In [32]:

```
#Data Cleaning and Preparation
```

In [33]:

```
import numpy as np
import pandas as pd
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
np.random.seed(12345)
```

In [34]:

```
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)
```

In [35]:

```
#Handling Missing Data
```

In [36]:

```
string_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])
string_data
string_data.isnull()
```

Out[36]:

```
0    False
1    False
2     True
3    False
dtype: bool
```

In [37]:

```
string_data[0] = None
string_data.isnull()
```

Out[37]:

```
0     True
1    False
2     True
3    False
dtype: bool
```

In [38]:

```
#Filtering Out Missing Data
```

In [39]:

```
from numpy import nan as NA
data = pd.Series([1, NA, 3.5, NA, 7])
data.dropna()
```

Out[39]:

```
0    1.0
2    3.5
4    7.0
dtype: float64
```

In [40]:

```
data[data.notnull()]
```

Out[40]:

```
0    1.0
2    3.5
4    7.0
dtype: float64
```

In [41]:

```
data = pd.DataFrame([[1., 6.5, 3.], [1., NA, NA],
                     [NA, NA, NA], [NA, 6.5, 3.]])
cleaned = data.dropna()
data
cleaned
```

Out[41]:

	0	1	2
0	1.0	6.5	3.0

In [42]:

```
data.dropna(how='all')
```

Out[42]:

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
3	NaN	6.5	3.0

In [43]:

```
data[4] = NA
data
data.dropna(axis=1, how='all')
```

Out[43]:

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

In [44]:

```
df = pd.DataFrame(np.random.randn(7, 3))
df.iloc[:4, 1] = NA
df.iloc[:2, 2] = NA
df
df.dropna()
df.dropna(thresh=2)
```

Out[44]:

	0	1	2
2	0.092908	NaN	0.769023
3	1.246435	NaN	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

In [45]:

```
#Filling In Missing Data
```

In [46]:

```
df.fillna(0)
```

Out[46]:

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

In [47]:

```
df.fillna({1: 0.5, 2: 0})
```

Out[47]:

	0	1	2
0	-0.204708	0.500000	0.000000
1	-0.555730	0.500000	0.000000
2	0.092908	0.500000	0.769023
3	1.246435	0.500000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

In [48]:

```
_ = df.fillna(0, inplace=True)
df
```

Out[48]:

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

In [49]:

```
df = pd.DataFrame(np.random.randn(6, 3))
df.iloc[2:, 1] = NA
df.iloc[4:, 2] = NA
df
df.fillna(method='ffill')
df.fillna(method='ffill', limit=2)
```

Out[49]:

	0	1	2
0	0.476985	3.248944	-1.021228
1	-0.577087	0.124121	0.302614
2	0.523772	0.124121	1.343810
3	-0.713544	0.124121	-2.370232
4	-1.860761	NaN	-2.370232
5	-1.265934	NaN	-2.370232

In [50]:

```
data = pd.Series([1., NA, 3.5, NA, 7])
data.fillna(data.mean())
```

Out[50]:

```
0    1.000000
1    3.833333
2    3.500000
3    3.833333
4    7.000000
dtype: float64
```

In [51]:

```
#Removing Duplicates
```

In [52]:

```
data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
                     'k2': [1, 1, 2, 3, 3, 4, 4]})
data
```

Out[52]:

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4
6	two	4

In [53]:

```
data.duplicated()
```

Out[53]:

```
0    False
1    False
2    False
3    False
4    False
5    False
6     True
dtype: bool
```

In [54]:

```
data.drop_duplicates()
```

Out[54]:

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4

In [55]:

```
data['v1'] = range(7)
data.drop_duplicates(['k1'])
```

Out[55]:

	k1	k2	v1
0	one	1	0
1	two	1	1

In [56]:

```
data.drop_duplicates(['k1', 'k2'], keep='last')
```

Out[56]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	3	4
6	two	4	6

In [57]:

```
#Transforming Data Using a Lambda Function
```

In [58]:

```
data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon',
                              'Pastrami', 'corned beef', 'Bacon',
                              'pastrami', 'honey ham', 'nova lox'],
                     'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
data
```

