

[Documentation](#)

search

Search

- [rocket launch](#)

[Get started](#)

- [Installation](#)
add
- [Fundamentals](#)
add
- [First steps](#)
add
- [code](#)

[Develop](#)

- [Concepts](#)
add
- [API reference](#)
remove

- PAGE ELEMENTS

- [Write and magic](#)

add

- [Text elements](#)

add

- [Data elements](#)

remove

- [st.dataframe](#)

- [st.data editor](#)

- [st.column config](#)

add

- [st.table](#)

- [st.metric](#)

- [st.json](#)

- [Chart elements](#)

add

- [Input widgets](#)

add

- [Media elements](#)

add

- [Layouts and containers](#)

add

- [Chat elements](#)

add

- [Status elements](#)

add

- [Third-party components](#)*open in new*

- APPLICATION LOGIC

- [Navigation and pages](#)

add

- [Execution flow](#)

add

- [Caching and state](#)

add

- [Connections and secrets](#)

add

- [Custom components](#)

add

- [Utilities](#)
add
- [Configuration](#)
add
- TOOLS

-
- [App testing](#)
add
 - [Command line](#)
add

- [Tutorials](#)
add
- [Quick reference](#)
add

- [web_asset](#)

[Deploy](#)

- [Concepts](#)
add
- [Streamlit Community Cloud](#)
add
- [Snowflake](#)
- [Other platforms](#)
add

- [school](#)

[Knowledge base](#)

- [FAQ](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Develop/](#)
- [API reference/](#)
- [Data elements/](#)
- [st.dataframe](#)

star

Tip

Learn more in our [Dataframes](#) guide and check out our tutorial, [Get dataframe row-selections from users](#).

st.dataframe



Streamlit Version 

Display a dataframe as an interactive table.

This command works with a wide variety of collection-like and dataframe-like object types.

```
st.dataframe(data=None, width=None, height=None, *, use_container_width=False, hide_index=None, column_order=None, column_config=None, key=None, on_select="ignore", selection_mode="multi-row")
```

Parameters

The data to display.

Dataframe-like objects include dataframe and series objects from popular libraries like Dask, Modin, Numpy, pandas, Polars, PyArrow, Snowpark, Xarray, and more. You can use database cursors and clients that comply with the [Python Database API Specification v2.0 \(PEP 249\)](#). Additionally, you can use anything that supports the [Python dataframe interchange protocol](#).

For example, you can use the following:

- `pandas.DataFrame`, `pandas.Series`, `pandas.Index`, `pandas.Styler`, and `pandas.Array`
- `polars.DataFrame`, `polars.LazyFrame`, and `polars.Series`
- `snowflake.snowpark.dataframe.DataFrame`, `snowflake.snowpark.table.Table`

data (dataframe-like, collection-like, or None)

If a data type is not recognized, Streamlit will convert the object to a `pandas.DataFrame` or `pyarrow.Table` using a `.to_pandas()` or `.to_arrow()` method, respectively, if available.

If data is a `pandas.Styler`, it will be used to style its underlying `pandas.DataFrame`. Streamlit supports custom cell values and colors. It does not support some of the more exotic styling options, like bar charts, hovering, and captions. For these styling options, use column configuration instead. Text and number formatting from `column_config` always takes precedence over text and number formatting from `pandas.Styler`.

Collection-like objects include all Python-native collection types, such as `dict`, `list`, and `set`.

If data is `None`, Streamlit renders an empty table.

width (int or None)

Desired width of the dataframe expressed in pixels. If `width` is `None` (default), Streamlit sets the dataframe width to fit its contents up to the width of the parent container. If `width` is greater than the width of the parent container, Streamlit sets the dataframe width to match the width of the parent container.

height (int or None)

Desired height of the dataframe expressed in pixels. If `height` is `None` (default), Streamlit sets the height to show at most ten rows. Vertical scrolling within the dataframe element is enabled when the height does not accomodate all rows.

use_container_width (bool)

Whether to override `width` with the width of the parent container. If `use_container_width` is `False` (default), Streamlit sets the dataframe's width according to `width`. If `use_container_width` is `True`, Streamlit sets the width of the dataframe to match the width of the parent container.

hide_index (bool or None)

Whether to hide the index column(s). If `hide_index` is `None` (default), the visibility of index columns is automatically determined based on the data.

column_order (Iterable of str or None)

The ordered list of columns to display. If `column_order` is `None` (default), Streamlit displays all columns in the order inherited from the underlying data structure. If `column_order` is a list, the indicated columns will display in the order they appear within the list. Columns may be omitted or repeated within the list.

