pandas-lecture-5-dec-batch

June 17, 2023

0.1 Pandas-Lecture - 5

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

[2]:
```
!gdown 173A59xh2mnpmljCCB9bhC4C5eP2IS6qZ
```

```
Downloading…
From: https://drive.google.com/uc?id=173A59xh2mnpmljCCB9bhC4C5eP2IS6qZ
To: /Users/satish/Desktop/scaler/Dec Tue Batch - DAV-1/Pfizer_1.csv
100%|                    | 1.51k/1.51k [00:00<00:00, 2.11MB/s]
```

[3]:
```
data = pd.read_csv('Pfizer_1.csv')
data
```

[3]:

	Date	Drug_Name	Parameter	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	8:30:00	9:30:00	10:30:00
0	15-10-2020	diltiazem hydrochloride	Temperature	23.0	22.0	NaN	21.0	21.0	22	23.0	21.0	22.0	20
1	15-10-2020	diltiazem hydrochloride	Pressure	12.0	13.0	NaN	11.0	13.0	14	16.0	16.0	24.0	18
2	15-10-2020	docetaxel injection	Temperature	NaN	17.0								
3	15-10-2020	docetaxel injection	Pressure	NaN	22.0								
4	15-10-2020	ketamine hydrochloride	Temperature	24.0	NaN								
5	15-10-2020	ketamine hydrochloride	Pressure	8.0	NaN								
6	16-10-2020	diltiazem hydrochloride	Temperature	34.0	35.0								
7	16-10-2020	diltiazem hydrochloride	Pressure	18.0	19.0								
8	16-10-2020	docetaxel injection	Temperature	46.0	47.0								
9	16-10-2020	docetaxel injection	Pressure	23.0	24.0								
10	16-10-2020	ketamine hydrochloride	Temperature	8.0	9.0								
11	16-10-2020	ketamine hydrochloride	Pressure	12.0	12.0								
12	17-10-2020	diltiazem hydrochloride	Temperature	20.0	19.0								
13	17-10-2020	diltiazem hydrochloride	Pressure	3.0	4.0								
14	17-10-2020	docetaxel injection	Temperature	12.0	13.0								
15	17-10-2020	docetaxel injection	Pressure	20.0	22.0								
16	17-10-2020	ketamine hydrochloride	Temperature	13.0	14.0								
17	17-10-2020	ketamine hydrochloride	Pressure	8.0	9.0								

```
2      18.0      NaN     17.0       18      NaN      NaN     23.0        23
3      22.0      NaN     22.0       23      NaN      NaN     27.0        26
4       NaN     27.0      NaN       26     25.0     24.0     23.0        22
5       NaN      7.0      NaN        9     10.0     11.0     10.0         9
6      36.0     36.0     37.0       38     37.0     38.0     39.0        40
7      20.0     21.0     22.0       23     24.0     25.0     25.0        24
8       NaN     48.0     48.0       49     50.0     52.0     55.0        56
9       NaN     25.0     26.0       27     28.0     29.0     28.0        28
10     10.0      NaN     11.0       12     12.0     11.0      NaN        13
11     13.0      NaN     15.0       15     15.0     15.0      NaN        16
12     19.0     18.0     17.0       16     15.0      NaN     13.0        14
13      4.0      4.0      6.0        8      9.0      NaN      9.0        11
14     14.0     15.0     16.0       17     18.0     19.0     20.0        21
15     22.0     22.0     22.0       23     25.0     26.0     27.0        28
16     15.0     16.0     17.0       18     19.0     20.0     21.0        22
17     10.0     11.0     11.0       12     12.0     11.0     12.0        13

    11:30:00  12:30:00
0       20.0        21
1       19.0        20
2       25.0        25
3       29.0        28
4       21.0        20
5        9.0        11
6        NaN        42
7        NaN        27
8       57.0        58
9       29.0        30
10      14.0        15
11      17.0        18
12      11.0        10
13      13.0        14
14      22.0        23
15      29.0        28
16      23.0        24
17      14.0        15
```

[4]:
```
data.shape
```

[4]:
```
(18, 15)
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype
```

```
---  ------      --------------  -----
 0   Date        18 non-null     object
 1   Drug_Name   18 non-null     object
 2   Parameter   18 non-null     object
 3   1:30:00     16 non-null     float64
 4   2:30:00     16 non-null     float64
 5   3:30:00     12 non-null     float64
 6   4:30:00     14 non-null     float64
 7   5:30:00     16 non-null     float64
 8   6:30:00     18 non-null     int64
 9   7:30:00     16 non-null     float64
 10  8:30:00     14 non-null     float64
 11  9:30:00     16 non-null     float64
 12  10:30:00    18 non-null     int64
 13  11:30:00    16 non-null     float64
 14  12:30:00    18 non-null     int64
dtypes: float64(9), int64(3), object(3)
memory usage: 2.2+ KB
```

[6]:
```
pd.melt(data, id_vars=['Date', 'Drug_Name', 'Parameter'])
```

[6]:

	Date	Drug_Name	Parameter	variable	value
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0
..	…	…	…	…	…
211	17-10-2020	diltiazem hydrochloride	Pressure	12:30:00	14.0
212	17-10-2020	docetaxel injection	Temperature	12:30:00	23.0
213	17-10-2020	docetaxel injection	Pressure	12:30:00	28.0
214	17-10-2020	ketamine hydrochloride	Temperature	12:30:00	24.0
215	17-10-2020	ketamine hydrochloride	Pressure	12:30:00	15.0

[216 rows x 5 columns]

[7]:
```
data_melt = pd.melt(data, id_vars=['Date', 'Drug_Name', 'Parameter'],
       var_name='time',
       value_name='reading')
data_melt
```

[7]:

	Date	Drug_Name	Parameter	time	reading
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0

```
..          …                        …           …       …       …
211  17-10-2020  diltiazem hydrochloride     Pressure  12:30:00     14.0
212  17-10-2020      docetaxel injection  Temperature  12:30:00     23.0
213  17-10-2020      docetaxel injection     Pressure  12:30:00     28.0
214  17-10-2020   ketamine hydrochloride  Temperature  12:30:00     24.0
215  17-10-2020   ketamine hydrochloride     Pressure  12:30:00     15.0

[216 rows x 5 columns]
```

