Pandas - Data Preparation and Cleaning

Handling Missing Values

In [1]:
```
import pandas as pd
import numpy  as np
```

In [2]:
```
np.random.seed(321)
df = pd.DataFrame(np.random.randint(1, 100, (5, 3)),
                  index=['a', 'c', 'e', 'f', 'h'],
                  columns=['one', 'two', 'three'])
df
```

Out[2]:

	one	two	three
a	27	32	42
c	73	18	41
e	27	89	73
f	84	86	2
h	28	27	62

In [3]:
```
df['four'] = 'abc'
df['five'] = df['one'] > 50
df['six'] =  pd.Timestamp('20190101')
df
```

Out[3]:

	one	two	three	four	five	six
a	27	32	42	abc	False	2019-01-01
c	73	18	41	abc	True	2019-01-01
e	27	89	73	abc	False	2019-01-01
f	84	86	2	abc	True	2019-01-01
h	28	27	62	abc	False	2019-01-01

In [4]:
```
df2 = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
df2
```

Out[4]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [5]:
```
df2['one']
```

Out[5]:
```
a    27.0
b     NaN
c    73.0
d     NaN
e    27.0
f    84.0
g     NaN
h    28.0
Name: one, dtype: float64
```

In [6]:
```
pd.isna(df2['one'])
```

Out[6]:
```
a    False
b     True
c    False
d     True
e    False
f    False
g     True
h    False
Name: one, dtype: bool
```

In [7]:
```
pd.notna(df2['one'])
```

Out[7]:
```
a     True
b    False
c     True
d    False
e     True
f     True
g    False
h     True
Name: one, dtype: bool
```

In [8]:
```
df2
```

Out[8]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [9]:
```
df2.isna()
```

Out[9]:

	one	two	three	four	five	six
a	False	False	False	False	False	False
b	True	True	True	True	True	True
c	False	False	False	False	False	False
d	True	True	True	True	True	True
e	False	False	False	False	False	False
f	False	False	False	False	False	False
g	True	True	True	True	True	True
h	False	False	False	False	False	False

In [10]:
```
df2['one']
```

Out[10]:
```
a    27.0
b     NaN
c    73.0
d     NaN
e    27.0
f    84.0
g     NaN
h    28.0
Name: one, dtype: float64
```

In [11]:
```
df2['one'].sum(), df2['one'].prod()
```

Out[11]:
```
(239.0, 125166384.0)
```

In [12]:
```
df2['one'].cumsum()
```

Out[12]:
```
a     27.0
b      NaN
c    100.0
d      NaN
e    127.0
f    211.0
g      NaN
h    239.0
Name: one, dtype: float64
```

In [13]:
```
df2['one'].mean()
```

Out[13]:
```
47.8
```

In [14]:
```
df2['one'].mean(skipna=False)
```

Out[14]:
```
nan
```

Filling missing values

In [15]:
```
df2
```

Out[15]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [16]:
```
df2.fillna(0)
```

Out[16]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01 00:00:00
b	0.0	0.0	0.0	0	0	0
c	73.0	18.0	41.0	abc	True	2019-01-01 00:00:00
d	0.0	0.0	0.0	0	0	0
e	27.0	89.0	73.0	abc	False	2019-01-01 00:00:00
f	84.0	86.0	2.0	abc	True	2019-01-01 00:00:00
g	0.0	0.0	0.0	0	0	0
h	28.0	27.0	62.0	abc	False	2019-01-01 00:00:00

In [17]:
```
df2['one']
```

Out[17]:
```
a    27.0
b     NaN
c    73.0
d     NaN
e    27.0
f    84.0
g     NaN
h    28.0
Name: one, dtype: float64
```

In [18]:
```
df2['one'].fillna('missing')
```

Out[18]:
```
a         27
b    missing
c         73
d    missing
e         27
f         84
g    missing
h         28
Name: one, dtype: object
```

In [19]:
```
df2
```

Out[19]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [20]:
```
df2.fillna(method='pad')  #ffill
```

Out[20]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	27.0	32.0	42.0	abc	False	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
d	73.0	18.0	41.0	abc	True	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	84.0	86.0	2.0	abc	True	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [21]:
```
df2.ffill()
```