For example, `column_order= ("col2", "col1")` will display "col2" first, followed by "col1", and will hide all other non-index columns.

Returns

(element or dict)

If `on_select` is "ignore" (default), this command returns an internal placeholder for the dataframe element that can be used with the `.add_rows()` method. Otherwise, this command returns a dictionary-like object that supports both key and attribute notation. The attributes are described by the `DataframeState` dictionary schema.

```
st.dataframe(data=None, width=None, height=None, *, use_container_width=False, hide_index=None, column_order=None,
             column_config=None, key=None, on_select="ignore", selection_mode="multi-row")
```

Configuration to customize how columns display. If `column_config` is `None` (default), columns are styled based on the underlying data type of each column.

Column configuration can modify column names, visibility, type, width, or format, among other things. `column_config` must be a dictionary where each key is a column name and the associated value is one of the following:

`column_config` (dict or `None`)

- `None`: Streamlit hides the column.
- A string: Streamlit changes the display label of the column to the given string.
- A column type within `st.column_config`: Streamlit applies the defined configuration to the column. For example, use `st.column_config.NumberColumn("Dollar values", format="$ %d")` to change the displayed name of the column to "Dollar values" and add a "\$" prefix in each cell. For more info on the available column types and config options, see [Column configuration](#).

To configure the index column(s), use `_index` as the column name.

An optional string to use for giving this element a stable identity. If `key` is `None` (default), this element's identity will be determined based on the values of the other parameters.

`key` (str)

Additionally, if selections are activated and `key` is provided, Streamlit will register the key in Session State to store the selection state. The selection state is read-only.

How the dataframe should respond to user selection events. This controls whether or not the dataframe behaves like an input widget. `on_select` can be one of the following:

`on_select` ("ignore" or "rerun" or callable)

- "ignore" (default): Streamlit will not react to any selection events in the dataframe. The dataframe will not behave like an input widget.
- "rerun": Streamlit will rerun the app when the user selects rows or columns in the dataframe. In this case, `st.dataframe` will return the selection data as a dictionary.
- A callable: Streamlit will rerun the app and execute the `callable` as a callback function before the rest of the app. In this case, `st.dataframe` will return the selection data as a dictionary.

The types of selections Streamlit should allow when selections are enabled with `on_select`. This can be one of the following:

`selection_mode` ("single-row", "multi-row", "single-column", "multi-column", or Iterable of these)

- "multi-row" (default): Multiple rows can be selected at a time.
- "single-row": Only one row can be selected at a time.
- "multi-column": Multiple columns can be selected at a time.
- "single-column": Only one column can be selected at a time.
- An Iterable of the above options: The table will allow selection based on the modes specified.

When column selections are enabled, column sorting is disabled.

Returns

If `on_select` is "ignore" (default), this command returns an internal placeholder for the dataframe element that can be used with the `.add_rows()` method. Otherwise, this command returns a dictionary-like object that supports both key and attribute notation. The attributes are described by the `DataframeState` dictionary schema.