[8]:
```
pd.melt?
```

```
Signature:
pd.melt(
    frame: 'DataFrame',
    id_vars=None,
    value_vars=None,
    var_name=None,
    value_name='value',
    col_level=None,
    ignore_index: 'bool' = True,
) -> 'DataFrame'
Docstring:
Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

This function is useful to massage a DataFrame into a format where one
or more columns are identifier variables (`id_vars`), while all other
columns, considered measured variables (`value_vars`), are "unpivoted" to
the row axis, leaving just two non-identifier columns, 'variable' and
'value'.

Parameters
----------
id_vars : tuple, list, or ndarray, optional
    Column(s) to use as identifier variables.
value_vars : tuple, list, or ndarray, optional
    Column(s) to unpivot. If not specified, uses all columns that
    are not set as `id_vars`.
var_name : scalar
    Name to use for the 'variable' column. If None it uses
    ``frame.columns.name`` or 'variable'.
value_name : scalar, default 'value'
    Name to use for the 'value' column.
col_level : int or str, optional
    If columns are a MultiIndex then use this level to melt.
ignore_index : bool, default True
    If True, original index is ignored. If False, the original index is retained.
    Index labels will be repeated as necessary.
```

```
    .. versionadded:: 1.1.0

Returns
-------
DataFrame
    Unpivoted DataFrame.

See Also
--------
DataFrame.melt : Identical method.
pivot_table : Create a spreadsheet-style pivot table as a DataFrame.
DataFrame.pivot : Return reshaped DataFrame organized
    by given index / column values.
DataFrame.explode : Explode a DataFrame from list-like
        columns to long format.

Notes
-----
Reference :ref:`the user guide <reshaping.melt>` for more examples.

Examples
--------
>>> df = pd.DataFrame({'A': {0: 'a', 1: 'b', 2: 'c'},
…                    'B': {0: 1, 1: 3, 2: 5},
…                    'C': {0: 2, 1: 4, 2: 6}})
>>> df
   A  B  C
0  a  1  2
1  b  3  4
2  c  5  6

>>> pd.melt(df, id_vars=['A'], value_vars=['B'])
   A variable  value
0  a        B      1
1  b        B      3
2  c        B      5

>>> pd.melt(df, id_vars=['A'], value_vars=['B', 'C'])
   A variable  value
0  a        B      1
1  b        B      3
2  c        B      5
3  a        C      2
4  b        C      4
5  c        C      6

The names of 'variable' and 'value' columns can be customized:
```

```
>>> pd.melt(df, id_vars=['A'], value_vars=['B'],
…         var_name='myVarname', value_name='myValname')
   A myVarname  myValname
0  a         B          1
1  b         B          3
2  c         B          5

Original index values can be kept around:

>>> pd.melt(df, id_vars=['A'], value_vars=['B', 'C'], ignore_index=False)
   A variable  value
0  a        B      1
1  b        B      3
2  c        B      5
0  a        C      2
1  b        C      4
2  c        C      6

If you have multi-index columns:

>>> df.columns = [list('ABC'), list('DEF')]
>>> df
   A  B  C
   D  E  F
0  a  1  2
1  b  3  4
2  c  5  6

>>> pd.melt(df, col_level=0, id_vars=['A'], value_vars=['B'])
   A variable  value
0  a        B      1
1  b        B      3
2  c        B      5

>>> pd.melt(df, id_vars=[('A', 'D')], value_vars=[('B', 'E')])
  (A, D) variable_0 variable_1  value
0      a          B          E      1
1      b          B          E      3
2      c          B          E      5
File:      /usr/local/lib/python3.9/site-packages/pandas/core/reshape/melt.py
Type:      function
```

[9]:
```
data_melt.shape
```

[9]:
```
(216, 5)
```

[10]:
```
data_melt
```

[10]:

	Date	Drug_Name	Parameter	time	reading
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0
..	…	…	…	…	…
211	17-10-2020	diltiazem hydrochloride	Pressure	12:30:00	14.0
212	17-10-2020	docetaxel injection	Temperature	12:30:00	23.0
213	17-10-2020	docetaxel injection	Pressure	12:30:00	28.0
214	17-10-2020	ketamine hydrochloride	Temperature	12:30:00	24.0
215	17-10-2020	ketamine hydrochloride	Pressure	12:30:00	15.0

[216 rows x 5 columns]

[ ]:

[11]:
```
data_melt.pivot(index=['Date', 'Drug_Name', 'Parameter'],
                columns='time',
                values='reading').reset_index()
```

[11]:

time	Date	Drug_Name	Parameter	10:30:00	11:30:00	\
0	15-10-2020	diltiazem hydrochloride	Pressure	18.0	19.0	
1	15-10-2020	diltiazem hydrochloride	Temperature	20.0	20.0	
2	15-10-2020	docetaxel injection	Pressure	26.0	29.0	
3	15-10-2020	docetaxel injection	Temperature	23.0	25.0	
4	15-10-2020	ketamine hydrochloride	Pressure	9.0	9.0	
5	15-10-2020	ketamine hydrochloride	Temperature	22.0	21.0	
6	16-10-2020	diltiazem hydrochloride	Pressure	24.0	NaN	
7	16-10-2020	diltiazem hydrochloride	Temperature	40.0	NaN	
8	16-10-2020	docetaxel injection	Pressure	28.0	29.0	
9	16-10-2020	docetaxel injection	Temperature	56.0	57.0	
10	16-10-2020	ketamine hydrochloride	Pressure	16.0	17.0	
11	16-10-2020	ketamine hydrochloride	Temperature	13.0	14.0	
12	17-10-2020	diltiazem hydrochloride	Pressure	11.0	13.0	
13	17-10-2020	diltiazem hydrochloride	Temperature	14.0	11.0	
14	17-10-2020	docetaxel injection	Pressure	28.0	29.0	
15	17-10-2020	docetaxel injection	Temperature	21.0	22.0	
16	17-10-2020	ketamine hydrochloride	Pressure	13.0	14.0	
17	17-10-2020	ketamine hydrochloride	Temperature	22.0	23.0	

time	12:30:00	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	\
0	20.0	12.0	13.0	NaN	11.0	13.0	14.0	16.0	
1	21.0	23.0	22.0	NaN	21.0	21.0	22.0	23.0	
2	28.0	NaN	22.0	22.0	NaN	22.0	23.0	NaN	

```
3         25.0      NaN     17.0     18.0      NaN     17.0     18.0      NaN
4         11.0      8.0      NaN      NaN      7.0      NaN      9.0     10.0
5         20.0     24.0      NaN      NaN     27.0      NaN     26.0     25.0
6         27.0     18.0     19.0     20.0     21.0     22.0     23.0     24.0
7         42.0     34.0     35.0     36.0     36.0     37.0     38.0     37.0
8         30.0     23.0     24.0      NaN     25.0     26.0     27.0     28.0
9         58.0     46.0     47.0      NaN     48.0     48.0     49.0     50.0
10        18.0     12.0     12.0     13.0      NaN     15.0     15.0     15.0
11        15.0      8.0      9.0     10.0      NaN     11.0     12.0     12.0
12        14.0      3.0      4.0      4.0      4.0      6.0      8.0      9.0
13        10.0     20.0     19.0     19.0     18.0     17.0     16.0     15.0
14        28.0     20.0     22.0     22.0     22.0     22.0     23.0     25.0
15        23.0     12.0     13.0     14.0     15.0     16.0     17.0     18.0
16        15.0      8.0      9.0     10.0     11.0     11.0     12.0     12.0
17        24.0     13.0     14.0     15.0     16.0     17.0     18.0     19.0

time  8:30:00  9:30:00
0        16.0     24.0
1        21.0     22.0
2         NaN     27.0
3         NaN     23.0
4        11.0     10.0
5        24.0     23.0
6        25.0     25.0
7        38.0     39.0
8        29.0     28.0
9        52.0     55.0
10       15.0      NaN
11       11.0      NaN
12        NaN      9.0
13        NaN     13.0
14       26.0     27.0
15       19.0     20.0
16       11.0     12.0
17       20.0     21.0
```

