datascience.tables.Table.where

`Table.`**`where`**(*column_or_label*, *value_or_predicate=None*, *other=None*) [source]

Return a new `Table` containing rows where `value_or_predicate` returns True for values in `column_or_label`.

Args:

`column_or_label`: A column of the `Table` either as a label (`str`) or an index (`int`). Can also be an array of booleans; only the rows where the array value is `True` are kept.

`value_or_predicate`: If a function, it is applied to every value in `column_or_label`. Only the rows where `value_or_predicate` returns True are kept. If a single value, only the rows where the values in `column_or_label` are equal to `value_or_predicate` are kept.

`other`: Optional additional column label for `value_or_predicate` to make pairwise comparisons. See the examples below for usage. When `other` is supplied, `value_or_predicate` must be a callable function.

Returns:

If `value_or_predicate` is a function, returns a new `Table` containing only the rows where `value_or_predicate(val)` is True for the `val``s in ``column_or_label`.

If `value_or_predicate` is a value, returns a new `Table` containing only the rows where the values in `column_or_label` are equal to `value_or_predicate`.

If `column_or_label` is an array of booleans, returns a new `Table` containing only the rows where `column_or_label` is `True`.

```
>>> marbles = Table().with_columns(
...     "Color", make_array("Red", "Green", "Blue",
...                         "Red", "Green", "Green"),
...     "Shape", make_array("Round", "Rectangular", "Rectangular",
...                         "Round", "Rectangular", "Round"),
...     "Amount", make_array(4, 6, 12, 7, 9, 2),
...     "Price", make_array(1.30, 1.20, 2.00, 1.75, 0, 3.00))
```

```
>>> marbles
Color | Shape       | Amount | Price
Red   | Round       | 4      | 1.3
Green | Rectangular | 6      | 1.2
Blue  | Rectangular | 12     | 2
Red   | Round       | 7      | 1.75
Green | Rectangular | 9      | 0
Green | Round       | 2      | 3
```

Use a value to select matching rows

```
>>> marbles.where("Price", 1.3)
Color | Shape | Amount | Price
Red   | Round | 4      | 1.3
```

In general, a higher order predicate function such as the functions in `datascience.predicates.are` can be used.

```
>>> from datascience.predicates import are
>>> # equivalent to previous example
>>> marbles.where("Price", are.equal_to(1.3))
Color | Shape | Amount | Price
Red   | Round | 4      | 1.3
```

```
>>> marbles.where("Price", are.above(1.5))
Color | Shape       | Amount | Price
Blue  | Rectangular | 12     | 2
Red   | Round       | 7      | 1.75
Green | Round       | 2      | 3
```

Use the optional argument `other` to apply predicates to compare columns.

```
>>> marbles.where("Price", are.above, "Amount")
Color | Shape | Amount | Price
Green | Round | 2      | 3
```

```
>>> marbles.where("Price", are.equal_to, "Amount") # empty table
Color | Shape | Amount | Price
```