4
availability      81
location        1305
size              31
society         2688
total_sqft      2117
bath              19
balcony            4
price           1994
dtype: int64
```

In [403]:
```
df.describe()
```

Out[403]:

	bath	balcony	price
count	13247.000000	12711.000000	13320.000000
mean	2.692610	1.584376	112.565627
std	1.341458	0.817263	148.971674
min	1.000000	0.000000	8.000000
25%	2.000000	1.000000	50.000000
50%	2.000000	2.000000	72.000000
75%	3.000000	2.000000	120.000000
max	40.000000	3.000000	3600.000000

In [409]:
```
df['balcony'].value_counts()
```

Out[409]:
```
2.0    5113
1.0    4897
3.0    1672
0.0    1029
Name: balcony, dtype: int64
```

In [410]:
```
print(dir(df))
```

```
['T', '_AXIS_LEN', '_AXIS_ORDERS', '_AXIS_TO_AXIS_NUMBER', '_HANDLED_TYPE
S', '__abs__', '__add__', '__and__', '__annotations__', '__array__', '__a
rray_priority__', '__array_ufunc__', '__array_wrap__', '__bool__', '__cla
ss__', '__contains__', '__copy__', '__dataframe__', '__deepcopy__', '__de
lattr__', '__delitem__', '__dict__', '__dir__', '__divmod__', '__doc__',
'__eq__', '__finalize__', '__floordiv__', '__format__', '__ge__', '__geta
ttr__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__h
ash__', '__iadd__', '__iand__', '__ifloordiv__', '__imod__', '__imul__',
'__init__', '__init_subclass__', '__invert__', '__ior__', '__ipow__', '__
isub__', '__iter__', '__itruediv__', '__ixor__', '__le__', '__len__', '__
lt__', '__matmul__', '__mod__', '__module__', '__mul__', '__ne__', '__neg
__', '__new__', '__nonzero__', '__or__', '__pos__', '__pow__', '__radd_
_', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__',
'__rfloordiv__', '__rmatmul__', '__rmod__', '__rmul__', '__ror__', '__rou
nd__', '__rpow__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__',
'__setitem__', '__setstate__', '__sizeof__', '__str__', '__sub__', '__sub
classhook__', '__truediv__', '__weakref__', '__xor__', '_accessors', '_ac
cum_func', '_add_numeric_operations', '_agg_by_level', '_agg_examples_do
c', '_agg_summary_and_see_also_doc', '_align_frame', '_align_series', '_a
ppend', '_arith_method', '_as_manager', '_attrs', '_box_col_values', '_ca
n_fast_transpose', '_check_inplace_and_allows_duplicate_labels', '_check_
inplace_setting', '_check_is_chained_assignment_possible', '_check_label_
or_level_ambiguity', '_check_setitem_copy', '_clear_item_cache', '_clip_w
ith_one_bound', '_clip_with_scalar', '_cmp_method', '_combine_frame', '_c
onsolidate', '_consolidate_inplace', '_construct_axes_dict', '_construct_
axes_from_arguments', '_construct_result', '_constructor', '_constructor_
sliced', '_convert', '_count_level', '_data', '_dir_additions', '_dir_del
etions', '_dispatch_frame_op', '_drop_axis', '_drop_labels_or_levels', '_
ensure_valid_index', '_find_valid_index', '_flags', '_from_arrays', '_get
_agg_axis', '_get_axis', '_get_axis_name', '_get_axis_number', '_get_axis
_resolvers', '_get_block_manager_axis', '_get_bool_data', '_get_cleaned_c
olumn_resolvers', '_get_column_array', '_get_index_resolvers', '_get_item
_cache', '_get_label_or_level_values', '_get_numeric_data', '_get_value',
'_getitem_bool_array', '_getitem_multilevel', '_gotitem', '_hidden_attr
s', '_indexed_same', '_info_axis', '_info_axis_name', '_info_axis_numbe
r', '_info_repr', '_init_mgr', '_inplace_method', '_internal_names', '_in
ternal_names_set', '_is_copy', '_is_homogeneous_type', '_is_label_or_leve
l_reference', '_is_label_reference', '_is_level_reference', '_is_mixed_ty
pe', '_is_view', '_iset_item', '_iset_item_mgr', '_iset_not_inplace', '_i
tem_cache', '_iter_column_arrays', '_ixs', '_join_compat', '_logical_fun
c', '_logical_method', '_maybe_cache_changed', '_maybe_update_cacher', '_
metadata', '_mgr', '_min_count_stat_function', '_needs_reindex_multi', '_
protect_consolidate', '_reduce', '_reduce_axis1', '_reindex_axes', '_rein
dex_columns', '_reindex_index', '_reindex_multi', '_reindex_with_indexer