[ ]:

[12]:
```
data_melt.pivot?
```

```
Signature: data_melt.pivot(index=None, columns=None, values=None) -> 'DataFrame'
Docstring:
Return reshaped DataFrame organized by given index / column values.

Reshape data (produce a "pivot" table) based on column values. Uses
unique values from specified `index` / `columns` to form axes of the
resulting DataFrame. This function does not support data
```

```
aggregation, multiple values will result in a MultiIndex in the
columns. See the :ref:`User Guide <reshaping>` for more on reshaping.

Parameters
----------
index : str or object or a list of str, optional
    Column to use to make new frame's index. If None, uses
    existing index.

    .. versionchanged:: 1.1.0
       Also accept list of index names.

columns : str or object or a list of str
    Column to use to make new frame's columns.

    .. versionchanged:: 1.1.0
       Also accept list of columns names.

values : str, object or a list of the previous, optional
    Column(s) to use for populating new frame's values. If not
    specified, all remaining columns will be used and the result will
    have hierarchically indexed columns.

Returns
-------
DataFrame
    Returns reshaped DataFrame.

Raises
------
ValueError:
    When there are any `index`, `columns` combinations with multiple
    values. `DataFrame.pivot_table` when you need to aggregate.

See Also
--------
DataFrame.pivot_table : Generalization of pivot that can handle
    duplicate values for one index/column pair.
DataFrame.unstack : Pivot based on the index values instead of a
    column.
wide_to_long : Wide panel to long format. Less flexible but more
    user-friendly than melt.

Notes
-----
For finer-tuned control, see hierarchical indexing documentation along
with the related stack/unstack methods.
```

```
Reference :ref:`the user guide <reshaping.pivot>` for more examples.

Examples
--------
>>> df = pd.DataFrame({'foo': ['one', 'one', 'one', 'two', 'two',
…                           'two'],
…                    'bar': ['A', 'B', 'C', 'A', 'B', 'C'],
…                    'baz': [1, 2, 3, 4, 5, 6],
…                    'zoo': ['x', 'y', 'z', 'q', 'w', 't']})
>>> df
    foo   bar  baz  zoo
0   one   A    1    x
1   one   B    2    y
2   one   C    3    z
3   two   A    4    q
4   two   B    5    w
5   two   C    6    t

>>> df.pivot(index='foo', columns='bar', values='baz')
bar  A   B   C
foo
one  1   2   3
two  4   5   6

>>> df.pivot(index='foo', columns='bar')['baz']
bar  A   B   C
foo
one  1   2   3
two  4   5   6

>>> df.pivot(index='foo', columns='bar', values=['baz', 'zoo'])
      baz       zoo
bar   A  B  C   A  B  C
foo
one   1  2  3   x  y  z
two   4  5  6   q  w  t

You could also assign a list of column names or a list of index names.

>>> df = pd.DataFrame({
…        "lev1": [1, 1, 1, 2, 2, 2],
…        "lev2": [1, 1, 2, 1, 1, 2],
…        "lev3": [1, 2, 1, 2, 1, 2],
…        "lev4": [1, 2, 3, 4, 5, 6],
…        "values": [0, 1, 2, 3, 4, 5]})
>>> df
    lev1 lev2 lev3 lev4 values
0   1    1    1    1    0
```

```
1   1    1    2    2    1
2   1    2    1    3    2
3   2    1    2    4    3
4   2    1    1    5    4
5   2    2    2    6    5

>>> df.pivot(index="lev1", columns=["lev2", "lev3"],values="values")
lev2    1         2
lev3    1    2    1    2
lev1
1     0.0  1.0  2.0  NaN
2     4.0  3.0  NaN  5.0

>>> df.pivot(index=["lev1", "lev2"], columns=["lev3"],values="values")
      lev3    1    2
lev1  lev2
   1     1  0.0  1.0
         2  2.0  NaN
   2     1  4.0  3.0
         2  NaN  5.0

A ValueError is raised if there are any duplicates.

>>> df = pd.DataFrame({"foo": ['one', 'one', 'two', 'two'],
…                    "bar": ['A', 'A', 'B', 'C'],
…                    "baz": [1, 2, 3, 4]})
>>> df
   foo bar  baz
0  one   A    1
1  one   A    2
2  two   B    3
3  two   C    4

Notice that the first two rows are the same for our `index`
and `columns` arguments.

>>> df.pivot(index='foo', columns='bar', values='baz')
Traceback (most recent call last):
   …
ValueError: Index contains duplicate entries, cannot reshape
File:      /usr/local/lib/python3.9/site-packages/pandas/core/frame.py
Type:      method
```

[13]:
```
data_tidy = data_melt.pivot(index=['Date', 'Drug_Name', 'time'],
                columns='Parameter',
                values='reading').reset_index()
```

```
data_tidy
```

[13]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
..                ...                      ...       ...       ...
103        17-10-2020   ketamine hydrochloride   5:30:00      11.0
104        17-10-2020   ketamine hydrochloride   6:30:00      12.0
105        17-10-2020   ketamine hydrochloride   7:30:00      12.0
106        17-10-2020   ketamine hydrochloride   8:30:00      11.0
107        17-10-2020   ketamine hydrochloride   9:30:00      12.0

Parameter  Temperature
0                 20.0
1                 20.0
2                 21.0
3                 23.0
4                 22.0
..                 ...
103               17.0
104               18.0
105               19.0
106               20.0
107               21.0

[108 rows x 5 columns]
```

[14]:
```
data.head()
```

[14]:
```
         Date                Drug_Name    Parameter  1:30:00  2:30:00  \
0  15-10-2020  diltiazem hydrochloride  Temperature     23.0     22.0
1  15-10-2020  diltiazem hydrochloride     Pressure     12.0     13.0
2  15-10-2020      docetaxel injection  Temperature      NaN     17.0
3  15-10-2020      docetaxel injection     Pressure      NaN     22.0
4  15-10-2020   ketamine hydrochloride  Temperature     24.0      NaN

   3:30:00  4:30:00  5:30:00  6:30:00  7:30:00  8:30:00  9:30:00  10:30:00  \
0      NaN     21.0     21.0       22     23.0     21.0     22.0        20
1      NaN     11.0     13.0       14     16.0     16.0     24.0        18
2     18.0      NaN     17.0       18      NaN      NaN     23.0        23
3     22.0      NaN     22.0       23      NaN      NaN     27.0        26
4      NaN     27.0      NaN       26     25.0     24.0     23.0        22