Out[21]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	27.0	32.0	42.0	abc	False	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
d	73.0	18.0	41.0	abc	True	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	84.0	86.0	2.0	abc	True	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [22]:
```
df2
```

Out[22]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [23]:
```
df2.fillna(method='bfill')
```

Out[23]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	73.0	18.0	41.0	abc	True	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
d	27.0	89.0	73.0	abc	False	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	28.0	27.0	62.0	abc	False	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [24]:
```
df2.bfill()
```

Out[24]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	73.0	18.0	41.0	abc	True	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
d	27.0	89.0	73.0	abc	False	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	28.0	27.0	62.0	abc	False	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [25]:
```
df2
```

Out[25]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [26]:
```
df2.fillna(df.mean())
```

Out[26]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	47.8	50.4	44.0	NaN	0.4	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	47.8	50.4	44.0	NaN	0.4	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	47.8	50.4	44.0	NaN	0.4	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [27]:
```
df2['five']
```

Out[27]:
```
a    False
b      NaN
c     True
d      NaN
e    False
f     True
g      NaN
h    False
Name: five, dtype: object
```

In [28]:
```
df2['five'].sum(), df2['five'].count(), df2['five'].mean()
```

Out[28]:
```
(2, 5, 0.4)
```

In [29]:
```
df2
```

Out[29]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [30]:
```
df.mean()
```

Out[30]:
```
one      47.8
two      50.4
three    44.0
five      0.4
dtype: float64
```

In [31]:
```
df.mean()['one':'three']
```

Out[31]:
```
one      47.8
two      50.4
three    44.0
dtype: float64
```

In [32]:
```
df2.fillna(df.mean()['one':'three'])
```

Out[32]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	47.8	50.4	44.0	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	47.8	50.4	44.0	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	47.8	50.4	44.0	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

dropna

In [33]:
```
df2
```

Out[33]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

In [34]:
```
df2.dropna()
```

Out[34]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [35]:
```
df2.dropna(axis=0)
```

Out[35]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
c	73.0	18.0	41.0	abc	True	2019-01-01
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
h	28.0	27.0	62.0	abc	False	2019-01-01

In [36]:
```
df2.dropna(axis=1)
```

Out[36]:

a
b
c
d
e
f
g
h

In [37]:
```
df2
```

Out[37]:

	one	two	three	four	five	six
a	27.0	32.0	42.0	abc	False	2019-01-01
b	NaN	NaN	NaN	NaN	NaN	NaT
c	73.0	18.0	41.0	abc	True	2019-01-01
d	NaN	NaN	NaN	NaN	NaN	NaT
e	27.0	89.0	73.0	abc	False	2019-01-01
f	84.0	86.0	2.0	abc	True	2019-01-01
g	NaN	NaN	NaN	NaN	NaN	NaT
h	28.0	27.0	62.0	abc	False	2019-01-01

Removing Duplicates

In [38]:
```
data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
                     'k2': [1, 1, 2, 3, 2, 4, 4]})
data
```

Out[38]:

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

In [39]:
```
data.duplicated()
```

Out[39]:
```
0    False
1    False
2    False
3    False
4     True
5    False
6     True
dtype: bool
```

In [40]:
```
data.drop_duplicates()
```

Out[40]:

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

In [41]:
```
data['v1'] = range(7)
data
```

Out[41]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	2	4
5	two	4	5
6	two	4	6

In [42]:
```
data.drop_duplicates(['k1'])
```

Out[42]:

	k1	k2	v1
0	one	1	0
1	two	1	1

In [43]:
```
data.drop_duplicates(['k1'], keep = 'last')
```

Out[43]:

	k1	k2	v1
4	one	2	4
6	two	4	6

In [44]:
```
data
```

Out[44]:

	k1	k2	v1
0	one	1	0
1	two	1	1
2	one	2	2
3	two	3	3
4	one	2	4
5	two	4	5
6	two	4	6

In [45]:
```
data.drop_duplicates(['k1', 'k2'], keep = 'last')
```

Out[45]:

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

Data Transformation with Function Mapping

In [46]:
```
data = pd.DataFrame({'course': ['cs1', 'cs2', 'CS1', 'Cs2'],
                     'grade': ['A', 'B', 'A-', 'B+']})
data
```