Out[58]:

	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0

In [59]:

```
meat_to_animal = {
  'bacon': 'pig',
  'pulled pork': 'pig',
  'pastrami': 'cow',
  'corned beef': 'cow',
  'honey ham': 'pig',
  'nova lox': 'salmon'
}
```

In [60]:

```
lowercased = data['food'].str.lower()
lowercased
data['animal'] = lowercased.map(meat_to_animal)
data
```

Out[60]:

	food	ounces	animal
0	bacon	4.0	pig
1	pulled pork	3.0	pig
2	bacon	12.0	pig
3	Pastrami	6.0	cow
4	corned beef	7.5	cow
5	Bacon	8.0	pig
6	pastrami	3.0	cow
7	honey ham	5.0	pig
8	nova lox	6.0	salmon

In [61]:

```
data['food'].map(lambda x: meat_to_animal[x.lower()])
```

Out[61]:

```
0       pig
1       pig
2       pig
3       cow
4       cow
5       pig
6       cow
7       pig
8    salmon
Name: food, dtype: object
```

In [62]:

```
#Replacing Values inplace of missing values
```

In [63]:

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

Out[63]:

```
0       1.0
1    -999.0
2       2.0
3    -999.0
4   -1000.0
5       3.0
dtype: float64
```

In [64]:

```
data.replace(-999, np.nan)
```

Out[64]:

```
0       1.0
1       NaN
2       2.0
3       NaN
4   -1000.0
5       3.0
dtype: float64
```

In [65]:

```
data.replace([-999, -1000], np.nan)
```

Out[65]:

```
0    1.0
1    NaN
2    2.0
3    NaN
4    NaN
5    3.0
dtype: float64
```

In [66]:

```
data.replace([-999, -1000], [np.nan, 0])
```

Out[66]:

```
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

In [67]:

```
data.replace({-999: np.nan, -1000: 0})
```

Out[67]:

```
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

In [68]:

```
#Renaming Axis Indexes
```

In [69]:

```
data = pd.DataFrame(np.arange(12).reshape((3, 4)),
                    index=['Ohio', 'Colorado', 'New York'],
                    columns=['one', 'two', 'three', 'four'])
```

In [70]:

```
data
```

Out[70]:

	one	**two**	**three**	**four**
Ohio	0	1	2	3
Colorado	4	5	6	7
New York	8	9	10	11

In [71]:

```
transform = lambda x: x[:4].upper()
data.index.map(transform)
```

Out[71]:

```
Index(['OHIO', 'COLO', 'NEW '], dtype='object')
```

In [72]:

```
data.index = data.index.map(transform)
data
```

Out[72]:

	one	two	three	four
OHIO	0	1	2	3
COLO	4	5	6	7
NEW	8	9	10	11

In [73]:

```
data.rename(index=str.title, columns=str.upper)
```

Out[73]:

	ONE	TWO	THREE	FOUR
Ohio	0	1	2	3
Colo	4	5	6	7
New	8	9	10	11

In [74]:

```
data.rename(index={'OHIO': 'INDIANA'},
            columns={'three': 'peekaboo'})
```

Out[74]:

	one	two	peekaboo	four
INDIANA	0	1	2	3
COLO	4	5	6	7
NEW	8	9	10	11

In [75]:

```
data.rename(index={'OHIO': 'INDIANA'}, inplace=True)
data
```

Out[75]:

	one	two	three	four
INDIANA	0	1	2	3
COLO	4	5	6	7
NEW	8	9	10	11

In [76]:

```
# binning or bucketizing for continuous variables
```

In [77]:

```
ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]
```

In [78]:

```
bins = [18, 25, 35, 60, 100]
cats = pd.cut(ages, bins)
cats
```

Out[78]:

```
[(18, 25], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25, 35], (60, 10
0], (35, 60], (35, 60], (25, 35]]
Length: 12
Categories (4, interval[int64]): [(18, 25] < (25, 35] < (35, 60] < (60, 10
0]]
```

In [79]:

```
cats.codes
cats.categories
pd.value_counts(cats)
```

Out[79]:

```
(18, 25]     5
(35, 60]     3
(25, 35]     3
(60, 100]    1
dtype: int64
```