   11:30:00  12:30:00
```

```
0      20.0        21
1      19.0        20
2      25.0        25
3      29.0        28
4      21.0        20
```

[15]:
```
data_melt.head()
```

[15]:
```
         Date                Drug_Name    Parameter     time  reading
0  15-10-2020  diltiazem hydrochloride  Temperature  1:30:00     23.0
1  15-10-2020  diltiazem hydrochloride     Pressure  1:30:00     12.0
2  15-10-2020      docetaxel injection  Temperature  1:30:00      NaN
3  15-10-2020      docetaxel injection     Pressure  1:30:00      NaN
4  15-10-2020   ketamine hydrochloride  Temperature  1:30:00     24.0
```

[16]:
```
data_tidy.head()
```

[16]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0

Parameter  Temperature
0                 20.0
1                 20.0
2                 21.0
3                 23.0
4                 22.0
```

0.1.1 Handling Missing Values

[17]:
```
data.head()
```

[17]:
```
         Date                Drug_Name    Parameter  1:30:00  2:30:00  \
0  15-10-2020  diltiazem hydrochloride  Temperature     23.0     22.0
1  15-10-2020  diltiazem hydrochloride     Pressure     12.0     13.0
2  15-10-2020      docetaxel injection  Temperature      NaN     17.0
3  15-10-2020      docetaxel injection     Pressure      NaN     22.0
4  15-10-2020   ketamine hydrochloride  Temperature     24.0      NaN

   3:30:00  4:30:00  5:30:00  6:30:00  7:30:00  8:30:00  9:30:00  10:30:00  \
0      NaN     21.0     21.0       22     23.0     21.0     22.0        20
1      NaN     11.0     13.0       14     16.0     16.0     24.0        18
2     18.0      NaN     17.0       18      NaN      NaN     23.0        23
3     22.0      NaN     22.0       23      NaN      NaN     27.0        26
```

```
4      NaN     27.0      NaN       26     25.0     24.0     23.0        22

   11:30:00  12:30:00
0      20.0        21
1      19.0        20
2      25.0        25
3      29.0        28
4      21.0        20
```

[18]:
```
type(None)
```

[18]:
```
NoneType
```

[19]:
```
type(np.nan)
```

[19]:
```
float
```

[20]:
```
pd.Series([1, np.nan, 3])
```

[20]:
```
0    1.0
1    NaN
2    3.0
dtype: float64
```

[21]:
```
pd.Series([1, np.nan, 3, None])
```

[21]:
```
0    1.0
1    NaN
2    3.0
3    NaN
dtype: float64
```

[22]:
```
pd.Series(['1', 'np.nan', '3', None])
```

[22]:
```
0         1
1    np.nan
2         3
3      None
dtype: object
```

[23]:
```
pd.Series(['1', 'np.nan', '3', np.nan])
```

[23]:
```
0         1
1    np.nan
2         3
3       NaN
dtype: object
```

How to know number of missing values/data in rows/colums?

[24]:
```
data.isnull().sum()
```

[24]:
```
Date          0
Drug_Name     0
Parameter     0
1:30:00       2
2:30:00       2
3:30:00       6
4:30:00       4
5:30:00       2
6:30:00       0
7:30:00       2
8:30:00       4
9:30:00       2
10:30:00      0
11:30:00      2
12:30:00      0
dtype: int64
```

[25]:
```
data.isnull().sum(axis=1)
```

[25]:
```
0     1
1     1
2     4
3     4
4     3
5     3
6     1
7     1
8     1
9     1
10    2
11    2
12    1
13    1
14    0
15    0
16    0
17    0
dtype: int64
```

[26]:
```
data.dropna()
```

[26]:

	Date	Drug_Name	Parameter	1:30:00	2:30:00	\
14	17-10-2020	docetaxel injection	Temperature	12.0	13.0	

```
15  17-10-2020     docetaxel injection     Pressure     20.0     22.0
16  17-10-2020  ketamine hydrochloride  Temperature     13.0     14.0
17  17-10-2020  ketamine hydrochloride     Pressure      8.0      9.0

    3:30:00  4:30:00  5:30:00  6:30:00  7:30:00  8:30:00  9:30:00  10:30:00  \
14     14.0     15.0     16.0       17     18.0     19.0     20.0        21
15     22.0     22.0     22.0       23     25.0     26.0     27.0        28
16     15.0     16.0     17.0       18     19.0     20.0     21.0        22
17     10.0     11.0     11.0       12     12.0     11.0     12.0        13

    11:30:00  12:30:00
14      22.0        23
15      29.0        28
16      23.0        24
17      14.0        15
```

[27]:
```
data.fillna(0)
```

[27]:
```
           Date                Drug_Name    Parameter  1:30:00  2:30:00  \
0   15-10-2020  diltiazem hydrochloride  Temperature     23.0     22.0
1   15-10-2020  diltiazem hydrochloride     Pressure     12.0     13.0
2   15-10-2020      docetaxel injection  Temperature      0.0     17.0
3   15-10-2020      docetaxel injection     Pressure      0.0     22.0
4   15-10-2020   ketamine hydrochloride  Temperature     24.0      0.0
5   15-10-2020   ketamine hydrochloride     Pressure      8.0      0.0
6   16-10-2020  diltiazem hydrochloride  Temperature     34.0     35.0
7   16-10-2020  diltiazem hydrochloride     Pressure     18.0     19.0
8   16-10-2020      docetaxel injection  Temperature     46.0     47.0
9   16-10-2020      docetaxel injection     Pressure     23.0     24.0
10  16-10-2020   ketamine hydrochloride  Temperature      8.0      9.0
11  16-10-2020   ketamine hydrochloride     Pressure     12.0     12.0
12  17-10-2020  diltiazem hydrochloride  Temperature     20.0     19.0
13  17-10-2020  diltiazem hydrochloride     Pressure      3.0      4.0
14  17-10-2020      docetaxel injection  Temperature     12.0     13.0
15  17-10-2020      docetaxel injection     Pressure     20.0     22.0
16  17-10-2020   ketamine hydrochloride  Temperature     13.0     14.0
17  17-10-2020   ketamine hydrochloride     Pressure      8.0      9.0

    3:30:00  4:30:00  5:30:00  6:30:00  7:30:00  8:30:00  9:30:00  10:30:00  \
0       0.0     21.0     21.0       22     23.0     21.0     22.0        20
1       0.0     11.0     13.0       14     16.0     16.0     24.0        18
2      18.0      0.0     17.0       18      0.0      0.0     23.0        23
3      22.0      0.0     22.0       23      0.0      0.0     27.0        26
4       0.0     27.0      0.0       26     25.0     24.0     23.0        22
5       0.0      7.0      0.0        9     10.0     11.0     10.0         9
6      36.0     36.0     37.0       38     37.0     38.0     39.0        40
7      20.0     21.0     22.0       23     24.0     25.0     25.0        24
```

```
8       0.0     48.0     48.0       49     50.0     52.0     55.0        56
9       0.0     25.0     26.0       27     28.0     29.0     28.0        28
10     10.0      0.0     11.0       12     12.0     11.0      0.0        13
11     13.0      0.0     15.0       15     15.0     15.0      0.0        16
12     19.0     18.0     17.0       16     15.0      0.0     13.0        14
13      4.0      4.0      6.0        8      9.0      0.0      9.0        11
14     14.0     15.0     16.0       17     18.0     19.0     20.0        21
15     22.0     22.0     22.0       23     25.0     26.0     27.0        28
16     15.0     16.0     17.0       18     19.0     20.0     21.0        22
17     10.0     11.0     11.0       12     12.0     11.0     12.0        13

    11:30:00  12:30:00
0       20.0        21
1       19.0        20
2       25.0        25
3       29.0        28
4       21.0        20
5        9.0        11
6        0.0        42
7        0.0        27
8       57.0        58
9       29.0        30
10      14.0        15
11      17.0        18
12      11.0        10
13      13.0        14
14      22.0        23
15      29.0        28
16      23.0        24
17      14.0        15
```