Out[46]:

	course	grade
0	cs1	A
1	cs2	B
2	CS1	A-
3	Cs2	B+

In [47]:
```
course_to_name = {'cs1' : 'Python', 'cs2' : 'Java'}
```

In [48]:
```
data.course
```

Out[48]:
```
0    cs1
1    cs2
2    CS1
3    Cs2
Name: course, dtype: object
```

In [49]:
```
lc = data.course.str.lower()
lc
```

Out[49]:
```
0    cs1
1    cs2
2    cs1
3    cs2
Name: course, dtype: object
```

In [50]:
```
data['language'] = lc.map(course_to_name)
data
```

Out[50]:

	course	grade	language
0	cs1	A	Python
1	cs2	B	Java
2	CS1	A-	Python
3	Cs2	B+	Java

In [51]:
```
# or
data['language'] = data['course'].map(
                          lambda x : course_to_name[x.lower()])
data
```

Out[51]:

	course	grade	language
0	cs1	A	Python
1	cs2	B	Java
2	CS1	A-	Python
3	Cs2	B+	Java

Replacing Values

In [52]:
```
data = pd.Series([1., -999., 2., -999., -1000., 3.])
data
```

Out[52]:
```
0       1.0
1    -999.0
2       2.0
3    -999.0
4   -1000.0
5       3.0
dtype: float64
```

In [53]:
```
data.replace(-999, np.nan)
```

Out[53]:
```
0       1.0
1       NaN
2       2.0
3       NaN
4   -1000.0
5       3.0
dtype: float64
```

In [54]:
```
data.replace([-999, -1000], np.nan)
```

Out[54]:
```
0    1.0
1    NaN
2    2.0
3    NaN
4    NaN
5    3.0
dtype: float64
```

In [55]:
```
data.replace([-999, -1000], [np.nan, 0])
```

Out[55]:
```
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

In [56]:
```
data.replace({-999: np.nan, -1000: 0})
```

Out[56]:
```
0    1.0
1    NaN
2    2.0
3    NaN
4    0.0
5    3.0
dtype: float64
```

In [57]:
```
data = pd.DataFrame({'course': ['cs1', 'cs2', 'CS1', 'Cs2'],
                     'grade': ['A', 'B', 'A-', 'B+']})
data
```

Out[57]:

	course	grade
0	cs1	A
1	cs2	B
2	CS1	A-
3	Cs2	B+

In [58]:
```
data.replace({'A': 100, 'A-' : 90, 'B+': 80, 'B': 70}, inplace=True)
data
```

Out[58]:

	course	grade
0	cs1	100
1	cs2	70
2	CS1	90
3	Cs2	80

In [59]:
```
data.replace([100,90], method='bfill')
```

Out[59]:

	course	grade
0	cs1	70
1	cs2	70
2	CS1	80
3	Cs2	80

Renaming Axis Indexes

In [60]:
```
data = pd.DataFrame(np.arange(12).reshape((3, 4)),
                    index=['Ohio', 'Colorado', 'Massachusetts'],
                    columns=['one', 'two', 'three', 'four'])
data
```

Out[60]:

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

In [61]:
```
data.index = data.index.map(lambda x: x[:2].upper())
data
```

Out[61]:

	one	two	three	four
OH	0	1	2	3
CO	4	5	6	7
MA	8	9	10	11

In [62]:
```
data.rename(index=str.title, columns=str.upper)
```

Out[62]:

	ONE	TWO	THREE	FOUR
Oh	0	1	2	3
Co	4	5	6	7
Ma	8	9	10	11

In [63]:
```
data
```

Out[63]:

	one	two	three	four
OH	0	1	2	3
CO	4	5	6	7
MA	8	9	10	11

In [64]:
```
data.rename(index={'OH': 'Ohio', 'MA': 'Mass'},
            columns={'two': '2', 'four': '4'})
```

Out[64]:

	one	2	three	4
Ohio	0	1	2	3
CO	4	5	6	7
Mass	8	9	10	11

In [65]:
```
data
```

Out[65]:

	one	two	three	four
OH	0	1	2	3
CO	4	5	6	7
MA	8	9	10	11

In [66]:
```
data.rename(index={'OH': 'Ohio', 'MA': 'Mass'},
            columns={'two': '2', 'four': '4'}, inplace=True)
data
```

Out[66]:

	one	2	three	4
Ohio	0	1	2	3
CO	4	5	6	7
Mass	8	9	10	11

Discretization and Binning

- pandas.cut() - Bin values into discrete intervals.

In [67]:
```
ages = [37, 22, 25, 27, 21, 23, 20, 31, 61, 45, 41, 32]
```

In [68]:
```
# Distribute into 3 bins

