pandas.pivot_table

pandas.pivot_table(*data*, *values=None*, *index=None*, *columns=None*, *aggfunc='mean'*, *fill_value=None*, *margins=False*, *dropna=True*, *margins_name='All'*, *observed=<no_default>*, *sort=True*) [source]

Create a spreadsheet-style pivot table as a DataFrame.

The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.

Parameters:

data : *DataFrame*

values : *list-like or scalar, optional*

Column or columns to aggregate.

index : *column, Grouper, array, or list of the previous*

Keys to group by on the pivot table index. If a list is passed, it can contain any of the other types (except list). If an array is passed, it must be the same length as the data and will be used in the same manner as column values.

columns : *column, Grouper, array, or list of the previous*

Keys to group by on the pivot table column. If a list is passed, it can contain any of the other types (except list). If an array is passed, it must be the same length as the data and will be used in the same manner as column values.

aggfunc : *function, list of functions, dict, default "mean"*

If a list of functions is passed, the resulting pivot table will have hierarchical columns whose top level are the function names (inferred from the function objects themselves). If a dict is passed, the key is column to aggregate and the value is function or list of functions. If `margin=True`, aggfunc will be used to calculate the partial aggregates.

Skip to main content

Value to replace missing values with (in the resulting pivot table, after aggregation).

margins : *bool, default False*

If `margins=True`, special `All` columns and rows will be added with partial group aggregates across the categories on the rows and columns.

dropna : *bool, default True*

Do not include columns whose entries are all NaN. If True, rows with a NaN value in any column will be omitted before computing margins.

margins_name : *str, default 'All'*

Name of the row / column that will contain the totals when margins is True.

observed : *bool, default False*

This only applies if any of the groupers are Categoricals. If True: only show observed values for categorical groupers. If False: show all values for categorical groupers.

> ***Deprecated since version 2.2.0:*** The default value of `False` is deprecated and will change to `True` in a future version of pandas.

sort : *bool, default True*

Specifies if the result should be sorted.

> ***Added in version 1.3.0.***

Returns:

DataFrame

An Excel style pivot table.

See also

`DataFrame.pivot`

Pivot without aggregation that can handle non-numeric data.

`DataFrame.melt`

Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

`wide_to_long`

Wide panel to long format. Less flexible but more user-friendly than melt.

Notes

Reference the user guide for more examples.

Examples

```
>>> df = pd.DataFrame({"A": ["foo", "foo", "foo", "foo", "foo",
...                          "bar", "bar", "bar", "bar"],
...                    "B": ["one", "one", "one", "two", "two",
...                          "one", "one", "two", "two"],
...                    "C": ["small", "large", "large", "small",
...                          "small", "large", "small", "small",
...                          "large"],
...                    "D": [1, 2, 2, 3, 3, 4, 5, 6, 7],
...                    "E": [2, 4, 5, 5, 6, 6, 8, 9, 9]})
>>> df
     A    B      C  D  E
0  foo  one  small  1  2
1  foo  one  large  2  4
2  foo  one  large  2  5
3  foo  two  small  3  5
4  foo  two  small  3  6
5  bar  one  large  4  6
6  bar  one  small  5  8
7  bar  two  small  6  9
8  bar  two  large  7  9
```

This first example aggregates values by taking the sum.

```
>>> table = pd.pivot_table(df, values='D', index=['A', 'B'],
...                        columns=['C'], aggfunc="sum")
>>> table
C        large  small
A   B
bar one    4.0    5.0
    two    7.0    6.0
foo one    4.0    1.0
    two    NaN    6.0
```

We can also fill missing values using the *fill_value* parameter.

```
>>> table = pd.pivot_table(df, values='D', index=['A', 'B'],
...                        columns=['C'], aggfunc="sum", fill_value=0)
>>> table
C        large  small
```

Skip to main content

```
    two      7      6
foo one      4      1
    two      0      6
```

The next example aggregates by taking the mean across multiple columns.

```
>>> table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
...                        aggfunc={'D': "mean", 'E': "mean"})
>>> table
                D         E
A   C
bar large  5.500000  7.500000
    small  5.500000  8.500000
foo large  2.000000  4.500000
    small  2.333333  4.333333
```

We can also calculate multiple types of aggregations for any given value column.

```
>>> table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
...                        aggfunc={'D': "mean",
...                                 'E': ["min", "max", "mean"]})
>>> table
                  D   E
               mean max      mean  min
A   C
bar large  5.500000   9  7.500000    6
    small  5.500000   9  8.500000    8
foo large  2.000000   5  4.500000    4
    small  2.333333   6  4.333333    2
```

Previous
pandas.pivot

Next
pandas.crosstab