s', '_rename', '_replace_columnwise', '_repr_data_resource_', '_repr_fits
_horizontal_', '_repr_fits_vertical_', '_repr_html_', '_repr_latex_', '_r
eset_cache', '_reset_cacher', '_sanitize_column', '_series', '_set_axis',
'_set_axis_name', '_set_axis_nocheck', '_set_is_copy', '_set_item', '_set
_item_frame_value', '_set_item_mgr', '_set_value', '_setitem_array', '_se
titem_frame', '_setitem_slice', '_slice', '_stat_axis', '_stat_axis_nam
e', '_stat_axis_number', '_stat_function', '_stat_function_ddof', '_tak
e', '_take_with_is_copy', '_to_dict_of_blocks', '_typ', '_update_inplac
e', '_validate_dtype', '_values', '_where', 'abs', 'add', 'add_prefix',
'add_suffix', 'agg', 'aggregate', 'align', 'all', 'any', 'append', 'appl
y', 'applymap', 'area_type', 'asfreq', 'asof', 'assign', 'astype', 'at',
'at_time', 'attrs', 'availability', 'axes', 'backfill', 'balcony', 'bat
h', 'between_time', 'bfill', 'bool', 'boxplot', 'clip', 'columns', 'combi
ne', 'combine_first', 'compare', 'convert_dtypes', 'copy', 'corr', 'corrw
ith', 'count', 'cov', 'cummax', 'cummin', 'cumprod', 'cumsum', 'describ
e', 'diff', 'div', 'divide', 'dot', 'drop', 'drop_duplicates', 'dropleve
l', 'dropna', 'dtypes', 'duplicated', 'empty', 'eq', 'equals', 'eval', 'e
wm', 'expanding', 'explode', 'ffill', 'fillna', 'filter', 'first', 'first
_valid_index', 'flags', 'floordiv', 'from_dict', 'from_records', 'ge', 'g
et', 'groupby', 'gt', 'head', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc',
'index', 'infer_objects', 'info', 'insert', 'interpolate', 'isetitem', 'i
sin', 'isna', 'isnull', 'items', 'iteritems', 'iterrows', 'itertuples',
'join', 'keys', 'kurt', 'kurtosis', 'last', 'last_valid_index', 'le', 'lo
c', 'location', 'lookup', 'lt', 'mad', 'mask', 'max', 'mean', 'median',
'melt', 'memory_usage', 'merge', 'min', 'mod', 'mode', 'mul', 'multiply',
'ndim', 'ne', 'nlargest', 'notna', 'notnull', 'nsmallest', 'nunique', 'pa
d', 'pct_change', 'pipe', 'pivot', 'pivot_table', 'plot', 'pop', 'pow',
'price', 'prod', 'product', 'quantile', 'query', 'radd', 'rank', 'rdiv',
'reindex', 'reindex_like', 'rename', 'rename_axis', 'reorder_levels', 're
place', 'resample', 'reset_index', 'rfloordiv', 'rmod', 'rmul', 'rollin
g', 'round', 'rpow', 'rsub', 'rtruediv', 'sample', 'select_dtypes', 'se
m', 'set_axis', 'set_flags', 'set_index', 'shape', 'shift', 'size', 'ske
w', 'slice_shift', 'society', 'sort_index', 'sort_values', 'squeeze', 'st
ack', 'std', 'style', 'sub', 'subtract', 'sum', 'swapaxes', 'swaplevel',
'tail', 'take', 'to_clipboard', 'to_csv', 'to_dict', 'to_excel', 'to_feat
her', 'to_gbq', 'to_hdf', 'to_html', 'to_json', 'to_latex', 'to_markdow
n', 'to_numpy', 'to_orc', 'to_parquet', 'to_period', 'to_pickle', 'to_rec
ords', 'to_sql', 'to_stata', 'to_string', 'to_timestamp', 'to_xarray', 't
o_xml', 'total_sqft', 'transform', 'transpose', 'truediv', 'truncate', 't
z_convert', 'tz_localize', 'unstack', 'update', 'value_counts', 'values',
'var', 'where', 'xs']
```

In [411]:
```
df.count()
```

Out[411]:
```
area_type       13320
availability    13320
location        13319
size            13304
society          7818
total_sqft      13320
bath            13247
balcony         12711
price           13320
dtype: int64
```

In [412]:
```
df.mean()
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\3698961737.py:1: Futur
eWarning: The default value of numeric_only in DataFrame.mean is deprecat
ed. In a future version, it will default to False. In addition, specifyin
g 'numeric_only=None' is deprecated. Select only valid columns or specify
the value of numeric_only to silence this warning.
  df.mean()
```

Out[412]:
```
bath         2.692610
balcony      1.584376
price      112.565627
dtype: float64
```

In [414]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   society       7818 non-null   object 
 5   total_sqft    13320 non-null  object 
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
```