cats = pd.cut(ages, 3)
cats
```

Out[68]:
```
[(33.667, 47.333], (19.959, 33.667], (19.959, 33.667], (19.959, 33.667], (19.959, 33.6
67], ..., (19.959, 33.667], (47.333, 61.0], (33.667, 47.333], (33.667, 47.333], (19.95
9, 33.667]]
Length: 12
Categories (3, interval[float64]): [(19.959, 33.667] < (33.667, 47.333] < (47.333, 61.
0]]
```

In [69]:
```
# Specify bin ranges

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

Out[69]:
```
[(35, 60], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25, 35], (60, 100], (35, 60],
(35, 60], (25, 35]]
Length: 12
Categories (4, interval[int64]): [(18, 25] < (25, 35] < (35, 60] < (60, 100]]
```

In [70]:
```
cats.codes
```

Out[70]:
```
array([2, 0, 0, 1, 0, 0, 0, 1, 3, 2, 2, 1], dtype=int8)
```

In [71]:
```
cats.categories
```

Out[71]:
```
IntervalIndex([(18, 25], (25, 35], (35, 60], (60, 100]],
              closed='right',
              dtype='interval[int64]')
```

In [72]:
```
pd.value_counts(cats)
```

Out[72]:
```
(18, 25]     5
(35, 60]     3
(25, 35]     3
(60, 100]    1
dtype: int64
```

In [73]:
```
pd.cut(ages, [18, 26, 36, 61, 100], right=False)
```

Out[73]:
```
[[36, 61), [18, 26), [18, 26), [26, 36), [18, 26), ..., [26, 36), [61, 100), [36, 61),
[36, 61), [26, 36)]
Length: 12
Categories (4, interval[int64]): [[18, 26) < [26, 36) < [36, 61) < [61, 100)]
```

In [74]:
```
group_names = ['Youth', 'YoungAdult', 'MiddleAged', 'Senior']
pd.cut(ages, bins, labels=group_names)
```

Out[74]:
```
[MiddleAged, Youth, Youth, YoungAdult, Youth, ..., YoungAdult, Senior, MiddleAged, Mid
dleAged, YoungAdult]
Length: 12
Categories (4, object): [Youth < YoungAdult < MiddleAged < Senior]
```

In [75]:
```
x,y = pd.cut(ages, bins, labels=group_names, retbins=True)
x
```

Out[75]:
```
[MiddleAged, Youth, Youth, YoungAdult, Youth, ..., YoungAdult, Senior, MiddleAged, Mid
dleAged, YoungAdult]
Length: 12
Categories (4, object): [Youth < YoungAdult < MiddleAged < Senior]
```

In [76]:
```
y
```

Out[76]:
```
array([ 18,  25,  35,  60, 100])
```

In [77]:
```
x.categories
```

Out[77]:
```
Index(['Youth', 'YoungAdult', 'MiddleAged', 'Senior'], dtype='object')
```

In [78]:
```
x.to_list()
```

Out[78]:
```
['MiddleAged',
 'Youth',
 'Youth',
 'YoungAdult',
 'Youth',
 'Youth',
 'Youth',
 'YoungAdult',
 'Senior',
 'MiddleAged',
 'MiddleAged',
 'YoungAdult']
```

In [79]:
```
pd.DataFrame({'age': ages, 'class':pd.cut(ages, bins, labels=group_names)})
```

Out[79]:

	age	class
0	37	MiddleAged
1	22	Youth
2	25	Youth
3	27	YoungAdult
4	21	Youth
5	23	Youth
6	20	Youth
7	31	YoungAdult
8	61	Senior
9	45	MiddleAged
10	41	MiddleAged
11	32	YoungAdult

qcut - Quantile-based discretization function

- Discretize variable into equal-sized buckets based on rank or based on sample quantiles