[28]:
```
data['2:30:00'].fillna(data['2:30:00'].mean())
```

[28]:
```
0     22.0000
1     13.0000
2     17.0000
3     22.0000
4     18.8125
5     18.8125
6     35.0000
7     19.0000
8     47.0000
9     24.0000
10     9.0000
11    12.0000
12    19.0000
13     4.0000
```

```
14    13.0000
15    22.0000
16    14.0000
17     9.0000
Name: 2:30:00, dtype: float64
```

[29]:
```
data_tidy
```

[29]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
..                ...                      ...       ...       ...
103        17-10-2020   ketamine hydrochloride   5:30:00      11.0
104        17-10-2020   ketamine hydrochloride   6:30:00      12.0
105        17-10-2020   ketamine hydrochloride   7:30:00      12.0
106        17-10-2020   ketamine hydrochloride   8:30:00      11.0
107        17-10-2020   ketamine hydrochloride   9:30:00      12.0

Parameter  Temperature
0                20.0
1                20.0
2                21.0
3                23.0
4                22.0
..                ...
103              17.0
104              18.0
105              19.0
106              20.0
107              21.0

[108 rows x 5 columns]
```

[30]:
```
def temp_mean(x):
    x['temp_avg'] = x['Temperature'].mean()
    return x

data_tidy = data_tidy.groupby('Drug_Name').apply(temp_mean)
data_tidy
```

[30]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
```

```
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
..                ...                      ...       ...       ...
103        17-10-2020   ketamine hydrochloride   5:30:00      11.0
104        17-10-2020   ketamine hydrochloride   6:30:00      12.0
105        17-10-2020   ketamine hydrochloride   7:30:00      12.0
106        17-10-2020   ketamine hydrochloride   8:30:00      11.0
107        17-10-2020   ketamine hydrochloride   9:30:00      12.0

Parameter  Temperature   temp_avg
0                 20.0  24.848485
1                 20.0  24.848485
2                 21.0  24.848485
3                 23.0  24.848485
4                 22.0  24.848485
..                 ...        ...
103               17.0  17.709677
104               18.0  17.709677
105               19.0  17.709677
106               20.0  17.709677
107               21.0  17.709677

[108 rows x 6 columns]
```

[31]:
```
data_tidy['Temperature'].fillna(data_tidy['temp_avg'], inplace=True)
data_tidy[:20]
```

[31]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
5          15-10-2020  diltiazem hydrochloride   3:30:00       NaN
6          15-10-2020  diltiazem hydrochloride   4:30:00      11.0
7          15-10-2020  diltiazem hydrochloride   5:30:00      13.0
8          15-10-2020  diltiazem hydrochloride   6:30:00      14.0
9          15-10-2020  diltiazem hydrochloride   7:30:00      16.0
10         15-10-2020  diltiazem hydrochloride   8:30:00      16.0
11         15-10-2020  diltiazem hydrochloride   9:30:00      24.0
12         15-10-2020      docetaxel injection  10:30:00      26.0
13         15-10-2020      docetaxel injection  11:30:00      29.0
14         15-10-2020      docetaxel injection  12:30:00      28.0
15         15-10-2020      docetaxel injection   1:30:00       NaN
16         15-10-2020      docetaxel injection   2:30:00      22.0
17         15-10-2020      docetaxel injection   3:30:00      22.0
18         15-10-2020      docetaxel injection   4:30:00       NaN
```

```
19          15-10-2020      docetaxel injection   5:30:00      22.0

Parameter  Temperature   temp_avg
0            20.000000  24.848485
1            20.000000  24.848485
2            21.000000  24.848485
3            23.000000  24.848485
4            22.000000  24.848485
5            24.848485  24.848485
6            21.000000  24.848485
7            21.000000  24.848485
8            22.000000  24.848485
9            23.000000  24.848485
10           21.000000  24.848485
11           22.000000  24.848485
12           23.000000  30.387097
13           25.000000  30.387097
14           25.000000  30.387097
15           30.387097  30.387097
16           17.000000  30.387097
17           18.000000  30.387097
18           30.387097  30.387097
19           17.000000  30.387097
```

[32]:
```
def pressure_mean(x):
    x['pressure_avg'] = x['Pressure'].mean()
    return x

data_tidy = data_tidy.groupby('Drug_Name').apply(pressure_mean)
data_tidy
```

[32]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
..                ...                      ...       ...       ...
103        17-10-2020   ketamine hydrochloride   5:30:00      11.0
104        17-10-2020   ketamine hydrochloride   6:30:00      12.0
105        17-10-2020   ketamine hydrochloride   7:30:00      12.0
106        17-10-2020   ketamine hydrochloride   8:30:00      11.0
107        17-10-2020   ketamine hydrochloride   9:30:00      12.0

Parameter  Temperature   temp_avg  pressure_avg
0                 20.0  24.848485     15.424242
1                 20.0  24.848485     15.424242
```

```
2                 21.0  24.848485     15.424242
3                 23.0  24.848485     15.424242
4                 22.0  24.848485     15.424242
..                 …        …            …
103               17.0  17.709677     11.935484
104               18.0  17.709677     11.935484
105               19.0  17.709677     11.935484
106               20.0  17.709677     11.935484
107               21.0  17.709677     11.935484

[108 rows x 7 columns]
```