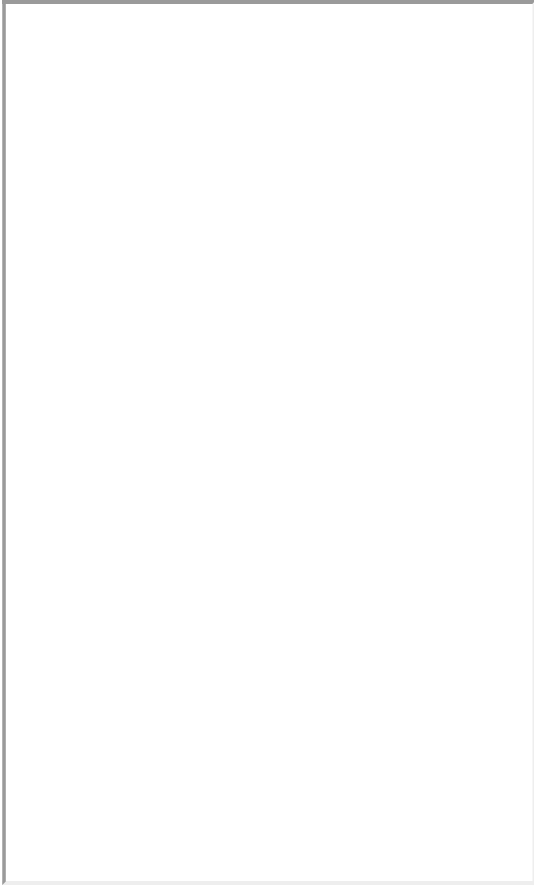
(element or dict)


Examples

Example 1: Display a dataframe

```
import streamlit as st
import pandas as pd
import numpy as np
```

```
df = pd.DataFrame(np.random.randn(50, 20), columns=["col %d" % i for i in range(20)])  
  
st.dataframe(df) # Same as st.write(df)
```



[Built with Streamlit](#) 
[Fullscreen open in new](#)

Example 2: Use Pandas Styler

You can also pass a Pandas Styler object to change the style of the rendered DataFrame:

```
import streamlit as st  
import pandas as pd  
import numpy as np  
  
df = pd.DataFrame(np.random.randn(10, 20), columns=["col %d" % i for i in range(20)])  
  
st.dataframe(df.style.highlight_max(axis=0))
```

[Built with Streamlit !\[\]\(b39c89771cd6fb2128a8c57aa7d97f9a_img.jpg\)](#)
[Fullscreen open in new](#)

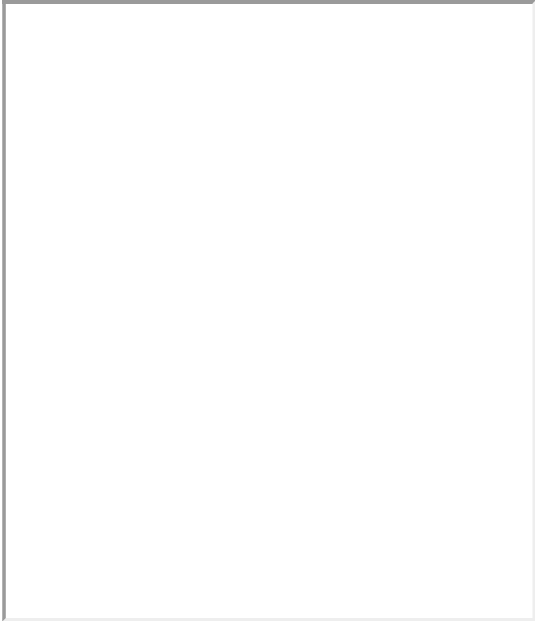
Example 3: Use column configuration

You can customize a dataframe via `column_config`, `hide_index`, or `column_order`.

```
import random
import pandas as pd
import streamlit as st

df = pd.DataFrame(
    {
        "name": ["Roadmap", "Extras", "Issues"],
        "url": ["https://roadmap.streamlit.app", "https://extras.streamlit.app", "https://issues.streamlit.app"],
        "stars": [random.randint(0, 1000) for _ in range(3)],
        "views_history": [[random.randint(0, 5000) for _ in range(30)] for _ in range(3)],
    }
)

st.dataframe(
    df,
    column_config={
        "name": "App name",
        "stars": st.column_config.NumberColumn(
            "Github Stars",
            help="Number of stars on GitHub",
            format="%d 🌟",
        ),
        "url": st.column_config.LinkColumn("App URL"),
        "views_history": st.column_config.LineChartColumn(
            "Views (past 30 days)", y_min=0, y_max=5000
        ),
    },
    hide_index=True,
)
```



[Built with Streamlit !\[\]\(35e4f762fc1cfea5610d92e2d225d5b4_img.jpg\)](#)
[Fullscreen open in new](#)