In [80]:
```
np.random.seed(123)

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

Out[80]:
```
(0.669, 3.572]                    250
(-0.0412, 0.669]                  250
(-0.685, -0.0412]                 250
(-3.2319999999999998, -0.685]     250
dtype: int64
```

In [81]:
```
data.describe()
```

Out[81]:
```
count    1000.000000
mean       -0.039564
std         1.001288
min        -3.231055
25%        -0.684557
50%        -0.041186
75%         0.668866
max         3.571579
dtype: float64
```

In [82]:
```
# Alternately, array of quantiles

cats = pd.qcut(data, [0, 0.25, 0.5, 0.75, 1.])
pd.value_counts(cats)
```

Out[82]:
```
(0.669, 3.572]                    250
(-0.0412, 0.669]                  250
(-0.685, -0.0412]                 250
(-3.2319999999999998, -0.685]     250
dtype: int64
```

In [83]:
```
# deciles

cats = pd.qcut(data, 10)
pd.value_counts(cats)
```

Out[83]:
```
(1.255, 3.572]                    100
(0.851, 1.255]                    100
(0.469, 0.851]                    100
(0.18, 0.469]                     100
(-0.0412, 0.18]                   100
(-0.273, -0.0412]                 100
(-0.548, -0.273]                  100
(-0.878, -0.548]                  100
(-1.326, -0.878]                  100
(-3.2319999999999998, -1.326]     100
dtype: int64
```

In [84]:
```
cats = pd.qcut(data, 4, labels=['A', 'B','C','D'])
pd.value_counts(cats)
```

Out[84]:
```
D    250
C    250
B    250
A    250
dtype: int64
```

In [ ]:

In [85]:
```
np.random.seed(123)

df = pd.DataFrame(np.random.randint(40, 101, (100, 3)),
                  columns=['Q1', 'Q2', 'Q3'])
df.head()
```

Out[85]:

	Q1	Q2	Q3
0	85	42	68
1	74	78	57
2	59	82	99
3	97	62	73
4	72	89	87

In [86]:
```
df['Average'] = np.round(df.mean(axis=1))
df.head()
```

Out[86]:

	Q1	Q2	Q3	Average
0	85	42	68	65.0
1	74	78	57	70.0
2	59	82	99	80.0
3	97	62	73	77.0
4	72	89	87	83.0

In [87]:
```
df['Average']
```

Out[87]:
```
0      65.0
1      70.0
2      80.0
3      77.0
4      83.0
       ...
95     69.0
96     58.0
97     82.0
98     71.0
99     73.0
Name: Average, Length: 100, dtype: float64
```

In [88]:
```
cats = pd.cut(df['Average'], bins=[0,40,60,80,100], labels=['D', 'C','B','A'])
pd.value_counts(cats, sort=False)
```

Out[88]:
```
D     0
C    19
B    66
A    15
Name: Average, dtype: int64
```

In [89]:
```
df['Grade'] = cats
df
```

Out[89]:

	Q1	Q2	Q3	Average	Grade
0	85	42	68	65.0	B
1	74	78	57	70.0	B
2	59	82	99	80.0	B
3	97	62	73	77.0	B
4	72	89	87	83.0	A
...	...	...	...	...	...
95	76	43	88	69.0	B
96	43	63	67	58.0	C
97	95	74	77	82.0	A
98	89	63	61	71.0	B
99	65	57	98	73.0	B

100 rows × 5 columns

In [90]:
```
df.sort_values(by='Average', ascending=False)
```

Out[90]:

	Q1	Q2	Q3	Average	Grade
62	92	98	98	96.0	A
90	88	96	100	95.0	A
87	99	94	88	94.0	A
78	86	94	95	92.0	A
76	86	85	97	89.0	A
...	...	...	...	...	...
53	44	70	47	54.0	C
94	68	42	50	53.0	C
36	43	51	61	52.0	C
79	49	57	43	50.0	C
64	54	46	41	47.0	C

100 rows × 5 columns

In [ ]:
```
```