[33]:
```
pd.cut?
```

```
Signature:
pd.cut(
    x,
    bins,
    right: 'bool' = True,
    labels=None,
    retbins: 'bool' = False,
    precision: 'int' = 3,
    include_lowest: 'bool' = False,
    duplicates: 'str' = 'raise',
    ordered: 'bool' = True,
)
Docstring:
Bin values into discrete intervals.

Use `cut` when you need to segment and sort data values into bins. This
function is also useful for going from a continuous variable to a
categorical variable. For example, `cut` could convert ages to groups of
age ranges. Supports binning into an equal number of bins, or a
pre-specified array of bins.

Parameters
----------
x : array-like
    The input array to be binned. Must be 1-dimensional.
bins : int, sequence of scalars, or IntervalIndex
    The criteria to bin by.

    * int : Defines the number of equal-width bins in the range of `x`. The
      range of `x` is extended by .1% on each side to include the minimum
      and maximum values of `x`.
    * sequence of scalars : Defines the bin edges allowing for non-uniform
      width. No extension of the range of `x` is done.
```

```
    * IntervalIndex : Defines the exact bins to be used. Note that
      IntervalIndex for `bins` must be non-overlapping.

right : bool, default True
    Indicates whether `bins` includes the rightmost edge or not. If
    ``right == True`` (the default), then the `bins` ``[1, 2, 3, 4]``
    indicate (1,2], (2,3], (3,4]. This argument is ignored when
    `bins` is an IntervalIndex.
labels : array or False, default None
    Specifies the labels for the returned bins. Must be the same length as
    the resulting bins. If False, returns only integer indicators of the
    bins. This affects the type of the output container (see below).
    This argument is ignored when `bins` is an IntervalIndex. If True,
    raises an error. When `ordered=False`, labels must be provided.
retbins : bool, default False
    Whether to return the bins or not. Useful when bins is provided
    as a scalar.
precision : int, default 3
    The precision at which to store and display the bins labels.
include_lowest : bool, default False
    Whether the first interval should be left-inclusive or not.
duplicates : {default 'raise', 'drop'}, optional
    If bin edges are not unique, raise ValueError or drop non-uniques.
ordered : bool, default True
    Whether the labels are ordered or not. Applies to returned types
    Categorical and Series (with Categorical dtype). If True,
    the resulting categorical will be ordered. If False, the resulting
    categorical will be unordered (labels must be provided).

    .. versionadded:: 1.1.0

Returns
-------
out : Categorical, Series, or ndarray
    An array-like object representing the respective bin for each value
    of `x`. The type depends on the value of `labels`.

    * None (default) : returns a Series for Series `x` or a
      Categorical for all other inputs. The values stored within
      are Interval dtype.

    * sequence of scalars : returns a Series for Series `x` or a
      Categorical for all other inputs. The values stored within
      are whatever the type in the sequence is.

    * False : returns an ndarray of integers.

bins : numpy.ndarray or IntervalIndex.
```

```
    The computed or specified bins. Only returned when `retbins=True`.
    For scalar or sequence `bins`, this is an ndarray with the computed
    bins. If set `duplicates=drop`, `bins` will drop non-unique bin. For
    an IntervalIndex `bins`, this is equal to `bins`.

See Also
--------
qcut : Discretize variable into equal-sized buckets based on rank
    or based on sample quantiles.
Categorical : Array type for storing data that come from a
    fixed set of values.
Series : One-dimensional array with axis labels (including time series).
IntervalIndex : Immutable Index implementing an ordered, sliceable set.

Notes
-----
Any NA values will be NA in the result. Out of bounds values will be NA in
the resulting Series or Categorical object.

Reference :ref:`the user guide <reshaping.tile.cut>` for more examples.

Examples
--------
Discretize into three equal-sized bins.

>>> pd.cut(np.array([1, 7, 5, 4, 6, 3]), 3)
… # doctest: +ELLIPSIS
[(0.994, 3.0], (5.0, 7.0], (3.0, 5.0], (3.0, 5.0], (5.0, 7.0], …
Categories (3, interval[float64, right]): [(0.994, 3.0] < (3.0, 5.0] …

>>> pd.cut(np.array([1, 7, 5, 4, 6, 3]), 3, retbins=True)
… # doctest: +ELLIPSIS
([(0.994, 3.0], (5.0, 7.0], (3.0, 5.0], (3.0, 5.0], (5.0, 7.0], …
Categories (3, interval[float64, right]): [(0.994, 3.0] < (3.0, 5.0] …
array([0.994, 3.   , 5.   , 7.   ]))

Discovers the same bins, but assign them specific labels. Notice that
the returned Categorical's categories are `labels` and is ordered.

>>> pd.cut(np.array([1, 7, 5, 4, 6, 3]),
…        3, labels=["bad", "medium", "good"])
['bad', 'good', 'medium', 'medium', 'good', 'bad']
Categories (3, object): ['bad' < 'medium' < 'good']

``ordered=False`` will result in unordered categories when labels are passed.
This parameter can be used to allow non-unique labels:

>>> pd.cut(np.array([1, 7, 5, 4, 6, 3]), 3,
```

```
…        labels=["B", "A", "B"], ordered=False)
['B', 'B', 'A', 'A', 'B', 'B']
Categories (2, object): ['A', 'B']

``labels=False`` implies you just want the bins back.

>>> pd.cut([0, 1, 1, 2], bins=4, labels=False)
array([0, 1, 1, 3])

Passing a Series as an input returns a Series with categorical dtype:

>>> s = pd.Series(np.array([2, 4, 6, 8, 10]),
…               index=['a', 'b', 'c', 'd', 'e'])
>>> pd.cut(s, 3)
… # doctest: +ELLIPSIS
a    (1.992, 4.667]
b    (1.992, 4.667]
c    (4.667, 7.333]
d     (7.333, 10.0]
e     (7.333, 10.0]
dtype: category
Categories (3, interval[float64, right]): [(1.992, 4.667] < (4.667, …

Passing a Series as an input returns a Series with mapping value.
It is used to map numerically to intervals based on bins.

>>> s = pd.Series(np.array([2, 4, 6, 8, 10]),
…               index=['a', 'b', 'c', 'd', 'e'])
>>> pd.cut(s, [0, 2, 4, 6, 8, 10], labels=False, retbins=True, right=False)
… # doctest: +ELLIPSIS
(a    1.0
 b    2.0
 c    3.0
 d    4.0
 e    NaN
 dtype: float64,
 array([ 0,  2,  4,  6,  8, 10]))

Use `drop` optional when bins is not unique

>>> pd.cut(s, [0, 2, 4, 6, 10, 10], labels=False, retbins=True,
…        right=False, duplicates='drop')
… # doctest: +ELLIPSIS
(a    1.0
 b    2.0
 c    3.0
 d    3.0
 e    NaN
```

```
 dtype: float64,
 array([ 0,  2,  4,  6, 10]))

Passing an IntervalIndex for `bins` results in those categories exactly.
Notice that values not covered by the IntervalIndex are set to NaN. 0
is to the left of the first bin (which is closed on the right), and 1.5
falls between two bins.

>>> bins = pd.IntervalIndex.from_tuples([(0, 1), (2, 3), (4, 5)])
>>> pd.cut([0, 0.5, 1.5, 2.5, 4.5], bins)
[NaN, (0.0, 1.0], NaN, (2.0, 3.0], (4.0, 5.0]]
Categories (3, interval[int64, right]): [(0, 1] < (2, 3] < (4, 5]]
File:      /usr/local/lib/python3.9/site-packages/pandas/core/reshape/tile.py
Type:      function
```

[34]:
```
temp_points = [5, 20, 35, 50, 60]
temp_labels = ['low', 'medium', 'high', 'very high']
data_tidy['temp_cat'] = pd.cut(data_tidy['Temperature'], bins=temp_points,
       labels=temp_labels)
data_tidy
```

[34]:

Parameter	Date	Drug_Name	time	Pressure	\
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	
..	…	…	…	…	
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	
105	17-10-2020	ketamine hydrochloride	7:30:00	12.0	
106	17-10-2020	ketamine hydrochloride	8:30:00	11.0	
107	17-10-2020	ketamine hydrochloride	9:30:00	12.0	