In [80]:

```
pd.cut(ages, [18, 26, 36, 61, 100], right=False)
```

Out[80]:

```
[[18, 26), [18, 26), [18, 26), [26, 36), [18, 26), ..., [26, 36), [61, 10
0), [36, 61), [36, 61), [26, 36)]
Length: 12
Categories (4, interval[int64]): [[18, 26) < [26, 36) < [36, 61) < [61, 10
0)]
```

In [81]:

```
group_names = ['Youth', 'YoungAdult', 'MiddleAged', 'Senior']
pd.cut(ages, bins, labels=group_names)
```

Out[81]:

```
[Youth, Youth, Youth, YoungAdult, Youth, ..., YoungAdult, Senior, MiddleAg
ed, MiddleAged, YoungAdult]
Length: 12
Categories (4, object): [Youth < YoungAdult < MiddleAged < Senior]
```

In [82]:

```
data = np.random.rand(20)
pd.cut(data, 4, precision=2)
```

Out[82]:

```
[(0.34, 0.55], (0.34, 0.55], (0.76, 0.97], (0.76, 0.97], (0.34, 0.55],
..., (0.34, 0.55], (0.34, 0.55], (0.55, 0.76], (0.34, 0.55], (0.12, 0.34]]
Length: 20
Categories (4, interval[float64]): [(0.12, 0.34] < (0.34, 0.55] < (0.55,
0.76] < (0.76, 0.97]]
```

In [83]:

```
data = np.random.randn(1000)  # Normally distributed
cats = pd.qcut(data, 4)  # Cut into quartiles
cats
pd.value_counts(cats)
```

Out[83]:

```
(0.62, 3.928]       250
(-0.0265, 0.62]     250
(-0.68, -0.0265]    250
(-2.95, -0.68]      250
dtype: int64
```

In [84]:

```
pd.qcut(data, [0, 0.1, 0.5, 0.9, 1.])
```

Out[84]:

```
[(-0.0265, 1.286], (-0.0265, 1.286], (-1.187, -0.0265], (-0.0265, 1.286],
(-0.0265, 1.286], ..., (-1.187, -0.0265], (-1.187, -0.0265], (-2.95, -1.18
7], (-0.0265, 1.286], (-1.187, -0.0265]]
Length: 1000
Categories (4, interval[float64]): [(-2.95, -1.187] < (-1.187, -0.0265] <
(-0.0265, 1.286] < (1.286, 3.928]]
```

In [85]:

```
#filter out the outliers
```

In [86]:

```
data = pd.DataFrame(np.random.randn(1000, 4))
data.describe()
```

Out[86]:

	0	1	2	3
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.049091	0.026112	-0.002544	-0.051827
std	0.996947	1.007458	0.995232	0.998311
min	-3.645860	-3.184377	-3.745356	-3.428254
25%	-0.599807	-0.612162	-0.687373	-0.747478
50%	0.047101	-0.013609	-0.022158	-0.088274
75%	0.756646	0.695298	0.699046	0.623331
max	2.653656	3.525865	2.735527	3.366626

In [87]:

```
col = data[2]
col[np.abs(col) > 3]
```

Out[87]:

```
41    -3.399312
136   -3.745356
Name: 2, dtype: float64
```

In [88]:

```
data[(np.abs(data) > 3).any(1)]
```

Out[88]:

	0	1	2	3
41	0.457246	-0.025907	-3.399312	-0.974657
60	1.951312	3.260383	0.963301	1.201206
136	0.508391	-0.196713	-3.745356	-1.520113
235	-0.242459	-3.056990	1.918403	-0.578828
258	0.682841	0.326045	0.425384	-3.428254
322	1.179227	-3.184377	1.369891	-1.074833
544	-3.548824	1.553205	-2.186301	1.277104
635	-0.578093	0.193299	1.397822	3.366626
782	-0.207434	3.525865	0.283070	0.544635
803	-3.645860	0.255475	-0.549574	-1.907459

In [89]:

```
data[np.abs(data) > 3] = np.sign(data) * 3
data.describe()
```

Out[89]:

	0	1	2	3
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.050286	0.025567	-0.001399	-0.051765
std	0.992920	1.004214	0.991414	0.995761
min	-3.000000	-3.000000	-3.000000	-3.000000
25%	-0.599807	-0.612162	-0.687373	-0.747478
50%	0.047101	-0.013609	-0.022158	-0.088274
75%	0.756646	0.695298	0.699046	0.623331
max	2.653656	3.000000	2.735527	3.000000

In [90]:

```
np.sign(data).head()
```

Out[90]:

	0	1	2	3
0	-1.0	1.0	-1.0	1.0
1	1.0	-1.0	1.0	-1.0
2	1.0	1.0	1.0	-1.0
3	-1.0	-1.0	1.0	-1.0
4	-1.0	1.0	-1.0	-1.0

In [91]:

```
#reshaping data
import numpy as np
import pandas as pd
pd.options.display.max_rows = 20
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)
```

In [92]:

```
data = pd.Series(np.random.randn(9),
                 index=[['a', 'a', 'a', 'b', 'b', 'c', 'c', 'd', 'd'],
                        [1, 2, 3, 1, 3, 1, 2, 2, 3]])
data
```

Out[92]:

```
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
d  2    0.281746
   3    0.769023
dtype: float64
```

In [93]:

```
data.index
```

Out[93]:

```
MultiIndex(levels=[['a', 'b', 'c', 'd'], [1, 2, 3]],
           labels=[[0, 0, 0, 1, 1, 2, 2, 3, 3], [0, 1, 2, 0, 2, 0, 1, 1,
2]])
```

In [94]:

```
data['b']
data['b':'c']
data.loc[['b', 'd']]
```

Out[94]:

```
b  1   -0.555730
   3    1.965781
d  2    0.281746
   3    0.769023
dtype: float64
```

In [95]:

```
data.loc[:, 2]
```

Out[95]:

```
a    0.478943
c    0.092908
d    0.281746
dtype: float64
```

In [96]:

```
data.unstack()
```

Out[96]:

	1	2	3
a	-0.204708	0.478943	-0.519439
b	-0.555730	NaN	1.965781
c	1.393406	0.092908	NaN
d	NaN	0.281746	0.769023

In [97]:

```
data.unstack().stack()
```

Out[97]:

```
a  1   -0.204708
   2    0.478943
   3   -0.519439
b  1   -0.555730
   3    1.965781
c  1    1.393406
   2    0.092908
d  2    0.281746
   3    0.769023
dtype: float64
```

In [98]:

```
frame = pd.DataFrame(np.arange(12).reshape((4, 3)),
                     index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],
                     columns=[['Ohio', 'Ohio', 'Colorado'],
                              ['Green', 'Red', 'Green']])
frame
```

Out[98]:

		Ohio		**Colorado**
		Green	**Red**	**Green**
a	**1**	0	1	2
	2	3	4	5
b	**1**	6	7	8
	2	9	10	11

In [99]:

```
frame.index.names = ['key1', 'key2']
frame.columns.names = ['state', 'color']
frame
```

Out[99]:

	state	**Ohio**		**Colorado**
	color	**Green**	**Red**	**Green**
key1	**key2**			
a	**1**	0	1	2
	2	3	4	5
b	**1**	6	7	8
	2	9	10	11

In [100]:

```
frame['Ohio']
```

Out[100]:

	color	Green	Red
key1	**key2**		
a	**1**	0	1
	2	3	4
b	**1**	6	7
	2	9	10

In [101]:

```
frame.swaplevel('key1', 'key2')
```

Out[101]:

	state	Ohio		Colorado
	color	**Green**	**Red**	**Green**
key2	**key1**			
1	**a**	0	1	2
2	**a**	3	4	5
1	**b**	6	7	8
2	**b**	9	10	11

In [102]:

```
frame.sort_index(level=1)
frame.swaplevel(0, 1).sort_index(level=0)
```

Out[102]:

	state	Ohio		Colorado
	color	**Green**	**Red**	**Green**
key2	**key1**			
1	**a**	0	1	2
	b	6	7	8
2	**a**	3	4	5
	b	9	10	11