In [415]:
```
df.shape
```

Out[415]:
```
(13320, 9)
```

In [416]:
```
df.size
```

Out[416]:
```
119880
```

In [417]:
```
df.ndim
```

Out[417]:
```
2
```

In [421]:
```
df.any()
```

Out[421]:
```
area_type       True
availability    True
location        True
size            True
society         True
total_sqft      True
bath            True
balcony         True
price           True
dtype: bool
```

In [426]:
```
df.corr()
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_10328\1134722465.py:1: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecat
ed. In a future version, it will default to False. Select only valid colu
mns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[426]:

	bath	balcony	price
bath	1.000000	0.204201	0.456345
balcony	0.204201	1.000000	0.120355
price	0.456345	0.120355	1.000000

In [428]:
```
df.
```

Out[428]:
```
<bound method DataFrame.diff of                   area_type   availabilit
y                  location  \
0      Super built-up  Area         19-Dec  Electronic City Phase II
1                Plot  Area  Ready To Move          Chikka Tirupathi
2            Built-up  Area  Ready To Move               Uttarahalli
3      Super built-up  Area  Ready To Move        Lingadheeranahalli
4      Super built-up  Area  Ready To Move                  Kothanur
...                     ...            ...                       ...
13315        Built-up  Area  Ready To Move                Whitefield
13316  Super built-up  Area  Ready To Move             Richards Town
13317        Built-up  Area  Ready To Move     Raja Rajeshwari Nagar
13318  Super built-up  Area         18-Jun           Padmanabhanagar
13319  Super built-up  Area  Ready To Move              Doddathoguru

            size  society total_sqft  bath  balcony   price
0          2 BHK  Coomee        1056   2.0      1.0   39.07
1      4 Bedroom  Theanmp       2600   5.0      3.0  120.00
2          3 BHK      NaN       1440   2.0      3.0   62.00
3          3 BHK  Soiewre       1521   3.0      1.0   95.00
4          2 BHK      NaN       1200   2.0      1.0   51.00
...          ...      ...        ...   ...      ...     ...
13315  5 Bedroom  ArsiaEx       3453   4.0      0.0  231.00
13316      4 BHK      NaN       3600   5.0      NaN  400.00
13317      2 BHK  Mahla T       1141   2.0      1.0   60.00
13318      4 BHK  SollyCl       4689   4.0      1.0  488.00
13319      1 BHK      NaN        550   1.0      1.0   17.00

[13320 rows x 9 columns]>
```

In [432]:
```
df[df.isnull()]
```

Out[432]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
13315	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13316	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13317	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13318	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13319	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

13320 rows × 9 columns

In [433]:
```
df[df.isna()]
```

Out[433]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
13315	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13316	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13317	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13318	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13319	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

13320 rows × 9 columns

In [449]:
```
# list(df.iterrows())
```

In [451]:
```
df.keys()
```

Out[451]:
```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

```
In [454]: df.columns
```

```
Out[454]: Index(['area_type', 'availability', 'location', 'size', 'society',
                 'total_sqft', 'bath', 'balcony', 'price'],
                dtype='object')
```

```
In [475]: # df[df.notnull()]
```

```
In [477]: df.nunique()
```

```
Out[477]: area_type          4
          availability      81
          location        1305
          size              31
          society         2688
          total_sqft      2117
          bath              19
          balcony            4
          price           1994
          dtype: int64
```

```
In [487]: df['size'].value_counts()
```

```
Out[487]: 2 BHK         5199
          3 BHK         4310
          4 Bedroom      826
          4 BHK          591
          3 Bedroom      547
          1 BHK          538
          2 Bedroom      329
          5 Bedroom      297
          6 Bedroom      191
          1 Bedroom      105
          8 Bedroom       84
          7 Bedroom       83
          5 BHK           59
          9 Bedroom       46
          6 BHK           30
          7 BHK           17
          1 RK            13
          10 Bedroom      12
          9 BHK            8
          8 BHK            5
          11 BHK           2
          11 Bedroom       2
          10 BHK           2
          14 BHK           1
          13 BHK           1
          12 Bedroom       1
          27 BHK           1
          43 Bedroom       1
          16 BHK           1
          19 BHK           1
          18 Bedroom       1
          Name: size, dtype: int64
```

```
In [ ]:
```

```
In [ ]:
```