Parameter	Temperature	temp_avg	pressure_avg	temp_cat
0	20.0	24.848485	15.424242	low
1	20.0	24.848485	15.424242	low
2	21.0	24.848485	15.424242	medium
3	23.0	24.848485	15.424242	medium
4	22.0	24.848485	15.424242	medium
..	…	…	…	…
103	17.0	17.709677	11.935484	low
104	18.0	17.709677	11.935484	low
105	19.0	17.709677	11.935484	low
106	20.0	17.709677	11.935484	low
107	21.0	17.709677	11.935484	medium

```
[108 rows x 8 columns]
```

[35]:
```
data_tidy['temp_cat'].value_counts()
```

[35]:
```
low          50
medium       38
high         15
very high     5
Name: temp_cat, dtype: int64
```

[37]:
```
data_tidy.loc[data_tidy['Drug_Name']=='hydrochloride']
```

[37]:
```
Empty DataFrame
Columns: [Date, Drug_Name, time, Pressure, Temperature, temp_avg, pressure_avg,
temp_cat]
Index: []
```

[39]:
```
data_tidy.loc[data_tidy['Drug_Name'].str.contains('hydrochloride')]
```

[39]:
```
Parameter        Date                Drug_Name      time  Pressure  \
0          15-10-2020  diltiazem hydrochloride  10:30:00      18.0
1          15-10-2020  diltiazem hydrochloride  11:30:00      19.0
2          15-10-2020  diltiazem hydrochloride  12:30:00      20.0
3          15-10-2020  diltiazem hydrochloride   1:30:00      12.0
4          15-10-2020  diltiazem hydrochloride   2:30:00      13.0
..                ...                      ...       ...       ...
103        17-10-2020   ketamine hydrochloride   5:30:00      11.0
104        17-10-2020   ketamine hydrochloride   6:30:00      12.0
105        17-10-2020   ketamine hydrochloride   7:30:00      12.0
106        17-10-2020   ketamine hydrochloride   8:30:00      11.0
107        17-10-2020   ketamine hydrochloride   9:30:00      12.0

Parameter  Temperature   temp_avg  pressure_avg temp_cat
0                 20.0  24.848485     15.424242      low
1                 20.0  24.848485     15.424242      low
2                 21.0  24.848485     15.424242   medium
3                 23.0  24.848485     15.424242   medium
4                 22.0  24.848485     15.424242   medium
..                 ...        ...           ...      ...
103               17.0  17.709677     11.935484      low
104               18.0  17.709677     11.935484      low
105               19.0  17.709677     11.935484      low
106               20.0  17.709677     11.935484      low
107               21.0  17.709677     11.935484   medium

[72 rows x 8 columns]
```

[43]:
```
def get_year(x):
    return x[2]

data_tidy['Date'].str.split('-').apply(get_year)
```

[43]:
```
0      2020
1      2020
2      2020
3      2020
4      2020
       …
103    2020
104    2020
105    2020
106    2020
107    2020
Name: Date, Length: 108, dtype: object
```

[45]:
```
data_tidy['timestamp'] = data_tidy['Date'] + ' '+data_tidy['time']
data_tidy.drop(['Date', 'time'], axis=1, inplace=True)
data_tidy
```

[45]:
```
Parameter                Drug_Name  Pressure  Temperature   temp_avg  \
0          diltiazem hydrochloride      18.0         20.0  24.848485
1          diltiazem hydrochloride      19.0         20.0  24.848485
2          diltiazem hydrochloride      20.0         21.0  24.848485
3          diltiazem hydrochloride      12.0         23.0  24.848485
4          diltiazem hydrochloride      13.0         22.0  24.848485
..                             …       …          …        …
103         ketamine hydrochloride      11.0         17.0  17.709677
104         ketamine hydrochloride      12.0         18.0  17.709677
105         ketamine hydrochloride      12.0         19.0  17.709677
106         ketamine hydrochloride      11.0         20.0  17.709677
107         ketamine hydrochloride      12.0         21.0  17.709677

Parameter  pressure_avg temp_cat            timestamp
0             15.424242      low  15-10-2020 10:30:00
1             15.424242      low  15-10-2020 11:30:00
2             15.424242   medium  15-10-2020 12:30:00
3             15.424242   medium   15-10-2020 1:30:00
4             15.424242   medium   15-10-2020 2:30:00
..                  …      …                   …
103           11.935484      low   17-10-2020 5:30:00
104           11.935484      low   17-10-2020 6:30:00
105           11.935484      low   17-10-2020 7:30:00
106           11.935484      low   17-10-2020 8:30:00
107           11.935484   medium   17-10-2020 9:30:00
```

```
[108 rows x 7 columns]
```

[46]:
```
data_tidy['timestamp']
```

[46]:
```
0      15-10-2020 10:30:00
1      15-10-2020 11:30:00
2      15-10-2020 12:30:00
3       15-10-2020 1:30:00
4       15-10-2020 2:30:00
              …
103     17-10-2020 5:30:00
104     17-10-2020 6:30:00
105     17-10-2020 7:30:00
106     17-10-2020 8:30:00
107     17-10-2020 9:30:00
Name: timestamp, Length: 108, dtype: object
```

[47]:
```
data_tidy['timestamp'] = pd.to_datetime(data_tidy['timestamp'])
data_tidy['timestamp']
```

[47]:
```
0     2020-10-15 10:30:00
1     2020-10-15 11:30:00
2     2020-10-15 12:30:00
3     2020-10-15 01:30:00
4     2020-10-15 02:30:00
              …
103   2020-10-17 05:30:00
104   2020-10-17 06:30:00
105   2020-10-17 07:30:00
106   2020-10-17 08:30:00
107   2020-10-17 09:30:00
Name: timestamp, Length: 108, dtype: datetime64[ns]
```

[48]:
```
ts = data_tidy['timestamp'][0]
ts
```

[48]:
```
Timestamp('2020-10-15 10:30:00')
```

[49]:
```
ts.year
```

[49]:
```
2020
```

[50]:
```
ts.month
```

[50]:
```
10
```

[51]:
```
ts.day
```

[51]:
```
15
```

[52]:
```
ts.hour
```

[52]:
```
10
```

[53]:
```
ts.minute
```

[53]:
```
30
```

[54]:
```
ts.month_name()
```

[54]:
```
'October'
```

[56]:
```
data_tidy['timesCtamp'].dt.month_name()
```

[56]:
```
0      October
1      October
2      October
3      October
4      October
        …
103    October
104    October
105    October
106    October
107    October
Name: timestamp, Length: 108, dtype: object
```