Example 4: Customize your index

You can use column configuration to format your index.

```
import streamlit as st
import pandas as pd
from datetime import date

df = pd.DataFrame(
    {
        "Date": [date(2024, 1, 1), date(2024, 2, 1), date(2024, 3, 1)],
        "Total": [13429, 23564, 23452],
    }
)
df.set_index("Date", inplace=True)

config = {
    "_index": st.column_config.DateColumn("Month", format="MMM YYYY"),
    "Total": st.column_config.NumberColumn("Total ($)"),
}

st.dataframe(df, column_config=config)
```



[Built with Streamlit !\[\]\(8d0f0e0fe25b320c33272c52aec1fbca_img.jpg\)](#)
[Fullscreen open in new](#)

Dataframe selections



DataframeState



The schema for the dataframe event state.

The event state is stored in a dictionary-like object that supports both key and attribute notation. Event states cannot be programmatically changed or set through Session State.

Only selection events are supported at this time.

Attributes

selection The state of the `on_select` event. This attribute returns a dictionary-like object that supports both key and attribute notation.
(dict) The attributes are described by the `DataframeSelectionState` dictionary schema.

DataframeSelectionState



The schema for the dataframe selection state.

The selection state is stored in a dictionary-like object that supports both key and attribute notation. Selection states cannot be programmatically changed or set through Session State.

Warning

If a user sorts a dataframe, row selections will be reset. If your users need to sort and filter the dataframe to make selections, direct them to use the search function in the dataframe toolbar instead.

Attributes

rows
(list[int]) The selected rows, identified by their integer position. The integer positions match the original dataframe, even if the user sorts the dataframe in their browser. For a `pandas.DataFrame`, you can retrieve data from its integer position using methods like `.iloc[]` or `.iat[]`.

columns
(list[str]) The selected columns, identified by their names.

Example

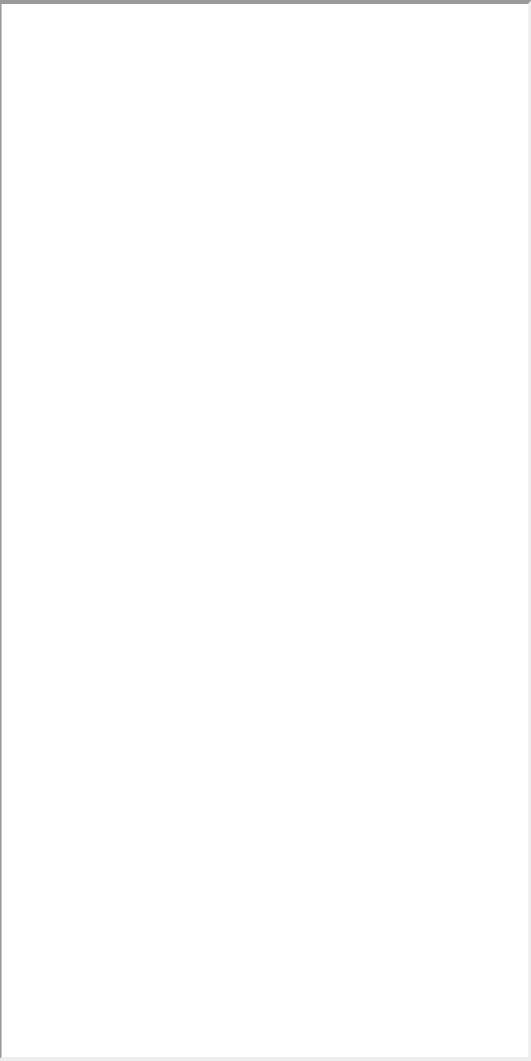
The following example has multi-row and multi-column selections enabled. Try selecting some rows. To select multiple columns, hold `ctrl` while selecting columns. Hold `shift` to select a range of columns.


```
import streamlit as st
import pandas as pd
import numpy as np

if "df" not in st.session_state:
    st.session_state.df = pd.DataFrame(
        np.random.randn(12, 5), columns=["a", "b", "c", "d", "e"]
    )

event = st.dataframe(
    st.session_state.df,
    key="data",
    on_select="rerun",
    selection_mode=["multi-row", "multi-column"],
)

event.selection
```

[Built with Streamlit](#) 
[Fullscreen open in new](#)

element.add_rows



Streamlit Version

Concatenate a dataframe to the bottom of the current one.

Function signature [\[source\]](#)

element.add_rows(data=None, **kwargs)

Parameters

data (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, pyspark.sql.DataFrame, snowflake.snowpark.dataframe.DataFrame, Iterable, dict, or None)	Table to concat. Optional.
**kwargs (pandas.DataFrame, numpy.ndarray, Iterable, dict, or None)	The named dataset to concat. Optional. You can only pass in 1 dataset (including the one in the data parameter).

Example

```
import streamlit as st
import pandas as pd
import numpy as np
```

```
df1 = pd.DataFrame(
    np.random.randn(50, 20), columns=("col %d" % i for i in range(20))
)

my_table = st.table(df1)

df2 = pd.DataFrame(
    np.random.randn(50, 20), columns=("col %d" % i for i in range(20))
)

my_table.add_rows(df2)
# Now the table shown in the Streamlit app contains the data for
# df1 followed by the data for df2.
```

You can do the same thing with plots. For example, if you want to add more data to a line chart:

```
# Assuming df1 and df2 from the example above still exist...
my_chart = st.line_chart(df1)
my_chart.add_rows(df2)
# Now the chart shown in the Streamlit app contains the data for
# df1 followed by the data for df2.
```

And for plots whose datasets are named, you can pass the data with a keyword argument where the key is the name:

```
my_chart = st.vega_lite_chart(
    {
        "mark": "line",
        "encoding": {"x": "a", "y": "b"},
        "datasets": {
            "some_fancy_name": df1, # <-- named dataset
        },
        "data": {"name": "some_fancy_name"},
    }
)
my_chart.add_rows(some_fancy_name=df2) # <-- name used as keyword
```

Interactivity

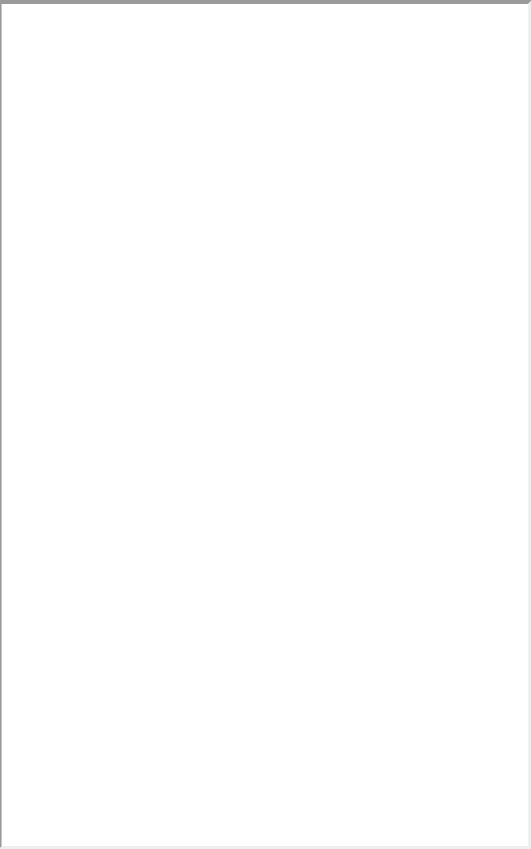



Dataframes displayed with `st.dataframe` are interactive. End users can sort, resize, search, and copy data to their clipboard. For an overview of features, read our [Dataframes](#) guide.

Configuring columns



You can configure the display and editing behavior of columns in `st.dataframe` and `st.data_editor` via the [Column configuration API](#). We have developed the API to let you add images, charts, and clickable URLs in dataframe and data editor columns. Additionally, you can make individual columns editable, set columns as categorical and specify which options they can take, hide the index of the dataframe, and much more.



[Built with Streamlit](#) 
[Fullscreen open in new](#)

[←Previous: Data elements](#)[Next: st.data_editor→](#)
forum

Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

[Home](#)[Contact Us](#)[Community](#)



© 2025 Snowflake Inc. [Cookie policy](#)

[forum](#) [Ask AI](#)