[58]:
```
ts = data_tidy['timestamp'][0]
ts
```

[58]:
```
Timestamp('2020-10-15 10:30:00')
```

[59]:
```
ts.strftime('%Y')
```

[59]:
```
'2020'
```

[63]:
```
ts.strftime('%m-%d-%Y')
```

[63]:
```
'10-15-2020'
```

[64]:
```
data_tidy.to_csv('pfizer_tidy.csv', sep=',')
```

[65]:
```
data_tidy.to_csv?
```

```
Signature:
data_tidy.to_csv(
    path_or_buf: 'FilePath | WriteBuffer[bytes] | WriteBuffer[str] | None' =␣
 ↪None,
    sep: 'str' = ',',
    na_rep: 'str' = '',
    float_format: 'str | None' = None,
    columns: 'Sequence[Hashable] | None' = None,
    header: 'bool_t | list[str]' = True,
    index: 'bool_t' = True,
    index_label: 'IndexLabel | None' = None,
    mode: 'str' = 'w',
    encoding: 'str | None' = None,
    compression: 'CompressionOptions' = 'infer',
    quoting: 'int | None' = None,
    quotechar: 'str' = '"',
    line_terminator: 'str | None' = None,
    chunksize: 'int | None' = None,
    date_format: 'str | None' = None,
    doublequote: 'bool_t' = True,
    escapechar: 'str | None' = None,
    decimal: 'str' = '.',
    errors: 'str' = 'strict',
    storage_options: 'StorageOptions' = None,
) -> 'str | None'
Docstring:
Write object to a comma-separated values (csv) file.

Parameters
----------
path_or_buf : str, path object, file-like object, or None, default None
    String, path object (implementing os.PathLike[str]), or file-like
    object implementing a write() function. If None, the result is
    returned as a string. If a non-binary file object is passed, it should
    be opened with `newline=''`, disabling universal newlines. If a binary
    file object is passed, `mode` might need to contain a `'b'`.

    .. versionchanged:: 1.2.0

       Support for binary file objects was introduced.

sep : str, default ','
    String of length 1. Field delimiter for the output file.
na_rep : str, default ''
    Missing data representation.
float_format : str, default None
    Format string for floating point numbers.
```

```
columns : sequence, optional
    Columns to write.
header : bool or list of str, default True
    Write out the column names. If a list of strings is given it is
    assumed to be aliases for the column names.
index : bool, default True
    Write row names (index).
index_label : str or sequence, or False, default None
    Column label for index column(s) if desired. If None is given, and
    `header` and `index` are True, then the index names are used. A
    sequence should be given if the object uses MultiIndex. If
    False do not print fields for index names. Use index_label=False
    for easier importing in R.
mode : str
    Python write mode, default 'w'.
encoding : str, optional
    A string representing the encoding to use in the output file,
    defaults to 'utf-8'. `encoding` is not supported if `path_or_buf`
    is a non-binary file object.
compression : str or dict, default 'infer'
    For on-the-fly compression of the output data. If 'infer' and '%s'
    path-like, then detect compression from the following extensions: '.gz',
    '.bz2', '.zip', '.xz', or '.zst' (otherwise no compression). Set to
    ``None`` for no compression. Can also be a dict with key ``'method'`` set
    to one of {``'zip'``, ``'gzip'``, ``'bz2'``, ``'zstd'``} and other
    key-value pairs are forwarded to ``zipfile.ZipFile``, ``gzip.GzipFile``,
    ``bz2.BZ2File``, or ``zstandard.ZstdDecompressor``, respectively. As an
    example, the following could be passed for faster compression and to create
    a reproducible gzip archive:
    ``compression={'method': 'gzip', 'compresslevel': 1, 'mtime': 1}``.

    .. versionchanged:: 1.0.0

       May now be a dict with key 'method' as compression mode
       and other entries as additional compression options if
       compression mode is 'zip'.

    .. versionchanged:: 1.1.0

       Passing compression options as keys in dict is
       supported for compression modes 'gzip', 'bz2', 'zstd', and 'zip'.

    .. versionchanged:: 1.2.0

        Compression is supported for binary file objects.

    .. versionchanged:: 1.2.0
```

```
        Previous versions forwarded dict entries for 'gzip' to
        `gzip.open` instead of `gzip.GzipFile` which prevented
        setting `mtime`.

quoting : optional constant from csv module
    Defaults to csv.QUOTE_MINIMAL. If you have set a `float_format`
    then floats are converted to strings and thus csv.QUOTE_NONNUMERIC
    will treat them as non-numeric.
quotechar : str, default '\"'
    String of length 1. Character used to quote fields.
line_terminator : str, optional
    The newline character or character sequence to use in the output
    file. Defaults to `os.linesep`, which depends on the OS in which
    this method is called ('\\n' for linux, '\\r\\n' for Windows, i.e.).
chunksize : int or None
    Rows to write at a time.
date_format : str, default None
    Format string for datetime objects.
doublequote : bool, default True
    Control quoting of `quotechar` inside a field.
escapechar : str, default None
    String of length 1. Character used to escape `sep` and `quotechar`
    when appropriate.
decimal : str, default '.'
    Character recognized as decimal separator. E.g. use ',' for
    European data.
errors : str, default 'strict'
    Specifies how encoding and decoding errors are to be handled.
    See the errors argument for :func:`open` for a full list
    of options.

    .. versionadded:: 1.1.0

storage_options : dict, optional
    Extra options that make sense for a particular storage connection, e.g.
    host, port, username, password, etc. For HTTP(S) URLs the key-value pairs
    are forwarded to ``urllib`` as header options. For other URLs (e.g.
    starting with "s3://", and "gcs://") the key-value pairs are forwarded to
    ``fsspec``. Please see ``fsspec`` and ``urllib`` for more details.

    .. versionadded:: 1.2.0

Returns
-------
None or str
    If path_or_buf is None, returns the resulting csv format as a
    string. Otherwise returns None.
```

```
See Also
--------
read_csv : Load a CSV file into a DataFrame.
to_excel : Write DataFrame to an Excel file.

Examples
--------
>>> df = pd.DataFrame({'name': ['Raphael', 'Donatello'],
…                    'mask': ['red', 'purple'],
…                    'weapon': ['sai', 'bo staff']})
>>> df.to_csv(index=False)
'name,mask,weapon\nRaphael,red,sai\nDonatello,purple,bo staff\n'

Create 'out.zip' containing 'out.csv'

>>> compression_opts = dict(method='zip',
…                         archive_name='out.csv')  # doctest: +SKIP
>>> df.to_csv('out.zip', index=False,
…           compression=compression_opts)  # doctest: +SKIP

To write a csv file to a new folder or nested folder you will first
need to create it using either Pathlib or os:

>>> from pathlib import Path  # doctest: +SKIP
>>> filepath = Path('folder/subfolder/out.csv')  # doctest: +SKIP
>>> filepath.parent.mkdir(parents=True, exist_ok=True)  # doctest: +SKIP
>>> df.to_csv(filepath)  # doctest: +SKIP

>>> import os  # doctest: +SKIP
>>> os.makedirs('folder/subfolder', exist_ok=True)  # doctest: +SKIP
>>> df.to_csv('folder/subfolder/out.csv')  # doctest: +SKIP
File:      /usr/local/lib/python3.9/site-packages/pandas/core/generic.py
Type:      method
```

[ ]: