hrl7ijdbr

November 26, 2025

```
[ ]: #### !!!! Exploratory Data Analysis with Pandas !!!! ####
```

```
[ ]: import kagglehub
```

```
[ ]: path = kagglehub.dataset_download('kashnitsky/mlcourse')
     print(path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/kashnitsky/mlco
urse?dataset_version_number=17…

100%|      | 51.1M/51.1M [00:00<00:00, 167MB/s]

Extracting files…


/root/.cache/kagglehub/datasets/kashnitsky/mlcourse/versions/17
```

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

     pd.set_option("display.precision", 2)
```

```
[ ]: ####
     help(pd.set_option)
     ####
```

```
Help on CallableDynamicDoc in module pandas._config.config:

<pandas._config.config.CallableDynamicDoc object>
    set_option(pat, value)

    Sets the value of the specified option.

    Available options:

    - compute.[use_bottleneck, use_numba, use_numexpr]
    - display.[chop_threshold, colheader_justify, date_dayfirst, date_yearfirst,
      encoding, expand_frame_repr, float_format]
    - display.html.[border, table_schema, use_mathjax]
    - display.[large_repr, max_categories, max_columns, max_colwidth,
max_dir_items,
      max_info_columns, max_info_rows, max_rows, max_seq_items, memory_usage,
      min_rows, multi_sparse, notebook_repr_html, pprint_nest_depth, precision,
      show_dimensions]
    - display.unicode.[ambiguous_as_wide, east_asian_width]
    - display.[width]
    - future.[infer_string, no_silent_downcasting]
    - io.excel.ods.[reader, writer]
    - io.excel.xls.[reader]
    - io.excel.xlsb.[reader]
    - io.excel.xlsm.[reader, writer]
    - io.excel.xlsx.[reader, writer]
    - io.hdf.[default_format, dropna_table]
    - io.parquet.[engine]
    - io.sql.[engine]
    - mode.[chained_assignment, copy_on_write, data_manager, sim_interactive,
      string_storage, use_inf_as_na]
    - plotting.[backend]
    - plotting.matplotlib.[register_converters]
    - styler.format.[decimal, escape, formatter, na_rep, precision, thousands]
    - styler.html.[mathjax]
    - styler.latex.[environment, hrules, multicol_align, multirow_align]
    - styler.render.[encoding, max_columns, max_elements, max_rows, repr]
    - styler.sparse.[columns, index]

    Parameters
    ----------
    pat : str
        Regexp which should match a single option.
        Note: partial matches are supported for convenience, but unless you use
the
        full option name (e.g. x.y.z.option_name), your code may break in future
        versions if new options with similar names are introduced.
    value : object
        New value of option.

    Returns
    -------
    None

    Raises
    ------
    OptionError if no such option exists

    Notes
    -----
    Please reference the :ref:`User Guide <options>` for more information.
    The available options with its descriptions:

    compute.use_bottleneck : bool
        Use the bottleneck library to accelerate if it is installed,
        the default is True
        Valid values: False,True
        [default: True] [currently: True]
    compute.use_numba : bool
        Use the numba engine option for select operations if it is installed,
        the default is False
        Valid values: False,True
        [default: False] [currently: False]
    compute.use_numexpr : bool
        Use the numexpr library to accelerate computation if it is installed,
        the default is True
        Valid values: False,True
        [default: True] [currently: True]
    display.chop_threshold : float or None
        if set to a float value, all float values smaller than the given
threshold
        will be displayed as exactly 0 by repr and friends.
        [default: None] [currently: None]
    display.colheader_justify : 'left'/'right'
        Controls the justification of column headers. used by
DataFrameFormatter.
        [default: right] [currently: right]
    display.date_dayfirst : boolean
        When True, prints and parses dates with the day first, eg 20/01/2005
        [default: False] [currently: False]
    display.date_yearfirst : boolean
        When True, prints and parses dates with the year first, eg 2005/01/20
        [default: False] [currently: False]
    display.encoding : str/unicode
        Defaults to the detected encoding of the console.
        Specifies the encoding to be used for strings returned by to_string,
        these are generally strings meant to be displayed on the console.
        [default: UTF-8] [currently: UTF-8]
    display.expand_frame_repr : boolean
        Whether to print out the full DataFrame repr for wide DataFrames across
        multiple lines, `max_columns` is still respected, but the output will
        wrap-around across multiple "pages" if its width exceeds
`display.width`.
        [default: True] [currently: True]
    display.float_format : callable
        The callable should accept a floating point number and return
        a string with the desired format of the number. This is used
        in some places like SeriesFormatter.
        See formats.format.EngFormatter for an example.
        [default: None] [currently: None]
    display.html.border : int
        A ``border=value`` attribute is inserted in the ``<table>`` tag
        for the DataFrame HTML repr.
        [default: 1] [currently: 1]
    display.html.table_schema : boolean
        Whether to publish a Table Schema representation for frontends
        that support it.
        (default: False)
        [default: False] [currently: False]
    display.html.use_mathjax : boolean
        When True, Jupyter notebook will process table contents using MathJax,
        rendering mathematical expressions enclosed by the dollar symbol.
        (default: True)
        [default: True] [currently: True]
    display.large_repr : 'truncate'/'info'
        For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can
        show a truncated table, or switch to the view from
        df.info() (the behaviour in earlier versions of pandas).
        [default: truncate] [currently: truncate]
    display.max_categories : int
        This sets the maximum number of categories pandas should output when
        printing out a `Categorical` or a Series of dtype "category".
        [default: 8] [currently: 8]
    display.max_columns : int
        If max_cols is exceeded, switch to truncate view. Depending on
        `large_repr`, objects are either centrally truncated or printed as
        a summary view. 'None' value means unlimited.

        In case python/IPython is running in a terminal and `large_repr`
        equals 'truncate' this can be set to 0 or None and pandas will auto-
detect
        the width of the terminal and print a truncated object which fits
        the screen width. The IPython notebook, IPython qtconsole, or IDLE
        do not run in a terminal and hence it is not possible to do
        correct auto-detection and defaults to 20.
        [default: 20] [currently: 20]
    display.max_colwidth : int or None
        The maximum width in characters of a column in the repr of
        a pandas data structure. When the column overflows, a "…"
        placeholder is embedded in the output. A 'None' value means unlimited.
        [default: 50] [currently: 50]
    display.max_dir_items : int
        The number of items that will be added to `dir(…)`. 'None' value means
        unlimited. Because dir is cached, changing this option will not
immediately
        affect already existing dataframes until a column is deleted or added.
        This is for instance used to suggest columns from a dataframe to tab
        completion.
        [default: 100] [currently: 100]
    display.max_info_columns : int
        max_info_columns is used in DataFrame.info method to decide if
        per column information will be printed.
        [default: 100] [currently: 100]
    display.max_info_rows : int
        df.info() will usually show null-counts for each column.
        For large frames this can be quite slow. max_info_rows and max_info_cols
        limit this null check only to frames with smaller dimensions than
        specified.
        [default: 1690785] [currently: 1690785]
    display.max_rows : int
        If max_rows is exceeded, switch to truncate view. Depending on
        `large_repr`, objects are either centrally truncated or printed as
        a summary view. 'None' value means unlimited.

        In case python/IPython is running in a terminal and `large_repr`
        equals 'truncate' this can be set to 0 and pandas will auto-detect
        the height of the terminal and print a truncated object which fits
        the screen height. The IPython notebook, IPython qtconsole, or
        IDLE do not run in a terminal and hence it is not possible to do
        correct auto-detection.
        [default: 60] [currently: 60]
    display.max_seq_items : int or None
        When pretty-printing a long sequence, no more then `max_seq_items`
        will be printed. If items are omitted, they will be denoted by the
        addition of "…" to the resulting string.

        If set to None, the number of items to be printed is unlimited.
        [default: 100] [currently: 100]
    display.memory_usage : bool, string or None
        This specifies if the memory usage of a DataFrame should be displayed
when
        df.info() is called. Valid values True,False,'deep'
        [default: True] [currently: True]
    display.min_rows : int
        The numbers of rows to show in a truncated view (when `max_rows` is
        exceeded). Ignored when `max_rows` is set to None or 0. When set to
        None, follows the value of `max_rows`.
        [default: 10] [currently: 10]
    display.multi_sparse : boolean
        "sparsify" MultiIndex display (don't display repeated
        elements in outer levels within groups)
        [default: True] [currently: True]
    display.notebook_repr_html : boolean
```

```
        When True, IPython notebook will use html representation for
        pandas objects (if it is available).
        [default: True] [currently: True]
    display.pprint_nest_depth : int
        Controls the number of nested levels to process when pretty-printing
        [default: 3] [currently: 3]
    display.precision : int
        Floating point output precision in terms of number of places after the
        decimal, for regular formatting as well as scientific notation. Similar
        to ``precision`` in :meth:`numpy.set_printoptions`.
        [default: 6] [currently: 2]
    display.show_dimensions : boolean or 'truncate'
        Whether to print out dimensions at the end of DataFrame repr.
        If 'truncate' is specified, only print out the dimensions if the
        frame is truncated (e.g. not display all rows and/or columns)
        [default: truncate] [currently: truncate]
    display.unicode.ambiguous_as_wide : boolean
        Whether to use the Unicode East Asian Width to calculate the display
text
        width.
        Enabling this may affect to the performance (default: False)
        [default: False] [currently: False]
    display.unicode.east_asian_width : boolean
        Whether to use the Unicode East Asian Width to calculate the display
text
        width.
        Enabling this may affect to the performance (default: False)
        [default: False] [currently: False]
    display.width : int
        Width of the display in characters. In case python/IPython is running in
        a terminal this can be set to None and pandas will correctly auto-detect
        the width.
        Note that the IPython notebook, IPython qtconsole, or IDLE do not run in
a
        terminal and hence it is not possible to correctly detect the width.
        [default: 80] [currently: 80]
    future.infer_string Whether to infer sequence of str objects as pyarrow
string dtype, which will be the default in pandas 3.0 (at which point this
option will be deprecated).
        [default: False] [currently: False]
    future.no_silent_downcasting Whether to opt-in to the future behavior which
will *not* silently downcast results from Series and DataFrame `where`, `mask`,
and `clip` methods. Silent downcasting will be removed in pandas 3.0 (at which
point this option will be deprecated).
        [default: False] [currently: False]
    io.excel.ods.reader : string
        The default Excel reader engine for 'ods' files. Available options:
        auto, odf, calamine.
        [default: auto] [currently: auto]
    io.excel.ods.writer : string
        The default Excel writer engine for 'ods' files. Available options:
        auto, odf.
        [default: auto] [currently: auto]
    io.excel.xls.reader : string
        The default Excel reader engine for 'xls' files. Available options:
        auto, xlrd, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsb.reader : string
        The default Excel reader engine for 'xlsb' files. Available options:
        auto, pyxlsb, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsm.reader : string
        The default Excel reader engine for 'xlsm' files. Available options:
        auto, xlrd, openpyxl, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsm.writer : string
        The default Excel writer engine for 'xlsm' files. Available options:
        auto, openpyxl.
        [default: auto] [currently: auto]
    io.excel.xlsx.reader : string
        The default Excel reader engine for 'xlsx' files. Available options:
        auto, xlrd, openpyxl, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsx.writer : string
        The default Excel writer engine for 'xlsx' files. Available options:
        auto, openpyxl, xlsxwriter.
        [default: auto] [currently: auto]
    io.hdf.default_format : format
        default format writing format, if None, then
        put will default to 'fixed' and append will default to 'table'
        [default: None] [currently: None]
    io.hdf.dropna_table : boolean
        drop ALL nan rows when appending to a table
        [default: False] [currently: False]
    io.parquet.engine : string
        The default parquet reader/writer engine. Available options:
        'auto', 'pyarrow', 'fastparquet', the default is 'auto'
        [default: auto] [currently: auto]
    io.sql.engine : string
        The default sql reader/writer engine. Available options:
        'auto', 'sqlalchemy', the default is 'auto'
        [default: auto] [currently: auto]
    mode.chained_assignment : string
        Raise an exception, warn, or no action if trying to use chained
assignment,
        The default is warn
        [default: warn] [currently: warn]
    mode.copy_on_write : bool
        Use new copy-view behaviour using Copy-on-Write. Defaults to False,
        unless overridden by the 'PANDAS_COPY_ON_WRITE' environment variable
        (if set to "1" for True, needs to be set before pandas is imported).
        [default: False] [currently: False]
    mode.data_manager : string
        Internal data manager type; can be "block" or "array". Defaults to
"block",
        unless overridden by the 'PANDAS_DATA_MANAGER' environment variable
(needs
        to be set before pandas is imported).
        [default: block] [currently: block]
        (Deprecated, use `` instead.)
    mode.sim_interactive : boolean
        Whether to simulate interactive mode for purposes of testing
        [default: False] [currently: False]
    mode.string_storage : string
        The default storage for StringDtype. This option is ignored if
        ``future.infer_string`` is set to True.
        [default: python] [currently: python]
    mode.use_inf_as_na : boolean
        True means treat None, NaN, INF, -INF as NA (old way),
        False means None and NaN are null, but INF, -INF are not NA
        (new way).

        This option is deprecated in pandas 2.1.0 and will be removed in 3.0.
        [default: False] [currently: False]
        (Deprecated, use `` instead.)
    plotting.backend : str
        The plotting backend to use. The default value is "matplotlib", the
        backend provided with pandas. Other backends can be specified by
        providing the name of the module that implements the backend.
        [default: matplotlib] [currently: matplotlib]
    plotting.matplotlib.register_converters : bool or 'auto'.
        Whether to register converters with matplotlib's units registry for
        dates, times, datetimes, and Periods. Toggling to False will remove
        the converters, restoring any converters that pandas overwrote.
        [default: auto] [currently: auto]
    styler.format.decimal : str
        The character representation for the decimal separator for floats and
complex.
        [default: .] [currently: .]
    styler.format.escape : str, optional
        Whether to escape certain characters according to the given context;
html or latex.
        [default: None] [currently: None]
    styler.format.formatter : str, callable, dict, optional
```

```
        A formatter object to be used as default within ``Styler.format``.
        [default: None] [currently: None]
    styler.format.na_rep : str, optional
        The string representation for values identified as missing.
        [default: None] [currently: None]
    styler.format.precision : int
        The precision for floats and complex numbers.
        [default: 6] [currently: 6]
    styler.format.thousands : str, optional
        The character representation for thousands separator for floats, int and
complex.
        [default: None] [currently: None]
    styler.html.mathjax : bool
        If False will render special CSS classes to table attributes that
indicate Mathjax
        will not be used in Jupyter Notebook.
        [default: True] [currently: True]
    styler.latex.environment : str
        The environment to replace ``\begin{table}``. If "longtable" is used
results
        in a specific longtable environment format.
        [default: None] [currently: None]
    styler.latex.hrules : bool
        Whether to add horizontal rules on top and bottom and below the headers.
        [default: False] [currently: False]
    styler.latex.multicol_align : {"r", "c", "l", "naive-l", "naive-r"}
        The specifier for horizontal alignment of sparsified LaTeX multicolumns.
Pipe
        decorators can also be added to non-naive values to draw vertical
        rules, e.g. "\|r" will draw a rule on the left side of right aligned
merged cells.
        [default: r] [currently: r]
    styler.latex.multirow_align : {"c", "t", "b"}
        The specifier for vertical alignment of sparsified LaTeX multirows.
        [default: c] [currently: c]
    styler.render.encoding : str
        The encoding used for output HTML and LaTeX files.
        [default: utf-8] [currently: utf-8]
    styler.render.max_columns : int, optional
        The maximum number of columns that will be rendered. May still be
reduced to
        satisfy ``max_elements``, which takes precedence.
        [default: None] [currently: None]
    styler.render.max_elements : int
        The maximum number of data-cell (<td>) elements that will be rendered
before
        trimming will occur over columns, rows or both if needed.
        [default: 262144] [currently: 262144]
    styler.render.max_rows : int, optional
        The maximum number of rows that will be rendered. May still be reduced
to
        satisfy ``max_elements``, which takes precedence.
        [default: None] [currently: None]
    styler.render.repr : str
        Determine which output to use in Jupyter Notebook in {"html", "latex"}.
        [default: html] [currently: html]
    styler.sparse.columns : bool
        Whether to sparsify the display of hierarchical columns. Setting to
False will
        display each explicit level element in a hierarchical key for each
column.
        [default: True] [currently: True]
    styler.sparse.index : bool
        Whether to sparsify the display of a hierarchical index. Setting to
False will
        display each explicit level element in a hierarchical key for each row.
        [default: True] [currently: True]

    Examples
    --------
    >>> pd.set_option('display.max_columns', 4)
    >>> df = pd.DataFrame([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
    >>> df
       0  1  …  3   4
    0  1  2  …  4   5
    1  6  7  …  9  10
    [2 rows x 5 columns]
    >>> pd.reset_option('display.max_columns')
```

[ ]:
```
####
print(path, type(path))
path
####
```

```
/root/.cache/kagglehub/datasets/kashnitsky/mlcourse/versions/17 <class 'str'>
```

[ ]:
```
df = pd.read_csv(path + "/telecom_churn.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
  State  Account length  Area code International plan Voice mail plan  \
0    KS             128        415                 No             Yes
1    OH             107        415                 No             Yes
2    NJ             137        415                 No              No
```

```
3    OH              84      408                 Yes               No
4    OK              75      415                 Yes               No

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   Customer service calls  Churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False
```

[ ]:
```
print(df.shape)
```

```
(3333, 20)
```

[ ]:
```
print(df.columns)
```

```
Index(['State', 'Account length', 'Area code', 'International plan',
       'Voice mail plan', 'Number vmail messages', 'Total day minutes',
       'Total day calls', 'Total day charge', 'Total eve minutes',
       'Total eve calls', 'Total eve charge', 'Total night minutes',
       'Total night calls', 'Total night charge', 'Total intl minutes',
       'Total intl calls', 'Total intl charge', 'Customer service calls',
       'Churn'],
      dtype='object')
```

```
[ ]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Account length          3333 non-null   int64
 2   Area code               3333 non-null   int64
 3   International plan      3333 non-null   object
 4   Voice mail plan         3333 non-null   object
 5   Number vmail messages   3333 non-null   int64
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64
 19  Churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
None
```

```
[ ]: df["Churn"] = df["Churn"].astype("int64")
```

```
[ ]: df.describe()
```

```
[ ]:        Account length  Area code  Number vmail messages  Total day minutes  \
     count         3333.00    3333.00                3333.00            3333.00
     mean           101.06     437.18                   8.10             179.78
     std             39.82      42.37                  13.69              54.47
     min              1.00     408.00                   0.00               0.00
     25%             74.00     408.00                   0.00             143.70
     50%            101.00     415.00                   0.00             179.40
```

```
75%          127.00     510.00                 20.00              216.40
max          243.00     510.00                 51.00              350.80

       Total day calls  Total day charge  Total eve minutes  Total eve calls  \
count          3333.00           3333.00            3333.00          3333.00
mean            100.44             30.56             200.98           100.11
std              20.07              9.26              50.71            19.92
min               0.00              0.00               0.00             0.00
25%              87.00             24.43             166.60            87.00
50%             101.00             30.50             201.40           100.00
75%             114.00             36.79             235.30           114.00
max             165.00             59.64             363.70           170.00

       Total eve charge  Total night minutes  Total night calls  \
count           3333.00              3333.00            3333.00
mean              17.08               200.87             100.11
std                4.31                50.57              19.57
min                0.00                23.20              33.00
25%               14.16               167.00              87.00
50%               17.12               201.20             100.00
75%               20.00               235.30             113.00
max               30.91               395.00             175.00

       Total night charge  Total intl minutes  Total intl calls  \
count             3333.00             3333.00           3333.00
mean                 9.04               10.24              4.48
std                  2.28                2.79              2.46
min                  1.04                0.00              0.00
25%                  7.52                8.50              3.00
50%                  9.05               10.30              4.00
75%                 10.59               12.10              6.00
max                 17.77               20.00             20.00

       Total intl charge  Customer service calls    Churn
count            3333.00                 3333.00  3333.00
mean                2.76                    1.56     0.14
std                 0.75                    1.32     0.35
min                 0.00                    0.00     0.00
25%                 2.30                    1.00     0.00
50%                 2.78                    1.00     0.00
75%                 3.27                    2.00     0.00
max                 5.40                    9.00     1.00
```

[ ]:
```
df.describe(include=["object", "bool"])
```

[ ]:
```
       State International plan Voice mail plan
count   3333               3333            3333
```

```
unique    51                  2                2
top       WV                 No               No
freq     106               3010             2411
```

[ ]:
```
df["Churn"].value_counts()
```

[ ]:
```
Churn
0    2850
1     483
Name: count, dtype: int64
```

[ ]:
```
# To check the distribtion of data in percentage in the range of 0 to 1
df["Churn"].value_counts(normalize=True)
```

[ ]:
```
Churn
0    0.86
1    0.14
Name: proportion, dtype: float64
```

[ ]:
```
####
df.head()
####
```

[ ]:
```
  State  Account length  Area code International plan Voice mail plan  \
0    KS             128        415                 No             Yes
1    OH             107        415                 No             Yes
2    NJ             137        415                 No              No
3    OH              84        408                Yes              No
4    OK              75        415                Yes              No

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
```

```
2                 162.6                104                 7.32
3                 196.9                 89                 8.86
4                 186.9                121                 8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   Customer service calls  Churn
0                       1      0
1                       1      0
2                       0      0
3                       2      0
4                       3      0
```

[ ]:
```
df.sort_values(by="Total day charge", ascending=False).head()
```

[ ]:
```
     State  Account length  Area code International plan Voice mail plan  \
365     CO             154        415                 No              No
985     NY              64        415                Yes              No
2594    OH             115        510                Yes              No
156     OH              83        415                 No              No
605     MO             112        415                 No              No

      Number vmail messages  Total day minutes  Total day calls  \
365                       0              350.8               75
985                       0              346.8               55
2594                      0              345.3               81
156                       0              337.4              120
605                       0              335.5               77

      Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
365              59.64              216.5               94             18.40
985              58.96              249.5               79             21.21
2594             58.70              203.4              106             17.29
156              57.36              227.4              116             19.33
605              57.04              212.5              109             18.06

      Total night minutes  Total night calls  Total night charge  \
365                 253.9                100               11.43
985                 275.4                102               12.39
2594                217.5                107                9.79
156                 153.9                114                6.93
605                 265.0                132               11.93
```

```
      Total intl minutes  Total intl calls  Total intl charge  \
365                 10.1                 9               2.73
985                 13.3                 9               3.59
2594                11.8                 8               3.19
156                 15.8                 7               4.27
605                 12.7                 8               3.43

      Customer service calls  Churn
365                        1      1
985                        1      1
2594                       1      1
156                        0      1
605                        2      1
```

[ ]:
```
df.sort_values(by=["Churn", "Total day charge"], ascending=[True, False]).head()
```

[ ]:
```
     State  Account length  Area code International plan Voice mail plan  \
688     MN              13        510                 No             Yes
2259    NC             210        415                 No             Yes
534     LA              67        510                 No              No
575     SD             114        415                 No             Yes
2858    AL             141        510                 No             Yes

      Number vmail messages  Total day minutes  Total day calls  \
688                      21              315.6              105
2259                     31              313.8               87
534                       0              310.4               97
575                      36              309.9               90
2858                     28              308.0              123

      Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
688              53.65              208.9               71             17.76
2259             53.35              147.7              103             12.55
534              52.77               66.5              123              5.65
575              52.68              200.3               89             17.03
2858             52.36              247.8              128             21.06

      Total night minutes  Total night calls  Total night charge  \
688                 260.1                123               11.70
2259                192.7                 97                8.67
534                 246.5                 99               11.09
575                 183.5                105                8.26
2858                152.9                103                6.88

      Total intl minutes  Total intl calls  Total intl charge  \
688                 12.1                 3               3.27
```

```
2259                 10.1                  7                2.73
534                   9.2                 10                2.48
575                  14.2                  2                3.83
2858                  7.4                  3                2.00

      Customer service calls  Churn
688                        3      0
2259                       3      0
534                        4      0
575                        1      0
2858                       1      0
```

[ ]:
```
df["Churn"].mean()
```

[ ]:
```
np.float64(0.14491449144914492)
```

[ ]:
```
####
df["Churn"].sum()/len(df["Churn"])
####
```

[ ]:
```
np.float64(0.14491449144914492)
```

[ ]:
```
#df[df["Churn"] == 1].mean()
```

[ ]:
```
####
df["Total eve minutes"][df["Churn"] == 1].mean()
####
```

[ ]:
```
np.float64(212.41014492753624)
```

[ ]:
```
####
df["Total eve minutes"].mean()
####
```

[ ]:
```
np.float64(200.98034803480348)
```

[ ]:
```
df[df["Churn"] == 1]["Total day minutes"].mean()
```

[ ]:
```
np.float64(206.91407867494823)
```

[ ]:
```
####
df[df["Churn"] == 1]["Total eve minutes"].mean()
####
```

[ ]:
```
np.float64(212.41014492753624)
```

```
[ ]: df[(df["Churn"] == 1) & (df["International plan"] == "No")]["Total intl␣
     ↪minutes"].max()
```

```
[ ]: 18.3
```

```
[ ]: df.loc[0:5, "State":"Area code"]
```

```
[ ]:   State  Account length  Area code
     0    KS             128        415
     1    OH             107        415
     2    NJ             137        415
     3    OH              84        408
     4    OK              75        415
     5    AL             118        510
```

```
[ ]: df.iloc[0:5, 0:3]
```

```
[ ]:   State  Account length  Area code
     0    KS             128        415
     1    OH             107        415
     2    NJ             137        415
     3    OH              84        408
     4    OK              75        415
```

```
[ ]: df[-1:]
```

```
[ ]:      State  Account length  Area code International plan Voice mail plan  \
     3332    TN              74        415                 No             Yes

           Number vmail messages  Total day minutes  Total day calls  \
     3332                     25              234.4              113

           Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
     3332             39.85              265.9               82              22.6

           Total night minutes  Total night calls  Total night charge  \
     3332                241.4                 77               10.86

           Total intl minutes  Total intl calls  Total intl charge  \
     3332                13.7                 4                3.7

           Customer service calls  Churn
     3332                       0      0
```

```
[ ]: df.apply(np.max)
[ ]: State                      WY
     Account length            243
     Area code                 510
     International plan        Yes
     Voice mail plan           Yes
     Number vmail messages      51
     Total day minutes       350.8
     Total day calls           165
     Total day charge        59.64
     Total eve minutes       363.7
     Total eve calls           170
     Total eve charge        30.91
     Total night minutes     395.0
     Total night calls         175
     Total night charge      17.77
     Total intl minutes       20.0
     Total intl calls           20
     Total intl charge         5.4
     Customer service calls      9
     Churn                       1
     dtype: object
```

```
[ ]: df[df["State"].apply(lambda state: state[0] == "W")].head()
```

```
[ ]:    State  Account length  Area code International plan Voice mail plan  \
     9     WV             141        415                Yes             Yes
     26    WY              57        408                 No             Yes
     44    WI              64        510                 No              No
     49    WY              97        415                 No             Yes
     54    WY              87        415                 No              No

         Number vmail messages  Total day minutes  Total day calls  \
     9                      37              258.6               84
     26                     39              213.0              115
     44                      0              154.0               67
     49                     24              133.2              135
     54                      0              151.0               83

         Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
     9              43.96              222.0              111             18.87
     26             36.21              191.1              112             16.24
     44             26.18              225.8              118             19.19
     49             22.64              217.2               58             18.46
     54             25.67              219.7              116             18.67

         Total night minutes  Total night calls  Total night charge  \
     9                 326.4                 97               14.69
```

```
26              182.7               115              8.22
44              265.3                86             11.94
49               70.6                79              3.18
54              203.9               127              9.18

    Total intl minutes  Total intl calls  Total intl charge  \
9                 11.2                 5               3.02
26                 9.5                 3               2.57
44                 3.5                 3               0.95
49                11.0                 3               2.97
54                 9.7                 3               2.62

    Customer service calls  Churn
9                        0      0
26                       0      0
44                       1      0
49                       1      0
54                       5      1
```

[ ]:
```
####
df["International plan"].head()
####
```

[ ]:
```
0     No
1     No
2     No
3    Yes
4    Yes
Name: International plan, dtype: object
```

[ ]:
```
d = {"No": False, "Yes": True}
df["International plan"] = df["International plan"].map(d)
df.head()
```

[ ]:
```
  State  Account length  Area code  International plan Voice mail plan  \
0    KS             128        415               False             Yes
1    OH             107        415               False             Yes
2    NJ             137        415               False              No
3    OH              84        408                True              No
4    OK              75        415                True              No

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113
```

```
   Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   Customer service calls  Churn
0                       1      0
1                       1      0
2                       0      0
3                       2      0
4                       3      0
```

[ ]:
```
df = df.replace({"Voice mail plan": d})
df.head()
```

```
/tmp/ipython-input-1212166602.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df = df.replace({"Voice mail plan": d})
```

[ ]:
```
  State  Account length  Area code  International plan  Voice mail plan  \
0    KS             128        415               False             True
1    OH             107        415               False             True
2    NJ             137        415               False            False
3    OH              84        408                True            False
4    OK              75        415                True            False

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
```

```
1                    26           161.6            123
2                     0           243.4            114
3                     0           299.4             71
4                     0           166.7            113

   Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   Customer service calls  Churn
0                       1      0
1                       1      0
2                       0      0
3                       2      0
4                       3      0
```

[ ]:
```
####
help(df.replace)
####
```

```
Help on method replace in module pandas.core.generic:

replace(to_replace=None, value=<no_default>, *, inplace: 'bool_t' = False,
limit: 'int | None' = None, regex: 'bool_t' = False, method: "Literal['pad',
'ffill', 'bfill'] | lib.NoDefault" = <no_default>) -> 'Self | None' method of
pandas.core.frame.DataFrame instance
    Replace values given in `to_replace` with `value`.

    Values of the Series/DataFrame are replaced with other values dynamically.
    This differs from updating with ``.loc`` or ``.iloc``, which require
    you to specify a location to update with some value.

    Parameters
    ----------
    to_replace : str, regex, list, dict, Series, int, float, or None
        How to find the values that will be replaced.

        * numeric, str or regex:

            - numeric: numeric values equal to `to_replace` will be
              replaced with `value`
            - str: string exactly matching `to_replace` will be replaced
              with `value`
            - regex: regexs matching `to_replace` will be replaced with
              `value`

        * list of str, regex, or numeric:

            - First, if `to_replace` and `value` are both lists, they
              **must** be the same length.
            - Second, if ``regex=True`` then all of the strings in **both**
              lists will be interpreted as regexs otherwise they will match
              directly. This doesn't matter much for `value` since there
              are only a few possible substitution regexes you can use.
            - str, regex and numeric rules apply as above.

        * dict:

            - Dicts can be used to specify different replacement values
              for different existing values. For example,
              ``{'a': 'b', 'y': 'z'}`` replaces the value 'a' with 'b' and
              'y' with 'z'. To use a dict in this way, the optional `value`
              parameter should not be given.
            - For a DataFrame a dict can specify that different values
              should be replaced in different columns. For example,
              ``{'a': 1, 'b': 'z'}`` looks for the value 1 in column 'a'
              and the value 'z' in column 'b' and replaces these values
              with whatever is specified in `value`. The `value` parameter
              should not be ``None`` in this case. You can treat this as a
              special case of passing two lists except that you are
              specifying the column to search in.
            - For a DataFrame nested dictionaries, e.g.,
              ``{'a': {'b': np.nan}}``, are read as follows: look in column
              'a' for the value 'b' and replace it with NaN. The optional
`value`
              parameter should not be specified to use a nested dict in this
              way. You can nest regular expressions as well. Note that
              column names (the top-level dictionary keys in a nested
              dictionary) **cannot** be regular expressions.

        * None:

            - This means that the `regex` argument must be a string,
              compiled regular expression, or list, dict, ndarray or
              Series of such elements. If `value` is also ``None`` then
              this **must** be a nested dictionary or Series.

        See the examples section for examples of each of these.
    value : scalar, dict, list, str, regex, default None
        Value to replace any values matching `to_replace` with.
        For a DataFrame a dict of values can be used to specify which
        value to use for each column (columns not in the dict will not be
        filled). Regular expressions, strings and lists or dicts of such
        objects are also allowed.

    inplace : bool, default False
        If True, performs operation inplace and returns None.
    limit : int, default None
        Maximum size gap to forward or backward fill.

        .. deprecated:: 2.1.0
    regex : bool or same types as `to_replace`, default False
        Whether to interpret `to_replace` and/or `value` as regular
        expressions. Alternatively, this could be a regular expression or a
        list, dict, or array of regular expressions in which case
        `to_replace` must be ``None``.
    method : {'pad', 'ffill', 'bfill'}
        The method to use when for replacement, when `to_replace` is a
        scalar, list or tuple and `value` is ``None``.

        .. deprecated:: 2.1.0

    Returns
    -------
    Series/DataFrame
        Object after replacement.

    Raises
    ------
    AssertionError
        * If `regex` is not a ``bool`` and `to_replace` is not
          ``None``.

    TypeError
        * If `to_replace` is not a scalar, array-like, ``dict``, or ``None``
        * If `to_replace` is a ``dict`` and `value` is not a ``list``,
          ``dict``, ``ndarray``, or ``Series``
        * If `to_replace` is ``None`` and `regex` is not compilable
          into a regular expression or is a list, dict, ndarray, or
          Series.
        * When replacing multiple ``bool`` or ``datetime64`` objects and
          the arguments to `to_replace` does not match the type of the
          value being replaced

    ValueError
        * If a ``list`` or an ``ndarray`` is passed to `to_replace` and
          `value` but they are not the same length.

    See Also
    --------
    Series.fillna : Fill NA values.
    DataFrame.fillna : Fill NA values.
    Series.where : Replace values based on boolean condition.
    DataFrame.where : Replace values based on boolean condition.
    DataFrame.map: Apply a function to a Dataframe elementwise.
    Series.map: Map values of Series according to an input mapping or function.
    Series.str.replace : Simple string replacement.

    Notes
    -----
    * Regex substitution is performed under the hood with ``re.sub``. The
      rules for substitution for ``re.sub`` are the same.
    * Regular expressions will only substitute on strings, meaning you
      cannot provide, for example, a regular expression matching floating
      point numbers and expect the columns in your frame that have a
      numeric dtype to be matched. However, if those floating point
      numbers *are* strings, then you can do this.
    * This method has *a lot* of options. You are encouraged to experiment
      and play with this method to gain intuition about how it works.
    * When dict is used as the `to_replace` value, it is like
      key(s) in the dict are the to_replace part and
      value(s) in the dict are the value parameter.

    Examples
    --------

    **Scalar `to_replace` and `value`**

    >>> s = pd.Series([1, 2, 3, 4, 5])
    >>> s.replace(1, 5)
    0    5
    1    2
    2    3
    3    4
```

```
4    5
dtype: int64

>>> df = pd.DataFrame({'A': [0, 1, 2, 3, 4],
…                    'B': [5, 6, 7, 8, 9],
…                    'C': ['a', 'b', 'c', 'd', 'e']})
>>> df.replace(0, 5)
    A  B  C
0  5  5  a
1  1  6  b
2  2  7  c
3  3  8  d
4  4  9  e
```

List-like `to_replace`

```
>>> df.replace([0, 1, 2, 3], 4)
    A  B  C
0  4  5  a
1  4  6  b
2  4  7  c
3  4  8  d
4  4  9  e

>>> df.replace([0, 1, 2, 3], [4, 3, 2, 1])
    A  B  C
0  4  5  a
1  3  6  b
2  2  7  c
3  1  8  d
4  4  9  e

>>> s.replace([1, 2], method='bfill')
0    3
1    3
2    3
3    4
4    5
dtype: int64
```

dict-like `to_replace`

```
>>> df.replace({0: 10, 1: 100})
        A  B  C
0   10  5  a
1  100  6  b
2    2  7  c
3    3  8  d
```

```
4    4  9  e

>>> df.replace({'A': 0, 'B': 5}, 100)
        A    B  C
0  100  100  a
1    1    6  b
2    2    7  c
3    3    8  d
4    4    9  e

>>> df.replace({'A': {0: 100, 4: 400}})
        A  B  C
0  100  5  a
1    1  6  b
2    2  7  c
3    3  8  d
4  400  9  e

**Regular expression `to_replace`**

>>> df = pd.DataFrame({'A': ['bat', 'foo', 'bait'],
…                    'B': ['abc', 'bar', 'xyz']})
>>> df.replace(to_replace=r'^ba.$', value='new', regex=True)
        A    B
0   new  abc
1   foo  new
2  bait  xyz

>>> df.replace({'A': r'^ba.$'}, {'A': 'new'}, regex=True)
        A    B
0   new  abc
1   foo  bar
2  bait  xyz

>>> df.replace(regex=r'^ba.$', value='new')
        A    B
0   new  abc
1   foo  new
2  bait  xyz

>>> df.replace(regex={r'^ba.$': 'new', 'foo': 'xyz'})
        A    B
0   new  abc
1   xyz  new
2  bait  xyz

>>> df.replace(regex=[r'^ba.$', 'foo'], value='new')
        A    B
```

```
0   new  abc
1   new  new
2  bait  xyz

Compare the behavior of ``s.replace({'a': None})`` and
``s.replace('a', None)`` to understand the peculiarities
of the `to_replace` parameter:

>>> s = pd.Series([10, 'a', 'a', 'b', 'a'])

When one uses a dict as the `to_replace` value, it is like the
value(s) in the dict are equal to the `value` parameter.
``s.replace({'a': None})`` is equivalent to
``s.replace(to_replace={'a': None}, value=None, method=None)``:

>>> s.replace({'a': None})
0      10
1    None
2    None
3       b
4    None
dtype: object

When ``value`` is not explicitly passed and `to_replace` is a scalar, list
or tuple, `replace` uses the method parameter (default 'pad') to do the
replacement. So this is why the 'a' values are being replaced by 10
in rows 1 and 2 and 'b' in row 4 in this case.

>>> s.replace('a')
0    10
1    10
2    10
3     b
4     b
dtype: object

    .. deprecated:: 2.1.0
        The 'method' parameter and padding behavior are deprecated.

On the other hand, if ``None`` is explicitly passed for ``value``, it will
be respected:

>>> s.replace('a', None)
0      10
1    None
2    None
3       b
4    None
```

```
    dtype: object

        .. versionchanged:: 1.4.0
            Previously the explicit ``None`` was silently ignored.

    When ``regex=True``, ``value`` is not ``None`` and `to_replace` is a string,
    the replacement will be applied in all columns of the DataFrame.

    >>> df = pd.DataFrame({'A': [0, 1, 2, 3, 4],
    …                      'B': ['a', 'b', 'c', 'd', 'e'],
    …                      'C': ['f', 'g', 'h', 'i', 'j']})

    >>> df.replace(to_replace='^[a-g]', value='e', regex=True)
        A  B  C
    0   0  e  e
    1   1  e  e
    2   2  e  h
    3   3  e  i
    4   4  e  j

    If ``value`` is not ``None`` and `to_replace` is a dictionary, the
dictionary
    keys will be the DataFrame columns that the replacement will be applied.

    >>> df.replace(to_replace={'B': '^[a-c]', 'C': '^[h-j]'}, value='e',
regex=True)
        A  B  C
    0   0  e  f
    1   1  e  g
    2   2  e  e
    3   3  d  e
    4   4  e  e
```

```
[ ]: columns_to_show = ["Total day minutes", "Total eve minutes", "Total night␣
      ↪minutes"]

     df.groupby(["Churn"])[columns_to_show].describe(percentiles=[])
```

```
[ ]:      Total day minutes                                   Total eve minutes  \
                     count    mean    std  min    50%    max             count
     Churn
     0              2850.0  175.18  50.18  0.0  177.2  315.6            2850.0
     1               483.0  206.91  69.00  0.0  217.6  350.8             483.0

                                             Total night minutes                 \
            mean     std   min    50%    max               count    mean    std
```

```
Churn
0      199.04  50.29   0.0  199.6  361.8              2850.0  200.13  51.11
1      212.41  51.73  70.9  211.3  363.7               483.0  205.23  47.13


        min     50%    max
Churn
0      23.2  200.25  395.0
1      47.4  204.80  354.9
```

[ ]:
```
####
help(df.groupby)
####
```

```
Help on method groupby in module pandas.core.frame:

groupby(by=None, axis: 'Axis | lib.NoDefault' = <no_default>, level: 'IndexLabel
| None' = None, as_index: 'bool' = True, sort: 'bool' = True, group_keys: 'bool'
= True, observed: 'bool | lib.NoDefault' = <no_default>, dropna: 'bool' = True)
-> 'DataFrameGroupBy' method of pandas.core.frame.DataFrame instance
    Group DataFrame using a mapper or by a Series of columns.

    A groupby operation involves some combination of splitting the
    object, applying a function, and combining the results. This can be
    used to group large amounts of data and compute operations on these
    groups.

    Parameters
    ----------
    by : mapping, function, label, pd.Grouper or list of such
        Used to determine the groups for the groupby.
        If ``by`` is a function, it's called on each value of the object's
        index. If a dict or Series is passed, the Series or dict VALUES
        will be used to determine the groups (the Series' values are first
        aligned; see ``.align()`` method). If a list or ndarray of length
        equal to the selected axis is passed (see the `groupby user guide
        <https://pandas.pydata.org/pandas-
docs/stable/user_guide/groupby.html#splitting-an-object-into-groups>`_),
        the values are used as-is to determine the groups. A label or list
        of labels may be passed to group by the columns in ``self``.
        Notice that a tuple is interpreted as a (single) key.
    axis : {0 or 'index', 1 or 'columns'}, default 0
        Split along rows (0) or columns (1). For `Series` this parameter
        is unused and defaults to 0.

        .. deprecated:: 2.1.0
```

```
            Will be removed and behave like axis=0 in a future version.
            For ``axis=1``, do ``frame.T.groupby(...)`` instead.

    level : int, level name, or sequence of such, default None
        If the axis is a MultiIndex (hierarchical), group by a particular
        level or levels. Do not specify both ``by`` and ``level``.
    as_index : bool, default True
        Return object with group labels as the
        index. Only relevant for DataFrame input. as_index=False is
        effectively "SQL-style" grouped output. This argument has no effect
        on filtrations (see the `filtrations in the user guide
<https://pandas.pydata.org/docs/dev/user_guide/groupby.html#filtration>`_),
        such as ``head()``, ``tail()``, ``nth()`` and in transformations
        (see the `transformations in the user guide
<https://pandas.pydata.org/docs/dev/user_guide/groupby.html#transformation>`_).
    sort : bool, default True
        Sort group keys. Get better performance by turning this off.
        Note this does not influence the order of observations within each
        group. Groupby preserves the order of rows within each group. If False,
        the groups will appear in the same order as they did in the original
DataFrame.
        This argument has no effect on filtrations (see the `filtrations in the
user guide
<https://pandas.pydata.org/docs/dev/user_guide/groupby.html#filtration>`_),
        such as ``head()``, ``tail()``, ``nth()`` and in transformations
        (see the `transformations in the user guide
<https://pandas.pydata.org/docs/dev/user_guide/groupby.html#transformation>`_).

        .. versionchanged:: 2.0.0

            Specifying ``sort=False`` with an ordered categorical grouper will
no
            longer sort the values.

    group_keys : bool, default True
        When calling apply and the ``by`` argument produces a like-indexed
        (i.e. :ref:`a transform <groupby.transform>`) result, add group keys to
        index to identify pieces. By default group keys are not included
        when the result's index (and column) labels match the inputs, and
        are included otherwise.

        .. versionchanged:: 1.5.0

           Warns that ``group_keys`` will no longer be ignored when the
           result from ``apply`` is a like-indexed Series or DataFrame.
           Specify ``group_keys`` explicitly to include the group keys or
           not.
        .. versionchanged:: 2.0.0

           ``group_keys`` now defaults to ``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:: 2.1.0

            The default value will change to True in a future version of pandas.

    dropna : bool, default True
        If True, and if group keys contain NA values, NA values together
        with row/column will be dropped.
        If False, NA values will also be treated as the key in groups.

    Returns
    -------
    pandas.api.typing.DataFrameGroupBy
        Returns a groupby object that contains information about the groups.

    See Also
    --------
    resample : Convenience method for frequency conversion and resampling
        of time series.

    Notes
    -----
    See the `user guide
    <https://pandas.pydata.org/pandas-docs/stable/groupby.html>`__ for more
    detailed usage and examples, including splitting an object into groups,
    iterating through groups, selecting a group, aggregation, and more.

    Examples
    --------
    >>> df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
    …                                 'Parrot', 'Parrot'],
    …                    'Max Speed': [380., 370., 24., 26.]})
    >>> df
       Animal  Max Speed
    0  Falcon      380.0
    1  Falcon      370.0
    2  Parrot       24.0
    3  Parrot       26.0
    >>> df.groupby(['Animal']).mean()
            Max Speed
```

```
Animal
Falcon      375.0
Parrot       25.0
```

Hierarchical Indexes

We can groupby different levels of a hierarchical index
using the `level` parameter:

```
>>> arrays = [['Falcon', 'Falcon', 'Parrot', 'Parrot'],
…           ['Captive', 'Wild', 'Captive', 'Wild']]
>>> index = pd.MultiIndex.from_arrays(arrays, names=('Animal', 'Type'))
>>> df = pd.DataFrame({'Max Speed': [390., 350., 30., 20.]},
…                   index=index)
>>> df
                Max Speed
Animal Type
Falcon Captive      390.0
       Wild         350.0
Parrot Captive       30.0
       Wild          20.0
>>> df.groupby(level=0).mean()
        Max Speed
Animal
Falcon      370.0
Parrot       25.0
>>> df.groupby(level="Type").mean()
         Max Speed
Type
Captive      210.0
Wild         185.0
```

We can also choose to include NA in group keys or not by setting
`dropna` parameter, the default setting is `True`.

```
>>> l = [[1, 2, 3], [1, None, 4], [2, 1, 3], [1, 2, 2]]
>>> df = pd.DataFrame(l, columns=["a", "b", "c"])

>>> df.groupby(by=["b"]).sum()
    a   c
b
1.0 2   3
2.0 2   5

>>> df.groupby(by=["b"], dropna=False).sum()
    a   c
b
1.0 2   3
```

```
2.0 2   5
NaN 1   4

>>> l = [["a", 12, 12], [None, 12.3, 33.], ["b", 12.3, 123], ["a", 1, 1]]
>>> df = pd.DataFrame(l, columns=["a", "b", "c"])

>>> df.groupby(by="a").sum()
    b     c
a
a   13.0   13.0
b   12.3  123.0

>>> df.groupby(by="a", dropna=False).sum()
    b     c
a
a   13.0   13.0
b   12.3  123.0
NaN 12.3   33.0

When using ``.apply()``, use ``group_keys`` to include or exclude the
group keys. The ``group_keys`` argument defaults to ``True`` (include).

>>> df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
…                                 'Parrot', 'Parrot'],
…                      'Max Speed': [380., 370., 24., 26.]})
>>> df.groupby("Animal", group_keys=True)[['Max Speed']].apply(lambda x: x)
          Max Speed
Animal
Falcon 0      380.0
       1      370.0
Parrot 2       24.0
       3       26.0

>>> df.groupby("Animal", group_keys=False)[['Max Speed']].apply(lambda x: x)
   Max Speed
0      380.0
1      370.0
2       24.0
3       26.0
```

[ ]:
```
####
df.groupby(["Churn"])
####
```

[ ]:
```
   State  Account length  Area code  International plan  Voice mail plan  \
0     KS             128        415               False             True
```

```
1     OH             107        415                False              True
2     NJ             137        415                False             False
3     OH              84        408                 True             False
4     OK              75        415                 True             False
10    IN              65        415                False             False
15    NY             161        415                False             False
21    CO              77        408                False             False
33    AZ              12        408                False             False
41    MD             135        408                 True              True

    Number vmail messages  Total day minutes  Total day calls  \
0                      25              265.1              110
1                      26              161.6              123
2                       0              243.4              114
3                       0              299.4               71
4                       0              166.7              113
10                      0              129.1              137
15                      0              332.9               67
21                      0               62.4               89
33                      0              249.6              118
41                     41              173.1               85

    Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0              45.07              197.4               99             16.78
1              27.47              195.5              103             16.62
2              41.38              121.2              110             10.30
3              50.90               61.9               88              5.26
4              28.34              148.3              122             12.61
10             21.95              228.5               83             19.42
15             56.59              317.8               97             27.01
21             10.61              169.9              121             14.44
33             42.43              252.4              119             21.45
41             29.43              203.9              107             17.33

    Total night minutes  Total night calls  Total night charge  \
0                 244.7                 91               11.01
1                 254.4                103               11.45
2                 162.6                104                7.32
3                 196.9                 89                8.86
4                 186.9                121                8.41
10                208.8                111                9.40
15                160.6                128                7.23
21                209.6                 64                9.43
33                280.2                 90               12.61
41                122.2                 78                5.50

    Total intl minutes  Total intl calls  Total intl charge  \
0                 10.0                  3               2.70
1                 13.7                  3               3.70
2                 12.2                  5               3.29
3                  6.6                  7               1.78
4                 10.1                  3               2.73
10                12.7                  6               3.43
15                 5.4                  9               1.46
21                 5.7                  6               1.54
33                11.8                  3               3.19
41                14.6                 15               3.94

    Customer service calls  Churn
0                        1      0
1                        1      0
2                        0      0
3                        2      0
4                        3      0
10                       4      1
15                       4      1
21                       5      1
33                       1      1
41                       0      1
```

[ ]:
```
####
help(df.describe)
####
```

```
Help on method describe in module pandas.core.generic:

describe(percentiles=None, include=None, exclude=None) -> 'Self' method of
pandas.core.frame.DataFrame instance
    Generate descriptive statistics.

    Descriptive statistics include those that summarize the central
    tendency, dispersion and shape of a
    dataset's distribution, excluding ``NaN`` values.

    Analyzes both numeric and object series, as well
    as ``DataFrame`` column sets of mixed data types. The output
    will vary depending on what is provided. Refer to the notes
    below for more detail.

    Parameters
    ----------
    percentiles : list-like of numbers, optional
        The percentiles to include in the output. All should
        fall between 0 and 1. The default is
        ``[.25, .5, .75]``, which returns the 25th, 50th, and
        75th percentiles.
    include : 'all', list-like of dtypes or None (default), optional
        A white list of data types to include in the result. Ignored
        for ``Series``. Here are the options:

        - 'all' : All columns of the input will be included in the output.
        - A list-like of dtypes : Limits the results to the
          provided data types.
          To limit the result to numeric types submit
          ``numpy.number``. To limit it instead to object columns submit
          the ``numpy.object`` data type. Strings
          can also be used in the style of
          ``select_dtypes`` (e.g. ``df.describe(include=['O'])``). To
          select pandas categorical columns, use ``'category'``
        - None (default) : The result will include all numeric columns.
    exclude : list-like of dtypes or None (default), optional,
        A black list of data types to omit from the result. Ignored
        for ``Series``. Here are the options:

        - A list-like of dtypes : Excludes the provided data types
          from the result. To exclude numeric types submit
          ``numpy.number``. To exclude object columns submit the data
          type ``numpy.object``. Strings can also be used in the style of
          ``select_dtypes`` (e.g. ``df.describe(exclude=['O'])``). To
          exclude pandas categorical columns, use ``'category'``
        - None (default) : The result will exclude nothing.

    Returns
    -------
    Series or DataFrame
        Summary statistics of the Series or Dataframe provided.

    See Also
    --------
    DataFrame.count: Count number of non-NA/null observations.
    DataFrame.max: Maximum of the values in the object.
    DataFrame.min: Minimum of the values in the object.
    DataFrame.mean: Mean of the values.
    DataFrame.std: Standard deviation of the observations.
    DataFrame.select_dtypes: Subset of a DataFrame including/excluding
        columns based on their dtype.

    Notes
    -----
    For numeric data, the result's index will include ``count``,
    ``mean``, ``std``, ``min``, ``max`` as well as lower, ``50`` and
    upper percentiles. By default the lower percentile is ``25`` and the
    upper percentile is ``75``. The ``50`` percentile is the
    same as the median.

    For object data (e.g. strings or timestamps), the result's index
    will include ``count``, ``unique``, ``top``, and ``freq``. The ``top``
    is the most common value. The ``freq`` is the most common value's
    frequency. Timestamps also include the ``first`` and ``last`` items.

    If multiple object values have the highest count, then the
    ``count`` and ``top`` results will be arbitrarily chosen from
    among those with the highest count.

    For mixed data types provided via a ``DataFrame``, the default is to
    return only an analysis of numeric columns. If the dataframe consists
    only of object and categorical data without any numeric columns, the
    default is to return an analysis of both the object and categorical
    columns. If ``include='all'`` is provided as an option, the result
    will include a union of attributes of each type.

    The `include` and `exclude` parameters can be used to limit
    which columns in a ``DataFrame`` are analyzed for the output.
    The parameters are ignored when analyzing a ``Series``.

    Examples
    --------
    Describing a numeric ``Series``.

    >>> s = pd.Series([1, 2, 3])
    >>> s.describe()
    count    3.0
    mean     2.0
    std      1.0
    min      1.0
    25%      1.5
    50%      2.0
    75%      2.5
    max      3.0
    dtype: float64

    Describing a categorical ``Series``.

    >>> s = pd.Series(['a', 'a', 'b', 'c'])
    >>> s.describe()
    count     4
    unique    3
    top       a
    freq      2
    dtype: object
```

```
Describing a timestamp ``Series``.

>>> s = pd.Series([
…     np.datetime64("2000-01-01"),
…     np.datetime64("2010-01-01"),
…     np.datetime64("2010-01-01")
… ])
>>> s.describe()
count                      3
mean     2006-09-01 08:00:00
min      2000-01-01 00:00:00
25%      2004-12-31 12:00:00
50%      2010-01-01 00:00:00
75%      2010-01-01 00:00:00
max      2010-01-01 00:00:00
dtype: object

Describing a ``DataFrame``. By default only numeric fields
are returned.

>>> df = pd.DataFrame({'categorical': pd.Categorical(['d', 'e', 'f']),
…                    'numeric': [1, 2, 3],
…                    'object': ['a', 'b', 'c']
…                    })
>>> df.describe()
       numeric
count      3.0
mean       2.0
std        1.0
min        1.0
25%        1.5
50%        2.0
75%        2.5
max        3.0

Describing all columns of a ``DataFrame`` regardless of data type.

>>> df.describe(include='all')  # doctest: +SKIP
       categorical  numeric object
count            3      3.0      3
unique           3      NaN      3
top              f      NaN      a
freq             1      NaN      1
mean           NaN      2.0    NaN
std            NaN      1.0    NaN
min            NaN      1.0    NaN
25%            NaN      1.5    NaN
```

```
50%            NaN      2.0    NaN
75%            NaN      2.5    NaN
max            NaN      3.0    NaN
```

Describing a column from a ``DataFrame`` by accessing it as
an attribute.

```
>>> df.numeric.describe()
count    3.0
mean     2.0
std      1.0
min      1.0
25%      1.5
50%      2.0
75%      2.5
max      3.0
Name: numeric, dtype: float64
```

Including only numeric columns in a ``DataFrame`` description.

```
>>> df.describe(include=[np.number])
       numeric
count      3.0
mean       2.0
std        1.0
min        1.0
25%        1.5
50%        2.0
75%        2.5
max        3.0
```

Including only string columns in a ``DataFrame`` description.

```
>>> df.describe(include=[object])  # doctest: +SKIP
       object
count       3
unique      3
top         a
freq        1
```

Including only categorical columns from a ``DataFrame`` description.

```
>>> df.describe(include=['category'])
       categorical
count            3
unique           3
top              d
freq             1
```

```
    Excluding numeric columns from a ``DataFrame`` description.

    >>> df.describe(exclude=[np.number])  # doctest: +SKIP
           categorical object
    count            3      3
    unique           3      3
    top              f      a
    freq             1      1

    Excluding object columns from a ``DataFrame`` description.

    >>> df.describe(exclude=[object])  # doctest: +SKIP
           categorical  numeric
    count            3      3.0
    unique           3      NaN
    top              f      NaN
    freq             1      NaN
    mean           NaN      2.0
    std            NaN      1.0
    min            NaN      1.0
    25%            NaN      1.5
    50%            NaN      2.0
    75%            NaN      2.5
    max            NaN      3.0
```

[ ]:
```
columns_to_show = ["Total day minutes", "Total eve minutes", "Total night␣
 ↪minutes"]
df.groupby(["Churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max])
```

```
/tmp/ipython-input-1186384480.py:2: FutureWarning: The provided callable
<function mean at 0x7fa0b7588220> is currently using SeriesGroupBy.mean. In a
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "mean" instead.
  df.groupby(["Churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max])
/tmp/ipython-input-1186384480.py:2: FutureWarning: The provided callable
<function std at 0x7fa0b7588360> is currently using SeriesGroupBy.std. In a
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "std" instead.
  df.groupby(["Churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max])
/tmp/ipython-input-1186384480.py:2: FutureWarning: The provided callable
<function min at 0x7fa0b7583920> is currently using SeriesGroupBy.min. In a
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "min" instead.
  df.groupby(["Churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max])
/tmp/ipython-input-1186384480.py:2: FutureWarning: The provided callable
<function max at 0x7fa0b75837e0> is currently using SeriesGroupBy.max. In a
```

```
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "max" instead.
  df.groupby(["Churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max])
```

[ ]:
```
      Total day minutes                       Total eve minutes               \
                   mean    std  min    max              mean    std   min
Churn
0                175.18  50.18  0.0  315.6            199.04  50.29   0.0
1                206.91  69.00  0.0  350.8            212.41  51.73  70.9

              Total night minutes
        max                  mean    std   min    max
Churn
0     361.8                200.13  51.11  23.2  395.0
1     363.7                205.23  47.13  47.4  354.9
```

[ ]:
```
####
help(df.agg)
####
```

```
Help on method aggregate in module pandas.core.frame:

aggregate(func=None, axis: 'Axis' = 0, *args, **kwargs) method of
pandas.core.frame.DataFrame instance
    Aggregate using one or more operations over the specified axis.

    Parameters
    ----------
    func : function, str, list or dict
        Function to use for aggregating the data. If a function, must either
        work when passed a DataFrame or when passed to DataFrame.apply.

        Accepted combinations are:

        - function
        - string function name
        - list of functions and/or function names, e.g. ``[np.sum, 'mean']``
        - dict of axis labels -> functions, function names or list of such.
    axis : {0 or 'index', 1 or 'columns'}, default 0
            If 0 or 'index': apply function to each column.
            If 1 or 'columns': apply function to each row.
    *args
        Positional arguments to pass to `func`.
    **kwargs
        Keyword arguments to pass to `func`.

    Returns
    -------
    scalar, Series or DataFrame

        The return can be:

        * scalar : when Series.agg is called with single function
        * Series : when DataFrame.agg is called with a single function
        * DataFrame : when DataFrame.agg is called with several functions

    See Also
    --------
    DataFrame.apply : Perform any type of operations.
    DataFrame.transform : Perform transformation type operations.
    pandas.DataFrame.groupby : Perform operations over groups.
    pandas.DataFrame.resample : Perform operations over resampled bins.
    pandas.DataFrame.rolling : Perform operations over rolling window.
    pandas.DataFrame.expanding : Perform operations over expanding window.
    pandas.core.window.ewm.ExponentialMovingWindow : Perform operation over
exponential
        weighted window.

    Notes
    -----
    The aggregation operations are always performed over an axis, either the
    index (default) or the column axis. This behavior is different from
    `numpy` aggregation functions (`mean`, `median`, `prod`, `sum`, `std`,
    `var`), where the default is to compute the aggregation of the flattened
    array, e.g., ``numpy.mean(arr_2d)`` as opposed to
    ``numpy.mean(arr_2d, axis=0)``.

    `agg` is an alias for `aggregate`. Use the alias.

    Functions that mutate the passed object can produce unexpected
    behavior or errors and are not supported. See :ref:`gotchas.udf-mutation`
    for more details.

    A passed user-defined-function will be passed a Series for evaluation.

    Examples
    --------
    >>> df = pd.DataFrame([[1, 2, 3],
    …                      [4, 5, 6],
    …                      [7, 8, 9],
    …                      [np.nan, np.nan, np.nan]],
    …                     columns=['A', 'B', 'C'])

    Aggregate these functions over the rows.

    >>> df.agg(['sum', 'min'])
```

```
             A     B     C
    sum   12.0  15.0  18.0
    min    1.0   2.0   3.0

    Different aggregations per column.

    >>> df.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']})
             A    B
    sum   12.0  NaN
    min    1.0  2.0
    max    NaN  8.0

    Aggregate different functions over the columns and rename the index of the
resulting
    DataFrame.

    >>> df.agg(x=('A', 'max'), y=('B', 'min'), z=('C', 'mean'))
         A    B    C
    x  7.0  NaN  NaN
    y  NaN  2.0  NaN
    z  NaN  NaN  6.0

    Aggregate over the columns.

    >>> df.agg("mean", axis="columns")
    0    2.0
    1    5.0
    2    8.0
    3    NaN
    dtype: float64
```

[ ]:
```
# Summary tables
```

[ ]:
```
pd.crosstab(df["Churn"], df["International plan"])
```

[ ]:
```
International plan  False  True
Churn
0                    2664   186
1                     346   137
```

[ ]:
```
####
len(df[df["Churn"]==1]), len(df[df["International plan"]==1])
####
```

[ ]:
```
(483, 323)
```

```
[ ]: ####
     help(pd.crosstab)
     ####
```

```
Help on function crosstab in module pandas.core.reshape.pivot:

crosstab(index, columns, values=None, rownames=None, colnames=None,
aggfunc=None, margins: 'bool' = False, margins_name: 'Hashable' = 'All', dropna:
'bool' = True, normalize: "bool | Literal[0, 1, 'all', 'index', 'columns']" =
False) -> 'DataFrame'
    Compute a simple cross tabulation of two (or more) factors.

    By default, computes a frequency table of the factors unless an
    array of values and an aggregation function are passed.

    Parameters
    ----------
    index : array-like, Series, or list of arrays/Series
        Values to group by in the rows.
    columns : array-like, Series, or list of arrays/Series
        Values to group by in the columns.
    values : array-like, optional
        Array of values to aggregate according to the factors.
        Requires `aggfunc` be specified.
    rownames : sequence, default None
        If passed, must match number of row arrays passed.
    colnames : sequence, default None
        If passed, must match number of column arrays passed.
    aggfunc : function, optional
        If specified, requires `values` be specified as well.
    margins : bool, default False
        Add row/column margins (subtotals).
    margins_name : str, default 'All'
        Name of the row/column that will contain the totals
        when margins is True.
    dropna : bool, default True
        Do not include columns whose entries are all NaN.
    normalize : bool, {'all', 'index', 'columns'}, or {0,1}, default False
        Normalize by dividing all values by the sum of values.

        - If passed 'all' or `True`, will normalize over all values.
        - If passed 'index' will normalize over each row.
        - If passed 'columns' will normalize over each column.
        - If margins is `True`, will also normalize margin values.

    Returns
    -------
    DataFrame
        Cross tabulation of the data.

    See Also
    --------
    DataFrame.pivot : Reshape data based on column values.
    pivot_table : Create a pivot table as a DataFrame.

    Notes
    -----
    Any Series passed will have their name attributes used unless row or column
    names for the cross-tabulation are specified.

    Any input passed containing Categorical data will have **all** of its
    categories included in the cross-tabulation, even if the actual data does
    not contain any instances of a particular category.

    In the event that there aren't overlapping indexes an empty DataFrame will
    be returned.

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

    Examples
    --------
    >>> a = np.array(["foo", "foo", "foo", "foo", "bar", "bar",
    …                "bar", "bar", "foo", "foo", "foo"], dtype=object)
    >>> b = np.array(["one", "one", "one", "two", "one", "one",
    …                "one", "two", "two", "two", "one"], dtype=object)
    >>> c = np.array(["dull", "dull", "shiny", "dull", "dull", "shiny",
    …                "shiny", "dull", "shiny", "shiny", "shiny"],
    …               dtype=object)
    >>> pd.crosstab(a, [b, c], rownames=['a'], colnames=['b', 'c'])
    b   one        two
    c   dull shiny dull shiny
    a
    bar    1     2    1     0
    foo    2     2    1     2

    Here 'c' and 'f' are not represented in the data and will not be
    shown in the output because dropna is True by default. Set
    dropna=False to preserve categories with no data.

    >>> foo = pd.Categorical(['a', 'b'], categories=['a', 'b', 'c'])
    >>> bar = pd.Categorical(['d', 'e'], categories=['d', 'e', 'f'])
    >>> pd.crosstab(foo, bar)
    col_0  d  e
    row_0
    a      1  0
    b      0  1
    >>> pd.crosstab(foo, bar, dropna=False)
    col_0  d  e  f
    row_0
    a      1  0  0
    b      0  1  0
    c      0  0  0
```

[ ]:
```
pd.crosstab(df["Churn"], df["Voice mail plan"], normalize=True)
```

[ ]:
```
Voice mail plan  False  True
Churn
0                 0.60  0.25
1                 0.12  0.02
```

[ ]:
```
df.pivot_table(
    ["Total day calls", "Total eve calls", "Total night calls"],
    ["Area code"],
    aggfunc="mean",
)
```

[ ]:
```
           Total day calls  Total eve calls  Total night calls
Area code
408                 100.50            99.79              99.04
415                 100.58           100.50             100.40
510                 100.10            99.67             100.60
```

[ ]:
```
####
help(df.pivot_table)
####
```

```
Help on method pivot_table in module pandas.core.frame:

pivot_table(values=None, index=None, columns=None, aggfunc: 'AggFuncType' =
'mean', fill_value=None, margins: 'bool' = False, dropna: 'bool' = True,
margins_name: 'Level' = 'All', observed: 'bool | lib.NoDefault' = <no_default>,
sort: 'bool' = True) -> 'DataFrame' method of pandas.core.frame.DataFrame
instance
    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
    ----------
    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.
fill_value : scalar, default None
    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:: 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.

    .. versionadded:: 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 :ref:`the user guide <reshaping.pivot>` 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
A   B
bar one      4      5
    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
```

[ ]:
```
# DataFrame transformations
```

```
[ ]: total_calls = (  df["Total day calls"]
                  + df["Total eve calls"]
                  + df["Total night calls"]
                  + df["Total intl calls"]
                  )
     df.insert(loc=len(df.columns), column="Total calls", value=total_calls)
```

```
[ ]: df.head()
```

```
[ ]:  State  Account length  Area code  International plan  Voice mail plan  \
     0    KS             128        415               False             True
     1    OH             107        415               False             True
     2    NJ             137        415               False            False
     3    OH              84        408                True            False
     4    OK              75        415                True            False

        Number vmail messages  Total day minutes  Total day calls  \
     0                     25              265.1              110
     1                     26              161.6              123
     2                      0              243.4              114
     3                      0              299.4               71
     4                      0              166.7              113

        Total day charge  Total eve minutes  …  Total eve charge  \
     0             45.07              197.4  …             16.78
     1             27.47              195.5  …             16.62
     2             41.38              121.2  …             10.30
     3             50.90               61.9  …              5.26
     4             28.34              148.3  …             12.61

        Total night minutes  Total night calls  Total night charge  \
     0                244.7                 91               11.01
     1                254.4                103               11.45
     2                162.6                104                7.32
     3                196.9                 89                8.86
     4                186.9                121                8.41

        Total intl minutes  Total intl calls  Total intl charge  \
     0                10.0                 3               2.70
     1                13.7                 3               3.70
     2                12.2                 5               3.29
     3                 6.6                 7               1.78
     4                10.1                 3               2.73

        Customer service calls  Churn  Total calls
     0                       1      0          303
     1                       1      0          332
```

```
2                       0      0          333
3                       2      0          255
4                       3      0          359

[5 rows x 21 columns]
```

```
[ ]: ####
     help(df.insert)
     ####
```

```
Help on method insert in module pandas.core.frame:

insert(loc: 'int', column: 'Hashable', value: 'Scalar | AnyArrayLike',
allow_duplicates: 'bool | lib.NoDefault' = <no_default>) -> 'None' method of
pandas.core.frame.DataFrame instance
    Insert column into DataFrame at specified location.

    Raises a ValueError if `column` is already contained in the DataFrame,
    unless `allow_duplicates` is set to True.

    Parameters
    ----------
    loc : int
        Insertion index. Must verify 0 <= loc <= len(columns).
    column : str, number, or hashable object
        Label of the inserted column.
    value : Scalar, Series, or array-like
        Content of the inserted column.
    allow_duplicates : bool, optional, default lib.no_default
        Allow duplicate column labels to be created.

    See Also
    --------
    Index.insert : Insert new item by index.

    Examples
    --------
    >>> df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})
    >>> df
       col1  col2
    0     1     3
    1     2     4
    >>> df.insert(1, "newcol", [99, 99])
    >>> df
       col1  newcol  col2
    0     1      99     3
    1     2      99     4
    >>> df.insert(0, "col1", [100, 100], allow_duplicates=True)
    >>> df
       col1  col1  newcol  col2
    0   100     1      99     3
    1   100     2      99     4

    Notice that pandas uses index alignment in case of `value` from type
`Series`:

    >>> df.insert(0, "col0", pd.Series([5, 6], index=[1, 2]))
    >>> df
       col0  col1  col1  newcol  col2
    0   NaN   100     1      99     3
    1   5.0   100     2      99     4
```

[ ]:
```
####
df["Total day calls"] + df["Total eve calls"]
####
```

[ ]:
```
0       209
1       226
2       224
3       159
4       235
       …
3328    203
3329    112
3330    167
3331    189
3332    195
Length: 3333, dtype: int64
```

[ ]:
```
df["Total charge"] = (
      df["Total day charge"]
    + df["Total eve charge"]
    + df["Total night charge"]
    + df["Total intl charge"]
)
df.head()
```

[ ]:
```
  State  Account length  Area code International plan Voice mail plan  \
0    KS             128        415                 No             Yes
1    OH             107        415                 No             Yes
2    NJ             137        415                 No              No
3    OH              84        408                Yes              No
4    OK              75        415                Yes              No
```

```
   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   Total day charge  Total eve minutes  …  Total eve charge  \
0             45.07              197.4  …             16.78
1             27.47              195.5  …             16.62
2             41.38              121.2  …             10.30
3             50.90               61.9  …              5.26
4             28.34              148.3  …             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   Customer service calls  Churn  Total charge
0                       1  False         75.56
1                       1  False         59.24
2                       0  False         62.29
3                       2  False         66.80
4                       3  False         52.09

[5 rows x 21 columns]
```

```
[ ]: ####
(
      df["Total day charge"]
    + df["Total eve charge"]
    + df["Total night charge"]
    + df["Total intl charge"]
)
####
```

[ ]:
```
0       75.56
1       59.24
2       62.29
3       66.80
4       52.09
        …
3328    60.10
3329    63.53
3330    67.74
3331    57.53
3332    77.01
Length: 3333, dtype: float64
```

[ ]:
```
####
df.head(5)
####
```

[ ]:
```
  State  Account length  Area code International plan Voice mail plan  \
0    KS             128        415                 No             Yes
1    OH             107        415                 No             Yes
2    NJ             137        415                 No              No
3    OH              84        408                Yes              No
4    OK              75        415                Yes              No

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   Total day charge  Total eve minutes  …  Total eve charge  \
0             45.07              197.4  …             16.78
1             27.47              195.5  …             16.62
2             41.38              121.2  …             10.30
3             50.90               61.9  …              5.26
4             28.34              148.3  …             12.61

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70
```

```
1                13.7                3               3.70
2                12.2                5               3.29
3                 6.6                7               1.78
4                10.1                3               2.73

   Customer service calls  Churn  Total charge
0                       1  False         75.56
1                       1  False         59.24
2                       0  False         62.29
3                       2  False         66.80
4                       3  False         52.09

[5 rows x 21 columns]
```

[ ]:
```python
# get rid of just created columns
df.drop(["Total charge", "Total calls"], axis=1, inplace=True)
# and here's how you can delete rows
df.drop([1, 2]).head()
```

[ ]:
```
  State  Account length  Area code International plan Voice mail plan  \
0    KS             128        415                 No             Yes
3    OH              84        408                Yes              No
4    OK              75        415                Yes              No
5    AL             118        510                Yes              No
6    MA             121        510                 No             Yes

   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
3                      0              299.4               71
4                      0              166.7              113
5                      0              223.4               98
6                     24              218.2               88

   Total day charge  Total eve minutes  Total eve calls  Total eve charge  \
0             45.07              197.4               99             16.78
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61
5             37.98              220.6              101             18.75
6             37.09              348.5              108             29.62

   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01
3                196.9                 89                8.86
4                186.9                121                8.41
5                203.9                118                9.18
6                212.6                118                9.57
```

```
   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70   
3                 6.6                 7               1.78   
4                10.1                 3               2.73   
5                 6.3                 6               1.70   
6                 7.5                 7               2.03   

   Customer service calls  Churn  
0                       1  False  
3                       2  False  
4                       3  False  
5                       0  False  
6                       3  False  
```

```
[ ]: # First attempt at predicting telecom churn
```

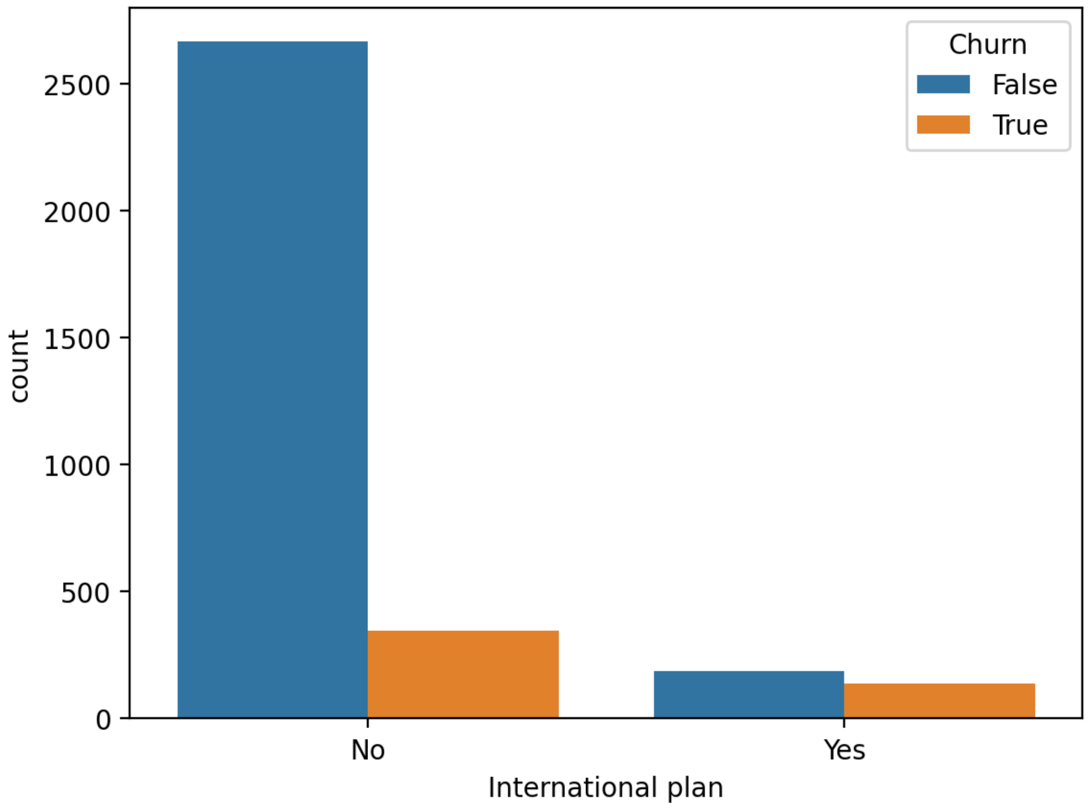
```
[ ]: pd.crosstab(df["Churn"], df["International plan"], margins=True)
```

```
[ ]: International plan    No  Yes   All
     Churn
     False               2664  186  2850
     True                 346  137   483
     All                 3010  323  3333
```

```
[ ]: # some import to set up plotting
     import matplotlib.pyplot as plt
     # pip install seaborn
     import seaborn as sns

     # Graphics in retina format arae more sharp and legible
     %config InlineBackend.figure_format = 'retina'
```

```
[ ]: sns.countplot(x="International plan", hue="Churn", data=df)
```

```
[ ]: <Axes: xlabel='International plan', ylabel='count'>
```



```
[ ]: ####
     df["International plan"], df["Churn"]
     ####
```

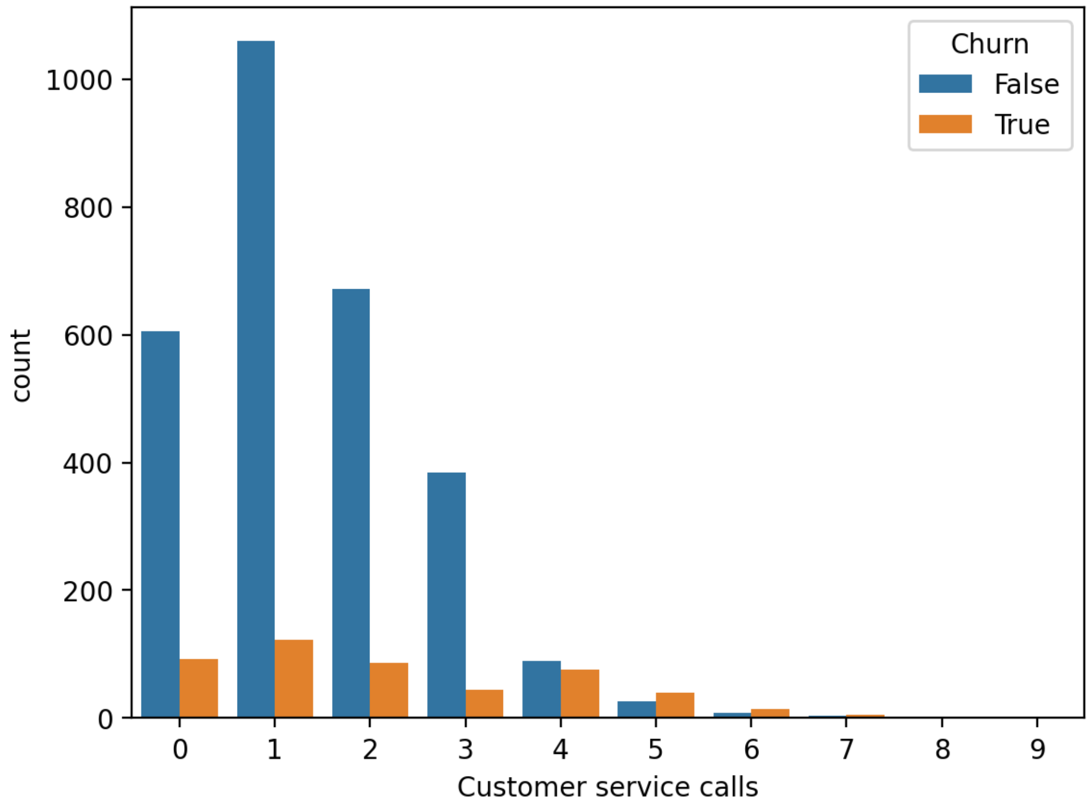
```
[ ]: (0         No
      1         No
      2         No
      3        Yes
      4        Yes
              …
      3328      No
      3329      No
      3330      No
      3331     Yes
      3332      No
      Name: International plan, Length: 3333, dtype: object,
      0        False
      1        False
      2        False
      3        False
      4        False
```

```
                 …
3328    False
3329    False
3330    False
3331    False
3332    False
Name: Churn, Length: 3333, dtype: bool)
```

[ ]:
```
pd.crosstab(df["Churn"], df["Customer service calls"], margins=True)
```

[ ]:
```
Customer service calls    0     1    2    3    4   5   6  7  8  9   All
Churn
False                   605  1059  672  385   90  26   8  4  1  0  2850
True                     92   122   87   44   76  40  14  5  1  2   483
All                     697  1181  759  429  166  66  22  9  2  2  3333
```

[ ]:
```
sns.countplot(x="Customer service calls", hue="Churn", data=df);
```



[ ]:
```
df["Many_service_calls"] = (df["Customer service calls"] > 3).astype("int")

pd.crosstab(df["Many_service_calls"], df["Churn"], margins=True)
```

```
[ ]: Churn               False  True   All
     Many_service_calls
     0                    2721   345  3066
     1                     129   138   267
     All                  2850   483  3333
```

```
[ ]: ####
     print(df["Many_service_calls"].unique(), "Unique values in Many_service_calls␣
      ↪column/series"), print(df["Churn"].unique(), "Unique values in Churn column/
      ↪series")
     ####
```

```
[0 1] Unique values in Many_service_calls column/series
[False  True] Unique values in Churn column/series
```

```
[ ]: (None, None)
```

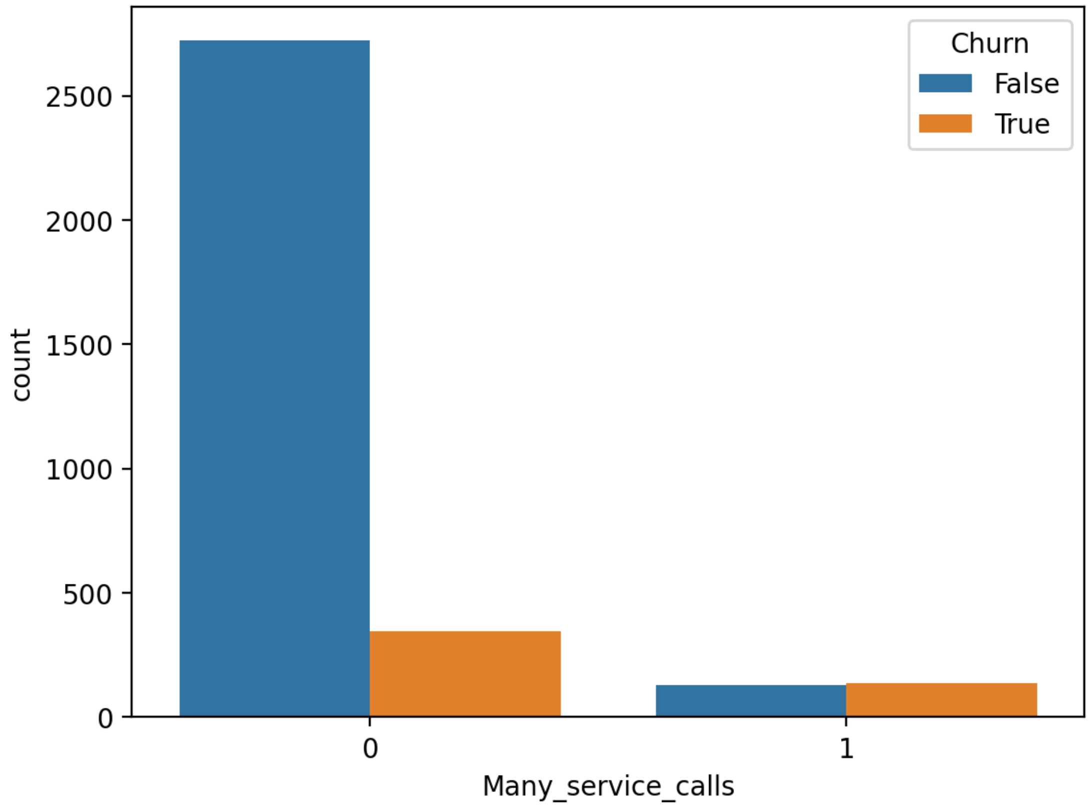
```
[ ]: ####
     df["Customer service calls"] > 3
     ####
```

```
[ ]: 0       False
     1       False
     2       False
     3       False
     4       False
             …
     3328    False
     3329    False
     3330    False
     3331    False
     3332    False
     Name: Customer service calls, Length: 3333, dtype: bool
```

```
[ ]: ####
     df["Many_service_calls"].info()
     ####
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3333 entries, 0 to 3332
Series name: Many_service_calls
Non-Null Count  Dtype
--------------  -----
3333 non-null   int64
dtypes: int64(1)
memory usage: 26.2 KB
```

```
[ ]: sns.countplot(x="Many_service_calls", hue="Churn", data=df);
```



```
[ ]: #pd.crosstab(df["Many_service_calls"] & df["International plan"], df["Churn"])
```

```
[ ]: #### !!!! Exploratory Data Analysis Amazon Sales Dataset EDA !!!! ####
     #https://www.kaggle.com/code/mehakiftikhar/amazon-sales-dataset-eda
```

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import scipy as sp

     %matplotlib inline
     # to display the plots inside jupyter-notebook
```

```
[ ]: # Data Loading and Exploration | Cleaning
```

```
[ ]: import kagglehub
[ ]: path = kagglehub.dataset_download("karkavelrajaj/amazon-sales-dataset")
     print(path)
```

```
Using Colab cache for faster access to the 'amazon-sales-dataset' dataset.
/kaggle/input/amazon-sales-dataset
```

```
[ ]: import os
```

```
[ ]: # load a csv file then creating a dataframe
     df = pd.read_csv(os.path.join(path, "amazon.csv"))
     df.head()
```

```
[ ]:   product_id                                       product_name  \
     0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha…
     1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5…
     2  B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U…
     3  B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…
     4  B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P…

                                                 category discounted_price  \
     0  Computers&Accessories|Accessories&Peripherals|…             399
     1  Computers&Accessories|Accessories&Peripherals|…             199
     2  Computers&Accessories|Accessories&Peripherals|…             199
     3  Computers&Accessories|Accessories&Peripherals|…             329
     4  Computers&Accessories|Accessories&Peripherals|…             154

       actual_price discount_percentage rating rating_count  \
     0       1,099                 64%    4.2       24,269
     1         349                 43%    4.0       43,994
     2       1,899                 90%    3.9        7,928
     3         699                 53%    4.2       94,363
     4         399                 61%    4.2       16,905

                                            about_product  \
     0  High Compatibility : Compatible With iPhone 12…
     1  Compatible with all Type C enabled devices, be…
     2   Fast Charger& Data Sync -With built-in safet…
     3  The boAt Deuce USB 300 2 in 1 cable is compati…
     4  [CHARGE & SYNC FUNCTION]- This cable comes wit…

                                                  user_id  \
     0  AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB…
     1  AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX…
     2  AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ…
     3  AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S…
     4  AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH…
```

```
                                           user_name  \
0  Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp…
1  ArdKn,Nirbhay kumar,Sagar Viswanathan,Asp,Plac…
2  Kunal,Himanshu,viswanath,sai niharka,saqib mal…
3  Omkar dhale,JD,HEMALATHA,Ajwadh a.,amar singh …
4  rahuls6099,Swasat Borah,Ajay Wadke,Pranali,RVK…

                                           review_id  \
0  R3HXWT0LRP0NMF,R2AJM3LFTLZHFO,R6AQJGUP6P86,R1K…
1  RGIQEG07R9HS2,R1SMWZQ86XIN8U,R2J3Y1WL29GWDE,RY…
2  R3J3EQQ9TZI5ZJ,R3E7WBGK7ID0KV,RWU79XKQ6I1QF,R2…
3  R3EEUZKKK9J36I,R3HJVYCLYOY554,REDECAZ7AMPQC,R1…
4  R1BP4L2HH9TFUP,R16PVJEXKV6QZS,R2UPDB81N66T4P,R…

                                        review_title  \
0  Satisfied,Charging is really fast,Value for mo…
1  A Good Braided Cable for Your Type C Device,Go…
2  Good speed for earlier versions,Good Product,W…
3  Good product,Good one,Nice,Really nice product…
4  As good as original,Decent,Good one for second…

                                      review_content  \
0  Looks durable Charging is fine tooNo complains…
1  I ordered this cable to connect my phone to An…
2  Not quite durable and sturdy,https://m.media-a…
3  Good product,long wire,Charges good,Nice,I bou…
4  Bought this instead of original apple, does th…

                                            img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237…
1  https://m.media-amazon.com/images/W/WEBP_40237…
2  https://m.media-amazon.com/images/W/WEBP_40237…
3  https://m.media-amazon.com/images/I/41V5FtEWPk…
4  https://m.media-amazon.com/images/W/WEBP_40237…

                                        product_link
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy…
1  https://www.amazon.in/Ambrane-Unbreakable-Char…
2  https://www.amazon.in/Sounce-iPhone-Charging-C…
3  https://www.amazon.in/Deuce-300-Resistant-Tang…
4  https://www.amazon.in/Portronics-Konnect-POR-1…
```

[ ]:
```python
# setting the option to show maximum columns
pd.set_option('display.max_columns', None)
```

[ ]:
```python
####
help(pd.set_option)
```

```
####
```

```
Help on CallableDynamicDoc in module pandas._config.config:

<pandas._config.config.CallableDynamicDoc object>
    set_option(pat, value)

    Sets the value of the specified option.

    Available options:

    - compute.[use_bottleneck, use_numba, use_numexpr]
    - display.[chop_threshold, colheader_justify, date_dayfirst, date_yearfirst,
      encoding, expand_frame_repr, float_format]
    - display.html.[border, table_schema, use_mathjax]
    - display.[large_repr, max_categories, max_columns, max_colwidth,
max_dir_items,
      max_info_columns, max_info_rows, max_rows, max_seq_items, memory_usage,
      min_rows, multi_sparse, notebook_repr_html, pprint_nest_depth, precision,
      show_dimensions]
    - display.unicode.[ambiguous_as_wide, east_asian_width]
    - display.[width]
    - future.[infer_string, no_silent_downcasting]
    - io.excel.ods.[reader, writer]
    - io.excel.xls.[reader]
    - io.excel.xlsb.[reader]
    - io.excel.xlsm.[reader, writer]
    - io.excel.xlsx.[reader, writer]
    - io.hdf.[default_format, dropna_table]
    - io.parquet.[engine]
    - io.sql.[engine]
    - mode.[chained_assignment, copy_on_write, data_manager, sim_interactive,
      string_storage, use_inf_as_na]
    - plotting.[backend]
    - plotting.matplotlib.[register_converters]
    - styler.format.[decimal, escape, formatter, na_rep, precision, thousands]
    - styler.html.[mathjax]
    - styler.latex.[environment, hrules, multicol_align, multirow_align]
    - styler.render.[encoding, max_columns, max_elements, max_rows, repr]
    - styler.sparse.[columns, index]

    Parameters
    ----------
    pat : str
        Regexp which should match a single option.
        Note: partial matches are supported for convenience, but unless you use
the
```

```
        full option name (e.g. x.y.z.option_name), your code may break in future
        versions if new options with similar names are introduced.
    value : object
        New value of option.

    Returns
    -------
    None

    Raises
    ------
    OptionError if no such option exists

    Notes
    -----
    Please reference the :ref:`User Guide <options>` for more information.

    The available options with its descriptions:

    compute.use_bottleneck : bool
        Use the bottleneck library to accelerate if it is installed,
        the default is True
        Valid values: False,True
        [default: True] [currently: True]
    compute.use_numba : bool
        Use the numba engine option for select operations if it is installed,
        the default is False
        Valid values: False,True
        [default: False] [currently: False]
    compute.use_numexpr : bool
        Use the numexpr library to accelerate computation if it is installed,
        the default is True
        Valid values: False,True
        [default: True] [currently: True]
    display.chop_threshold : float or None
        if set to a float value, all float values smaller than the given
threshold
        will be displayed as exactly 0 by repr and friends.
        [default: None] [currently: None]
    display.colheader_justify : 'left'/'right'
        Controls the justification of column headers. used by
DataFrameFormatter.
        [default: right] [currently: right]
    display.date_dayfirst : boolean
        When True, prints and parses dates with the day first, eg 20/01/2005
        [default: False] [currently: False]
    display.date_yearfirst : boolean
        When True, prints and parses dates with the year first, eg 2005/01/20
        [default: False] [currently: False]
    display.encoding : str/unicode
        Defaults to the detected encoding of the console.
        Specifies the encoding to be used for strings returned by to_string,
        these are generally strings meant to be displayed on the console.
        [default: UTF-8] [currently: UTF-8]
    display.expand_frame_repr : boolean
        Whether to print out the full DataFrame repr for wide DataFrames across
        multiple lines, `max_columns` is still respected, but the output will
        wrap-around across multiple "pages" if its width exceeds
`display.width`.
        [default: True] [currently: True]
    display.float_format : callable
        The callable should accept a floating point number and return
        a string with the desired format of the number. This is used
        in some places like SeriesFormatter.
        See formats.format.EngFormatter for an example.
        [default: None] [currently: None]
    display.html.border : int
        A ``border=value`` attribute is inserted in the ``<table>`` tag
        for the DataFrame HTML repr.
        [default: 1] [currently: 1]
    display.html.table_schema : boolean
        Whether to publish a Table Schema representation for frontends
        that support it.
        (default: False)
        [default: False] [currently: False]
    display.html.use_mathjax : boolean
        When True, Jupyter notebook will process table contents using MathJax,
        rendering mathematical expressions enclosed by the dollar symbol.
        (default: True)
        [default: True] [currently: True]
    display.large_repr : 'truncate'/'info'
        For DataFrames exceeding max_rows/max_cols, the repr (and HTML repr) can
        show a truncated table, or switch to the view from
        df.info() (the behaviour in earlier versions of pandas).
        [default: truncate] [currently: truncate]
    display.max_categories : int
        This sets the maximum number of categories pandas should output when
        printing out a `Categorical` or a Series of dtype "category".
        [default: 8] [currently: 8]
    display.max_columns : int
        If max_cols is exceeded, switch to truncate view. Depending on
        `large_repr`, objects are either centrally truncated or printed as
        a summary view. 'None' value means unlimited.

        In case python/IPython is running in a terminal and `large_repr`
        equals 'truncate' this can be set to 0 or None and pandas will auto-
```

```
detect
        the width of the terminal and print a truncated object which fits
        the screen width. The IPython notebook, IPython qtconsole, or IDLE
        do not run in a terminal and hence it is not possible to do
        correct auto-detection and defaults to 20.
        [default: 20] [currently: 20]
    display.max_colwidth : int or None
        The maximum width in characters of a column in the repr of
        a pandas data structure. When the column overflows, a "…"
        placeholder is embedded in the output. A 'None' value means unlimited.
        [default: 50] [currently: 50]
    display.max_dir_items : int
        The number of items that will be added to `dir(…)`. 'None' value means
        unlimited. Because dir is cached, changing this option will not
immediately
        affect already existing dataframes until a column is deleted or added.

        This is for instance used to suggest columns from a dataframe to tab
        completion.
        [default: 100] [currently: 100]
    display.max_info_columns : int
        max_info_columns is used in DataFrame.info method to decide if
        per column information will be printed.
        [default: 100] [currently: 100]
    display.max_info_rows : int
        df.info() will usually show null-counts for each column.
        For large frames this can be quite slow. max_info_rows and max_info_cols
        limit this null check only to frames with smaller dimensions than
        specified.
        [default: 1690785] [currently: 1690785]
    display.max_rows : int
        If max_rows is exceeded, switch to truncate view. Depending on
        `large_repr`, objects are either centrally truncated or printed as
        a summary view. 'None' value means unlimited.

        In case python/IPython is running in a terminal and `large_repr`
        equals 'truncate' this can be set to 0 and pandas will auto-detect
        the height of the terminal and print a truncated object which fits
        the screen height. The IPython notebook, IPython qtconsole, or
        IDLE do not run in a terminal and hence it is not possible to do
        correct auto-detection.
        [default: 60] [currently: 60]
    display.max_seq_items : int or None
        When pretty-printing a long sequence, no more then `max_seq_items`
        will be printed. If items are omitted, they will be denoted by the
        addition of "…" to the resulting string.

        If set to None, the number of items to be printed is unlimited.
        [default: 100] [currently: 100]
    display.memory_usage : bool, string or None
        This specifies if the memory usage of a DataFrame should be displayed
when
        df.info() is called. Valid values True,False,'deep'
        [default: True] [currently: True]
    display.min_rows : int
        The numbers of rows to show in a truncated view (when `max_rows` is
        exceeded). Ignored when `max_rows` is set to None or 0. When set to
        None, follows the value of `max_rows`.
        [default: 10] [currently: 10]
    display.multi_sparse : boolean
        "sparsify" MultiIndex display (don't display repeated
        elements in outer levels within groups)
        [default: True] [currently: True]
    display.notebook_repr_html : boolean
        When True, IPython notebook will use html representation for
        pandas objects (if it is available).
        [default: True] [currently: True]
    display.pprint_nest_depth : int
        Controls the number of nested levels to process when pretty-printing
        [default: 3] [currently: 3]
    display.precision : int
        Floating point output precision in terms of number of places after the
        decimal, for regular formatting as well as scientific notation. Similar
        to ``precision`` in :meth:`numpy.set_printoptions`.
        [default: 6] [currently: 2]
    display.show_dimensions : boolean or 'truncate'
        Whether to print out dimensions at the end of DataFrame repr.
        If 'truncate' is specified, only print out the dimensions if the
        frame is truncated (e.g. not display all rows and/or columns)
        [default: truncate] [currently: truncate]
    display.unicode.ambiguous_as_wide : boolean
        Whether to use the Unicode East Asian Width to calculate the display
text
        width.
        Enabling this may affect to the performance (default: False)
        [default: False] [currently: False]
    display.unicode.east_asian_width : boolean
        Whether to use the Unicode East Asian Width to calculate the display
text
        width.
        Enabling this may affect to the performance (default: False)
        [default: False] [currently: False]
    display.width : int
        Width of the display in characters. In case python/IPython is running in
        a terminal this can be set to None and pandas will correctly auto-detect
        the width.
        Note that the IPython notebook, IPython qtconsole, or IDLE do not run in
a
        terminal and hence it is not possible to correctly detect the width.
        [default: 80] [currently: 80]
    future.infer_string Whether to infer sequence of str objects as pyarrow
string dtype, which will be the default in pandas 3.0 (at which point this
option will be deprecated).
        [default: False] [currently: False]
    future.no_silent_downcasting Whether to opt-in to the future behavior which
will *not* silently downcast results from Series and DataFrame `where`, `mask`,
and `clip` methods. Silent downcasting will be removed in pandas 3.0 (at which
point this option will be deprecated).
        [default: False] [currently: False]
    io.excel.ods.reader : string
        The default Excel reader engine for 'ods' files. Available options:
        auto, odf, calamine.
        [default: auto] [currently: auto]
    io.excel.ods.writer : string
        The default Excel writer engine for 'ods' files. Available options:
        auto, odf.
        [default: auto] [currently: auto]
    io.excel.xls.reader : string
        The default Excel reader engine for 'xls' files. Available options:
        auto, xlrd, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsb.reader : string
        The default Excel reader engine for 'xlsb' files. Available options:
        auto, pyxlsb, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsm.reader : string
        The default Excel reader engine for 'xlsm' files. Available options:
        auto, xlrd, openpyxl, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsm.writer : string
        The default Excel writer engine for 'xlsm' files. Available options:
        auto, openpyxl.
        [default: auto] [currently: auto]
    io.excel.xlsx.reader : string
        The default Excel reader engine for 'xlsx' files. Available options:
        auto, xlrd, openpyxl, calamine.
        [default: auto] [currently: auto]
    io.excel.xlsx.writer : string
        The default Excel writer engine for 'xlsx' files. Available options:
        auto, openpyxl, xlsxwriter.
        [default: auto] [currently: auto]
    io.hdf.default_format : format
        default format writing format, if None, then
        put will default to 'fixed' and append will default to 'table'
        [default: None] [currently: None]
    io.hdf.dropna_table : boolean
        drop ALL nan rows when appending to a table
        [default: False] [currently: False]
    io.parquet.engine : string
        The default parquet reader/writer engine. Available options:
        'auto', 'pyarrow', 'fastparquet', the default is 'auto'
        [default: auto] [currently: auto]
    io.sql.engine : string
        The default sql reader/writer engine. Available options:
        'auto', 'sqlalchemy', the default is 'auto'
        [default: auto] [currently: auto]
    mode.chained_assignment : string
        Raise an exception, warn, or no action if trying to use chained
assignment,
        The default is warn
        [default: warn] [currently: warn]
    mode.copy_on_write : bool
        Use new copy-view behaviour using Copy-on-Write. Defaults to False,
        unless overridden by the 'PANDAS_COPY_ON_WRITE' environment variable
        (if set to "1" for True, needs to be set before pandas is imported).
        [default: False] [currently: False]
    mode.data_manager : string
        Internal data manager type; can be "block" or "array". Defaults to
"block",
        unless overridden by the 'PANDAS_DATA_MANAGER' environment variable
(needs
        to be set before pandas is imported).
        [default: block] [currently: block]
        (Deprecated, use `` instead.)
    mode.sim_interactive : boolean
        Whether to simulate interactive mode for purposes of testing
        [default: False] [currently: False]
    mode.string_storage : string
        The default storage for StringDtype. This option is ignored if
        ``future.infer_string`` is set to True.
        [default: python] [currently: python]
    mode.use_inf_as_na : boolean
        True means treat None, NaN, INF, -INF as NA (old way),
        False means None and NaN are null, but INF, -INF are not NA
        (new way).

        This option is deprecated in pandas 2.1.0 and will be removed in 3.0.
        [default: False] [currently: False]
        (Deprecated, use `` instead.)
    plotting.backend : str
        The plotting backend to use. The default value is "matplotlib", the
        backend provided with pandas. Other backends can be specified by
        providing the name of the module that implements the backend.
        [default: matplotlib] [currently: matplotlib]
    plotting.matplotlib.register_converters : bool or 'auto'.
        Whether to register converters with matplotlib's units registry for
        dates, times, datetimes, and Periods. Toggling to False will remove
        the converters, restoring any converters that pandas overwrote.
        [default: auto] [currently: auto]
    styler.format.decimal : str
        The character representation for the decimal separator for floats and
complex.
        [default: .] [currently: .]
    styler.format.escape : str, optional
        Whether to escape certain characters according to the given context;
html or latex.
        [default: None] [currently: None]
    styler.format.formatter : str, callable, dict, optional
        A formatter object to be used as default within ``Styler.format``.
        [default: None] [currently: None]
    styler.format.na_rep : str, optional
        The string representation for values identified as missing.
        [default: None] [currently: None]
    styler.format.precision : int
        The precision for floats and complex numbers.
        [default: 6] [currently: 6]
    styler.format.thousands : str, optional
        The character representation for thousands separator for floats, int and
complex.
        [default: None] [currently: None]
    styler.html.mathjax : bool
        If False will render special CSS classes to table attributes that
indicate Mathjax
        will not be used in Jupyter Notebook.
        [default: True] [currently: True]
    styler.latex.environment : str
        The environment to replace ``\begin{table}``. If "longtable" is used
results
        in a specific longtable environment format.
        [default: None] [currently: None]
    styler.latex.hrules : bool
        Whether to add horizontal rules on top and bottom and below the headers.
        [default: False] [currently: False]
    styler.latex.multicol_align : {"r", "c", "l", "naive-l", "naive-r"}
        The specifier for horizontal alignment of sparsified LaTeX multicolumns.
Pipe
        decorators can also be added to non-naive values to draw vertical
        rules, e.g. "\|r" will draw a rule on the left side of right aligned
merged cells.
        [default: r] [currently: r]
    styler.latex.multirow_align : {"c", "t", "b"}
        The specifier for vertical alignment of sparsified LaTeX multirows.
        [default: c] [currently: c]
    styler.render.encoding : str
        The encoding used for output HTML and LaTeX files.
        [default: utf-8] [currently: utf-8]
    styler.render.max_columns : int, optional
        The maximum number of columns that will be rendered. May still be
reduced to
        satisfy ``max_elements``, which takes precedence.
        [default: None] [currently: None]
    styler.render.max_elements : int
        The maximum number of data-cell (<td>) elements that will be rendered
before
        trimming will occur over columns, rows or both if needed.
        [default: 262144] [currently: 262144]
    styler.render.max_rows : int, optional
        The maximum number of rows that will be rendered. May still be reduced
to
        satisfy ``max_elements``, which takes precedence.
        [default: None] [currently: None]
    styler.render.repr : str
        Determine which output to use in Jupyter Notebook in {"html", "latex"}.
        [default: html] [currently: html]
    styler.sparse.columns : bool
        Whether to sparsify the display of hierarchical columns. Setting to
False will
        display each explicit level element in a hierarchical key for each
column.
        [default: True] [currently: True]
    styler.sparse.index : bool
        Whether to sparsify the display of a hierarchical index. Setting to
False will
        display each explicit level element in a hierarchical key for each row.
        [default: True] [currently: True]

    Examples
    --------
    >>> pd.set_option('display.max_columns', 4)
    >>> df = pd.DataFrame([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
    >>> df
       0  1  …  3   4
    0  1  2  …  4   5
    1  6  7  …  9  10
    [2 rows x 5 columns]
    >>> pd.reset_option('display.max_columns')
```

```
[ ]: df.head(5)
```

```
[ ]:   product_id                                       product_name  \
    0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha…
    1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5…
    2  B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U…
    3  B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S…
    4  B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P…

                                                category discounted_price  \
    0  Computers&Accessories|Accessories&Peripherals|…              399
    1  Computers&Accessories|Accessories&Peripherals|…              199
    2  Computers&Accessories|Accessories&Peripherals|…              199
    3  Computers&Accessories|Accessories&Peripherals|…              329
    4  Computers&Accessories|Accessories&Peripherals|…              154

      actual_price discount_percentage rating rating_count  \
    0        1,099                 64%    4.2       24,269
    1          349                 43%    4.0       43,994
    2        1,899                 90%    3.9        7,928
    3          699                 53%    4.2       94,363
    4          399                 61%    4.2       16,905

                                           about_product  \
    0  High Compatibility : Compatible With iPhone 12…
    1  Compatible with all Type C enabled devices, be…
    2   Fast Charger& Data Sync -With built-in safet…
    3  The boAt Deuce USB 300 2 in 1 cable is compati…
    4  [CHARGE & SYNC FUNCTION]- This cable comes wit…

                                                 user_id  \
    0  AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB…
    1  AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX…
    2  AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ…
    3  AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S…
    4  AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH…

                                               user_name  \
    0  Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp…
    1  ArdKn,Nirbhay kumar,Sagar Viswanathan,Asp,Plac…
    2  Kunal,Himanshu,viswanath,sai niharka,saqib mal…
    3  Omkar dhale,JD,HEMALATHA,Ajwadh a.,amar singh …
    4  rahuls6099,Swasat Borah,Ajay Wadke,Pranali,RVK…

                                               review_id  \
    0  R3HXWT0LRP0NMF,R2AJM3LFTLZHFO,R6AQJGUP6P86,R1K…
    1  RGIQEG07R9HS2,R1SMWZQ86XIN8U,R2J3Y1WL29GWDE,RY…
```

```
2  R3J3EQQ9TZI5ZJ,R3E7WBGK7IDOKV,RWU79XKQ6I1QF,R2…
3  R3EEUZKKK9J36I,R3HJVYCLYOY554,REDECAZ7AMPQC,R1…
4  R1BP4L2HH9TFUP,R16PVJEXKV6QZS,R2UPDB81N66T4P,R…

                                        review_title  \
0  Satisfied,Charging is really fast,Value for mo…
1  A Good Braided Cable for Your Type C Device,Go…
2  Good speed for earlier versions,Good Product,W…
3  Good product,Good one,Nice,Really nice product…
4  As good as original,Decent,Good one for second…

                                      review_content  \
0  Looks durable Charging is fine tooNo complains…
1  I ordered this cable to connect my phone to An…
2  Not quite durable and sturdy,https://m.media-a…
3  Good product,long wire,Charges good,Nice,I bou…
4  Bought this instead of original apple, does th…

                                            img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237…
1  https://m.media-amazon.com/images/W/WEBP_40237…
2  https://m.media-amazon.com/images/W/WEBP_40237…
3  https://m.media-amazon.com/images/I/41V5FtEWPk…
4  https://m.media-amazon.com/images/W/WEBP_40237…

                                        product_link
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy…
1  https://www.amazon.in/Ambrane-Unbreakable-Char…
2  https://www.amazon.in/Sounce-iPhone-Charging-C…
3  https://www.amazon.in/Deuce-300-Resistant-Tang…
4  https://www.amazon.in/Portronics-Konnect-POR-1…
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['product_id', 'product_name', 'category', 'discounted_price',
       'actual_price', 'discount_percentage', 'rating', 'rating_count',
       'about_product', 'user_id', 'user_name', 'review_id', 'review_title',
       'review_content', 'img_link', 'product_link'],
      dtype='object')
```

[ ]:
```
# Let's have a look on the shape of the dataset
```

[ ]:
```
print(f"The Number of Rows are {df.shape[0]}, and columns are {df.shape[1]}.")
```

```
The Number of Rows are 1465, and columns are 16.
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.3+ KB
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
product_id             0
product_name           0
category               0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           2
about_product          0
user_id                0
user_name              0
review_id              0
review_title           0
review_content         0
img_link               0
product_link           0
dtype: int64
```

[ ]:
```
####
df.isnull()
####
```

```
[ ]:      product_id  product_name  category  discounted_price  actual_price  \
    0          False         False     False             False         False
    1          False         False     False             False         False
    2          False         False     False             False         False
    3          False         False     False             False         False
    4          False         False     False             False         False
    …            …             …         …                 …             …
    1460       False         False     False             False         False
    1461       False         False     False             False         False
    1462       False         False     False             False         False
    1463       False         False     False             False         False
    1464       False         False     False             False         False

          discount_percentage  rating  rating_count  about_product  user_id  \
    0                   False   False         False          False    False
    1                   False   False         False          False    False
    2                   False   False         False          False    False
    3                   False   False         False          False    False
    4                   False   False         False          False    False
    …                     …       …             …              …        …
    1460                False   False         False          False    False
    1461                False   False         False          False    False
    1462                False   False         False          False    False
    1463                False   False         False          False    False
    1464                False   False         False          False    False

          user_name  review_id  review_title  review_content  img_link  \
    0         False      False         False           False     False
    1         False      False         False           False     False
    2         False      False         False           False     False
    3         False      False         False           False     False
    4         False      False         False           False     False
    …           …          …             …               …         …
    1460      False      False         False           False     False
    1461      False      False         False           False     False
    1462      False      False         False           False     False
    1463      False      False         False           False     False
    1464      False      False         False           False     False

          product_link
    0            False
    1            False
    2            False
    3            False
    4            False
    …              …
    1460         False
```

```
1461         False
1462         False
1463         False
1464         False

[1465 rows x 16 columns]
```

```
[ ]: # Changing Data Types of Columns from object to flaot

     df['discounted_price'] = df['discounted_price'].str.replace(' ', '')
     df['discounted_price'] = df['discounted_price'].str.replace(',', '')
     df['discounted_price'] = df['discounted_price'].astype('float64')

     df['actual_price'] = df['actual_price'].str.replace(' ', '')
     df['actual_price'] = df['actual_price'].str.replace(',', '')
     df['actual_price'] = df['actual_price'].astype('float64')
```

```
[ ]: ####
     df['discounted_price']
     ####
```

```
[ ]: 0         399
     1         199
     2         199
     3         329
     4         154
             …
     1460      379
     1461    2,280
     1462    2,219
     1463    1,399
     1464    2,863
     Name: discounted_price, Length: 1465, dtype: object
```

```
[ ]:
```

```
[ ]: ####
     help(df['discounted_price'].str)
     ####
```

```
Help on StringMethods in module pandas.core.strings.accessor object:

class StringMethods(pandas.core.base.NoNewAttributesMixin)
 |  StringMethods(data) -> 'None'
 |
 |  Vectorized string functions for Series and Index.
 |
 |  NAs stay NA unless handled otherwise by a particular method.
 |  Patterned after Python's string methods, with some inspiration from
 |  R's stringr package.
 |
 |  Examples
 |  --------
 |  >>> s = pd.Series(["A_Str_Series"])
 |  >>> s
 |  0    A_Str_Series
 |  dtype: object
 |
 |  >>> s.str.split("_")
 |  0    [A, Str, Series]
 |  dtype: object
 |
 |  >>> s.str.replace("_", "")
 |  0    AStrSeries
 |  dtype: object
 |
 |  Method resolution order:
 |      StringMethods
 |      pandas.core.base.NoNewAttributesMixin
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __getitem__(self, key)
 |
 |  __init__(self, data) -> 'None'
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self) -> 'Iterator'
 |
 |  capitalize(self)
 |      Convert strings in the Series/Index to be capitalized.
 |
 |      Equivalent to :meth:`str.capitalize`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
```

```
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1              CAPITALS
 |      2    this is a sentence
 |      3              SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                 lower
 |      1              capitals
 |      2    this is a sentence
 |      3              swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                 LOWER
 |      1              CAPITALS
 |      2    THIS IS A SENTENCE
 |      3              SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                 Lower
 |      1              Capitals
 |      2    This Is A Sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                 Lower
 |      1              Capitals
 |      2    This is a sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                 LOWER
 |      1              capitals
 |      2    THIS IS A SENTENCE
 |      3              sWaPcAsE
 |      dtype: object
 |
 |  casefold(self)
 |      Convert strings in the Series/Index to be casefolded.
 |
 |      Equivalent to :meth:`str.casefold`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1              CAPITALS
 |      2    this is a sentence
 |      3              SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                 lower
 |      1              capitals
 |      2    this is a sentence
 |      3              swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                 LOWER
 |      1              CAPITALS
 |      2    THIS IS A SENTENCE
 |      3              SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                 Lower
 |      1              Capitals
 |      2    This Is A Sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                 Lower
 |      1              Capitals
 |      2    This is a sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                 LOWER
 |      1              capitals
 |      2    THIS IS A SENTENCE
 |      3              sWaPcAsE
 |      dtype: object
 |
 |  cat(self, others=None, sep: 'str | None' = None, na_rep=None, join:
'AlignJoin' = 'left') -> 'str | Series | Index'
 |      Concatenate strings in the Series/Index with given separator.
 |
 |      If `others` is specified, this function concatenates the Series/Index
 |      and elements of `others` element-wise.
 |      If `others` is not passed, then all values in the Series/Index are
 |      concatenated into a single string with a given `sep`.
 |
 |      Parameters
 |      ----------
 |      others : Series, Index, DataFrame, np.ndarray or list-like
 |          Series, Index, DataFrame, np.ndarray (one- or two-dimensional) and
 |          other list-likes of strings must have the same length as the
 |          calling Series/Index, with the exception of indexed objects (i.e.
 |          Series/Index/DataFrame) if `join` is not None.
 |
 |          If others is a list-like that contains a combination of Series,
 |          Index or np.ndarray (1-dim), then all elements will be unpacked and
 |          must satisfy the above criteria individually.
 |
 |          If others is None, the method returns the concatenation of all
 |          strings in the calling Series/Index.
 |      sep : str, default ''
 |          The separator between the different elements/columns. By default
 |          the empty string `''` is used.
 |      na_rep : str or None, default None
 |          Representation that is inserted for all missing values:
 |
 |          - If `na_rep` is None, and `others` is None, missing values in the
 |            Series/Index are omitted from the result.
 |          - If `na_rep` is None, and `others` is not None, a row containing a
 |            missing value in any of the columns (before concatenation) will
 |            have a missing value in the result.
 |      join : {'left', 'right', 'outer', 'inner'}, default 'left'
 |          Determines the join-style between the calling Series/Index and any
 |          Series/Index/DataFrame in `others` (objects without an index need
 |          to match the length of the calling Series/Index). To disable
 |          alignment, use `.values` on any Series/Index/DataFrame in `others`.
 |
 |      Returns
 |      -------
 |      str, Series or Index
 |          If `others` is None, `str` is returned, otherwise a `Series/Index`
 |          (same type as caller) of objects is returned.
 |
 |      See Also
 |      --------
 |      split : Split each string in the Series/Index.
 |      join : Join lists contained as elements in the Series/Index.
 |
 |      Examples
 |      --------
 |      When not passing `others`, all values are concatenated into a single
 |      string:
 |
 |      >>> s = pd.Series(['a', 'b', np.nan, 'd'])
 |      >>> s.str.cat(sep=' ')
 |      'a b d'
 |
 |      By default, NA values in the Series are ignored. Using `na_rep`, they
 |      can be given a representation:
 |
 |      >>> s.str.cat(sep=' ', na_rep='?')
 |      'a b ? d'
 |
 |      If `others` is specified, corresponding values are concatenated with
 |      the separator. Result will be a Series of strings.
 |
 |      >>> s.str.cat(['A', 'B', 'C', 'D'], sep=',')
 |      0    a,A
```

```
|      1    b,B
|      2    NaN
|      3    d,D
|      dtype: object
|
|      Missing values will remain missing in the result, but can again be
|      represented using `na_rep`
|
|      >>> s.str.cat(['A', 'B', 'C', 'D'], sep=',', na_rep='-')
|      0    a,A
|      1    b,B
|      2    -,C
|      3    d,D
|      dtype: object
|
|      If `sep` is not specified, the values are concatenated without
|      separation.
|
|      >>> s.str.cat(['A', 'B', 'C', 'D'], na_rep='-')
|      0    aA
|      1    bB
|      2    -C
|      3    dD
|      dtype: object
|
|      Series with different indexes can be aligned before concatenation. The
|      `join`-keyword works as in other methods.
|
|      >>> t = pd.Series(['d', 'a', 'e', 'c'], index=[3, 0, 4, 2])
|      >>> s.str.cat(t, join='left', na_rep='-')
|      0    aa
|      1    b-
|      2    -c
|      3    dd
|      dtype: object
|      >>>
|      >>> s.str.cat(t, join='outer', na_rep='-')
|      0    aa
|      1    b-
|      2    -c
|      3    dd
|      4    -e
|      dtype: object
|      >>>
|      >>> s.str.cat(t, join='inner', na_rep='-')
|      0    aa
|      2    -c
|      3    dd
```

```
 |      dtype: object
 |      >>>
 |      >>> s.str.cat(t, join='right', na_rep='-')
 |      3    dd
 |      0    aa
 |      4    -e
 |      2    -c
 |      dtype: object
 |
 |      For more examples, see :ref:`here <text.concatenate>`.
 |
 |  center(self, width: 'int', fillchar: 'str' = ' ')
 |      Pad left and right side of strings in the Series/Index.
 |
 |      Equivalent to :meth:`str.center`.
 |
 |      Parameters
 |      ----------
 |      width : int
 |          Minimum width of resulting string; additional characters will be
filled
 |          with ``fillchar``.
 |      fillchar : str
 |          Additional character for filling, default is whitespace.
 |
 |      Returns
 |      -------
 |      Series/Index of objects.
 |
 |      Examples
 |      --------
 |      For Series.str.center:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.center(8, fillchar='.')
 |      0   ..dog…
 |      1   ..bird..
 |      2   .mouse..
 |      dtype: object
 |
 |      For Series.str.ljust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.ljust(8, fillchar='.')
 |      0   dog…
 |      1   bird…
 |      2   mouse…
 |      dtype: object
 |
 |      For Series.str.rjust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.rjust(8, fillchar='.')
 |      0   …dog
 |      1   …bird
 |      2   …mouse
 |      dtype: object
 |
 |  contains(self, pat, case: 'bool' = True, flags: 'int' = 0, na=None, regex:
'bool' = True)
 |      Test if pattern or regex is contained within a string of a Series or
Index.
 |
 |      Return boolean Series or Index based on whether a given pattern or regex
is
 |      contained within a string of a Series or Index.
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Character sequence or regular expression.
 |      case : bool, default True
 |          If True, case sensitive.
 |      flags : int, default 0 (no flags)
 |          Flags to pass through to the re module, e.g. re.IGNORECASE.
 |      na : scalar, optional
 |          Fill value for missing values. The default depends on dtype of the
 |          array. For object-dtype, ``numpy.nan`` is used. For ``StringDtype``,
 |          ``pandas.NA`` is used.
 |      regex : bool, default True
 |          If True, assumes the pat is a regular expression.
 |
 |          If False, treats the pat as a literal string.
 |
 |      Returns
 |      -------
 |      Series or Index of boolean values
 |          A Series or Index of boolean values indicating whether the
 |          given pattern is contained within the string of each element
 |          of the Series or Index.
 |
 |      See Also
 |      --------
 |      match : Analogous, but stricter, relying on re.match instead of
re.search.
 |      Series.str.startswith : Test if the start of each string element matches
```

```
a
 |          pattern.
 |      Series.str.endswith : Same as startswith, but tests the end of string.
 |
 |      Examples
 |      --------
 |      Returning a Series of booleans using only a literal pattern.
 |
 |      >>> s1 = pd.Series(['Mouse', 'dog', 'house and parrot', '23', np.nan])
 |      >>> s1.str.contains('og', regex=False)
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      4      NaN
 |      dtype: object
 |
 |      Returning an Index of booleans using only a literal pattern.
 |
 |      >>> ind = pd.Index(['Mouse', 'dog', 'house and parrot', '23.0', np.nan])
 |      >>> ind.str.contains('23', regex=False)
 |      Index([False, False, False, True, nan], dtype='object')
 |
 |      Specifying case sensitivity using `case`.
 |
 |      >>> s1.str.contains('oG', case=True, regex=True)
 |      0    False
 |      1    False
 |      2    False
 |      3    False
 |      4      NaN
 |      dtype: object
 |
 |      Specifying `na` to be `False` instead of `NaN` replaces NaN values
 |      with `False`. If Series or Index does not contain NaN values
 |      the resultant dtype will be `bool`, otherwise, an `object` dtype.
 |
 |      >>> s1.str.contains('og', na=False, regex=True)
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      4    False
 |      dtype: bool
 |
 |      Returning 'house' or 'dog' when either expression occurs in a string.
 |
 |      >>> s1.str.contains('house|dog', regex=True)
 |      0    False
 |      1     True
 |      2     True
 |      3    False
 |      4      NaN
 |      dtype: object
 |
 |      Ignoring case sensitivity using `flags` with regex.
 |
 |      >>> import re
 |      >>> s1.str.contains('PARROT', flags=re.IGNORECASE, regex=True)
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      4      NaN
 |      dtype: object
 |
 |      Returning any digit using regular expression.
 |
 |      >>> s1.str.contains('\\d', regex=True)
 |      0    False
 |      1    False
 |      2    False
 |      3     True
 |      4      NaN
 |      dtype: object
 |
 |      Ensure `pat` is a not a literal pattern when `regex` is set to True.
 |      Note in the following example one might expect only `s2[1]` and `s2[3]`
to
 |      return `True`. However, '.0' as a regex matches any character
 |      followed by a 0.
 |
 |      >>> s2 = pd.Series(['40', '40.0', '41', '41.0', '35'])
 |      >>> s2.str.contains('.0', regex=True)
 |      0     True
 |      1     True
 |      2    False
 |      3     True
 |      4    False
 |      dtype: bool
 |
 |  count(self, pat, flags: 'int' = 0)
 |      Count occurrences of pattern in each string of the Series/Index.
 |
 |      This function is used to count the number of times a particular regex
 |      pattern is repeated in each of the string elements of the
 |      :class:`~pandas.Series`.
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Valid regular expression.
 |      flags : int, default 0, meaning no flags
 |          Flags for the `re` module. For a complete list, `see here
 |          <https://docs.python.org/3/howto/regex.html#compilation-flags>`_.
 |      **kwargs
 |          For compatibility with other string methods. Not used.
 |
 |      Returns
 |      -------
 |      Series or Index
 |          Same type as the calling object containing the integer counts.
 |
 |      See Also
 |      --------
 |      re : Standard library module for regular expressions.
 |      str.count : Standard library version, without regular expression
support.
 |
 |      Notes
 |      -----
 |      Some characters need to be escaped when passing in `pat`.
 |      eg. ``'$'`` has a special meaning in regex and must be escaped when
 |      finding this literal character.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['A', 'B', 'Aaba', 'Baca', np.nan, 'CABA', 'cat'])
 |      >>> s.str.count('a')
 |      0    0.0
 |      1    0.0
 |      2    2.0
 |      3    2.0
 |      4    NaN
 |      5    0.0
 |      6    1.0
 |      dtype: float64
 |
 |      Escape ``'$'`` to find the literal dollar sign.
 |
 |      >>> s = pd.Series(['$', 'B', 'Aab$', '$$ca', 'C$B$', 'cat'])
 |      >>> s.str.count('\\$')
 |      0    1
 |      1    0
 |      2    1
 |      3    2
 |      4    2
 |      5    0
 |      dtype: int64
 |
 |      This is also available on Index
 |
 |      >>> pd.Index(['A', 'A', 'Aaba', 'cat']).str.count('a')
 |      Index([0, 0, 2, 1], dtype='int64')
 |
 |  decode(self, encoding, errors: 'str' = 'strict')
 |      Decode character string in the Series/Index using indicated encoding.
 |
 |      Equivalent to :meth:`str.decode` in python2 and :meth:`bytes.decode` in
 |      python3.
 |
 |      Parameters
 |      ----------
 |      encoding : str
 |      errors : str, optional
 |
 |      Returns
 |      -------
 |      Series or Index
 |
 |      Examples
 |      --------
 |      For Series:
 |
 |      >>> ser = pd.Series([b'cow', b'123', b'()'])
 |      >>> ser.str.decode('ascii')
 |      0   cow
 |      1   123
 |      2   ()
 |      dtype: object
 |
 |  encode(self, encoding, errors: 'str' = 'strict')
 |      Encode character string in the Series/Index using indicated encoding.
 |
 |      Equivalent to :meth:`str.encode`.
 |
 |      Parameters
 |      ----------
 |      encoding : str
 |      errors : str, optional
 |
 |      Returns
 |      -------
 |      Series/Index of objects
 |
 |      Examples
 |      --------
 |      >>> ser = pd.Series(['cow', '123', '()'])
 |      >>> ser.str.encode(encoding='ascii')
 |      0     b'cow'
 |      1     b'123'
 |      2      b'()'
 |      dtype: object
 |
 |  endswith(self, pat: 'str | tuple[str, …]', na: 'Scalar | None' = None) ->
'Series | Index'
 |      Test if the end of each string element matches a pattern.
 |
 |      Equivalent to :meth:`str.endswith`.
 |
 |      Parameters
 |      ----------
 |      pat : str or tuple[str, …]
 |          Character sequence or tuple of strings. Regular expressions are not
 |          accepted.
 |      na : object, default NaN
 |          Object shown if element tested is not a string. The default depends
 |          on dtype of the array. For object-dtype, ``numpy.nan`` is used.
 |          For ``StringDtype``, ``pandas.NA`` is used.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          A Series of booleans indicating whether the given pattern matches
 |          the end of each string element.
 |
 |      See Also
 |      --------
 |      str.endswith : Python standard library string method.
 |      Series.str.startswith : Same as endswith, but tests the start of string.
 |      Series.str.contains : Tests if string element contains a pattern.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['bat', 'bear', 'caT', np.nan])
 |      >>> s
 |      0     bat
 |      1    bear
 |      2     caT
 |      3     NaN
 |      dtype: object
 |
 |      >>> s.str.endswith('t')
 |      0     True
 |      1    False
 |      2    False
 |      3      NaN
 |      dtype: object
 |
 |      >>> s.str.endswith(('t', 'T'))
 |      0     True
 |      1    False
 |      2     True
 |      3      NaN
 |      dtype: object
 |
 |      Specifying `na` to be `False` instead of `NaN`.
 |
 |      >>> s.str.endswith('t', na=False)
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  extract(self, pat: 'str', flags: 'int' = 0, expand: 'bool' = True) ->
'DataFrame | Series | Index'
 |      Extract capture groups in the regex `pat` as columns in a DataFrame.
 |
 |      For each subject string in the Series, extract groups from the
 |      first match of regular expression `pat`.
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Regular expression pattern with capturing groups.
 |      flags : int, default 0 (no flags)
 |          Flags from the ``re`` module, e.g. ``re.IGNORECASE``, that
 |          modify regular expression matching for things like case,
 |          spaces, etc. For more details, see :mod:`re`.
 |      expand : bool, default True
 |          If True, return DataFrame with one column per capture group.
 |          If False, return a Series/Index if there is one capture group
 |          or DataFrame if there are multiple capture groups.
 |
 |      Returns
 |      -------
 |      DataFrame or Series or Index
 |           A DataFrame with one row for each subject string, and one
 |           column for each group. Any capture group names in regular
 |           expression pat will be used for column names; otherwise
 |           capture group numbers will be used. The dtype of each result
 |           column is always object, even when no match is found. If
 |           ``expand=False`` and pat has only one capture group, then
 |           return a Series (if subject is a Series) or Index (if subject
 |           is an Index).
 |
 |       See Also
 |       --------
 |       extractall : Returns all matches (not just the first match).
 |
 |       Examples
 |       --------
 |       A pattern with two groups will return a DataFrame with two columns.
 |       Non-matches will be NaN.
 |
 |       >>> s = pd.Series(['a1', 'b2', 'c3'])
 |       >>> s.str.extract(r'([ab])(\d)')
 |           0    1
 |       0    a    1
 |       1    b    2
 |       2  NaN  NaN
 |
 |       A pattern may contain optional groups.
 |
 |       >>> s.str.extract(r'([ab])?(\d)')
 |           0  1
 |       0    a  1
 |       1    b  2
 |       2  NaN  3
 |
 |       Named groups will become column names in the result.
 |
 |       >>> s.str.extract(r'(?P<letter>[ab])(?P<digit>\d)')
 |       letter digit
 |       0      a     1
 |       1      b     2
 |       2    NaN   NaN
 |
 |       A pattern with one group will return a DataFrame with one column
 |       if expand=True.
 |
 |       >>> s.str.extract(r'[ab](\d)', expand=True)
 |           0
 |       0    1
 |       1    2
```

```
 |      2  NaN
 |
 |      A pattern with one group will return a Series if expand=False.
 |
 |      >>> s.str.extract(r'[ab](\d)', expand=False)
 |      0      1
 |      1      2
 |      2    NaN
 |      dtype: object
 |
 |  extractall(self, pat, flags: 'int' = 0) -> 'DataFrame'
 |      Extract capture groups in the regex `pat` as columns in DataFrame.
 |
 |      For each subject string in the Series, extract groups from all
 |      matches of regular expression pat. When each subject string in the
 |      Series has exactly one match, extractall(pat).xs(0, level='match')
 |      is the same as extract(pat).
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Regular expression pattern with capturing groups.
 |      flags : int, default 0 (no flags)
 |          A ``re`` module flag, for example ``re.IGNORECASE``. These allow
 |          to modify regular expression matching for things like case, spaces,
 |          etc. Multiple flags can be combined with the bitwise OR operator,
 |          for example ``re.IGNORECASE | re.MULTILINE``.
 |
 |      Returns
 |      -------
 |      DataFrame
 |          A ``DataFrame`` with one row for each match, and one column for each
 |          group. Its rows have a ``MultiIndex`` with first levels that come
from
 |          the subject ``Series``. The last level is named 'match' and indexes
the
 |          matches in each item of the ``Series``. Any capture group names in
 |          regular expression pat will be used for column names; otherwise
capture
 |          group numbers will be used.
 |
 |      See Also
 |      --------
 |      extract : Returns first match only (not all matches).
 |
 |      Examples
 |      --------
 |      A pattern with one group will return a DataFrame with one column.
 |      Indices with no matches will not appear in the result.
 |
 |      >>> s = pd.Series(["a1a2", "b1", "c1"], index=["A", "B", "C"])
 |      >>> s.str.extractall(r"[ab](\d)")
 |              0
 |      match
 |      A 0      1
 |        1      2
 |      B 0      1
 |
 |      Capture group names are used for column names of the result.
 |
 |      >>> s.str.extractall(r"[ab](?P<digit>\d)")
 |              digit
 |      match
 |      A 0         1
 |        1         2
 |      B 0         1
 |
 |      A pattern with two groups will return a DataFrame with two columns.
 |
 |      >>> s.str.extractall(r"(?P<letter>[ab])(?P<digit>\d)")
 |              letter digit
 |      match
 |      A 0          a     1
 |        1          a     2
 |      B 0          b     1
 |
 |      Optional groups that do not match are NaN in the result.
 |
 |      >>> s.str.extractall(r"(?P<letter>[ab])?(?P<digit>\d)")
 |              letter digit
 |      match
 |      A 0          a     1
 |        1          a     2
 |      B 0          b     1
 |      C 0        NaN     1
 |
 |  find(self, sub, start: 'int' = 0, end=None)
 |      Return lowest indexes in each strings in the Series/Index.
 |
 |      Each of returned indexes corresponds to the position where the
 |      substring is fully contained between [start:end]. Return -1 on
 |      failure. Equivalent to standard :meth:`str.find`.
 |
 |      Parameters
 |      ----------
 |      sub : str
 |          Substring being searched.
 |      start : int
 |          Left edge index.
 |      end : int
 |          Right edge index.
 |
 |      Returns
 |      -------
 |      Series or Index of int.
 |
 |      See Also
 |      --------
 |      rfind : Return highest indexes in each strings.
 |
 |      Examples
 |      --------
 |      For Series.str.find:
 |
 |      >>> ser = pd.Series(["cow_", "duck_", "do_ve"])
 |      >>> ser.str.find("_")
 |      0   3
 |      1   4
 |      2   2
 |      dtype: int64
 |
 |      For Series.str.rfind:
 |
 |      >>> ser = pd.Series(["_cow_", "duck_", "do_v_e"])
 |      >>> ser.str.rfind("_")
 |      0   4
 |      1   4
 |      2   4
 |      dtype: int64
 |
 |  findall(self, pat, flags: 'int' = 0)
 |      Find all occurrences of pattern or regular expression in the
Series/Index.
 |
 |      Equivalent to applying :func:`re.findall` to all the elements in the
 |      Series/Index.
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Pattern or regular expression.
 |      flags : int, default 0
 |          Flags from ``re`` module, e.g. `re.IGNORECASE` (default is 0, which
 |          means no flags).
 |
 |      Returns
 |      -------
 |      Series/Index of lists of strings
 |          All non-overlapping matches of pattern or regular expression in each
 |          string of this Series/Index.
 |
 |      See Also
 |      --------
 |      count : Count occurrences of pattern or regular expression in each
string
 |          of the Series/Index.
 |      extractall : For each string in the Series, extract groups from all
matches
 |          of regular expression and return a DataFrame with one row for each
 |          match and one column for each group.
 |      re.findall : The equivalent ``re`` function to all non-overlapping
matches
 |          of pattern or regular expression in string, as a list of strings.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['Lion', 'Monkey', 'Rabbit'])
 |
 |      The search for the pattern 'Monkey' returns one match:
 |
 |      >>> s.str.findall('Monkey')
 |      0          []
 |      1    [Monkey]
 |      2          []
 |      dtype: object
 |
 |      On the other hand, the search for the pattern 'MONKEY' doesn't return
any
 |      match:
 |
 |      >>> s.str.findall('MONKEY')
 |      0    []
 |      1    []
 |      2    []
 |      dtype: object
 |
 |      Flags can be added to the pattern or regular expression. For instance,
 |      to find the pattern 'MONKEY' ignoring the case:
 |
 |      >>> import re
 |      >>> s.str.findall('MONKEY', flags=re.IGNORECASE)
 |      0          []
 |      1    [Monkey]
 |      2          []
 |      dtype: object
 |
 |      When the pattern matches more than one string in the Series, all matches
 |      are returned:
 |
 |      >>> s.str.findall('on')
 |      0    [on]
 |      1    [on]
 |      2      []
 |      dtype: object
 |
 |      Regular expressions are supported too. For instance, the search for all
the
 |      strings ending with the word 'on' is shown next:
 |
 |      >>> s.str.findall('on$')
 |      0    [on]
 |      1      []
 |      2      []
 |      dtype: object
 |
 |      If the pattern is found more than once in the same string, then a list
of
 |      multiple strings is returned:
 |
 |      >>> s.str.findall('b')
 |      0        []
 |      1        []
 |      2    [b, b]
 |      dtype: object
 |
 |  fullmatch(self, pat, case: 'bool' = True, flags: 'int' = 0, na=None)
 |      Determine if each string entirely matches a regular expression.
 |
 |      Parameters
 |      ----------
 |      pat : str
 |          Character sequence or regular expression.
 |      case : bool, default True
 |          If True, case sensitive.
 |      flags : int, default 0 (no flags)
 |          Regex module flags, e.g. re.IGNORECASE.
 |      na : scalar, optional
 |          Fill value for missing values. The default depends on dtype of the
 |          array. For object-dtype, ``numpy.nan`` is used. For ``StringDtype``,
 |          ``pandas.NA`` is used.
 |
 |      Returns
 |      -------
 |      Series/Index/array of boolean values
 |
 |      See Also
 |      --------
 |      match : Similar, but also returns `True` when only a *prefix* of the
string
 |          matches the regular expression.
 |      extract : Extract matched groups.
 |
 |      Examples
 |      --------
 |      >>> ser = pd.Series(["cat", "duck", "dove"])
 |      >>> ser.str.fullmatch(r'd.+')
 |      0   False
 |      1    True
 |      2    True
 |      dtype: bool
 |
 |  get(self, i)
 |      Extract element from each component at specified position or with
specified key.
 |
 |      Extract element from lists, tuples, dict, or strings in each element in
the
 |      Series/Index.
 |
 |      Parameters
 |      ----------
 |      i : int or hashable dict label
 |          Position or key of element to extract.
 |
 |      Returns
 |      -------
 |      Series or Index
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(["String",
 |      …               (1, 2, 3),
 |      …               ["a", "b", "c"],
 |      …               123,
 |      …               -456,
 |      …               {1: "Hello", "2": "World"}])
 |      >>> s
 |      0                        String
 |      1                     (1, 2, 3)
 |      2                     [a, b, c]
 |      3                           123
 |      4                          -456
 |      5    {1: 'Hello', '2': 'World'}
 |      dtype: object
 |
 |      >>> s.str.get(1)
 |      0        t
 |      1        2
 |      2        b
 |      3      NaN
 |      4      NaN
 |      5    Hello
 |      dtype: object
 |
 |      >>> s.str.get(-1)
 |      0      g
 |      1      3
 |      2      c
 |      3    NaN
 |      4    NaN
 |      5    None
 |      dtype: object
 |
 |      Return element with given key
 |
 |      >>> s = pd.Series([{"name": "Hello", "value": "World"},
 |      …               {"name": "Goodbye", "value": "Planet"}])
 |      >>> s.str.get('name')
 |      0      Hello
 |      1    Goodbye
 |      dtype: object
 |
 |  get_dummies(self, sep: 'str' = '|')
 |      Return DataFrame of dummy/indicator variables for Series.
 |
 |      Each string in Series is split by sep and returned as a DataFrame
 |      of dummy/indicator variables.
 |
 |      Parameters
 |      ----------
 |      sep : str, default "|"
 |          String to split on.
 |
 |      Returns
 |      -------
 |      DataFrame
 |          Dummy variables corresponding to values of the Series.
 |
 |      See Also
 |      --------
 |      get_dummies : Convert categorical variable into dummy/indicator
 |          variables.
 |
 |      Examples
 |      --------
 |      >>> pd.Series(['a|b', 'a', 'a|c']).str.get_dummies()
 |         a  b  c
 |      0  1  1  0
 |      1  1  0  0
 |      2  1  0  1
 |
 |      >>> pd.Series(['a|b', np.nan, 'a|c']).str.get_dummies()
 |         a  b  c
 |      0  1  1  0
 |      1  0  0  0
 |      2  1  0  1
 |
 |  index(self, sub, start: 'int' = 0, end=None)
 |      Return lowest indexes in each string in Series/Index.
 |
 |      Each of the returned indexes corresponds to the position where the
 |      substring is fully contained between [start:end]. This is the same
 |      as ``str.find`` except instead of returning -1, it raises a
 |      ValueError when the substring is not found. Equivalent to standard
 |      ``str.index``.
 |
 |      Parameters
 |      ----------
 |      sub : str
 |          Substring being searched.
 |      start : int
 |          Left edge index.
 |      end : int
 |          Right edge index.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      rindex : Return highest indexes in each strings.
 |
 |      Examples
```

```
 |      --------
 |      For Series.str.index:
 |
 |      >>> ser = pd.Series(["horse", "eagle", "donkey"])
 |      >>> ser.str.index("e")
 |      0   4
 |      1   0
 |      2   4
 |      dtype: int64
 |
 |      For Series.str.rindex:
 |
 |      >>> ser = pd.Series(["Deer", "eagle", "Sheep"])
 |      >>> ser.str.rindex("e")
 |      0   2
 |      1   4
 |      2   3
 |      dtype: int64
 |
 |  isalnum(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are alphanumeric.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isalnum` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isalpha(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are alphabetic.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isalpha` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', '⅕', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
```

```
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isdecimal(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are decimal.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isdecimal` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isdigit(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are digits.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isdigit` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
```

```
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  islower(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are lowercase.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.islower` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isnumeric(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are numeric.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isnumeric` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', '⅕', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isspace(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are whitespace.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isspace` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
```

```
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  istitle(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are titlecase.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.istitle` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  isupper(self) from pandas.core.strings.accessor._map_and_wrap.<locals>
 |      Check whether all characters in each string are uppercase.
 |
 |      This is equivalent to running the Python string method
 |      :meth:`str.isupper` for each element of the Series/Index. If a string
 |      has zero characters, ``False`` is returned for that check.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          Series or Index of boolean values with the same length as the
original
 |          Series/Index.
 |
 |      See Also
 |      --------
 |      Series.str.isalpha : Check whether all characters are alphabetic.
 |      Series.str.isnumeric : Check whether all characters are numeric.
 |      Series.str.isalnum : Check whether all characters are alphanumeric.
 |      Series.str.isdigit : Check whether all characters are digits.
 |      Series.str.isdecimal : Check whether all characters are decimal.
 |      Series.str.isspace : Check whether all characters are whitespace.
 |      Series.str.islower : Check whether all characters are lowercase.
 |      Series.str.isupper : Check whether all characters are uppercase.
 |      Series.str.istitle : Check whether all characters are titlecase.
 |
 |      Examples
 |      --------
 |      **Checks for Alphabetic and Numeric Characters**
 |
 |      >>> s1 = pd.Series(['one', 'one1', '1', ''])
 |
 |      >>> s1.str.isalpha()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isnumeric()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      >>> s1.str.isalnum()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      Note that checks against characters mixed with any additional
punctuation
 |      or whitespace will evaluate to false for an alphanumeric check.
 |
 |      >>> s2 = pd.Series(['A B', '1.5', '3,000'])
 |      >>> s2.str.isalnum()
 |      0    False
 |      1    False
 |      2    False
 |      dtype: bool
 |
 |      **More Detailed Checks for Numeric Characters**
 |
 |      There are several different but overlapping sets of numeric characters
that
 |      can be checked for.
 |
 |      >>> s3 = pd.Series(['23', '³', ' ', ''])
 |
 |      The ``s3.str.isdecimal`` method checks for characters used to form
numbers
 |      in base 10.
 |
 |      >>> s3.str.isdecimal()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isdigit`` method is the same as ``s3.str.isdecimal`` but
also
 |      includes special digits, like superscripted and subscripted digits in
 |      unicode.
 |
 |      >>> s3.str.isdigit()
 |      0     True
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      The ``s.str.isnumeric`` method is the same as ``s3.str.isdigit`` but
also
 |      includes other characters that can represent quantities such as unicode
 |      fractions.
 |
 |      >>> s3.str.isnumeric()
 |      0     True
 |      1     True
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      **Checks for Whitespace**
 |
 |      >>> s4 = pd.Series([' ', '\t\r\n ', ''])
 |      >>> s4.str.isspace()
 |      0     True
 |      1     True
 |      2    False
 |      dtype: bool
 |
 |      **Checks for Character Case**
 |
 |      >>> s5 = pd.Series(['leopard', 'Golden Eagle', 'SNAKE', ''])
 |
 |      >>> s5.str.islower()
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |      >>> s5.str.isupper()
 |      0    False
 |      1    False
 |      2     True
 |      3    False
 |      dtype: bool
 |
 |      The ``s5.str.istitle`` method checks for whether all words are in title
 |      case (whether only the first letter of each word is capitalized). Words
are
 |      assumed to be as any sequence of non-numeric characters separated by
 |      whitespace characters.
 |
 |      >>> s5.str.istitle()
 |      0    False
 |      1     True
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  join(self, sep: 'str')
 |      Join lists contained as elements in the Series/Index with passed
```

```
delimiter.
 |
 |      If the elements of a Series are lists themselves, join the content of
these
 |      lists using the delimiter passed to the function.
 |      This function is an equivalent to :meth:`str.join`.
 |
 |      Parameters
 |      ----------
 |      sep : str
 |          Delimiter to use between list entries.
 |
 |      Returns
 |      -------
 |      Series/Index: object
 |          The list entries concatenated by intervening occurrences of the
 |          delimiter.
 |
 |      Raises
 |      ------
 |      AttributeError
 |          If the supplied Series contains neither strings nor lists.
 |
 |      See Also
 |      --------
 |      str.join : Standard library version of this method.
 |      Series.str.split : Split strings around given separator/delimiter.
 |
 |      Notes
 |      -----
 |      If any of the list items is not a string object, the result of the join
 |      will be `NaN`.
 |
 |      Examples
 |      --------
 |      Example with a list that contains non-string elements.
 |
 |      >>> s = pd.Series([['lion', 'elephant', 'zebra'],
 |      …                 [1.1, 2.2, 3.3],
 |      …                 ['cat', np.nan, 'dog'],
 |      …                 ['cow', 4.5, 'goat'],
 |      …                 ['duck', ['swan', 'fish'], 'guppy']])
 |      >>> s
 |      0        [lion, elephant, zebra]
 |      1                [1.1, 2.2, 3.3]
 |      2                [cat, nan, dog]
 |      3               [cow, 4.5, goat]
 |      4    [duck, [swan, fish], guppy]
 |      dtype: object
 |
 |      Join all lists using a '-'. The lists containing object(s) of types
other
 |      than str will produce a NaN.
 |
 |      >>> s.str.join('-')
 |      0    lion-elephant-zebra
 |      1                    NaN
 |      2                    NaN
 |      3                    NaN
 |      4                    NaN
 |      dtype: object
 |
 |  len(self)
 |      Compute the length of each element in the Series/Index.
 |
 |      The element may be a sequence (such as a string, tuple or list) or a
collection
 |      (such as a dictionary).
 |
 |      Returns
 |      -------
 |      Series or Index of int
 |          A Series or Index of integer values indicating the length of each
 |          element in the Series or Index.
 |
 |      See Also
 |      --------
 |      str.len : Python built-in function returning the length of an object.
 |      Series.size : Returns the length of the Series.
 |
 |      Examples
 |      --------
 |      Returns the length (number of characters) in a string. Returns the
 |      number of entries for dictionaries, lists or tuples.
 |
 |      >>> s = pd.Series(['dog',
 |      …                 '',
 |      …                 5,
 |      …                 {'foo' : 'bar'},
 |      …                 [2, 3, 5, 7],
 |      …                 ('one', 'two', 'three')])
 |      >>> s
 |      0                  dog
 |      1
 |      2                    5
 |      3       {'foo': 'bar'}
 |      4         [2, 3, 5, 7]
 |      5    (one, two, three)
 |      dtype: object
 |      >>> s.str.len()
 |      0    3.0
 |      1    0.0
 |      2    NaN
 |      3    1.0
 |      4    4.0
 |      5    3.0
 |      dtype: float64
 |
 |  ljust(self, width: 'int', fillchar: 'str' = ' ')
 |      Pad right side of strings in the Series/Index.
 |
 |      Equivalent to :meth:`str.ljust`.
 |
 |      Parameters
 |      ----------
 |      width : int
 |          Minimum width of resulting string; additional characters will be
filled
 |          with ``fillchar``.
 |      fillchar : str
 |          Additional character for filling, default is whitespace.
 |
 |      Returns
 |      -------
 |      Series/Index of objects.
 |
 |      Examples
 |      --------
 |      For Series.str.center:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.center(8, fillchar='.')
 |      0   ..dog…
 |      1   ..bird..
 |      2   .mouse..
 |      dtype: object
 |
 |      For Series.str.ljust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.ljust(8, fillchar='.')
 |      0   dog…
 |      1   bird…
 |      2   mouse…
 |      dtype: object
 |
 |      For Series.str.rjust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.rjust(8, fillchar='.')
 |      0   …dog
 |      1   …bird
 |      2   …mouse
 |      dtype: object
 |
 |  lower(self)
 |      Convert strings in the Series/Index to lowercase.
 |
 |      Equivalent to :meth:`str.lower`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1              CAPITALS
 |      2    this is a sentence
 |      3              SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                 lower
 |      1              capitals
 |      2    this is a sentence
 |      3              swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                 LOWER
 |      1              CAPITALS
 |      2    THIS IS A SENTENCE
 |      3              SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                 Lower
 |      1              Capitals
 |      2    This Is A Sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                 Lower
 |      1              Capitals
 |      2    This is a sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                 LOWER
 |      1              capitals
 |      2    THIS IS A SENTENCE
 |      3              sWaPcAsE
 |      dtype: object
 |
 |  lstrip(self, to_strip=None)
 |      Remove leading characters.
 |
 |      Strip whitespaces (including newlines) or a set of specified characters
 |      from each string in the Series/Index from left side.
 |      Replaces any non-strings in Series with NaNs.
 |      Equivalent to :meth:`str.lstrip`.
 |
 |      Parameters
 |      ----------
 |      to_strip : str or None, default None
 |          Specifying the set of characters to be removed.
 |          All combinations of this set of characters will be stripped.
 |          If None then whitespaces are removed.
 |
 |      Returns
 |      -------
```

```
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.strip : Remove leading and trailing characters in
Series/Index.
 |      Series.str.lstrip : Remove leading characters in Series/Index.
 |      Series.str.rstrip : Remove trailing characters in Series/Index.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['1. Ant.  ', '2. Bee!\n', '3. Cat?\t', np.nan, 10,
True])
 |      >>> s
 |      0    1. Ant.
 |      1    2. Bee!\n
 |      2    3. Cat?\t
 |      3          NaN
 |      4           10
 |      5         True
 |      dtype: object
 |
 |      >>> s.str.strip()
 |      0    1. Ant.
 |      1    2. Bee!
 |      2    3. Cat?
 |      3        NaN
 |      4        NaN
 |      5        NaN
 |      dtype: object
 |
 |      >>> s.str.lstrip('123.')
 |      0    Ant.
 |      1    Bee!\n
 |      2    Cat?\t
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
 |
 |      >>> s.str.rstrip('.!? \n\t')
 |      0    1. Ant
 |      1    2. Bee
 |      2    3. Cat
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
```

```
|
|      >>> s.str.strip('123.!? \n\t')
|      0    Ant
|      1    Bee
|      2    Cat
|      3    NaN
|      4    NaN
|      5    NaN
|      dtype: object
|
|  match(self, pat: 'str', case: 'bool' = True, flags: 'int' = 0, na=None)
|      Determine if each string starts with a match of a regular expression.
|
|      Parameters
|      ----------
|      pat : str
|          Character sequence.
|      case : bool, default True
|          If True, case sensitive.
|      flags : int, default 0 (no flags)
|          Regex module flags, e.g. re.IGNORECASE.
|      na : scalar, optional
|          Fill value for missing values. The default depends on dtype of the
|          array. For object-dtype, ``numpy.nan`` is used. For ``StringDtype``,
|          ``pandas.NA`` is used.
|
|      Returns
|      -------
|      Series/Index/array of boolean values
|
|      See Also
|      --------
|      fullmatch : Stricter matching that requires the entire string to match.
|      contains : Analogous, but less strict, relying on re.search instead of
|          re.match.
|      extract : Extract matched groups.
|
|      Examples
|      --------
|      >>> ser = pd.Series(["horse", "eagle", "donkey"])
|      >>> ser.str.match("e")
|      0   False
|      1   True
|      2   False
|      dtype: bool
|
|  normalize(self, form)
|      Return the Unicode normal form for the strings in the Series/Index.
```

```
 |
 |      For more information on the forms, see the
 |      :func:`unicodedata.normalize`.
 |
 |      Parameters
 |      ----------
 |      form : {'NFC', 'NFKC', 'NFD', 'NFKD'}
 |          Unicode form.
 |
 |      Returns
 |      -------
 |      Series/Index of objects
 |
 |      Examples
 |      --------
 |      >>> ser = pd.Series(['ñ'])
 |      >>> ser.str.normalize('NFC') == ser.str.normalize('NFD')
 |      0   False
 |      dtype: bool
 |
 |  pad(self, width: 'int', side: "Literal['left', 'right', 'both']" = 'left',
fillchar: 'str' = ' ')
 |      Pad strings in the Series/Index up to width.
 |
 |      Parameters
 |      ----------
 |      width : int
 |          Minimum width of resulting string; additional characters will be
filled
 |          with character defined in `fillchar`.
 |      side : {'left', 'right', 'both'}, default 'left'
 |          Side from which to fill resulting string.
 |      fillchar : str, default ' '
 |          Additional character for filling, default is whitespace.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |          Returns Series or Index with minimum number of char in object.
 |
 |      See Also
 |      --------
 |      Series.str.rjust : Fills the left side of strings with an arbitrary
 |          character. Equivalent to ``Series.str.pad(side='left')``.
 |      Series.str.ljust : Fills the right side of strings with an arbitrary
 |          character. Equivalent to ``Series.str.pad(side='right')``.
 |      Series.str.center : Fills both sides of strings with an arbitrary
 |          character. Equivalent to ``Series.str.pad(side='both')``.
 |      Series.str.zfill : Pad strings in the Series/Index by prepending '0'
 |          character. Equivalent to ``Series.str.pad(side='left',
fillchar='0')``.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(["caribou", "tiger"])
 |      >>> s
 |      0    caribou
 |      1      tiger
 |      dtype: object
 |
 |      >>> s.str.pad(width=10)
 |      0       caribou
 |      1         tiger
 |      dtype: object
 |
 |      >>> s.str.pad(width=10, side='right', fillchar='-')
 |      0    caribou---
 |      1    tiger-----
 |      dtype: object
 |
 |      >>> s.str.pad(width=10, side='both', fillchar='-')
 |      0    -caribou--
 |      1    --tiger---
 |      dtype: object
 |
 |  partition(self, sep: 'str' = ' ', expand: 'bool' = True)
 |      Split the string at the first occurrence of `sep`.
 |
 |      This method splits the string at the first occurrence of `sep`,
 |      and returns 3 elements containing the part before the separator,
 |      the separator itself, and the part after the separator.
 |      If the separator is not found, return 3 elements containing the string
itself, followed by two empty strings.
 |
 |      Parameters
 |      ----------
 |      sep : str, default whitespace
 |          String to split on.
 |      expand : bool, default True
 |          If True, return DataFrame/MultiIndex expanding dimensionality.
 |          If False, return Series/Index.
 |
 |      Returns
 |      -------
 |      DataFrame/MultiIndex or Series/Index of objects
 |
 |      See Also
 |      --------
 |      rpartition : Split the string at the last occurrence of `sep`.
 |      Series.str.split : Split strings around given separators.
 |      str.partition : Standard library version.
 |
 |      Examples
 |      --------
 |
 |      >>> s = pd.Series(['Linda van der Berg', 'George Pitt-Rivers'])
 |      >>> s
 |      0    Linda van der Berg
 |      1    George Pitt-Rivers
 |      dtype: object
 |
 |      >>> s.str.partition()
 |              0  1             2
 |      0   Linda     van der Berg
 |      1  George      Pitt-Rivers
 |
 |      To partition by the last space instead of the first one:
 |
 |      >>> s.str.rpartition()
 |                     0  1            2
 |      0  Linda van der            Berg
 |      1         George     Pitt-Rivers
 |
 |      To partition by something different than a space:
 |
 |      >>> s.str.partition('-')
 |                          0  1       2
 |      0  Linda van der Berg
 |      1         George Pitt  -  Rivers
 |
 |      To return a Series containing tuples instead of a DataFrame:
 |
 |      >>> s.str.partition('-', expand=False)
 |      0    (Linda van der Berg, , )
 |      1    (George Pitt, -, Rivers)
 |      dtype: object
 |
 |      Also available on indices:
 |
 |      >>> idx = pd.Index(['X 123', 'Y 999'])
 |      >>> idx
 |      Index(['X 123', 'Y 999'], dtype='object')
 |
 |      Which will create a MultiIndex:
```

```
|
|      >>> idx.str.partition()
|      MultiIndex([('X', ' ', '123'),
|                  ('Y', ' ', '999')],
|                 )
|
|      Or an index with tuples with ``expand=False``:
|
|      >>> idx.str.partition(expand=False)
|      Index([('X', ' ', '123'), ('Y', ' ', '999')], dtype='object')
|
|  removeprefix(self, prefix: 'str')
|      Remove a prefix from an object series.
|
|      If the prefix is not present, the original string will be returned.
|
|      Parameters
|      ----------
|      prefix : str
|          Remove the prefix of the string.
|
|      Returns
|      -------
|      Series/Index: object
|          The Series or Index with given prefix removed.
|
|      See Also
|      --------
|      Series.str.removesuffix : Remove a suffix from an object series.
|
|      Examples
|      --------
|      >>> s = pd.Series(["str_foo", "str_bar", "no_prefix"])
|      >>> s
|      0    str_foo
|      1    str_bar
|      2    no_prefix
|      dtype: object
|      >>> s.str.removeprefix("str_")
|      0    foo
|      1    bar
|      2    no_prefix
|      dtype: object
|
|      >>> s = pd.Series(["foo_str", "bar_str", "no_suffix"])
|      >>> s
|      0    foo_str
|      1    bar_str
```

```
 |      2    no_suffix
 |      dtype: object
 |      >>> s.str.removesuffix("_str")
 |      0    foo
 |      1    bar
 |      2    no_suffix
 |      dtype: object
 |
 |  removesuffix(self, suffix: 'str')
 |      Remove a suffix from an object series.
 |
 |      If the suffix is not present, the original string will be returned.
 |
 |      Parameters
 |      ----------
 |      suffix : str
 |          Remove the suffix of the string.
 |
 |      Returns
 |      -------
 |      Series/Index: object
 |          The Series or Index with given suffix removed.
 |
 |      See Also
 |      --------
 |      Series.str.removeprefix : Remove a prefix from an object series.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(["str_foo", "str_bar", "no_prefix"])
 |      >>> s
 |      0    str_foo
 |      1    str_bar
 |      2    no_prefix
 |      dtype: object
 |      >>> s.str.removeprefix("str_")
 |      0    foo
 |      1    bar
 |      2    no_prefix
 |      dtype: object
 |
 |      >>> s = pd.Series(["foo_str", "bar_str", "no_suffix"])
 |      >>> s
 |      0    foo_str
 |      1    bar_str
 |      2    no_suffix
 |      dtype: object
 |      >>> s.str.removesuffix("_str")
 |      0    foo
 |      1    bar
 |      2    no_suffix
 |      dtype: object
 |
 |  repeat(self, repeats)
 |      Duplicate each string in the Series or Index.
 |
 |      Parameters
 |      ----------
 |      repeats : int or sequence of int
 |          Same value for all (int) or different value per (sequence).
 |
 |      Returns
 |      -------
 |      Series or pandas.Index
 |          Series or Index of repeated string objects specified by
 |          input parameter repeats.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['a', 'b', 'c'])
 |      >>> s
 |      0    a
 |      1    b
 |      2    c
 |      dtype: object
 |
 |      Single int repeats string in Series
 |
 |      >>> s.str.repeat(repeats=2)
 |      0    aa
 |      1    bb
 |      2    cc
 |      dtype: object
 |
 |      Sequence of int repeats corresponding string in Series
 |
 |      >>> s.str.repeat(repeats=[1, 2, 3])
 |      0      a
 |      1     bb
 |      2    ccc
 |      dtype: object
 |
 |  replace(self, pat: 'str | re.Pattern', repl: 'str | Callable', n: 'int' =
-1, case: 'bool | None' = None, flags: 'int' = 0, regex: 'bool' = False)
 |      Replace each occurrence of pattern/regex in the Series/Index.
 |
 |      Equivalent to :meth:`str.replace` or :func:`re.sub`, depending on
 |      the regex value.
 |
 |      Parameters
 |      ----------
 |      pat : str or compiled regex
 |          String can be a character sequence or regular expression.
 |      repl : str or callable
 |          Replacement string or a callable. The callable is passed the regex
 |          match object and must return a replacement string to be used.
 |          See :func:`re.sub`.
 |      n : int, default -1 (all)
 |          Number of replacements to make from start.
 |      case : bool, default None
 |          Determines if replace is case sensitive:
 |
 |          - If True, case sensitive (the default if `pat` is a string)
 |          - Set to False for case insensitive
 |          - Cannot be set if `pat` is a compiled regex.
 |
 |      flags : int, default 0 (no flags)
 |          Regex module flags, e.g. re.IGNORECASE. Cannot be set if `pat` is a
compiled
 |          regex.
 |      regex : bool, default False
 |          Determines if the passed-in pattern is a regular expression:
 |
 |          - If True, assumes the passed-in pattern is a regular expression.
 |          - If False, treats the pattern as a literal string
 |          - Cannot be set to False if `pat` is a compiled regex or `repl` is
 |            a callable.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |          A copy of the object with all matching occurrences of `pat` replaced
by
 |          `repl`.
 |
 |      Raises
 |      ------
 |      ValueError
 |          * if `regex` is False and `repl` is a callable or `pat` is a
compiled
 |            regex
 |          * if `pat` is a compiled regex and `case` or `flags` is set
 |
 |      Notes
 |      -----
 |      When `pat` is a compiled regex, all flags should be included in the
 |      compiled regex. Use of `case`, `flags`, or `regex=False` with a compiled
 |      regex will raise an error.
 |
 |      Examples
 |      --------
 |      When `pat` is a string and `regex` is True, the given `pat`
 |      is compiled as a regex. When `repl` is a string, it replaces matching
 |      regex patterns as with :meth:`re.sub`. NaN value(s) in the Series are
 |      left as is:
 |
 |      >>> pd.Series(['foo', 'fuz', np.nan]).str.replace('f.', 'ba',
regex=True)
 |      0    bao
 |      1    baz
 |      2    NaN
 |      dtype: object
 |
 |      When `pat` is a string and `regex` is False, every `pat` is replaced
with
 |      `repl` as with :meth:`str.replace`:
 |
 |      >>> pd.Series(['f.o', 'fuz', np.nan]).str.replace('f.', 'ba',
regex=False)
 |      0    bao
 |      1    fuz
 |      2    NaN
 |      dtype: object
 |
 |      When `repl` is a callable, it is called on every `pat` using
 |      :func:`re.sub`. The callable should expect one positional argument
 |      (a regex object) and return a string.
 |
 |      To get the idea:
 |
 |      >>> pd.Series(['foo', 'fuz', np.nan]).str.replace('f', repr, regex=True)
 |      0    <re.Match object; span=(0, 1), match='f'>oo
 |      1    <re.Match object; span=(0, 1), match='f'>uz
 |      2                                            NaN
 |      dtype: object
 |
 |      Reverse every lowercase alphabetic word:
 |
 |      >>> repl = lambda m: m.group(0)[::-1]
 |      >>> ser = pd.Series(['foo 123', 'bar baz', np.nan])
 |      >>> ser.str.replace(r'[a-z]+', repl, regex=True)
 |      0    oof 123
 |      1    rab zab
 |      2        NaN
 |      dtype: object
 |
 |      Using regex groups (extract second group and swap case):
 |
 |      >>> pat = r"(?P<one>\w+) (?P<two>\w+) (?P<three>\w+)"
 |      >>> repl = lambda m: m.group('two').swapcase()
 |      >>> ser = pd.Series(['One Two Three', 'Foo Bar Baz'])
 |      >>> ser.str.replace(pat, repl, regex=True)
 |      0    tWO
 |      1    bAR
 |      dtype: object
 |
 |      Using a compiled regex with flags
 |
 |      >>> import re
 |      >>> regex_pat = re.compile(r'FUZ', flags=re.IGNORECASE)
 |      >>> pd.Series(['foo', 'fuz', np.nan]).str.replace(regex_pat, 'bar',
regex=True)
 |      0    foo
 |      1    bar
 |      2    NaN
 |      dtype: object
 |
 |  rfind(self, sub, start: 'int' = 0, end=None)
 |      Return highest indexes in each strings in the Series/Index.
 |
 |      Each of returned indexes corresponds to the position where the
 |      substring is fully contained between [start:end]. Return -1 on
 |      failure. Equivalent to standard :meth:`str.rfind`.
 |
 |      Parameters
 |      ----------
 |      sub : str
 |          Substring being searched.
 |      start : int
 |          Left edge index.
 |      end : int
 |          Right edge index.
 |
 |      Returns
 |      -------
 |      Series or Index of int.
 |
 |      See Also
 |      --------
 |      find : Return lowest indexes in each strings.
```

```
|
|      Examples
|      --------
|      For Series.str.find:
|
|      >>> ser = pd.Series(["cow_", "duck_", "do_ve"])
|      >>> ser.str.find("_")
|      0   3
|      1   4
|      2   2
|      dtype: int64
|
|      For Series.str.rfind:
|
|      >>> ser = pd.Series(["_cow_", "duck_", "do_v_e"])
|      >>> ser.str.rfind("_")
|      0   4
|      1   4
|      2   4
|      dtype: int64
|
|  rindex(self, sub, start: 'int' = 0, end=None)
|      Return highest indexes in each string in Series/Index.
|
|      Each of the returned indexes corresponds to the position where the
|      substring is fully contained between [start:end]. This is the same
|      as ``str.rfind`` except instead of returning -1, it raises a
|      ValueError when the substring is not found. Equivalent to standard
|      ``str.rindex``.
|
|      Parameters
|      ----------
|      sub : str
|          Substring being searched.
|      start : int
|          Left edge index.
|      end : int
|          Right edge index.
|
|      Returns
|      -------
|      Series or Index of object
|
|      See Also
|      --------
|      index : Return lowest indexes in each strings.
|
|      Examples
```

```
 |      --------
 |      For Series.str.index:
 |
 |      >>> ser = pd.Series(["horse", "eagle", "donkey"])
 |      >>> ser.str.index("e")
 |      0   4
 |      1   0
 |      2   4
 |      dtype: int64
 |
 |      For Series.str.rindex:
 |
 |      >>> ser = pd.Series(["Deer", "eagle", "Sheep"])
 |      >>> ser.str.rindex("e")
 |      0   2
 |      1   4
 |      2   3
 |      dtype: int64
 |
 |  rjust(self, width: 'int', fillchar: 'str' = ' ')
 |      Pad left side of strings in the Series/Index.
 |
 |      Equivalent to :meth:`str.rjust`.
 |
 |      Parameters
 |      ----------
 |      width : int
 |          Minimum width of resulting string; additional characters will be
filled
 |          with ``fillchar``.
 |      fillchar : str
 |          Additional character for filling, default is whitespace.
 |
 |      Returns
 |      -------
 |      Series/Index of objects.
 |
 |      Examples
 |      --------
 |      For Series.str.center:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.center(8, fillchar='.')
 |      0   ..dog…
 |      1   ..bird..
 |      2   .mouse..
 |      dtype: object
 |
 |      For Series.str.ljust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.ljust(8, fillchar='.')
 |      0   dog…
 |      1   bird…
 |      2   mouse…
 |      dtype: object
 |
 |      For Series.str.rjust:
 |
 |      >>> ser = pd.Series(['dog', 'bird', 'mouse'])
 |      >>> ser.str.rjust(8, fillchar='.')
 |      0   …dog
 |      1   …bird
 |      2   …mouse
 |      dtype: object
 |
 |  rpartition(self, sep: 'str' = ' ', expand: 'bool' = True)
 |      Split the string at the last occurrence of `sep`.
 |
 |      This method splits the string at the last occurrence of `sep`,
 |      and returns 3 elements containing the part before the separator,
 |      the separator itself, and the part after the separator.
 |      If the separator is not found, return 3 elements containing two empty
strings, followed by the string itself.
 |
 |      Parameters
 |      ----------
 |      sep : str, default whitespace
 |          String to split on.
 |      expand : bool, default True
 |          If True, return DataFrame/MultiIndex expanding dimensionality.
 |          If False, return Series/Index.
 |
 |      Returns
 |      -------
 |      DataFrame/MultiIndex or Series/Index of objects
 |
 |      See Also
 |      --------
 |      partition : Split the string at the first occurrence of `sep`.
 |      Series.str.split : Split strings around given separators.
 |      str.partition : Standard library version.
 |
 |      Examples
 |      --------
 |
 |       >>> s = pd.Series(['Linda van der Berg', 'George Pitt-Rivers'])
 |       >>> s
 |       0    Linda van der Berg
 |       1    George Pitt-Rivers
 |       dtype: object
 |
 |       >>> s.str.partition()
 |               0  1             2
 |       0   Linda     van der Berg
 |       1  George      Pitt-Rivers
 |
 |       To partition by the last space instead of the first one:
 |
 |       >>> s.str.rpartition()
 |                      0  1            2
 |       0  Linda van der            Berg
 |       1         George     Pitt-Rivers
 |
 |       To partition by something different than a space:
 |
 |       >>> s.str.partition('-')
 |                           0  1       2
 |       0  Linda van der Berg
 |       1         George Pitt  -  Rivers
 |
 |       To return a Series containing tuples instead of a DataFrame:
 |
 |       >>> s.str.partition('-', expand=False)
 |       0    (Linda van der Berg, , )
 |       1    (George Pitt, -, Rivers)
 |       dtype: object
 |
 |       Also available on indices:
 |
 |       >>> idx = pd.Index(['X 123', 'Y 999'])
 |       >>> idx
 |       Index(['X 123', 'Y 999'], dtype='object')
 |
 |       Which will create a MultiIndex:
 |
 |       >>> idx.str.partition()
 |       MultiIndex([('X', ' ', '123'),
 |                   ('Y', ' ', '999')],
 |                  )
 |
 |       Or an index with tuples with ``expand=False``:
 |
 |       >>> idx.str.partition(expand=False)
 |      Index([('X', ' ', '123'), ('Y', ' ', '999')], dtype='object')
 |
 |  rsplit(self, pat=None, *, n=-1, expand: 'bool' = False)
 |      Split strings around given separator/delimiter.
 |
 |      Splits the string in the Series/Index from the end,
 |      at the specified delimiter string.
 |
 |      Parameters
 |      ----------
 |      pat : str, optional
 |          String to split on.
 |          If not specified, split on whitespace.
 |      n : int, default -1 (all)
 |          Limit number of splits in output.
 |          ``None``, 0 and -1 will be interpreted as return all splits.
 |      expand : bool, default False
 |          Expand the split strings into separate columns.
 |
 |          - If ``True``, return DataFrame/MultiIndex expanding dimensionality.
 |          - If ``False``, return Series/Index, containing lists of strings.
 |
 |      Returns
 |      -------
 |      Series, Index, DataFrame or MultiIndex
 |          Type matches caller unless ``expand=True`` (see Notes).
 |
 |      See Also
 |      --------
 |      Series.str.split : Split strings around given separator/delimiter.
 |      Series.str.rsplit : Splits string around given separator/delimiter,
 |          starting from the right.
 |      Series.str.join : Join lists contained as elements in the Series/Index
 |          with passed delimiter.
 |      str.split : Standard library version for split.
 |      str.rsplit : Standard library version for rsplit.
 |
 |      Notes
 |      -----
 |      The handling of the `n` keyword depends on the number of found splits:
 |
 |      - If found splits > `n`,  make first `n` splits only
 |      - If found splits <= `n`, make all splits
 |      - If for a certain row the number of found splits < `n`,
 |        append `None` for padding up to `n` if ``expand=True``
 |
 |      If using ``expand=True``, Series and Index callers return DataFrame and
 |      MultiIndex objects, respectively.
```

```
|
|       Examples
|       --------
|       >>> s = pd.Series(
|       …     [
|       …         "this is a regular sentence",
|       …         "https://docs.python.org/3/tutorial/index.html",
|       …         np.nan
|       …     ]
|       … )
|       >>> s
|       0                       this is a regular sentence
|       1    https://docs.python.org/3/tutorial/index.html
|       2                                              NaN
|       dtype: object
|
|       In the default setting, the string is split by whitespace.
|
|       >>> s.str.split()
|       0                   [this, is, a, regular, sentence]
|       1    [https://docs.python.org/3/tutorial/index.html]
|       2                                                NaN
|       dtype: object
|
|       Without the `n` parameter, the outputs of `rsplit` and `split`
|       are identical.
|
|       >>> s.str.rsplit()
|       0                   [this, is, a, regular, sentence]
|       1    [https://docs.python.org/3/tutorial/index.html]
|       2                                                NaN
|       dtype: object
|
|       The `n` parameter can be used to limit the number of splits on the
|       delimiter. The outputs of `split` and `rsplit` are different.
|
|       >>> s.str.split(n=2)
|       0                     [this, is, a regular sentence]
|       1    [https://docs.python.org/3/tutorial/index.html]
|       2                                                NaN
|       dtype: object
|
|       >>> s.str.rsplit(n=2)
|       0                     [this is a, regular, sentence]
|       1    [https://docs.python.org/3/tutorial/index.html]
|       2                                                NaN
|       dtype: object
|
 |      The `pat` parameter can be used to split by other characters.
 |
 |      >>> s.str.split(pat="/")
 |      0                         [this is a regular sentence]
 |      1    [https:, , docs.python.org, 3, tutorial, index…
 |      2                                                  NaN
 |      dtype: object
 |
 |      When using ``expand=True``, the split elements will expand out into
 |      separate columns. If NaN is present, it is propagated throughout
 |      the columns during the split.
 |
 |      >>> s.str.split(expand=True)
 |                                                     0     1     2        3
4
 |      0                                           this    is     a  regular
sentence
 |      1  https://docs.python.org/3/tutorial/index.html  None  None     None
None
 |      2                                            NaN   NaN   NaN      NaN
NaN
 |
 |      For slightly more complex use cases like splitting the html document
name
 |      from a url, a combination of parameter settings can be used.
 |
 |      >>> s.str.rsplit("/", n=1, expand=True)
 |                                          0           1
 |      0          this is a regular sentence        None
 |      1  https://docs.python.org/3/tutorial  index.html
 |      2                                 NaN         NaN
 |
 |  rstrip(self, to_strip=None)
 |      Remove trailing characters.
 |
 |      Strip whitespaces (including newlines) or a set of specified characters
 |      from each string in the Series/Index from right side.
 |      Replaces any non-strings in Series with NaNs.
 |      Equivalent to :meth:`str.rstrip`.
 |
 |      Parameters
 |      ----------
 |      to_strip : str or None, default None
 |          Specifying the set of characters to be removed.
 |          All combinations of this set of characters will be stripped.
 |          If None then whitespaces are removed.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.strip : Remove leading and trailing characters in
Series/Index.
 |      Series.str.lstrip : Remove leading characters in Series/Index.
 |      Series.str.rstrip : Remove trailing characters in Series/Index.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['1. Ant.  ', '2. Bee!\n', '3. Cat?\t', np.nan, 10,
True])
 |      >>> s
 |      0    1. Ant.
 |      1    2. Bee!\n
 |      2    3. Cat?\t
 |      3          NaN
 |      4           10
 |      5         True
 |      dtype: object
 |
 |      >>> s.str.strip()
 |      0    1. Ant.
 |      1    2. Bee!
 |      2    3. Cat?
 |      3        NaN
 |      4        NaN
 |      5        NaN
 |      dtype: object
 |
 |      >>> s.str.lstrip('123.')
 |      0    Ant.
 |      1    Bee!\n
 |      2    Cat?\t
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
 |
 |      >>> s.str.rstrip('.!? \n\t')
 |      0    1. Ant
 |      1    2. Bee
 |      2    3. Cat
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
 |
 |      >>> s.str.strip('123.!? \n\t')
 |      0    Ant
 |      1    Bee
 |      2    Cat
 |      3    NaN
 |      4    NaN
 |      5    NaN
 |      dtype: object
 |
 |  slice(self, start=None, stop=None, step=None)
 |      Slice substrings from each element in the Series or Index.
 |
 |      Parameters
 |      ----------
 |      start : int, optional
 |          Start position for slice operation.
 |      stop : int, optional
 |          Stop position for slice operation.
 |      step : int, optional
 |          Step size for slice operation.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |          Series or Index from sliced substring from original string object.
 |
 |      See Also
 |      --------
 |      Series.str.slice_replace : Replace a slice with a string.
 |      Series.str.get : Return element at position.
 |          Equivalent to `Series.str.slice(start=i, stop=i+1)` with `i`
 |          being the position.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(["koala", "dog", "chameleon"])
 |      >>> s
 |      0        koala
 |      1          dog
 |      2    chameleon
 |      dtype: object
 |
 |      >>> s.str.slice(start=1)
 |      0        oala
 |      1          og
 |      2    hameleon
 |      dtype: object
 |
 |      >>> s.str.slice(start=-1)
 |      0           a
 |      1           g
 |      2           n
 |      dtype: object
 |
 |      >>> s.str.slice(stop=2)
 |      0    ko
 |      1    do
 |      2    ch
 |      dtype: object
 |
 |      >>> s.str.slice(step=2)
 |      0      kaa
 |      1       dg
 |      2    caeen
 |      dtype: object
 |
 |      >>> s.str.slice(start=0, stop=5, step=3)
 |      0    kl
 |      1     d
 |      2    cm
 |      dtype: object
 |
 |      Equivalent behaviour to:
 |
 |      >>> s.str[0:5:3]
 |      0    kl
 |      1     d
 |      2    cm
 |      dtype: object
 |
 |  slice_replace(self, start=None, stop=None, repl=None)
 |      Replace a positional slice of a string with another value.
 |
 |      Parameters
 |      ----------
 |      start : int, optional
 |          Left index position to use for the slice. If not specified (None),
 |          the slice is unbounded on the left, i.e. slice from the start
 |          of the string.
 |      stop : int, optional
 |          Right index position to use for the slice. If not specified (None),
 |          the slice is unbounded on the right, i.e. slice until the
 |          end of the string.
 |      repl : str, optional
```

```
 |          String for replacement. If not specified (None), the sliced region
 |          is replaced with an empty string.
 |
 |      Returns
 |      -------
 |      Series or Index
 |          Same type as the original object.
 |
 |      See Also
 |      --------
 |      Series.str.slice : Just slicing without replacement.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['a', 'ab', 'abc', 'abdc', 'abcde'])
 |      >>> s
 |      0        a
 |      1       ab
 |      2      abc
 |      3     abdc
 |      4    abcde
 |      dtype: object
 |
 |      Specify just `start`, meaning replace `start` until the end of the
 |      string with `repl`.
 |
 |      >>> s.str.slice_replace(1, repl='X')
 |      0    aX
 |      1    aX
 |      2    aX
 |      3    aX
 |      4    aX
 |      dtype: object
 |
 |      Specify just `stop`, meaning the start of the string to `stop` is
replaced
 |      with `repl`, and the rest of the string is included.
 |
 |      >>> s.str.slice_replace(stop=2, repl='X')
 |      0       X
 |      1       X
 |      2      Xc
 |      3     Xdc
 |      4    Xcde
 |      dtype: object
 |
 |      Specify `start` and `stop`, meaning the slice from `start` to `stop` is
 |      replaced with `repl`. Everything before or after `start` and `stop` is
 |      included as is.
 |
 |      >>> s.str.slice_replace(start=1, stop=3, repl='X')
 |      0      aX
 |      1      aX
 |      2      aX
 |      3     aXc
 |      4    aXde
 |      dtype: object
 |
 |  split(self, pat: 'str | re.Pattern | None' = None, *, n=-1, expand: 'bool' =
False, regex: 'bool | None' = None)
 |      Split strings around given separator/delimiter.
 |
 |      Splits the string in the Series/Index from the beginning,
 |      at the specified delimiter string.
 |
 |      Parameters
 |      ----------
 |      pat : str or compiled regex, optional
 |          String or regular expression to split on.
 |          If not specified, split on whitespace.
 |      n : int, default -1 (all)
 |          Limit number of splits in output.
 |          ``None``, 0 and -1 will be interpreted as return all splits.
 |      expand : bool, default False
 |          Expand the split strings into separate columns.
 |
 |          - If ``True``, return DataFrame/MultiIndex expanding dimensionality.
 |          - If ``False``, return Series/Index, containing lists of strings.
 |
 |      regex : bool, default None
 |          Determines if the passed-in pattern is a regular expression:
 |
 |          - If ``True``, assumes the passed-in pattern is a regular expression
 |          - If ``False``, treats the pattern as a literal string.
 |          - If ``None`` and `pat` length is 1, treats `pat` as a literal
string.
 |          - If ``None`` and `pat` length is not 1, treats `pat` as a regular
expression.
 |          - Cannot be set to False if `pat` is a compiled regex
 |
 |          .. versionadded:: 1.4.0
 |
 |      Returns
 |      -------
 |      Series, Index, DataFrame or MultiIndex
 |          Type matches caller unless ``expand=True`` (see Notes).
 |
 |                        Raises
 |                        ------
 |                        ValueError
 |                            * if `regex` is False and `pat` is a compiled
regex
 |
 |       See Also
 |       --------
 |       Series.str.split : Split strings around given separator/delimiter.
 |       Series.str.rsplit : Splits string around given separator/delimiter,
 |           starting from the right.
 |       Series.str.join : Join lists contained as elements in the Series/Index
 |           with passed delimiter.
 |       str.split : Standard library version for split.
 |       str.rsplit : Standard library version for rsplit.
 |
 |       Notes
 |       -----
 |       The handling of the `n` keyword depends on the number of found splits:
 |
 |       - If found splits > `n`,  make first `n` splits only
 |       - If found splits <= `n`, make all splits
 |       - If for a certain row the number of found splits < `n`,
 |         append `None` for padding up to `n` if ``expand=True``
 |
 |       If using ``expand=True``, Series and Index callers return DataFrame and
 |       MultiIndex objects, respectively.
 |
 |       Use of `regex =False` with a `pat` as a compiled regex will raise an
error.
 |
 |       Examples
 |       --------
 |       >>> s = pd.Series(
 |       …     [
 |       …         "this is a regular sentence",
 |       …         "https://docs.python.org/3/tutorial/index.html",
 |       …         np.nan
 |       …     ]
 |       … )
 |       >>> s
 |       0                       this is a regular sentence
 |       1    https://docs.python.org/3/tutorial/index.html
 |       2                                              NaN
 |       dtype: object
 |
 |       In the default setting, the string is split by whitespace.
 |
 |      >>> s.str.split()
 |      0                  [this, is, a, regular, sentence]
 |      1    [https://docs.python.org/3/tutorial/index.html]
 |      2                                                NaN
 |      dtype: object
 |
 |      Without the `n` parameter, the outputs of `rsplit` and `split`
 |      are identical.
 |
 |      >>> s.str.rsplit()
 |      0                  [this, is, a, regular, sentence]
 |      1    [https://docs.python.org/3/tutorial/index.html]
 |      2                                                NaN
 |      dtype: object
 |
 |      The `n` parameter can be used to limit the number of splits on the
 |      delimiter. The outputs of `split` and `rsplit` are different.
 |
 |      >>> s.str.split(n=2)
 |      0                    [this, is, a regular sentence]
 |      1    [https://docs.python.org/3/tutorial/index.html]
 |      2                                                NaN
 |      dtype: object
 |
 |      >>> s.str.rsplit(n=2)
 |      0                    [this is a, regular, sentence]
 |      1    [https://docs.python.org/3/tutorial/index.html]
 |      2                                                NaN
 |      dtype: object
 |
 |      The `pat` parameter can be used to split by other characters.
 |
 |      >>> s.str.split(pat="/")
 |      0                        [this is a regular sentence]
 |      1    [https:, , docs.python.org, 3, tutorial, index…
 |      2                                                  NaN
 |      dtype: object
 |
 |      When using ``expand=True``, the split elements will expand out into
 |      separate columns. If NaN is present, it is propagated throughout
 |      the columns during the split.
 |
 |      >>> s.str.split(expand=True)
 |                                                     0     1     2        3
4
 |      0                                           this    is     a  regular
sentence
```

```
 |      1  https://docs.python.org/3/tutorial/index.html  None  None     None
None
 |      2                                            NaN   NaN   NaN      NaN
NaN
 |
 |      For slightly more complex use cases like splitting the html document
name
 |      from a url, a combination of parameter settings can be used.
 |
 |      >>> s.str.rsplit("/", n=1, expand=True)
 |                                          0           1
 |      0          this is a regular sentence        None
 |      1  https://docs.python.org/3/tutorial  index.html
 |      2                                 NaN         NaN
 |
 |      Remember to escape special characters when explicitly using regular
expressions.
 |
 |      >>> s = pd.Series(["foo and bar plus baz"])
 |      >>> s.str.split(r"and|plus", expand=True)
 |          0   1   2
 |      0 foo bar baz
 |
 |      Regular expressions can be used to handle urls or file names.
 |      When `pat` is a string and ``regex=None`` (the default), the given `pat`
is compiled
 |      as a regex only if ``len(pat) != 1``.
 |
 |      >>> s = pd.Series(['foojpgbar.jpg'])
 |      >>> s.str.split(r".", expand=True)
 |                0    1
 |      0  foojpgbar  jpg
 |
 |      >>> s.str.split(r"\.jpg", expand=True)
 |                0 1
 |      0  foojpgbar
 |
 |      When ``regex=True``, `pat` is interpreted as a regex
 |
 |      >>> s.str.split(r"\.jpg", regex=True, expand=True)
 |                0 1
 |      0  foojpgbar
 |
 |      A compiled regex can be passed as `pat`
 |
 |      >>> import re
 |      >>> s.str.split(re.compile(r"\.jpg"), expand=True)
 |                0 1
 |      0  foojpgbar
 |
 |      When ``regex=False``, `pat` is interpreted as the string itself
 |
 |      >>> s.str.split(r"\.jpg", regex=False, expand=True)
 |                    0
 |      0  foojpgbar.jpg
 |
 |  startswith(self, pat: 'str | tuple[str, …]', na: 'Scalar | None' = None)
-> 'Series | Index'
 |      Test if the start of each string element matches a pattern.
 |
 |      Equivalent to :meth:`str.startswith`.
 |
 |      Parameters
 |      ----------
 |      pat : str or tuple[str, …]
 |          Character sequence or tuple of strings. Regular expressions are not
 |          accepted.
 |      na : object, default NaN
 |          Object shown if element tested is not a string. The default depends
 |          on dtype of the array. For object-dtype, ``numpy.nan`` is used.
 |          For ``StringDtype``, ``pandas.NA`` is used.
 |
 |      Returns
 |      -------
 |      Series or Index of bool
 |          A Series of booleans indicating whether the given pattern matches
 |          the start of each string element.
 |
 |      See Also
 |      --------
 |      str.startswith : Python standard library string method.
 |      Series.str.endswith : Same as startswith, but tests the end of string.
 |      Series.str.contains : Tests if string element contains a pattern.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['bat', 'Bear', 'cat', np.nan])
 |      >>> s
 |      0     bat
 |      1    Bear
 |      2     cat
 |      3     NaN
 |      dtype: object
 |
 |      >>> s.str.startswith('b')
 |      0     True
 |      1    False
 |      2    False
 |      3      NaN
 |      dtype: object
 |
 |      >>> s.str.startswith(('b', 'B'))
 |      0     True
 |      1     True
 |      2    False
 |      3      NaN
 |      dtype: object
 |
 |      Specifying `na` to be `False` instead of `NaN`.
 |
 |      >>> s.str.startswith('b', na=False)
 |      0     True
 |      1    False
 |      2    False
 |      3    False
 |      dtype: bool
 |
 |  strip(self, to_strip=None)
 |      Remove leading and trailing characters.
 |
 |      Strip whitespaces (including newlines) or a set of specified characters
 |      from each string in the Series/Index from left and right sides.
 |      Replaces any non-strings in Series with NaNs.
 |      Equivalent to :meth:`str.strip`.
 |
 |      Parameters
 |      ----------
 |      to_strip : str or None, default None
 |          Specifying the set of characters to be removed.
 |          All combinations of this set of characters will be stripped.
 |          If None then whitespaces are removed.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.strip : Remove leading and trailing characters in
Series/Index.
 |      Series.str.lstrip : Remove leading characters in Series/Index.
 |      Series.str.rstrip : Remove trailing characters in Series/Index.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['1. Ant.  ', '2. Bee!\n', '3. Cat?\t', np.nan, 10,
True])
 |      >>> s
 |      0    1. Ant.
 |      1    2. Bee!\n
 |      2    3. Cat?\t
 |      3          NaN
 |      4           10
 |      5         True
 |      dtype: object
 |
 |      >>> s.str.strip()
 |      0    1. Ant.
 |      1    2. Bee!
 |      2    3. Cat?
 |      3        NaN
 |      4        NaN
 |      5        NaN
 |      dtype: object
 |
 |      >>> s.str.lstrip('123.')
 |      0    Ant.
 |      1    Bee!\n
 |      2    Cat?\t
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
 |
 |      >>> s.str.rstrip('.!? \n\t')
 |      0    1. Ant
 |      1    2. Bee
 |      2    3. Cat
 |      3       NaN
 |      4       NaN
 |      5       NaN
 |      dtype: object
 |
 |      >>> s.str.strip('123.!? \n\t')
 |      0    Ant
 |      1    Bee
 |      2    Cat
 |      3    NaN
 |      4    NaN
 |      5    NaN
 |      dtype: object
 |
 |  swapcase(self)
 |      Convert strings in the Series/Index to be swapcased.
 |
 |      Equivalent to :meth:`str.swapcase`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1              CAPITALS
 |      2    this is a sentence
 |      3              SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                 lower
 |      1              capitals
 |      2    this is a sentence
 |      3              swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                 LOWER
 |      1              CAPITALS
 |      2    THIS IS A SENTENCE
 |      3              SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                 Lower
 |      1              Capitals
 |      2    This Is A Sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                 Lower
 |      1              Capitals
 |      2    This is a sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                 LOWER
 |      1              capitals
 |      2    THIS IS A SENTENCE
 |      3              sWaPcAsE
 |      dtype: object
 |
 |  title(self)
 |      Convert strings in the Series/Index to titlecase.
 |
 |      Equivalent to :meth:`str.title`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1               CAPITALS
 |      2    this is a sentence
 |      3               SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                  lower
 |      1               capitals
 |      2    this is a sentence
 |      3               swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                  LOWER
 |      1               CAPITALS
 |      2    THIS IS A SENTENCE
 |      3               SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                  Lower
 |      1               Capitals
 |      2    This Is A Sentence
 |      3               Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                  Lower
 |      1               Capitals
 |      2    This is a sentence
 |      3               Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                  LOWER
 |      1               capitals
 |      2    THIS IS A SENTENCE
 |      3               sWaPcAsE
 |      dtype: object
 |
 |  translate(self, table)
 |      Map all characters in the string through the given mapping table.
 |
 |      Equivalent to standard :meth:`str.translate`.
 |
 |      Parameters
 |      ----------
 |      table : dict
```

```
 |          Table is a mapping of Unicode ordinals to Unicode ordinals, strings,
or
 |          None. Unmapped characters are left untouched.
 |          Characters mapped to None are deleted. :meth:`str.maketrans` is a
 |          helper function for making translation tables.
 |
 |      Returns
 |      -------
 |      Series or Index
 |
 |      Examples
 |      --------
 |      >>> ser = pd.Series(["El niño", "Françoise"])
 |      >>> mytable = str.maketrans({'ñ': 'n', 'ç': 'c'})
 |      >>> ser.str.translate(mytable)
 |      0   El nino
 |      1   Francoise
 |      dtype: object
 |
 |  upper(self)
 |      Convert strings in the Series/Index to uppercase.
 |
 |      Equivalent to :meth:`str.upper`.
 |
 |      Returns
 |      -------
 |      Series or Index of object
 |
 |      See Also
 |      --------
 |      Series.str.lower : Converts all characters to lowercase.
 |      Series.str.upper : Converts all characters to uppercase.
 |      Series.str.title : Converts first character of each word to uppercase
and
 |          remaining to lowercase.
 |      Series.str.capitalize : Converts first character to uppercase and
 |          remaining to lowercase.
 |      Series.str.swapcase : Converts uppercase to lowercase and lowercase to
 |          uppercase.
 |      Series.str.casefold: Removes all case distinctions in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['lower', 'CAPITALS', 'this is a sentence',
'SwApCaSe'])
 |      >>> s
 |      0                 lower
 |      1              CAPITALS
 |      2    this is a sentence
 |      3              SwApCaSe
 |      dtype: object
 |
 |      >>> s.str.lower()
 |      0                 lower
 |      1              capitals
 |      2    this is a sentence
 |      3              swapcase
 |      dtype: object
 |
 |      >>> s.str.upper()
 |      0                 LOWER
 |      1              CAPITALS
 |      2    THIS IS A SENTENCE
 |      3              SWAPCASE
 |      dtype: object
 |
 |      >>> s.str.title()
 |      0                 Lower
 |      1              Capitals
 |      2    This Is A Sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.capitalize()
 |      0                 Lower
 |      1              Capitals
 |      2    This is a sentence
 |      3              Swapcase
 |      dtype: object
 |
 |      >>> s.str.swapcase()
 |      0                 LOWER
 |      1              capitals
 |      2    THIS IS A SENTENCE
 |      3              sWaPcAsE
 |      dtype: object
 |
 |  wrap(self, width: 'int', **kwargs)
 |      Wrap strings in Series/Index at specified line width.
 |
 |      This method has the same keyword parameters and defaults as
 |      :class:`textwrap.TextWrapper`.
 |
 |      Parameters
 |      ----------
 |      width : int
```

```
 |          Maximum line width.
 |      expand_tabs : bool, optional
 |          If True, tab characters will be expanded to spaces (default: True).
 |      replace_whitespace : bool, optional
 |          If True, each whitespace character (as defined by string.whitespace)
 |          remaining after tab expansion will be replaced by a single space
 |          (default: True).
 |      drop_whitespace : bool, optional
 |          If True, whitespace that, after wrapping, happens to end up at the
 |          beginning or end of a line is dropped (default: True).
 |      break_long_words : bool, optional
 |          If True, then words longer than width will be broken in order to
ensure
 |          that no lines are longer than width. If it is false, long words will
 |          not be broken, and some lines may be longer than width (default:
True).
 |      break_on_hyphens : bool, optional
 |          If True, wrapping will occur preferably on whitespace and right
after
 |          hyphens in compound words, as it is customary in English. If false,
 |          only whitespaces will be considered as potentially good places for
line
 |          breaks, but you need to set break_long_words to false if you want
truly
 |          insecable words (default: True).
 |
 |      Returns
 |      -------
 |      Series or Index
 |
 |      Notes
 |      -----
 |      Internally, this method uses a :class:`textwrap.TextWrapper` instance
with
 |      default settings. To achieve behavior matching R's stringr library
str_wrap
 |      function, use the arguments:
 |
 |      - expand_tabs = False
 |      - replace_whitespace = True
 |      - drop_whitespace = True
 |      - break_long_words = False
 |      - break_on_hyphens = False
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['line to be wrapped', 'another line to be wrapped'])
 |      >>> s.str.wrap(12)
 |      0             line to be\nwrapped
 |      1    another line\nto be\nwrapped
 |      dtype: object
 |
 |  zfill(self, width: 'int')
 |      Pad strings in the Series/Index by prepending '0' characters.
 |
 |      Strings in the Series/Index are padded with '0' characters on the
 |      left of the string to reach a total string length  `width`. Strings
 |      in the Series/Index with length greater or equal to `width` are
 |      unchanged.
 |
 |      Parameters
 |      ----------
 |      width : int
 |          Minimum length of resulting string; strings with length less
 |          than `width` be prepended with '0' characters.
 |
 |      Returns
 |      -------
 |      Series/Index of objects.
 |
 |      See Also
 |      --------
 |      Series.str.rjust : Fills the left side of strings with an arbitrary
 |          character.
 |      Series.str.ljust : Fills the right side of strings with an arbitrary
 |          character.
 |      Series.str.pad : Fills the specified sides of strings with an arbitrary
 |          character.
 |      Series.str.center : Fills both sides of strings with an arbitrary
 |          character.
 |
 |      Notes
 |      -----
 |      Differs from :meth:`str.zfill` which has special handling
 |      for '+'/'-' in the string.
 |
 |      Examples
 |      --------
 |      >>> s = pd.Series(['-1', '1', '1000', 10, np.nan])
 |      >>> s
 |      0      -1
 |      1       1
 |      2    1000
 |      3      10
 |      4     NaN
 |      dtype: object
```

```
|
|       Note that ``10`` and ``NaN`` are not strings, therefore they are
|       converted to ``NaN``. The minus sign in ``'-1'`` is treated as a
|       special character and the zero is added to the right of it
|       (:meth:`str.zfill` would have moved it to the left). ``1000``
|       remains unchanged as it is longer than `width`.
|
|       >>> s.str.zfill(3)
|       0     -01
|       1     001
|       2    1000
|       3     NaN
|       4     NaN
|       dtype: object
|
|   ----------------------------------------------------------------------
|   Data and other attributes defined here:
|
|   __annotations__ = {'_doc_args': 'dict[str, dict[str, str]]'}
|
|   ----------------------------------------------------------------------
|   Methods inherited from pandas.core.base.NoNewAttributesMixin:
|
|   __setattr__(self, key: 'str', value) -> 'None'
|       Implement setattr(self, name, value).
|
|   ----------------------------------------------------------------------
|   Data descriptors inherited from pandas.core.base.NoNewAttributesMixin:
|
|   __dict__
|       dictionary for instance variables
|
|   __weakref__
|       list of weak references to the object
```

[ ]:
```
####
df['discounted_price']
####
```

[ ]:
```
0         399.0
1         199.0
2         199.0
3         329.0
4         154.0
          …
1460      379.0
```

```
1461    2280.0
1462    2219.0
1463    1399.0
1464    2863.0
Name: discounted_price, Length: 1465, dtype: float64
```

[ ]:
```
####
df['discount_percentage']
####
```

[ ]:
```
0       64%
1       43%
2       90%
3       53%
4       61%
       …
1460    59%
1461    25%
1462    28%
1463    26%
1464    22%
Name: discount_percentage, Length: 1465, dtype: object
```

[ ]:
```
# Changing Datatype and values in Discount Percentage

df['discount_percentage'] = df['discount_percentage'].str.replace('%', '').
 ↪astype('float64')

df['discount_percentage'] = df['discount_percentage'] / 100

df['discount_percentage'].head()
```

[ ]:
```
0    0.64
1    0.43
2    0.90
3    0.53
4    0.61
Name: discount_percentage, dtype: float64
```

[ ]:
```
# Finding unusual string in rating column
df['rating'].value_counts()
```

[ ]:
```
rating
4.1    244
4.3    230
4.2    228
4.0    129
```

```
3.9    123
4.4    123
3.8     86
4.5     75
4       52
3.7     42
3.6     35
3.5     26
4.6     17
3.3     16
3.4     10
4.7      6
3.1      4
3.0      3
4.8      3
5.0      3
2.8      2
3.2      2
2.3      1
|        1
2        1
3        1
2.6      1
2.9      1
Name: count, dtype: int64
```

[ ]:
```
####
df['rating']
####
```

[ ]:
```
0       4.2
1       4.0
2       3.9
3       4.2
4       4.2
       …
1460      4
1461    4.1
1462    3.6
1463      4
1464    4.3
Name: rating, Length: 1465, dtype: object
```

[ ]:
```
# Check the strange row
df.query('rating == "|"')
```

```
[ ]:       product_id                                       product_name  \
     1279  B08L12N5H1  Eureka Forbes car Vac 100 Watts Powerful Sucti…

                                                    category  discounted_price  \
     1279  Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cle…            2099.0

           actual_price  discount_percentage rating rating_count  \
     1279        2499.0                 0.16      |          992

                                              about_product  \
     1279  No Installation is provided for this product|1…

                                                    user_id  \
     1279  AGTDSNT2FKVYEPDPXAA673AIS44A,AER2XFSWNN4LAUCJ5…

                                                  user_name  \
     1279  Divya,Dr Nefario,Deekshith,Preeti,Prasanth R,P…

                                                  review_id  \
     1279  R2KKTKM4M9RDVJ,R1O692MZOBTE79,R2WRSEWL56SOS4,R…

                                               review_title  \
     1279  Decent product,doesn't pick up sand,Ok ok,Must…

                                             review_content  \
     1279  Does the job well,doesn't work on sand. though…

                                                   img_link  \
     1279  https://m.media-amazon.com/images/W/WEBP_40237…

                                               product_link
     1279  https://www.amazon.in/Eureka-Forbes-Vacuum-Cle…
```

```
[ ]: # Changing Rating Columns Data Type

     df['rating'] = df['rating'].str.replace('|', '3.9').astype('float64')
```

```
[ ]: # Changing 'rating_count' Column Data Type

     df['rating_count'] = df['rating_count'].str.replace(',','').astype('float64')
```

```
[ ]: ####
     df['rating_count']
     ####
```

```
[ ]: 0       24,269
     1       43,994
```

```
2        7,928
3       94,363
4       16,905
          …
1460     1,090
1461     4,118
1462       468
1463     8,031
1464     6,987
Name: rating_count, Length: 1465, dtype: object
```

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   float64
 4   actual_price         1465 non-null   float64
 5   discount_percentage  1465 non-null   float64
 6   rating               1465 non-null   float64
 7   rating_count         1463 non-null   float64
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: float64(5), object(11)
memory usage: 183.3+ KB
```

```
[ ]: # Descriptive Statistics
```

```
[ ]: # Descriptive statistics are a collection of quantitative measures that␣
      ↪summarize and descrive the main characteristice of a dataset
```

```
[ ]: df.describe() # Note there are four columns only displayed because we converted␣
      ↪only 4 to float datatype and statistics functions are only applied to␣
      ↪numerical values so only 4 are displayed
```

```
[ ]:        discounted_price  actual_price  discount_percentage   rating  \
     count            1465.00       1465.00              1465.00  1465.00
     mean             3125.31       5444.99                 0.48     4.10
     std              6944.30      10874.83                 0.22     0.29
     min                39.00         39.00                 0.00     2.00
     25%               325.00        800.00                 0.32     4.00
     50%               799.00       1650.00                 0.50     4.10
     75%              1999.00       4295.00                 0.63     4.30
     max             77990.00     139900.00                 0.94     5.00

            rating_count
     count       1463.00
     mean       18295.54
     std        42753.86
     min            2.00
     25%         1186.00
     50%         5179.00
     75%        17336.50
     max       426973.00
```

```
[ ]: # Dealing with the missing values
```

```
[ ]: # Missing Values
     df.isnull().sum().sort_values(ascending = False)
```

```
[ ]: rating_count           2
     product_id             0
     category               0
     product_name           0
     discounted_price       0
     actual_price           0
     discount_percentage    0
     rating                 0
     about_product          0
     user_id                0
     user_name              0
     review_id              0
     review_title           0
     review_content         0
     img_link               0
     product_link           0
     dtype: int64
```

```
[ ]: # Finding missing values percentage in the data
     round(df.isnull().sum() / len(df) * 100, 2).sort_values(ascending=False)
```

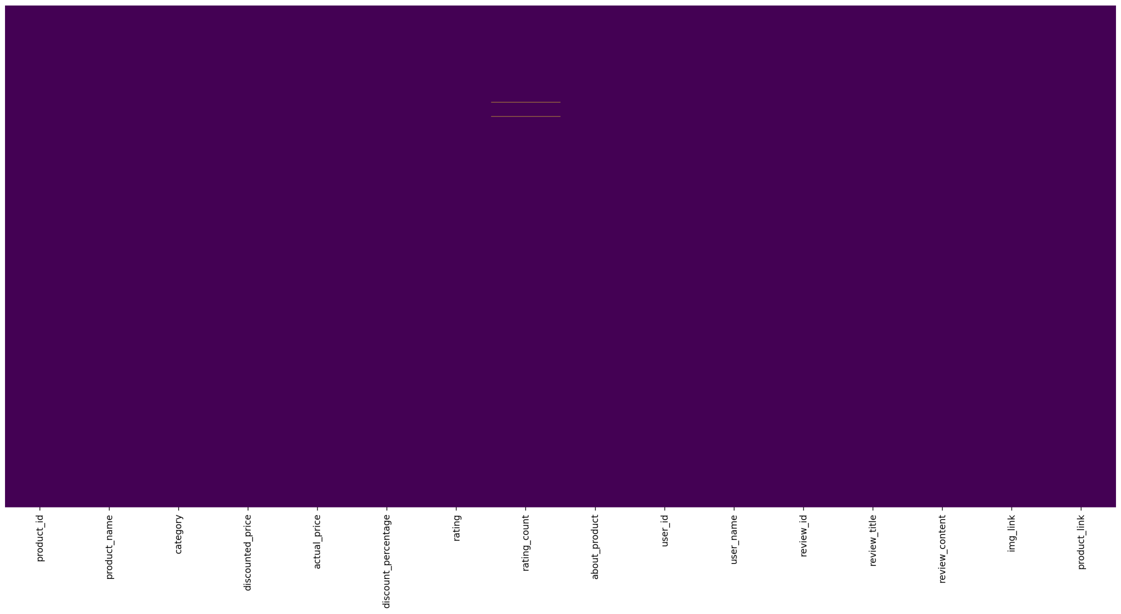
[ ]:
```
rating_count           0.14
product_id             0.00
category               0.00
product_name           0.00
discounted_price       0.00
actual_price           0.00
discount_percentage    0.00
rating                 0.00
about_product          0.00
user_id                0.00
user_name              0.00
review_id              0.00
review_title           0.00
review_content         0.00
img_link               0.00
product_link           0.00
dtype: float64
```

[ ]:
```
# Find total number of missing values
df.isnull().sum().sum()
```

[ ]:
```
np.int64(2)
```

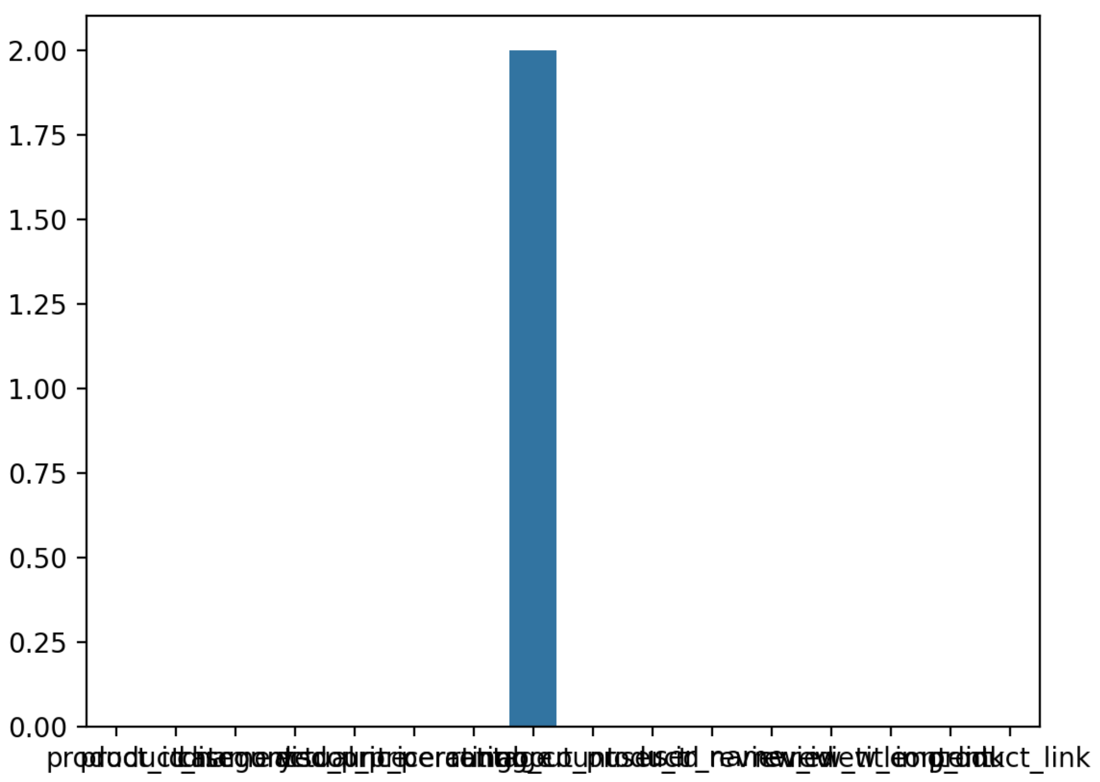
[ ]:
```
# Let's plot the missing values

# make a figure size
plt.figure(figsize=(22, 10))
# plot the null values in each column
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

[ ]:
```
<Axes: >
```



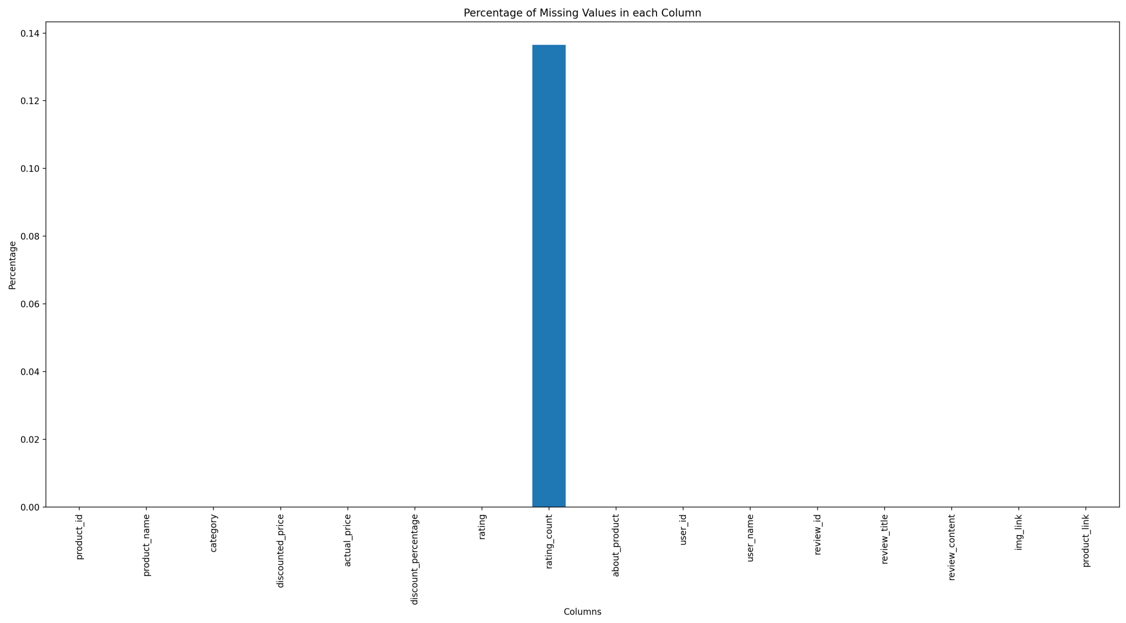
```
[ ]: ####
     sns.barplot(df.isnull().sum())
     ####
```

```
[ ]: <Axes: >
```



```
[ ]: # Plot the missing values by percentage

     plt.figure(figsize=(22, 10))
     # plot the null values by their percentage in each column
     missing_percentage = df.isnull().sum()/len(df)*100
     missing_percentage.plot(kind='bar')
     # add the labels
     plt.xlabel('Columns')
     plt.ylabel('Percentage')
     plt.title('Percentage of Missing Values in each Column')
```

```
[ ]: Text(0.5, 1.0, 'Percentage of Missing Values in each Column')
```



```
[ ]: ####
     missing_percentage
     ####
```

```
[ ]: product_id              0.00
     product_name            0.00
     category                0.00
     discounted_price        0.00
     actual_price            0.00
     discount_percentage     0.00
     rating                  0.00
     rating_count            0.14
     about_product           0.00
     user_id                 0.00
     user_name               0.00
     review_id               0.00
     review_title            0.00
     review_content          0.00
     img_link                0.00
     product_link            0.00
     dtype: float64
```

```
[ ]: # We are only viewing the rows where there are null values in the column
```

```
[ ]: df[df['rating_count'].isnull()].head(5)
[ ]:     product_id                                       product_name  \
    282  B0B94JPY2N  Amazon Brand - Solimo 65W Fast Charging Braide…
    324  B0BQRJ3C47  REDTECH USB-C to Lightning Cable 3.3FT, [Apple…

                                                  category  discounted_price  \
    282  Computers&Accessories|Accessories&Peripherals|…             199.0
    324  Computers&Accessories|Accessories&Peripherals|…             249.0

         actual_price  discount_percentage  rating  rating_count  \
    282         999.0                 0.80     3.0           NaN
    324         999.0                 0.75     5.0           NaN

                                             about_product  \
    282  USB C to C Cable: This cable has type C connec…
    324   [The Fastest Charge] - This iPhone USB C cabl…

                              user_id    user_name      review_id  \
    282  AE7CFHY23VAJT2FI4NZKKP6GS2UQ       Pranav   RUB7U91HVZ30
    324  AGJC5O5H5BBXWUV7WRIEIOOR3TVQ  Abdul Gafur  RQXD5SAMMPC6L

                                  review_title  \
    282  The cable works but is not 65W as advertised
    324                            Awesome Product

                                            review_content  \
    282  I have a pd supported car charger and I bought…
    324  Quick delivery.Awesome ProductPacking was good…

                                                  img_link  \
    282  https://m.media-amazon.com/images/W/WEBP_40237…
    324  https://m.media-amazon.com/images/I/31-q0xhaTA…

                                              product_link
    282  https://www.amazon.in/Amazon-Brand-Charging-Su…
    324  https://www.amazon.in/REDTECH-Lightning-Certif…
```

```
[ ]: # Impute missing values
     df['rating_count'] = df.rating_count.fillna(value=df['rating_count'].median())
```

```
[ ]: df.isnull().sum().sort_values(ascending=False)
```

```
[ ]: product_id             0
     product_name           0
     category               0
     discounted_price       0
     actual_price           0
     discount_percentage    0
```

```
rating                 0
rating_count           0
about_product          0
user_id                0
user_name              0
review_id              0
review_title           0
review_content         0
img_link               0
product_link           0
dtype: int64
```

```
[ ]: # Milestone 1: We have cleaned the dataset from null values
```

```
[ ]: # Finding Duplications and Analyse them
```

```
[ ]: # Find Duplicate
     df.duplicated().any()
```

```
[ ]: np.False_
```

```
[ ]: ####
     df.duplicated()
     ####
```

```
[ ]: 0       False
     1       False
     2       False
     3       False
     4       False
             …
     1460    False
     1461    False
     1462    False
     1463    False
     1464    False
     Length: 1465, dtype: bool
```

```
[ ]: ####
     help(df.any)
     ####
```

```
Help on method any in module pandas.core.frame:

any(*, axis: 'Axis | None' = 0, bool_only: 'bool' = False, skipna: 'bool' =
True, **kwargs) -> 'Series | bool' method of pandas.core.frame.DataFrame
instance
```

```
    Return whether any element is True, potentially over an axis.

    Returns False unless there is at least one element within a series or
    along a Dataframe axis that is True or equivalent (e.g. non-zero or
    non-empty).

    Parameters
    ----------
    axis : {0 or 'index', 1 or 'columns', None}, default 0
        Indicate which axis or axes should be reduced. For `Series` this
parameter
        is unused and defaults to 0.

        * 0 / 'index' : reduce the index, return a Series whose index is the
          original column labels.
        * 1 / 'columns' : reduce the columns, return a Series whose index is the
          original index.
        * None : reduce all axes, return a scalar.

    bool_only : bool, default False
        Include only boolean columns. Not implemented for Series.
    skipna : bool, default True
        Exclude NA/null values. If the entire row/column is NA and skipna is
        True, then the result will be False, as for an empty row/column.
        If skipna is False, then NA are treated as True, because these are not
        equal to zero.
    **kwargs : any, default None
        Additional keywords have no effect but might be accepted for
        compatibility with NumPy.

    Returns
    -------
    Series or DataFrame
        If level is specified, then, DataFrame is returned; otherwise, Series
        is returned.

    See Also
    --------
    numpy.any : Numpy version of this method.
    Series.any : Return whether any element is True.
    Series.all : Return whether all elements are True.
    DataFrame.any : Return whether any element is True over requested axis.
    DataFrame.all : Return whether all elements are True over requested axis.

    Examples
    --------
    **Series**
```

```
For Series input, the output is a scalar indicating whether any element
is True.

>>> pd.Series([False, False]).any()
False
>>> pd.Series([True, False]).any()
True
>>> pd.Series([], dtype="float64").any()
False
>>> pd.Series([np.nan]).any()
False
>>> pd.Series([np.nan]).any(skipna=False)
True

**DataFrame**

Whether each column contains at least one True element (the default).

>>> df = pd.DataFrame({"A": [1, 2], "B": [0, 2], "C": [0, 0]})
>>> df
   A  B  C
0  1  0  0
1  2  2  0

>>> df.any()
A     True
B     True
C    False
dtype: bool

Aggregating over the columns.

>>> df = pd.DataFrame({"A": [True, False], "B": [1, 2]})
>>> df
       A  B
0   True  1
1  False  2

>>> df.any(axis='columns')
0    True
1    True
dtype: bool

>>> df = pd.DataFrame({"A": [True, False], "B": [1, 0]})
>>> df
       A  B
0   True  1
1  False  0
```

```
    >>> df.any(axis='columns')
    0     True
    1    False
    dtype: bool

    Aggregating over the entire DataFrame with ``axis=None``.

    >>> df.any(axis=None)
    True

    `any` for an empty DataFrame is an empty Series.

    >>> pd.DataFrame([]).any()
    Series([], dtype: bool)
```

[ ]:
```
####
help(df.duplicated)
####
```

```
Help on method duplicated in module pandas.core.frame:

duplicated(subset: 'Hashable | Sequence[Hashable] | None' = None, keep:
'DropKeep' = 'first') -> 'Series' method of pandas.core.frame.DataFrame instance
    Return boolean Series denoting duplicate rows.

    Considering certain columns is optional.

    Parameters
    ----------
    subset : column label or sequence of labels, optional
        Only consider certain columns for identifying duplicates, by
        default use all of the columns.
    keep : {'first', 'last', False}, default 'first'
        Determines which duplicates (if any) to mark.

        - ``first`` : Mark duplicates as ``True`` except for the first
occurrence.
        - ``last`` : Mark duplicates as ``True`` except for the last occurrence.
        - False : Mark all duplicates as ``True``.

    Returns
    -------
    Series
        Boolean series for each duplicated rows.

    See Also
```

```
--------
Index.duplicated : Equivalent method on index.
Series.duplicated : Equivalent method on Series.
Series.drop_duplicates : Remove duplicate values from Series.
DataFrame.drop_duplicates : Remove duplicate values from DataFrame.

Examples
--------
Consider dataset containing ramen rating.

>>> df = pd.DataFrame({
…     'brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie'],
…     'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
…     'rating': [4, 4, 3.5, 15, 5]
… })
>>> df
    brand style  rating
0  Yum Yum   cup     4.0
1  Yum Yum   cup     4.0
2  Indomie   cup     3.5
3  Indomie  pack    15.0
4  Indomie  pack     5.0

By default, for each set of duplicated values, the first occurrence
is set on False and all others on True.

>>> df.duplicated()
0    False
1     True
2    False
3    False
4    False
dtype: bool

By using 'last', the last occurrence of each set of duplicated values
is set on False and all others on True.

>>> df.duplicated(keep='last')
0     True
1    False
2    False
3    False
4    False
dtype: bool

By setting ``keep`` on False, all duplicates are True.

>>> df.duplicated(keep=False)
```

```
0     True
1     True
2    False
3    False
4    False
dtype: bool

To find duplicates on specific column(s), use ``subset``.

>>> df.duplicated(subset=['brand'])
0    False
1     True
2    False
3     True
4     True
dtype: bool
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['product_id', 'product_name', 'category', 'discounted_price',
       'actual_price', 'discount_percentage', 'rating', 'rating_count',
       'about_product', 'user_id', 'user_name', 'review_id', 'review_title',
       'review_content', 'img_link', 'product_link'],
      dtype='object')
```

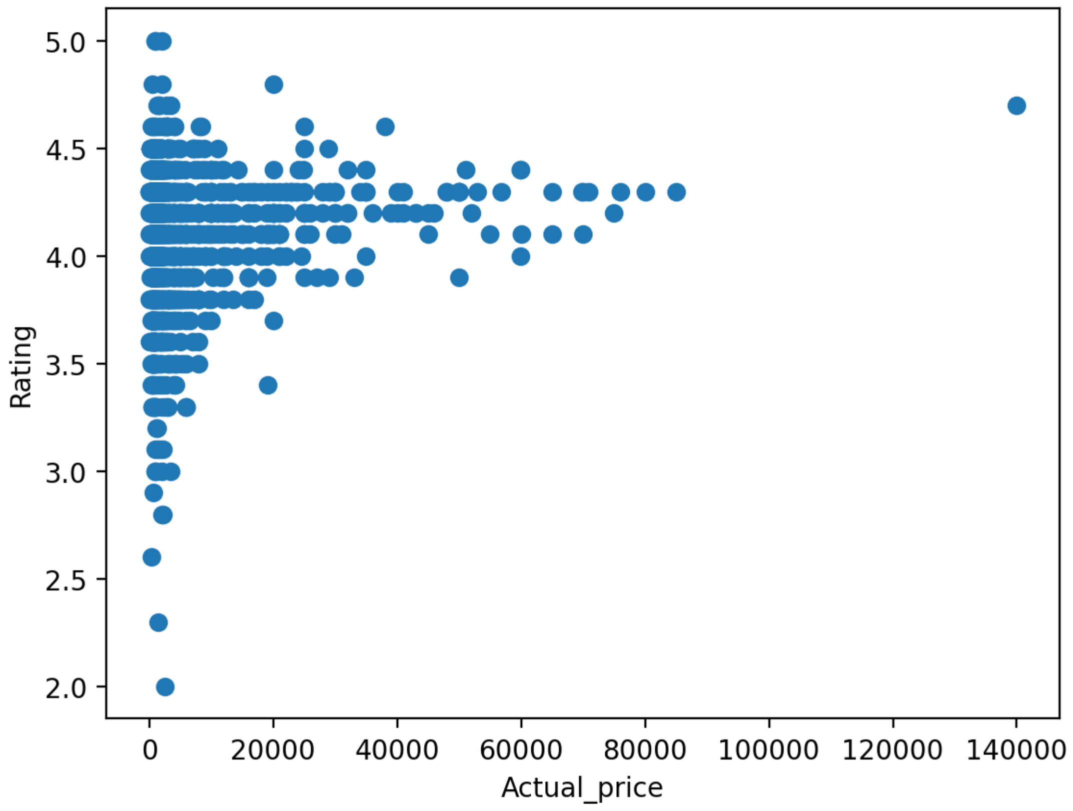
[ ]:
```
any_duplicates = df.duplicated(subset=['product_id', 'product_name',␣
↪'category', 'discounted_price',
                                       'actual_price', 'discount_percentage',␣
↪'rating',
                                       'rating_count', 'about_product',␣
↪'user_id', 'user_name',
                                       'review_id', 'review_title',␣
↪'review_content', 'img_link', 'product_link']).any()
```

[ ]:
```
any_duplicates
```

[ ]:
```
np.False_
```

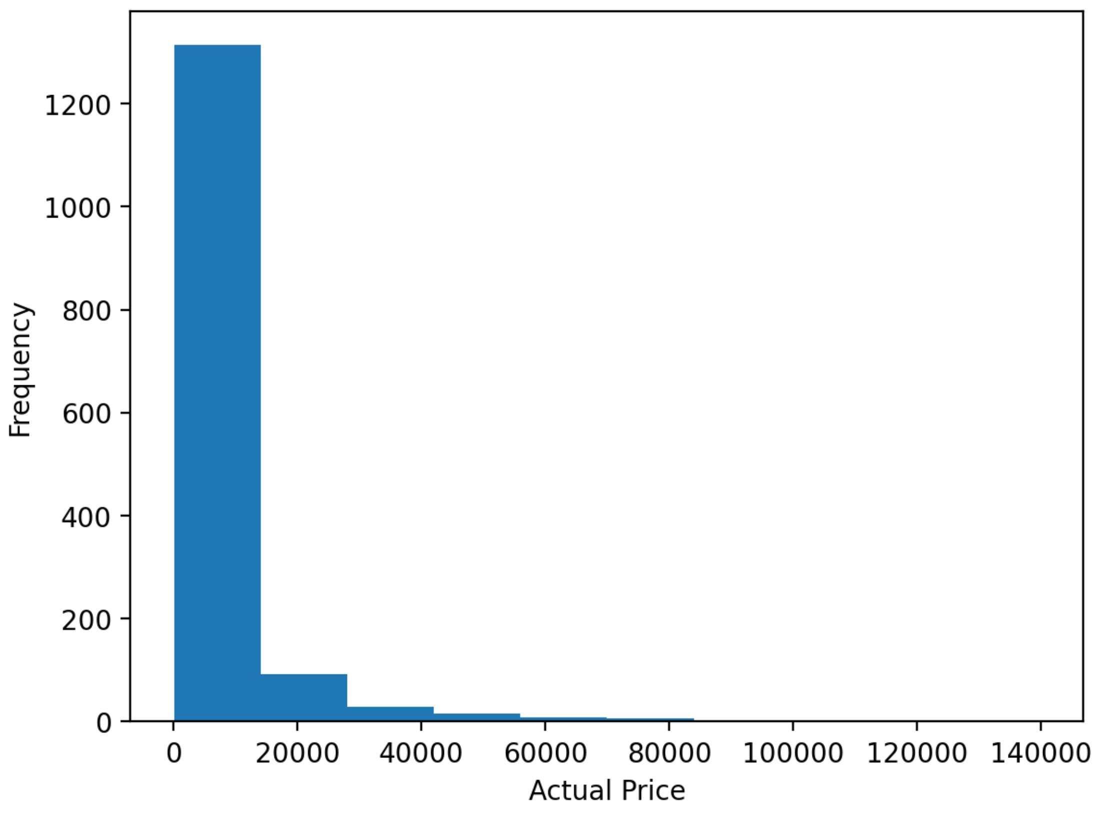
[ ]:
```
# Milestone 2: Hence no duplicates found
```

[ ]:
```
# Data Visualization
# Scatter Plot

# Plot actual_price vs. rating
plt.scatter(df['actual_price'], df['rating'])
plt.xlabel('Actual_price')
plt.ylabel('Rating')
plt.show()
```



[ ]:
```
# p.s inorder to remove warning do this,
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
# Histogram

# Plot distributions of actual_price
plt.hist(df['actual_price'])
plt.xlabel('Actual Price')
plt.ylabel('Frequency')
plt.show()
```



```
[ ]: from sklearn.preprocessing import LabelEncoder
     # label encode categorical variables

     le_product_id = LabelEncoder()
     le_category   = LabelEncoder()
     le_review_id  = LabelEncoder()
     le_review_content = LabelEncoder()
     le_product_name = LabelEncoder()
     le_user_name = LabelEncoder()
     le_about_product = LabelEncoder()
     le_user_id = LabelEncoder()
     le_review_title = LabelEncoder()
     le_img_link = LabelEncoder()
     le_product_link = LabelEncoder()

     df['product_id'] = le_product_id.fit_transform(df['product_id'])
     df['category']   = le_category.fit_transform(df['category'])
     df['review_id'] = le_review_id.fit_transform(df['review_id'])
     df['review_content'] = le_review_content.fit_transform(df['review_content'])
     df['product_name'] = le_product_name.fit_transform(df['product_name'])
     df['user_name'] = le_user_name.fit_transform(df['user_name'])
```

```
df['about_product'] = le_about_product.fit_transform(df['about_product'])
df['user_id'] = le_user_id.fit_transform(df['user_id'])
df['review_title'] = le_review_title.fit_transform(df['review_title'])
df['img_link'] = le_img_link.fit_transform(df['img_link'])
df['product_link'] = le_product_link.fit_transform(df['product_link'])
```

[ ]:
```
####
df['product_id']
####
```

[ ]:
```
0       B07JW9H4J1
1       B098NS6PVG
2       B096MSW6CT
3       B08HDJ86NZ
4       B08CF3B7N1
           …
1460    B08L7J3T31
1461    B01M6453MB
1462    B009P2LIL4
1463    B00J5DYCCA
1464    B01486F4G6
Name: product_id, Length: 1465, dtype: object
```

[ ]:
```
df['product_id']
```

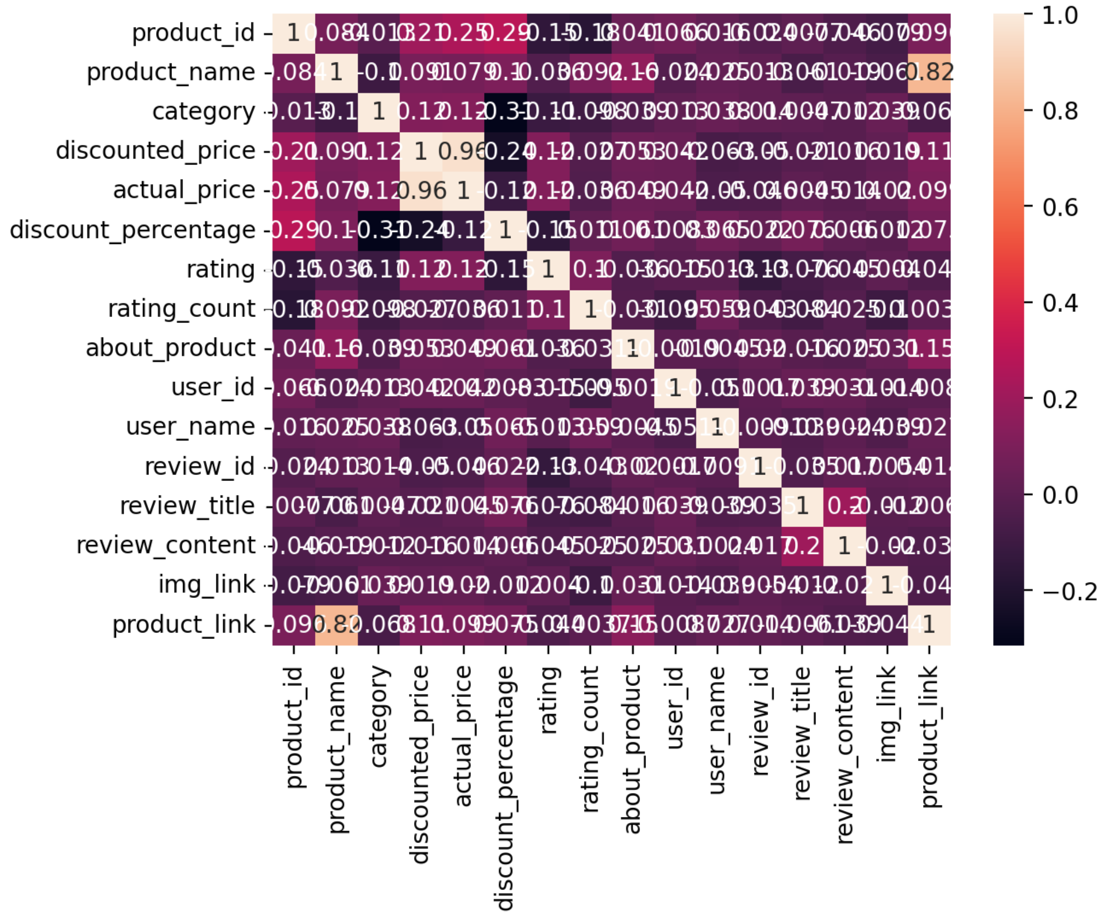
[ ]:
```
0       346
1       848
2       819
3       643
4       588
       …
1460    673
1461    201
1462     27
1463     61
1464    134
Name: product_id, Length: 1465, dtype: int64
```

[ ]:
```
####
help(LabelEncoder)
####
```

```
Help on class LabelEncoder in module sklearn.preprocessing._label:

class LabelEncoder(sklearn.base.TransformerMixin, sklearn.base.BaseEstimator)
 |  Encode target labels with value between 0 and n_classes-1.
 |
 |  This transformer should be used to encode target values, *i.e.* `y`, and
 |  not the input `X`.
 |
 |  Read more in the :ref:`User Guide <preprocessing_targets>`.
 |
 |  .. versionadded:: 0.12
 |
 |  Attributes
 |  ----------
 |  classes_ : ndarray of shape (n_classes,)
 |      Holds the label for each class.
 |
 |  See Also
 |  --------
 |  OrdinalEncoder : Encode categorical features using an ordinal encoding
 |      scheme.
 |  OneHotEncoder : Encode categorical features as a one-hot numeric array.
 |
 |  Examples
 |  --------
 |  `LabelEncoder` can be used to normalize labels.
 |
 |  >>> from sklearn.preprocessing import LabelEncoder
 |  >>> le = LabelEncoder()
 |  >>> le.fit([1, 2, 2, 6])
 |  LabelEncoder()
 |  >>> le.classes_
 |  array([1, 2, 6])
 |  >>> le.transform([1, 1, 2, 6])
 |  array([0, 0, 1, 2]…)
 |  >>> le.inverse_transform([0, 0, 1, 2])
 |  array([1, 1, 2, 6])
 |
 |  It can also be used to transform non-numerical labels (as long as they are
 |  hashable and comparable) to numerical labels.
 |
 |  >>> le = LabelEncoder()
 |  >>> le.fit(["paris", "paris", "tokyo", "amsterdam"])
 |  LabelEncoder()
 |  >>> list(le.classes_)
 |  [np.str_('amsterdam'), np.str_('paris'), np.str_('tokyo')]
 |  >>> le.transform(["tokyo", "tokyo", "paris"])
 |  array([2, 2, 1]…)
 |  >>> list(le.inverse_transform([2, 2, 1]))
 |  [np.str_('tokyo'), np.str_('tokyo'), np.str_('paris')]
 |
 |  Method resolution order:
 |      LabelEncoder
 |      sklearn.base.TransformerMixin
 |      sklearn.utils._set_output._SetOutputMixin
 |      sklearn.base.BaseEstimator
 |      sklearn.utils._estimator_html_repr._HTMLDocumentationLinkMixin
 |      sklearn.utils._metadata_requests._MetadataRequester
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __sklearn_tags__(self)
 |
 |  fit(self, y)
 |      Fit label encoder.
 |
 |      Parameters
 |      ----------
 |      y : array-like of shape (n_samples,)
 |          Target values.
 |
 |      Returns
 |      -------
 |      self : returns an instance of self.
 |          Fitted label encoder.
 |
 |  fit_transform(self, y)
 |      Fit label encoder and return encoded labels.
 |
 |      Parameters
 |      ----------
 |      y : array-like of shape (n_samples,)
 |          Target values.
 |
 |      Returns
 |      -------
 |      y : array-like of shape (n_samples,)
 |          Encoded labels.
 |
 |  inverse_transform(self, y)
 |      Transform labels back to original encoding.
 |
 |      Parameters
 |      ----------
 |      y : array-like of shape (n_samples,)
 |          Target values.
 |
 |      Returns
 |      -------
 |      y : ndarray of shape (n_samples,)
```

```
 |          Original encoding.
 |
 |  transform(self, y)
 |      Transform labels to normalized encoding.
 |
 |      Parameters
 |      ----------
 |      y : array-like of shape (n_samples,)
 |          Target values.
 |
 |      Returns
 |      -------
 |      y : array-like of shape (n_samples,)
 |          Labels as normalized encodings.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from sklearn.utils._set_output._SetOutputMixin:
 |
 |  set_output(self, *, transform=None)
 |      Set output container.
 |
 |      See :ref:`sphx_glr_auto_examples_miscellaneous_plot_set_output.py`
 |      for an example on how to use the API.
 |
 |      Parameters
 |      ----------
 |      transform : {"default", "pandas", "polars"}, default=None
 |          Configure output of `transform` and `fit_transform`.
 |
 |          - `"default"`: Default output format of a transformer
 |          - `"pandas"`: DataFrame output
 |          - `"polars"`: Polars output
 |          - `None`: Transform configuration is unchanged
 |
 |          .. versionadded:: 1.4
 |              `"polars"` option was added.
 |
 |      Returns
 |      -------
 |      self : estimator instance
 |          Estimator instance.
 |
 |  ----------------------------------------------------------------------
 |  Class methods inherited from sklearn.utils._set_output._SetOutputMixin:
 |
 |  __init_subclass__(auto_wrap_output_keys=('transform',), **kwargs)
 |      Set the ``set_{method}_request`` methods.
 |
 |      This uses PEP-487 [1]_ to set the ``set_{method}_request`` methods. It
 |      looks for the information available in the set default values which are
 |      set using ``__metadata_request__*`` class attributes, or inferred
 |      from method signatures.
 |
 |      The ``__metadata_request__*`` class attributes are used when a method
 |      does not explicitly accept a metadata through its arguments or if the
 |      developer would like to specify a request value for those metadata
 |      which are different from the default ``None``.
 |
 |      References
 |      ----------
 |      .. [1] https://www.python.org/dev/peps/pep-0487
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from sklearn.utils._set_output._SetOutputMixin:
 |
 |  __dict__
 |      dictionary for instance variables
 |
 |  __weakref__
 |      list of weak references to the object
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from sklearn.base.BaseEstimator:
 |
 |  __getstate__(self)
 |      Helper for pickle.
 |
 |  __repr__(self, N_CHAR_MAX=700)
 |      Return repr(self).
 |
 |  __setstate__(self, state)
 |
 |  __sklearn_clone__(self)
 |
 |  get_params(self, deep=True)
 |      Get parameters for this estimator.
 |
 |      Parameters
 |      ----------
 |      deep : bool, default=True
 |          If True, will return the parameters for this estimator and
 |          contained subobjects that are estimators.
 |
 |      Returns
 |      -------
 |      params : dict
 |          Parameter names mapped to their values.
 |
 |  set_params(self, **params)
 |      Set the parameters of this estimator.
 |
 |      The method works on simple estimators as well as on nested objects
 |      (such as :class:`~sklearn.pipeline.Pipeline`). The latter have
 |      parameters of the form ``<component>__<parameter>`` so that it's
 |      possible to update each component of a nested object.
 |
 |      Parameters
 |      ----------
 |      **params : dict
 |          Estimator parameters.
 |
 |      Returns
 |      -------
 |      self : estimator instance
 |          Estimator instance.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from sklearn.utils._metadata_requests._MetadataRequester:
 |
 |  get_metadata_routing(self)
 |      Get metadata routing of this object.
 |
 |      Please check :ref:`User Guide <metadata_routing>` on how the routing
 |      mechanism works.
 |
 |      Returns
 |      -------
 |      routing : MetadataRequest
 |          A :class:`~sklearn.utils.metadata_routing.MetadataRequest`
encapsulating
 |          routing information.
```

[ ]:
```
# Heatmap

# Plot correlations between variables
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```



```
[ ]: ####
     correlation_matrix
     ####
```

```
[ ]:                      product_id  product_name  category  discounted_price  \
     product_id             1.00e+00          0.08 -1.26e-02              0.21
     product_name           8.41e-02          1.00 -1.04e-01              0.09
     category              -1.26e-02         -0.10  1.00e+00              0.12
     discounted_price       2.06e-01          0.09  1.19e-01              1.00
     actual_price           2.47e-01          0.08  1.22e-01              0.96
     discount_percentage    2.90e-01          0.10 -3.14e-01             -0.24
     rating                -1.49e-01         -0.04 -1.09e-01              0.12
     rating_count          -1.76e-01          0.09 -9.84e-02             -0.03
     about_product          4.14e-02          0.16 -3.88e-02              0.05
     user_id                6.57e-02         -0.02  1.27e-02              0.04
     user_name              1.61e-02          0.02  3.78e-02             -0.06
     review_id             -2.43e-02          0.01  1.40e-02             -0.05
     review_title           7.65e-03         -0.06  4.71e-03             -0.02
```

```
review_content        -4.63e-02          -0.02 -1.21e-02              -0.02
img_link              -7.88e-02          -0.06  3.89e-02               0.02
product_link           9.62e-02           0.82 -6.77e-02               0.11

                    actual_price  discount_percentage    rating  \
product_id              2.47e-01             2.90e-01 -1.49e-01
product_name            7.86e-02             1.02e-01 -3.56e-02
category                1.22e-01            -3.14e-01 -1.09e-01
discounted_price        9.62e-01            -2.42e-01  1.20e-01
actual_price            1.00e+00            -1.18e-01  1.22e-01
discount_percentage    -1.18e-01             1.00e+00 -1.55e-01
rating                  1.22e-01            -1.55e-01  1.00e+00
rating_count           -3.60e-02             1.11e-02  1.02e-01
about_product           4.85e-02             6.08e-02 -3.61e-02
user_id                 4.15e-02             8.29e-03 -1.45e-02
user_name              -4.96e-02             6.46e-02 -1.27e-02
review_id              -4.56e-02             2.24e-02 -1.34e-01
review_title            4.52e-03             7.60e-02 -7.57e-02
review_content         -1.39e-02             5.99e-03 -4.46e-02
img_link                2.01e-02            -1.17e-02  3.99e-03
product_link            9.93e-02             7.45e-02 -4.41e-02

                    rating_count  about_product   user_id  user_name  \
product_id             -1.76e-01       4.14e-02  6.57e-02   1.61e-02
product_name            9.24e-02       1.58e-01 -2.41e-02   2.46e-02
category               -9.84e-02      -3.88e-02  1.27e-02   3.78e-02
discounted_price       -2.71e-02       5.26e-02  4.17e-02  -6.31e-02
actual_price           -3.60e-02       4.85e-02  4.15e-02  -4.96e-02
discount_percentage     1.11e-02       6.08e-02  8.29e-03   6.46e-02
rating                  1.02e-01      -3.61e-02 -1.45e-02  -1.27e-02
rating_count            1.00e+00      -3.08e-02 -9.45e-02   5.92e-02
about_product          -3.08e-02       1.00e+00 -1.95e-03  -4.55e-03
user_id                -9.45e-02      -1.95e-03  1.00e+00  -5.14e-02
user_name               5.92e-02      -4.55e-03 -5.14e-02   1.00e+00
review_id              -4.31e-02       1.97e-02  1.69e-03  -9.06e-03
review_title           -8.42e-02      -1.58e-02  3.90e-02  -3.89e-02
review_content         -2.53e-02      -2.46e-02  3.13e-02   2.43e-03
img_link               -1.01e-01       3.09e-02 -1.41e-02  -3.88e-02
product_link            3.67e-03       1.47e-01 -8.66e-03   2.70e-02

                    review_id  review_title  review_content  img_link  \
product_id          -2.43e-02      7.65e-03       -4.63e-02 -7.88e-02
product_name         1.35e-02     -6.06e-02       -1.85e-02 -6.09e-02
category             1.40e-02      4.71e-03       -1.21e-02  3.89e-02
discounted_price    -4.98e-02     -2.10e-02       -1.59e-02  1.89e-02
actual_price        -4.56e-02      4.52e-03       -1.39e-02  2.01e-02
discount_percentage  2.24e-02      7.60e-02        5.99e-03 -1.17e-02
```

```
rating              -1.34e-01    -7.57e-02       -4.46e-02  3.99e-03
rating_count        -4.31e-02    -8.42e-02       -2.53e-02 -1.01e-01
about_product        1.97e-02    -1.58e-02       -2.46e-02  3.09e-02
user_id              1.69e-03     3.90e-02        3.13e-02 -1.41e-02
user_name           -9.06e-03    -3.89e-02        2.43e-03 -3.88e-02
review_id            1.00e+00    -3.46e-02        1.67e-02  5.42e-03
review_title        -3.46e-02     1.00e+00        2.01e-01 -1.21e-02
review_content       1.67e-02     2.01e-01        1.00e+00 -2.03e-02
img_link             5.42e-03    -1.21e-02       -2.03e-02  1.00e+00
product_link         1.44e-02    -6.08e-03       -3.86e-02 -4.39e-02

                     product_link
product_id               9.62e-02
product_name             8.24e-01
category                -6.77e-02
discounted_price         1.10e-01
actual_price             9.93e-02
discount_percentage      7.45e-02
rating                  -4.41e-02
rating_count             3.67e-03
about_product            1.47e-01
user_id                 -8.66e-03
user_name                2.70e-02
review_id                1.44e-02
review_title            -6.08e-03
review_content          -3.86e-02
img_link                -4.39e-02
product_link             1.00e+00
```

[ ]:
```
#### Data Wrangling of Electoral Data
# https://www.kaggle.com/code/zusmani/election-data-wrangling
```

[ ]:
```
import numpy as np # linear algebra library
import pandas as pd # data preprocessing, CSV files etc

!pip install fuzzywuzzy
import fuzzywuzzy
from fuzzywuzzy import process
import chardet

from subprocess import check_output
```

```
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)
Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
```

```
/usr/local/lib/python3.12/dist-packages/fuzzywuzzy/fuzz.py:11: UserWarning:
Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove
this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning')
```

[ ]:
```
import kagglehub
```

[ ]:
```
path = kagglehub.dataset_download("zusmani/predict-pakistan-elections-2018")
print(path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/zusmani/predict-pakistan-
elections-2018?dataset_version_number=15…

100%|      | 46.5M/46.5M [00:00<00:00, 64.7MB/s]

Extracting files…


/root/.cache/kagglehub/datasets/zusmani/predict-pakistan-
elections-2018/versions/15
```

[ ]:
```
import os.path
```

[ ]:
```
NA2 = pd.read_csv(os.path.join(path, 'National Assembly 2002 - Updated.csv'),␣
 ↪encoding = "ISO-8859-1")
NA8 = pd.read_csv(os.path.join(path, 'National Assembly 2008.csv'), encoding =␣
 ↪"ISO-8859-1")
NA13 = pd.read_csv(os.path.join(path, "National Assembly 2013.csv"), encoding =␣
 ↪"ISO-8859-1")

print("Data Dimensions are: ", NA2.shape)
print("Data Dimensions are: ", NA8.shape)
print("Data Dimensions are: ", NA13.shape)
```

```
Data Dimensions are:  (1793, 11)
Data Dimensions are:  (2316, 11)
Data Dimensions are:  (4510, 11)
```

[ ]:
```
print("NA 2002.csv")
NA2.info()
print("\nNA 2008.csv")
NA8.info()
print("\nNA 2013.csv")
NA13.info()
```

```
NA 2002.csv
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1793 entries, 0 to 1792
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   District                 1793 non-null   object
 1   Seat                     1793 non-null   object
 2   Constituency_title       1793 non-null   object
 3   Candidate_Name           1793 non-null   object
 4   Party                    1793 non-null   object
 5   Votes                    1793 non-null   int64
 6   Total_Valid_Votes        1793 non-null   int64
 7   Total_Rejected_Votes     1793 non-null   int64
 8   Total_Votes              1793 non-null   int64
 9   Total_Registered_Voters  1793 non-null   int64
 10  Turnout                  1793 non-null   float64
dtypes: float64(1), int64(5), object(5)
memory usage: 154.2+ KB

NA 2008.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2316 entries, 0 to 2315
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             2316 non-null   int64
 1   Seat                   2316 non-null   object
 2   ConstituencyTitle      2316 non-null   object
 3   CandidateName          2316 non-null   object
 4   Party                  2316 non-null   object
 5   Votes                  2316 non-null   int64
 6   TotalValidVotes        2316 non-null   int64
 7   TotalRejectedVotes     2316 non-null   int64
 8   TotalVotes             2316 non-null   int64
 9   TotalRegisteredVoters  2316 non-null   int64
 10  Turnout                2316 non-null   object
dtypes: int64(6), object(5)
memory usage: 199.2+ KB

NA 2013.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4510 entries, 0 to 4509
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             4510 non-null   int64
 1   Seat                   4510 non-null   object
 2   ConstituencyTitle      4510 non-null   object
 3   CandidateName          4510 non-null   object
 4   Party                 4510 non-null   object
 5   Votes                 4510 non-null   int64
 6   TotalValidVotes       4510 non-null   int64
 7   TotalRejectedVotes    4510 non-null   int64
 8   TotalVotes            4510 non-null   int64
 9   TotalRegisteredVoters 4510 non-null   int64
 10  Turnout               4510 non-null   object
dtypes: int64(6), object(5)
memory usage: 387.7+ KB
```

[ ]:
```
print(NA2.head())
print(NA8.head())
print(NA13.head())
print(NA8.columns, "\n>>\n", NA13.columns)
```

```
   District        Seat Constituency_title  \
0  PESHAWAR  PESHAWAR-I               NA-1
1  PESHAWAR  PESHAWAR-I               NA-1
2  PESHAWAR  PESHAWAR-I               NA-1
3  PESHAWAR  PESHAWAR-I               NA-1
4  PESHAWAR  PESHAWAR-I               NA-1

                        Candidate_Name                               Party  \
0                    Mr Sajid Abdullah           Pakistan Tehreek-e-Insaf
1                     Mr Shabir Ahmad  Muttahidda Majlis-e-Amal Pakistan
2               Mr Usman Bashir Bilour               Awami National Party
3  Mr Muhammad Khurshid Khan Advocate                        Independent
4             Mr Muhammad Muazzam Butt         Pakistan Muslim League(QA)

   Votes  Total_Valid_Votes  Total_Rejected_Votes  Total_Votes  \
0   2029              65642                  1552        67194
1  37179              65642                  1552        67194
2  23002              65642                  1552        67194
3   1537              65642                  1552        67194
4   1417              65642                  1552        67194

   Total_Registered_Voters  Turnout
0                   233907    28.73
1                   233907    28.73
2                   233907    28.73
3                   233907    28.73
4                   233907    28.73
   Unnamed: 0        Seat ConstituencyTitle             CandidateName  \
0           0  Peshawar-1              NA-1              Abdullah Jan
1           1  Peshawar-1              NA-1              Ashoni Kumar
2           2  Peshawar-1              NA-1                 Aurangzeb
3           3  Peshawar-1              NA-1                 Ayub Shah
4           4  Peshawar-1              NA-1  Fakhri Alam Khan  Paracha
```

```
                             Party  Votes  TotalValidVotes  \
0                      Independent    313            88325
1                      Independent    156            88325
2                      Independent    261            88325
3  Pakistan Peoples Party Parliamentarians  37682            88325
4                      Independent    184            88325

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout
0                 629       88913                 387083  22.97 %
1                 629       88913                 387083  22.97 %
2                 629       88913                 387083  22.97 %
3                 629       88913                 387083  22.97 %
4                 629       88913                 387083  22.97 %
   Unnamed: 0        Seat ConstituencyTitle             CandidateName  \
0           0  PESHAWAR-I              NA-1        Aamir Shehzad Hashmi
1           1  PESHAWAR-I              NA-1                  Akram Khan
2           2  PESHAWAR-I              NA-1  Alhaaj Ghulam Ahmad Bilour
3           3  PESHAWAR-I              NA-1                   Amir Syed
4           4  PESHAWAR-I              NA-1         Bashir Ahmad Afridi

                             Party  Votes  TotalValidVotes  \
0               Mustaqbil Pakistan     77           145924
1                      Independent    182           145924
2              Awami National Party  24468           145924
3  Pakistan Peoples Party (Shaheed Bhutto)    454           145924
4        Muttahida Qaumi Movement Pakistan    117           145924

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout
0                2103      146044                 320578   46.18%
1                2103      146044                 320578  46.18 %
2                2103      146044                 320578  46.18 %
3                2103      146044                 320578  46.18 %
4                2103      146044                 320578  46.18 %
Index(['Unnamed: 0', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout'],
      dtype='object')
>>
 Index(['Unnamed: 0', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout'],
      dtype='object')
```

```
[ ]: NA8.rename(columns={'Unnamed: 0': 'District'}, inplace=True)
     NA13.rename(columns={'Unnamed: 0': 'District'}, inplace=True)
     print("NA 8: ", NA8.columns, "\nNA 13: ", NA13.columns)
```

```
NA 8:  Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout'],
      dtype='object')
NA 13:  Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName',
'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout'],
      dtype='object')
```

```
[ ]: ####
     NA8.District
     ####
```

```
[ ]: 0          0
     1          1
     2          2
     3          3
     4          4
             …
     2311    2311
     2312    2312
     2313    2313
     2314    2314
     2315    2315
     Name: District, Length: 2316, dtype: int64
```

```
[ ]: ####
     NA8.Seat
     ####
```

```
[ ]: 0              Peshawar-1
     1              Peshawar-1
     2              Peshawar-1
     3              Peshawar-1
     4              Peshawar-1
                    …
     2311    Kech-cum-Gwadar.
     2312    Kech-cum-Gwadar.
     2313    Kech-cum-Gwadar.
     2314    Kech-cum-Gwadar.
     2315    Kech-cum-Gwadar.
     Name: Seat, Length: 2316, dtype: object
```

```
[ ]: ####
     NA8.District.unique()
     ####
```

```
[ ]: array([   0,    1,    2, …, 2313, 2314, 2315])
```

```
[ ]: ####
     NA8['District'].unique()
     ####
```

```
[ ]: array([   0,    1,    2, …, 2313, 2314, 2315])
```

```
[ ]: ####
     NA8['District'].unique()
     ####
```

```
[ ]: array([   0,    1,    2, …, 2313, 2314, 2315])
```

```
[ ]: NA8.District = NA8.Seat
     NA8.District.head()
```

```
[ ]: 0    Peshawar-1
     1    Peshawar-1
     2    Peshawar-1
     3    Peshawar-1
     4    Peshawar-1
     Name: District, dtype: object
```

```
[ ]: ####
     NA8.District.unique()
     ####
```

```
[ ]: array(['Peshawar-1', 'Peshawar-II', 'Peshawar-III', 'Peshawar-IV',
            'Nowshera-I', 'Nowshera-II', 'Charsadda-I', 'Mardan-I',
            'Mardan-II', 'Mardan-III', 'Swabi-I', 'Swabi-II', 'Kohat', 'Karak',
            'Hangu', 'Abbottabad-I', 'Abbottabad-II', 'Haripur', 'Mansehra-I',
            'Mansehra-II', 'Battagram', 'Kohistan', 'D.I.Khan',
            'D.I.Khan-com-Tank', 'Bannu', 'Lakki Marwat', 'Buner', 'Swat-I',
            'Swat-II', 'Shangla', 'Chitral',
            'Upper Dir-cum-Lower Dir (Old Upper Dir', 'Lower Dir',
            'Malakand P.A', 'Tribal Area-I', 'Tribal Area-II',
            'Tribal Area-III', 'Tribal Area-IV', 'Tribal Area-V',
            'Tribal Area-VI', 'Tribal Area-VII', 'Tribal Area-VIII',
            'Tribal Area-IX', 'Tribal Area-X', 'Tribal Area-XI',
            'Tribal Area-XII', 'Islamabad-I', 'Islamabad-II', 'Rawalpindi-I',
            'Rawalpindi -II', 'Rawalpindi-III', 'Rawalpindi-IV',
            'Rawalpindi \x96V', 'Rawalpindi-VI', 'Rawalpindi-VII', 'Attock-I',
            'Attock-II', 'Attock-III', 'Chakwal-I', 'Chakwal-II', 'Jhelum-I',
            'Jhelum-II', 'Sargodha-I', 'Sargodha-II', 'Sargodha-III',
            'Sargodha-IV', 'Sargodha-V', 'Khushab-I', 'Khushab-II',
            'Mianwali-I', 'Mianwali-II', 'Bhakkar-I', 'Bhakkar-II',
            'Faisalabad-I', 'Faisalabad-II', 'Faisalabad-III', 'Faisalabad-IV',
            'Faisalabad-V', 'Faisalabad-VI', 'Faisalabad-VII',
            'Faisalabad-VIII', 'Faisalabad-IX', 'Faisalabad-X',
            'Faisalabad-XI', 'Jhang-I', 'Jhang-II', 'Jhang-III', 'Jhang-IV',
            'Jhang-V', 'Jhang-VI', 'Toba Tek Singh-I', 'Toba Tek Singh-II',
            'Toba Tek Singh-III', 'Gujranwala-I', 'Gujranwala-II',
            'Gujranwala-III', 'Gujranwala-IV', 'Gujranwala-V', 'Gujranwala-VI',
            'Gujranwala-VII', 'Hafizabad-I', 'Hafizabad-II', 'Gujrat-I',
            'Gujrat-II', 'Gujrat-III', 'Gujrat-IV', 'M.B.Din-I', 'M.B.Din-II',
            'Sialkot-I', 'Sialkot-II', 'Sialkot-III', 'Sialkot-IV',
            'Sialkot-V', 'Narowal-I', 'Narowal-II', 'Narowal-III', 'Lahore-I',
            'Lahore-III', 'Lahore-IV', 'Lahore-V', 'Lahore-VI', 'Lahore-VII',
            'Lahore-VIII', 'Lahore-IX', 'Lahore-X', 'Lahore-XI', 'Lahore-XII',
            'Lahore-XIII', 'Sheikhupura-I',
            'Sheikhupura-Cum-Nankana Sahib-I (Old Sheikhupura-I',
            'Sheikhupura-II (Old Sheikhupura-III',
            'Sheikhupura-Cum-Nankana Sahib-II (Old Sheikhupura-',
            'Nanka Sahib-I (Old Sheikhupura-V',
            'Nankana Sahib-cum- Sheikhupura (Old Sheikhupura-VI',
            'Nankana Sahib-II (Old Sheikhupura-VII', 'Kasur-I', 'Kasur-II',
            'Kasur-III', 'Kasur-IV', 'Kasur-V', 'Okara-I', 'Okara-II',
            'Okara-III', 'Okara-IV', 'Okara-V', 'Multan-I', 'Multan-II',
            'Multan-III', 'Multan-IV', 'Multan-V', 'Multan-VI', 'Lodhran-I',
            'Lodhran-II', 'Khanewal-I', 'Khanewal-II', 'Khanewal-III',
            'Khanewal-IV', 'Sahiwal-I', 'Sahiwal-II', 'Sahiwal-III',
            'Sahiwal-IV', 'Pakpattan-I', 'Pakpattan-II', 'Pakpattan-III',
            'Vehari-I', 'Vehari-II', 'Vehari-III', 'Vehari-IV',
            'Dera Ghazi Khan-I', 'Dera Ghazi Khan-II', 'Dera Ghazi Khan-III',
            'Rajanpur-I', 'Rajanpur-II', 'Muzaffargarh-I', 'Muzaffargarh-II',
            'Muzaffargarh-III', 'Muzaffargarh-IV', 'Muzaffargarh-V',
            'Layyah-I', 'Layyah-II', 'Bahawalpur-I', 'Bahawalpur-II',
            'Bahawalpur-III', 'Bahawalpur-IV', 'Bahawalpur-V',
            'Bahawalnagar-I', 'Bahawalnagar-II', 'Bahawalnagar-III',
            'Bahawalnagar-IV', 'Rahimyar Khan-I', 'Rahimyar Khan-II',
            'Rahimyar Khan-III', 'Rahimyar Khan-IV', 'Rahimyar Khan-V',
            'Rahimyar Khan-VI', 'Sukkur cum Shikarpur(I',
            'Sukkur cum Shikarpur(II', 'Ghotki-I', 'Ghotki-II',
            'Shikarpur (Old Shikarpur-I',
            'Shikarpur-cum-Sukkur-cum-Larkana (Old Shikarpur-II',
            'Larkana (Old Larkana I',
            'Larka cum-Kamber Shahdadkot (Old Larkana II',
            'Kamber Shahdadkot (Old Larkana III',
            'Jacobabad  (Old Jacobabad-I',
            'Jacobabad-cum-Kashmore (Old Jacobabad-II',
            'Kashmore (Old Jacobabad-III', 'Naushero Feroze-I',
            'Naushero Feroze-II', 'Nawabshah-I', 'Nawabshah-II', 'Khairpur-I',
            'Khairpur-II', 'Khairpur-III',
            'Matiari-cum-Hyderabad (Old Hyderabad-I',
            'Hyderabad-I (Old Hyderabad-II', 'Hyderabad-II (Old Hyderabad-III',
            'Hyderabad-cum-Matiari (Old Hyderabad-IV',
            'Tando Muhammad Khan-cum-Hyderabad-cum-Badin (Old H',
            'Tando Allahyar-cum-Matiari (Old Hyderabad-VI',
            'Badin-cum-Tando Muhammad Khan-I (Old Badin-I',
            'Badin-cum-Tando Muhammad Khan-II', 'Mirpurkhas-cum-Umerkot(I',
            'Mirpurkhas-cum-Umerkot(II', 'Umerkot (Old Mirpurkhas-III',
            'Tharparkar-I', 'Tharparkar-II.', 'Jamshoro (Old Dadu-I',
            'Dadu-I (Old Dadu-II', 'Dadu-II (Old Dadu-III', 'Sanghar-I',
            'Sanghar-II (Old Sanghar-III', 'Thatta-I', 'Thatta-II',
            'Karachi-I', 'Karachi-II', 'Karachi-III', 'Karachi-IV',
            'Karachi-V', 'Karachi-VI', 'Karachi-VII', 'Karachi-VIII',
            'Karachi-IX', 'Karachi-X', 'Karachi-XI', 'Karachi-XII',
            'Karachi-XIII', 'Karachi-XIV', 'Karachi-XV', 'Karachi-XVI',
            'Karachi-XVII', 'Karachi-XVIII', 'Karachi-XIX', 'Karachi-XX',
            'Quetta', 'Quetta-cum-Chagai-cum-Nushki (Old Quetta-cum-Chaga',
            'Pishin-cum-Ziarat', 'Killa Abdullah',
            'Loralai-cum-Musakhel-cum-Barkhan (Old Loralai',
            'Zhob-cum-Sherani-cum-Killa Saifullah(Old Zhob-cum-',
            'Sibi-cum-Kohlu-cum-Dera Bugti-cum-Hernai(Old Sibi-',
            'Bolan-cum-Jhal Magsi (Old Kachhi', 'Kalat-cum-Mastung.',
            'Khuzdar.', 'Awaran-cum-Lasbela.',
            'Kharan-cum-Washuk-cum-Panjgur(Old Kharan-cum-Panjg',
            'Kech-cum-Gwadar.'], dtype=object)
```

```
[ ]: ####
     help(str.split)
     ####
```

```
Help on method_descriptor:

split(self, /, sep=None, maxsplit=-1) unbound builtins.str method
    Return a list of the substrings in the string, using sep as the separator
string.

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \n \r \t \f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits.
        -1 (the default value) means no limit.

    Splitting starts at the front of the string and works to the end.
```

```
Note, str.split() is mainly useful for data that has been intentionally
delimited.  With natural text that includes punctuation, consider using
the regular expression module.
```

```
[ ]: NA8.District.str.split("-", expand=True)[0]
```

```
[ ]: 0
     1
     2
     3
     4
                    …
     2311
     2312
     2313
     2314
     2315
     Name: 0, Length: 2316, dtype: object
```

```
[ ]: ####
     #NA8['District'] = NA8['District'].str.replace("."," ")
     tim = "D.I. Khan"
     tim.replace(".", " ")
     ####
```

```
[ ]: 'D I  Khan'
```

```
[ ]: ####
     tim.replace(r"\(.*\)","")
     ####
```

```
[ ]: 'D.I. Khan'
```

```
[ ]: ####
     NA8.District.unique()
     ####
```

```
[ ]: array(['Peshawar-1', 'Peshawar-II', 'Peshawar-III', 'Peshawar-IV',
            'Nowshera-I', 'Nowshera-II', 'Charsadda-I', 'Mardan-I',
            'Mardan-II', 'Mardan-III', 'Swabi-I', 'Swabi-II', 'Kohat', 'Karak',
            'Hangu', 'Abbottabad-I', 'Abbottabad-II', 'Haripur', 'Mansehra-I',
            'Mansehra-II', 'Battagram', 'Kohistan', 'D.I.Khan',
            'D.I.Khan-com-Tank', 'Bannu', 'Lakki Marwat', 'Buner', 'Swat-I',
            'Swat-II', 'Shangla', 'Chitral',
            'Upper Dir-cum-Lower Dir (Old Upper Dir', 'Lower Dir',
            'Malakand P.A', 'Tribal Area-I', 'Tribal Area-II',
            'Tribal Area-III', 'Tribal Area-IV', 'Tribal Area-V',
            'Tribal Area-VI', 'Tribal Area-VII', 'Tribal Area-VIII',
            'Tribal Area-IX', 'Tribal Area-X', 'Tribal Area-XI',
            'Tribal Area-XII', 'Islamabad-I', 'Islamabad-II', 'Rawalpindi-I',
            'Rawalpindi -II', 'Rawalpindi-III', 'Rawalpindi-IV',
            'Rawalpindi \x96V', 'Rawalpindi-VI', 'Rawalpindi-VII', 'Attock-I',
            'Attock-II', 'Attock-III', 'Chakwal-I', 'Chakwal-II', 'Jhelum-I',
            'Jhelum-II', 'Sargodha-I', 'Sargodha-II', 'Sargodha-III',
            'Sargodha-IV', 'Sargodha-V', 'Khushab-I', 'Khushab-II',
            'Mianwali-I', 'Mianwali-II', 'Bhakkar-I', 'Bhakkar-II',
            'Faisalabad-I', 'Faisalabad-II', 'Faisalabad-III', 'Faisalabad-IV',
            'Faisalabad-V', 'Faisalabad-VI', 'Faisalabad-VII',
            'Faisalabad-VIII', 'Faisalabad-IX', 'Faisalabad-X',
            'Faisalabad-XI', 'Jhang-I', 'Jhang-II', 'Jhang-III', 'Jhang-IV',
            'Jhang-V', 'Jhang-VI', 'Toba Tek Singh-I', 'Toba Tek Singh-II',
            'Toba Tek Singh-III', 'Gujranwala-I', 'Gujranwala-II',
            'Gujranwala-III', 'Gujranwala-IV', 'Gujranwala-V', 'Gujranwala-VI',
            'Gujranwala-VII', 'Hafizabad-I', 'Hafizabad-II', 'Gujrat-I',
            'Gujrat-II', 'Gujrat-III', 'Gujrat-IV', 'M.B.Din-I', 'M.B.Din-II',
            'Sialkot-I', 'Sialkot-II', 'Sialkot-III', 'Sialkot-IV',
            'Sialkot-V', 'Narowal-I', 'Narowal-II', 'Narowal-III', 'Lahore-I',
            'Lahore-III', 'Lahore-IV', 'Lahore-V', 'Lahore-VI', 'Lahore-VII',
            'Lahore-VIII', 'Lahore-IX', 'Lahore-X', 'Lahore-XI', 'Lahore-XII',
            'Lahore-XIII', 'Sheikhupura-I',
            'Sheikhupura-Cum-Nankana Sahib-I (Old Sheikhupura-I',
            'Sheikhupura-II (Old Sheikhupura-III',
            'Sheikhupura-Cum-Nankana Sahib-II (Old Sheikhupura-',
            'Nanka Sahib-I (Old Sheikhupura-V',
            'Nankana Sahib-cum- Sheikhupura (Old Sheikhupura-VI',
            'Nankana Sahib-II (Old Sheikhupura-VII', 'Kasur-I', 'Kasur-II',
            'Kasur-III', 'Kasur-IV', 'Kasur-V', 'Okara-I', 'Okara-II',
            'Okara-III', 'Okara-IV', 'Okara-V', 'Multan-I', 'Multan-II',
            'Multan-III', 'Multan-IV', 'Multan-V', 'Multan-VI', 'Lodhran-I',
            'Lodhran-II', 'Khanewal-I', 'Khanewal-II', 'Khanewal-III',
            'Khanewal-IV', 'Sahiwal-I', 'Sahiwal-II', 'Sahiwal-III',
            'Sahiwal-IV', 'Pakpattan-I', 'Pakpattan-II', 'Pakpattan-III',
            'Vehari-I', 'Vehari-II', 'Vehari-III', 'Vehari-IV',
            'Dera Ghazi Khan-I', 'Dera Ghazi Khan-II', 'Dera Ghazi Khan-III',
            'Rajanpur-I', 'Rajanpur-II', 'Muzaffargarh-I', 'Muzaffargarh-II',
            'Muzaffargarh-III', 'Muzaffargarh-IV', 'Muzaffargarh-V',
            'Layyah-I', 'Layyah-II', 'Bahawalpur-I', 'Bahawalpur-II',
            'Bahawalpur-III', 'Bahawalpur-IV', 'Bahawalpur-V',
            'Bahawalnagar-I', 'Bahawalnagar-II', 'Bahawalnagar-III',
            'Bahawalnagar-IV', 'Rahimyar Khan-I', 'Rahimyar Khan-II',
            'Rahimyar Khan-III', 'Rahimyar Khan-IV', 'Rahimyar Khan-V',
            'Rahimyar Khan-VI', 'Sukkur cum Shikarpur(I',
            'Sukkur cum Shikarpur(II', 'Ghotki-I', 'Ghotki-II',
            'Shikarpur (Old Shikarpur-I',
            'Shikarpur-cum-Sukkur-cum-Larkana (Old Shikarpur-II',
            'Larkana (Old Larkana I',
            'Larka cum-Kamber Shahdadkot (Old Larkana II',
            'Kamber Shahdadkot (Old Larkana III',
            'Jacobabad  (Old Jacobabad-I',
            'Jacobabad-cum-Kashmore (Old Jacobabad-II',
            'Kashmore (Old Jacobabad-III', 'Naushero Feroze-I',
            'Naushero Feroze-II', 'Nawabshah-I', 'Nawabshah-II', 'Khairpur-I',
            'Khairpur-II', 'Khairpur-III',
            'Matiari-cum-Hyderabad (Old Hyderabad-I',
            'Hyderabad-I (Old Hyderabad-II', 'Hyderabad-II (Old Hyderabad-III',
            'Hyderabad-cum-Matiari (Old Hyderabad-IV',
            'Tando Muhammad Khan-cum-Hyderabad-cum-Badin (Old H',
            'Tando Allahyar-cum-Matiari (Old Hyderabad-VI',
            'Badin-cum-Tando Muhammad Khan-I (Old Badin-I',
            'Badin-cum-Tando Muhammad Khan-II', 'Mirpurkhas-cum-Umerkot(I',
            'Mirpurkhas-cum-Umerkot(II', 'Umerkot (Old Mirpurkhas-III',
            'Tharparkar-I', 'Tharparkar-II.', 'Jamshoro (Old Dadu-I',
            'Dadu-I (Old Dadu-II', 'Dadu-II (Old Dadu-III', 'Sanghar-I',
            'Sanghar-II (Old Sanghar-III', 'Thatta-I', 'Thatta-II',
            'Karachi-I', 'Karachi-II', 'Karachi-III', 'Karachi-IV',
            'Karachi-V', 'Karachi-VI', 'Karachi-VII', 'Karachi-VIII',
            'Karachi-IX', 'Karachi-X', 'Karachi-XI', 'Karachi-XII',
            'Karachi-XIII', 'Karachi-XIV', 'Karachi-XV', 'Karachi-XVI',
            'Karachi-XVII', 'Karachi-XVIII', 'Karachi-XIX', 'Karachi-XX',
            'Quetta', 'Quetta-cum-Chagai-cum-Nushki (Old Quetta-cum-Chaga',
            'Pishin-cum-Ziarat', 'Killa Abdullah',
            'Loralai-cum-Musakhel-cum-Barkhan (Old Loralai',
            'Zhob-cum-Sherani-cum-Killa Saifullah(Old Zhob-cum-',
            'Sibi-cum-Kohlu-cum-Dera Bugti-cum-Hernai(Old Sibi-',
            'Bolan-cum-Jhal Magsi (Old Kachhi', 'Kalat-cum-Mastung.',
            'Khuzdar.', 'Awaran-cum-Lasbela.',
            'Kharan-cum-Washuk-cum-Panjgur(Old Kharan-cum-Panjg',
            'Kech-cum-Gwadar.'], dtype=object)
```

```
[ ]: ####
     tim = "hello (Old Jacobabad-I)"
     tim.replace(r"\(.*\)", "")
     ####
```

```
[ ]: 'hello (Old Jacobabad-I)'
```

```
[ ]: ####
     tim = "M.B.Din-I"
     print(tim, "before first")
```

```
tim = tim.replace(".", "")
print(tim, "before second")
tim = re.sub(r"(XX|IX|X?I{0,3})(IX|IV|V?I{0,3})$", '', tim)
tim
####
```

```
M.B.Din-I before first
MBDin-I before second
```

[ ]: 'MBDin-'

[ ]:
```
####
import regex as re
tim = "Yes you are (You how are you)"
re.sub(r"\(.*\)", "", tim)
####
```

[ ]: 'Yes you are '

[ ]:
```
####
tim = "Yo boy, how You ar.e doing ,!s1 -"
tim = re.sub("[^a-zA-Z -]", "", tim)
tim
####
```

[ ]: 'Yo boy how You are doing s -'

[ ]:
```
####
tim = "timmy yo lets go to London-II and get back to France-IIIGood"
tim = re.sub(r"-.*", "", tim)
tim
####
```

[ ]: 'timmy yo lets go to London'

[ ]:
```
####
help(str.replace)
####
```

```
Help on method_descriptor:

replace(self, old, new, count=-1, /) unbound builtins.str method
    Return a copy with all occurrences of substring old replaced by new.

      count
        Maximum number of occurrences to replace.
        -1 (the default value) means replace all occurrences.
    If the optional argument count is given, only the first count occurrences
are
    replaced.
```

```
[ ]: ####
     NA8.District = NA8.Seat#.str.split("-", expand=True)[0]
     #Add District column
     #NA8['District'] = NA8['Seat']
     NA8['District'] = NA8['District'].str.replace("."," ") # to deal with D.I. Khan
     # remove all those substring with ()
     NA8['District'] = NA8['District'].str.replace(r"\(.*\)","")
     # remove numeric
     NA8['District']  = NA8['District'] .str.replace('[^a-zA-Z -]', '')
     #NA8['District'] = NA8['District'].str.replace(r"Cum.*","")
     #NA8['District'] = NA8['District'].str.replace(r"cum.*","")
     #na18['District'] = na18['District'].str.replace(r"KUM.*","")
     # to convert Tribal Area III - Mohman into Tribal Area III
     NA8['District'] = NA8['District'].str.replace(r"-.*","")
     NA8['District']  = NA8['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA8['District']  = NA8['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA8['District'].unique()
     ####
```

```
[ ]: #####
     help(str.split)
     #####
```

```
Help on method_descriptor:

split(self, /, sep=None, maxsplit=-1) unbound builtins.str method
    Return a list of the substrings in the string, using sep as the separator
string.

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \n \r \t \f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits.
        -1 (the default value) means no limit.

    Splitting starts at the front of the string and works to the end.
```

```
Note, str.split() is mainly useful for data that has been intentionally
delimited.  With natural text that includes punctuation, consider using
the regular expression module.
```

```
[ ]: NA8.District = NA8.Seat#.str.split("-", expand=True)[0]
     #Add District column
     #NA8['District'] = NA8['Seat']
     NA8['District'] = NA8['District'].str.replace("."," ") # to deal with D.I. Khan
     # remove all those substring with ()
     NA8['District'] = NA8['District'].str.replace(r"\(.*\)","")
     # remove numeric
     NA8['District']  = NA8['District'] .str.replace('[^a-zA-Z -]', '')
     #NA8['District'] = NA8['District'].str.replace(r"Cum.*","")
     NA8['District'] = NA8['District'].str.replace(r"cum.*","")
     #na18['District'] = na18['District'].str.replace(r"KUM.*","")
     # to convert Tribal Area III - Mohman into Tribal Area III
     NA8['District'] = NA8['District'].str.replace(r"-.*","")
     NA8['District']  = NA8['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA8['District']  = NA8['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA8['District'].unique()
```

```
[ ]: array(['Peshawar-1', 'Peshawar-II', 'Peshawar-III', 'Peshawar-IV',
            'Nowshera-I', 'Nowshera-II', 'Charsadda-I', 'Mardan-I',
            'Mardan-II', 'Mardan-III', 'Swabi-I', 'Swabi-II', 'Kohat', 'Karak',
            'Hangu', 'Abbottabad-I', 'Abbottabad-II', 'Haripur', 'Mansehra-I',
            'Mansehra-II', 'Battagram', 'Kohistan', 'D I Khan',
            'D I Khan-com-Tank', 'Bannu', 'Lakki Marwat', 'Buner', 'Swat-I',
            'Swat-II', 'Shangla', 'Chitral',
            'Upper Dir-cum-Lower Dir (Old Upper Dir', 'Lower Dir',
            'Malakand P A', 'Tribal Area-I', 'Tribal Area-II',
            'Tribal Area-III', 'Tribal Area-IV', 'Tribal Area-V',
            'Tribal Area-VI', 'Tribal Area-VII', 'Tribal Area-VIII',
            'Tribal Area-IX', 'Tribal Area-X', 'Tribal Area-XI',
            'Tribal Area-XII', 'Islamabad-I', 'Islamabad-II', 'Rawalpindi-I',
            'Rawalpindi -II', 'Rawalpindi-III', 'Rawalpindi-IV',
            'Rawalpindi \x96V', 'Rawalpindi-VI', 'Rawalpindi-VII', 'Attock-I',
            'Attock-II', 'Attock-III', 'Chakwal-I', 'Chakwal-II', 'Jhelum-I',
            'Jhelum-II', 'Sargodha-I', 'Sargodha-II', 'Sargodha-III',
            'Sargodha-IV', 'Sargodha-V', 'Khushab-I', 'Khushab-II',
            'Mianwali-I', 'Mianwali-II', 'Bhakkar-I', 'Bhakkar-II',
            'Faisalabad-I', 'Faisalabad-II', 'Faisalabad-III', 'Faisalabad-IV',
            'Faisalabad-V', 'Faisalabad-VI', 'Faisalabad-VII',
            'Faisalabad-VIII', 'Faisalabad-IX', 'Faisalabad-X',
            'Faisalabad-XI', 'Jhang-I', 'Jhang-II', 'Jhang-III', 'Jhang-IV',
            'Jhang-V', 'Jhang-VI', 'Toba Tek Singh-I', 'Toba Tek Singh-II',
            'Toba Tek Singh-III', 'Gujranwala-I', 'Gujranwala-II',
            'Gujranwala-III', 'Gujranwala-IV', 'Gujranwala-V', 'Gujranwala-VI',
            'Gujranwala-VII', 'Hafizabad-I', 'Hafizabad-II', 'Gujrat-I',
            'Gujrat-II', 'Gujrat-III', 'Gujrat-IV', 'M B Din-I', 'M B Din-II',
            'Sialkot-I', 'Sialkot-II', 'Sialkot-III', 'Sialkot-IV',
            'Sialkot-V', 'Narowal-I', 'Narowal-II', 'Narowal-III', 'Lahore-I',
            'Lahore-III', 'Lahore-IV', 'Lahore-V', 'Lahore-VI', 'Lahore-VII',
            'Lahore-VIII', 'Lahore-IX', 'Lahore-X', 'Lahore-XI', 'Lahore-XII',
            'Lahore-XIII', 'Sheikhupura-I',
            'Sheikhupura-Cum-Nankana Sahib-I (Old Sheikhupura-I',
            'Sheikhupura-II (Old Sheikhupura-III',
            'Sheikhupura-Cum-Nankana Sahib-II (Old Sheikhupura-',
            'Nanka Sahib-I (Old Sheikhupura-V',
            'Nankana Sahib-cum- Sheikhupura (Old Sheikhupura-VI',
            'Nankana Sahib-II (Old Sheikhupura-VII', 'Kasur-I', 'Kasur-II',
            'Kasur-III', 'Kasur-IV', 'Kasur-V', 'Okara-I', 'Okara-II',
            'Okara-III', 'Okara-IV', 'Okara-V', 'Multan-I', 'Multan-II',
            'Multan-III', 'Multan-IV', 'Multan-V', 'Multan-VI', 'Lodhran-I',
            'Lodhran-II', 'Khanewal-I', 'Khanewal-II', 'Khanewal-III',
            'Khanewal-IV', 'Sahiwal-I', 'Sahiwal-II', 'Sahiwal-III',
            'Sahiwal-IV', 'Pakpattan-I', 'Pakpattan-II', 'Pakpattan-III',
            'Vehari-I', 'Vehari-II', 'Vehari-III', 'Vehari-IV',
            'Dera Ghazi Khan-I', 'Dera Ghazi Khan-II', 'Dera Ghazi Khan-III',
            'Rajanpur-I', 'Rajanpur-II', 'Muzaffargarh-I', 'Muzaffargarh-II',
            'Muzaffargarh-III', 'Muzaffargarh-IV', 'Muzaffargarh-V',
            'Layyah-I', 'Layyah-II', 'Bahawalpur-I', 'Bahawalpur-II',
            'Bahawalpur-III', 'Bahawalpur-IV', 'Bahawalpur-V',
            'Bahawalnagar-I', 'Bahawalnagar-II', 'Bahawalnagar-III',
            'Bahawalnagar-IV', 'Rahimyar Khan-I', 'Rahimyar Khan-II',
            'Rahimyar Khan-III', 'Rahimyar Khan-IV', 'Rahimyar Khan-V',
            'Rahimyar Khan-VI', 'Sukkur cum Shikarpur(I',
            'Sukkur cum Shikarpur(II', 'Ghotki-I', 'Ghotki-II',
            'Shikarpur (Old Shikarpur-I',
            'Shikarpur-cum-Sukkur-cum-Larkana (Old Shikarpur-II',
            'Larkana (Old Larkana I',
            'Larka cum-Kamber Shahdadkot (Old Larkana II',
            'Kamber Shahdadkot (Old Larkana III',
            'Jacobabad  (Old Jacobabad-I',
            'Jacobabad-cum-Kashmore (Old Jacobabad-II',
            'Kashmore (Old Jacobabad-III', 'Naushero Feroze-I',
            'Naushero Feroze-II', 'Nawabshah-I', 'Nawabshah-II', 'Khairpur-I',
            'Khairpur-II', 'Khairpur-III',
            'Matiari-cum-Hyderabad (Old Hyderabad-I',
            'Hyderabad-I (Old Hyderabad-II', 'Hyderabad-II (Old Hyderabad-III',
            'Hyderabad-cum-Matiari (Old Hyderabad-IV',
            'Tando Muhammad Khan-cum-Hyderabad-cum-Badin (Old H',
            'Tando Allahyar-cum-Matiari (Old Hyderabad-VI',
            'Badin-cum-Tando Muhammad Khan-I (Old Badin-I',
            'Badin-cum-Tando Muhammad Khan-II', 'Mirpurkhas-cum-Umerkot(I',
            'Mirpurkhas-cum-Umerkot(II', 'Umerkot (Old Mirpurkhas-III',
            'Tharparkar-I', 'Tharparkar-II ', 'Jamshoro (Old Dadu-I',
            'Dadu-I (Old Dadu-II', 'Dadu-II (Old Dadu-III', 'Sanghar-I',
            'Sanghar-II (Old Sanghar-III', 'Thatta-I', 'Thatta-II',
            'Karachi-I', 'Karachi-II', 'Karachi-III', 'Karachi-IV',
            'Karachi-V', 'Karachi-VI', 'Karachi-VII', 'Karachi-VIII',
            'Karachi-IX', 'Karachi-X', 'Karachi-XI', 'Karachi-XII',
            'Karachi-XIII', 'Karachi-XIV', 'Karachi-XV', 'Karachi-XVI',
            'Karachi-XVII', 'Karachi-XVIII', 'Karachi-XIX', 'Karachi-XX',
            'Quetta', 'Quetta-cum-Chagai-cum-Nushki (Old Quetta-cum-Chaga',
            'Pishin-cum-Ziarat', 'Killa Abdullah',
            'Loralai-cum-Musakhel-cum-Barkhan (Old Loralai',
            'Zhob-cum-Sherani-cum-Killa Saifullah(Old Zhob-cum-',
            'Sibi-cum-Kohlu-cum-Dera Bugti-cum-Hernai(Old Sibi-',
            'Bolan-cum-Jhal Magsi (Old Kachhi', 'Kalat-cum-Mastung ',
            'Khuzdar ', 'Awaran-cum-Lasbela ',
            'Kharan-cum-Washuk-cum-Panjgur(Old Kharan-cum-Panjg',
            'Kech-cum-Gwadar '], dtype=object)
```

```
[ ]: ####
     tim = "Peshawar-1"
     tim = (re.sub("."," ", tim))
     tim
     ####
```

```
[ ]: '          '
```

```
[ ]: import regex as re

     NA8.District = NA8.Seat#.str.split("-", expand=True)[0]
     #Add District column

     #NA8['District'] = NA8['District'].map(lambda x: (re.sub("."," ", x))) # to␣
      ↪deal with D.I. Khan
     # remove all those substring with ()
     NA8['District'] = NA8['District'].map(lambda x: (re.sub(r"\(.*\)","", x)))
     # remove numeric
     NA8['District']  = NA8['District'] .map(lambda x: (re.sub('[^a-zA-Z -]', '',␣
      ↪x)))
     # to convert Tribal Area III - Mohman into Tribal Area III
     #NA8['District'] = NA8['District'].str.replace(r"Cum.*","")
     #NA8['District'] = NA8['District'].str.replace(r"cum.*","")
     #na18['District'] = na18['District'].str.replace(r"KUM.*","")
     NA8['District'] = NA8['District'].map(lambda x: (re.sub(r"-.*","", x)))
```

```
NA8['District']  = NA8['District'].map(lambda x: (re.sub(r" (XX|IX|X?
↪I{0,3})(IX|IV|V?I{0,3})$", '', x)))
#NA8['District']  = NA8['District'].map(lambda x: (re.sub(r" (XX|IX|X?
↪I{0,3})(IX|IV|V?I{0,3})$", '', x)))
NA8['District'].unique()
```

```
[ ]: array(['Peshawar', 'Nowshera', 'Charsadda', 'Mardan', 'Swabi', 'Kohat',
            'Karak', 'Hangu', 'Abbottabad', 'Haripur', 'Mansehra', 'Battagram',
            'Kohistan', 'DIKhan', 'Bannu', 'Lakki Marwat', 'Buner', 'Swat',
            'Shangla', 'Chitral', 'Upper Dir', 'Lower Dir', 'Malakand PA',
            'Tribal Area', 'Islamabad', 'Rawalpindi', 'Attock', 'Chakwal',
            'Jhelum', 'Sargodha', 'Khushab', 'Mianwali', 'Bhakkar',
            'Faisalabad', 'Jhang', 'Toba Tek Singh', 'Gujranwala', 'Hafizabad',
            'Gujrat', 'MBDin', 'Sialkot', 'Narowal', 'Lahore', 'Sheikhupura',
            'Nanka Sahib', 'Nankana Sahib', 'Kasur', 'Okara', 'Multan',
            'Lodhran', 'Khanewal', 'Sahiwal', 'Pakpattan', 'Vehari',
            'Dera Ghazi Khan', 'Rajanpur', 'Muzaffargarh', 'Layyah',
            'Bahawalpur', 'Bahawalnagar', 'Rahimyar Khan',
            'Sukkur cum ShikarpurI', 'Sukkur cum ShikarpurII', 'Ghotki',
            'Shikarpur Old Shikarpur', 'Shikarpur', 'Larkana Old Larkana',
            'Larka cum', 'Kamber Shahdadkot Old Larkana',
            'Jacobabad  Old Jacobabad', 'Jacobabad', 'Kashmore Old Jacobabad',
            'Naushero Feroze', 'Nawabshah', 'Khairpur', 'Matiari', 'Hyderabad',
            'Tando Muhammad Khan', 'Tando Allahyar', 'Badin', 'Mirpurkhas',
            'Umerkot Old Mirpurkhas', 'Tharparkar', 'Jamshoro Old Dadu',
            'Dadu', 'Sanghar', 'Thatta', 'Karachi', 'Quetta', 'Pishin',
            'Killa Abdullah', 'Loralai', 'Zhob', 'Sibi', 'Bolan', 'Kalat',
            'Khuzdar', 'Awaran', 'Kharan', 'Kech'], dtype=object)
```

```
[ ]: import regex as re

NA13.District = NA13.Seat#.str.split("-", expand=True)[0]
#Add District column

#NA8['District'] = NA8['District'].map(lambda x: (re.sub(".","  ", x))) # to␣
↪deal with D.I. Khan
# remove all those substring with ()
NA13['District'] = NA13['District'].map(lambda x: (re.sub(r"\(.*\)","", x)))
# remove numeric
NA13['District']  = NA13['District'] .map(lambda x: (re.sub('[^a-zA-Z -]', '',␣
↪x)))
# to convert Tribal Area III - Mohman into Tribal Area III
NA8['District'] = NA8['District'].map(lambda x: (re.sub(r"Cum.*","", x)))
#NA8['District'] = NA8['District'].str.replace(r"cum.*","")
#na18['District'] = na18['District'].str.replace(r"KUM.*","")
NA13['District'] = NA13['District'].map(lambda x: (re.sub(r"-.*","", x)))
```

```
NA13['District']  = NA13['District'].map(lambda x: (re.sub(r" (XX|IX|X?
↪I{0,3})(IX|IV|V?I{0,3})$", '', x)))
#NA8['District']  = NA8['District'].map(lambda x: (re.sub(r" (XX|IX|X?
↪I{0,3})(IX|IV|V?I{0,3})$", '', x)))
NA13['District'].unique()
```

```
[ ]: array(['PESHAWAR', 'NOWSHERA', 'CHARSADDA', 'MARDAN', 'SWABI', 'KOHAT',
        'KARAK', 'HANGU', 'ABBOTTABAD', 'HARIPUR', 'MANSEHRA',
        'Tor Ghar cum Mansehra', 'BATTAGRAM', 'KOHISTAN',
        'DERA ISMAIL KHAN', 'DIKHAN', 'BANNU', 'LAKKI MARWAT', 'BUNER',
        'SWAT', 'SHANGLA', 'CHITRAL', 'UPPER DIR', 'LOWER DIR',
        'MALAKAND P AREA', 'TRIBAL AREA', 'ISLAMABAD', 'RAWALPINDI',
        'ATTOCK', 'CHAKWAL', 'JHELUM', 'SARGODHA', 'KHUSHAB', 'MIANWALI',
        'BHAKKAR', 'FAISALABAD', 'Chiniot', 'Jhang', 'TOBA TEK SINGH',
        'GUJRANWALA', 'HAFIZABAD', 'GUJRAT', 'MANDI BAHAUDDIN', 'SIALKOT',
        'NAROWAL', 'LAHORE', 'SHEIKHUPURA', 'Sheikhupura', 'Nankana Sahib',
        'Kasur', 'KASUR', 'OKARA', 'MULTAN', 'LODHRAN', 'KHANEWAL',
        'SAHIWAL', 'PAKPATTAN', 'VEHARI', 'DERA GHAZI KHAN', 'RAJANPUR',
        'MUZAFFARGARH', 'LAYYAH', 'BAHAWALPUR', 'BAHAWALNAGAR',
        'RAHIM YAR KHAN', 'SUKKUR', 'GHOTKI', 'SHIKARPUR OLD SHIKARPUR',
        'SHEIKHUPUR', 'Larkana Old Larkana', 'LARKANA',
        'KAMBER SHAHDADKOT OLD LARKANA', 'JACOBABAD OLD JACOBABAD',
        'JACOBABAD', 'KASHMORE OLD JACOBABAD', 'NAUSHEHRO FEROZ',
        'NAWABSHAH', 'KHAIRPUR', 'MATIARI', 'HYDERABAD',
        'TANDO MUHAMMAD KHAN', 'TANDO ALLAHYAR', 'BADIN', 'MIRPUR KHAS',
        'MIRPURKHAS', 'UMERKOTOLD MIRPURKHAS', 'THARPARKAR',
        'JAMSHOROOLD DADU', 'DADU', 'SANGHAR', 'THATTA', 'KARACHI',
        'QUETTA', 'Quetta', 'Pishin', 'Killa Abdullah', 'Loralai', 'Zhob',
        'Sibi', 'Nasirabad', 'Kachhi', 'Kalat', 'Khuzdar', 'Awaran',
        'Kharan', 'Kech'], dtype=object)
```

```
[ ]: NA13.head()
```

```
[ ]:    District        Seat ConstituencyTitle              CandidateName  \
     0  PESHAWAR  PESHAWAR-I              NA-1        Aamir Shehzad Hashmi
     1  PESHAWAR  PESHAWAR-I              NA-1                  Akram Khan
     2  PESHAWAR  PESHAWAR-I              NA-1  Alhaaj Ghulam Ahmad Bilour
     3  PESHAWAR  PESHAWAR-I              NA-1                   Amir Syed
     4  PESHAWAR  PESHAWAR-I              NA-1         Bashir Ahmad Afridi

                                       Party  Votes  TotalValidVotes  \
     0                     Mustaqbil Pakistan     77           145924
     1                            Independent    182           145924
     2                    Awami National Party  24468           145924
     3  Pakistan Peoples Party (Shaheed Bhutto)    454           145924
     4        Muttahida Qaumi Movement Pakistan    117           145924
```

```
   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout
0                2103      146044                 320578   46.18%
1                2103      146044                 320578  46.18 %
2                2103      146044                 320578  46.18 %
3                2103      146044                 320578  46.18 %
4                2103      146044                 320578  46.18 %
```

[ ]:
```
NA8['Turnout']  = NA8['Turnout'].str.rstrip('%').str.rstrip(' ')
NA13['Turnout'] = NA13['Turnout'].str.rstrip('%').str.rstrip(' ')
NA8['Turnout']  = pd.to_numeric(NA8['Turnout'], errors='coerce')
NA13['Turnout'] = pd.to_numeric(NA13['Turnout'], errors='coerce')
```

[ ]:
```
####
NA8['Turnout'].str.rstrip('%').head(1).item(), (NA8['Turnout'].str.rstrip('%').
 ↪str.rstrip(' ').head(1).item())
####
```

[ ]:
```
('22.97 ', '22.97')
```

[ ]:
```
####
NA8['Turnout']
####
```

[ ]:
```
0       22.97
1       22.97
2       22.97
3       22.97
4       22.97
        …
2311    33.75
2312    33.75
2313    33.75
2314    33.75
2315    33.75
Name: Turnout, Length: 2316, dtype: float64
```

[ ]:
```
####
help(pd.to_numeric)
####
```

```
Help on function to_numeric in module pandas.core.tools.numeric:

to_numeric(arg, errors: 'DateTimeErrorChoices' = 'raise', downcast:
"Literal['integer', 'signed', 'unsigned', 'float'] | None" = None,
dtype_backend: 'DtypeBackend | lib.NoDefault' = <no_default>)
    Convert argument to a numeric type.
```

```
The default return dtype is `float64` or `int64`
depending on the data supplied. Use the `downcast` parameter
to obtain other dtypes.

Please note that precision loss may occur if really large numbers
are passed in. Due to the internal limitations of `ndarray`, if
numbers smaller than `-9223372036854775808` (np.iinfo(np.int64).min)
or larger than `18446744073709551615` (np.iinfo(np.uint64).max) are
passed in, it is very likely they will be converted to float so that
they can be stored in an `ndarray`. These warnings apply similarly to
`Series` since it internally leverages `ndarray`.

Parameters
----------
arg : scalar, list, tuple, 1-d array, or Series
    Argument to be converted.
errors : {'ignore', 'raise', 'coerce'}, default 'raise'
    - If 'raise', then invalid parsing will raise an exception.
    - If 'coerce', then invalid parsing will be set as NaN.
    - If 'ignore', then invalid parsing will return the input.

    .. versionchanged:: 2.2

    "ignore" is deprecated. Catch exceptions explicitly instead.

downcast : str, default None
    Can be 'integer', 'signed', 'unsigned', or 'float'.
    If not None, and if the data has been successfully cast to a
    numerical dtype (or if the data was numeric to begin with),
    downcast that resulting data to the smallest numerical dtype
    possible according to the following rules:

    - 'integer' or 'signed': smallest signed int dtype (min.: np.int8)
    - 'unsigned': smallest unsigned int dtype (min.: np.uint8)
    - 'float': smallest float dtype (min.: np.float32)

    As this behaviour is separate from the core conversion to
    numeric values, any errors raised during the downcasting
    will be surfaced regardless of the value of the 'errors' input.

    In addition, downcasting will only occur if the size
    of the resulting data's dtype is strictly larger than
    the dtype it is to be cast to, so if none of the dtypes
    checked satisfy that specification, no downcasting will be
    performed on the data.
dtype_backend : {'numpy_nullable', 'pyarrow'}, default 'numpy_nullable'
    Back-end data type applied to the resultant :class:`DataFrame`
    (still experimental). Behaviour is as follows:
```

```
    * ``"numpy_nullable"``: returns nullable-dtype-backed :class:`DataFrame`
      (default).
    * ``"pyarrow"``: returns pyarrow-backed nullable :class:`ArrowDtype`
      DataFrame.

    .. versionadded:: 2.0

Returns
-------
ret
    Numeric if parsing succeeded.
    Return type depends on input.  Series if Series, otherwise ndarray.

See Also
--------
DataFrame.astype : Cast argument to a specified dtype.
to_datetime : Convert argument to datetime.
to_timedelta : Convert argument to timedelta.
numpy.ndarray.astype : Cast a numpy array to a specified type.
DataFrame.convert_dtypes : Convert dtypes.

Examples
--------
Take separate series and convert to numeric, coercing when told to

>>> s = pd.Series(['1.0', '2', -3])
>>> pd.to_numeric(s)
0    1.0
1    2.0
2   -3.0
dtype: float64
>>> pd.to_numeric(s, downcast='float')
0    1.0
1    2.0
2   -3.0
dtype: float32
>>> pd.to_numeric(s, downcast='signed')
0    1
1    2
2   -3
dtype: int8
>>> s = pd.Series(['apple', '1.0', '2', -3])
>>> pd.to_numeric(s, errors='coerce')
0    NaN
1    1.0
2    2.0
3   -3.0
```

```
dtype: float64

Downcasting of nullable integer and floating dtypes is supported:

>>> s = pd.Series([1, 2, 3], dtype="Int64")
>>> pd.to_numeric(s, downcast="integer")
0    1
1    2
2    3
dtype: Int8
>>> s = pd.Series([1.0, 2.1, 3.0], dtype="Float64")
>>> pd.to_numeric(s, downcast="float")
0    1.0
1    2.1
2    3.0
dtype: Float32
```

```
[ ]: ####
     NA2.head(1)
     ####
```

```
[ ]:   District        Seat Constituency_title     Candidate_Name  \
     0  PESHAWAR  PESHAWAR-I               NA-1  Mr Sajid Abdullah

                           Party  Votes  Total_Valid_Votes  Total_Rejected_Votes  \
     0  Pakistan Tehreek-e-Insaf   2029              65642                  1552

        Total_Votes  Total_Registered_Voters  Turnout
     0        67194                   233907    28.73
```

```
[ ]: NA2['Year']  = "2002"
     NA8['Year']  = "2008"
     NA13['Year'] = "2013"
```

```
[ ]: ####
     NA2.head(3)
     ####
```

```
[ ]:   District        Seat Constituency_title          Candidate_Name  \
     0  PESHAWAR  PESHAWAR-I               NA-1       Mr Sajid Abdullah
     1  PESHAWAR  PESHAWAR-I               NA-1         Mr Shabir Ahmad
     2  PESHAWAR  PESHAWAR-I               NA-1  Mr Usman Bashir Bilour

                                   Party  Votes  Total_Valid_Votes  \
     0           Pakistan Tehreek-e-Insaf   2029              65642
     1  Muttahidda Majlis-e-Amal Pakistan  37179              65642
```

```
2               Awami National Party  23002             65642

   Total_Rejected_Votes  Total_Votes  Total_Registered_Voters  Turnout  Year
0                  1552        67194                   233907    28.73  2002
1                  1552        67194                   233907    28.73  2002
2                  1552        67194                   233907    28.73  2002
```

```
[ ]: print(NA2.head(), "\n", NA8.head(), "\n", NA13.head())
```

```
   District        Seat Constituency_title  \
0  PESHAWAR  PESHAWAR-I               NA-1
1  PESHAWAR  PESHAWAR-I               NA-1
2  PESHAWAR  PESHAWAR-I               NA-1
3  PESHAWAR  PESHAWAR-I               NA-1
4  PESHAWAR  PESHAWAR-I               NA-1

                     Candidate_Name                              Party  \
0                 Mr Sajid Abdullah          Pakistan Tehreek-e-Insaf
1                   Mr Shabir Ahmad  Muttahidda Majlis-e-Amal Pakistan
2            Mr Usman Bashir Bilour               Awami National Party
3  Mr Muhammad Khurshid Khan Advocate                       Independent
4           Mr Muhammad Muazzam Butt        Pakistan Muslim League(QA)

   Votes  Total_Valid_Votes  Total_Rejected_Votes  Total_Votes  \
0   2029              65642                  1552        67194
1  37179              65642                  1552        67194
2  23002              65642                  1552        67194
3   1537              65642                  1552        67194
4   1417              65642                  1552        67194

   Total_Registered_Voters  Turnout  Year
0                   233907    28.73  2002
1                   233907    28.73  2002
2                   233907    28.73  2002
3                   233907    28.73  2002
4                   233907    28.73  2002
    District        Seat ConstituencyTitle             CandidateName  \
0  Peshawar  Peshawar-1              NA-1              Abdullah Jan
1  Peshawar  Peshawar-1              NA-1              Ashoni Kumar
2  Peshawar  Peshawar-1              NA-1                 Aurangzeb
3  Peshawar  Peshawar-1              NA-1                 Ayub Shah
4  Peshawar  Peshawar-1              NA-1  Fakhri Alam Khan  Paracha

                                 Party  Votes  TotalValidVotes  \
0                          Independent    313            88325
1                          Independent    156            88325
2                          Independent    261            88325
```

```
3  Pakistan Peoples Party Parliamentarians  37682            88325
4                              Independent    184            88325

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout  Year
0                 629       88913                 387083    22.97  2008
1                 629       88913                 387083    22.97  2008
2                 629       88913                 387083    22.97  2008
3                 629       88913                 387083    22.97  2008
4                 629       88913                 387083    22.97  2008
    District        Seat ConstituencyTitle              CandidateName  \
0  PESHAWAR  PESHAWAR-I              NA-1        Aamir Shehzad Hashmi
1  PESHAWAR  PESHAWAR-I              NA-1                  Akram Khan
2  PESHAWAR  PESHAWAR-I              NA-1  Alhaaj Ghulam Ahmad Bilour
3  PESHAWAR  PESHAWAR-I              NA-1                   Amir Syed
4  PESHAWAR  PESHAWAR-I              NA-1          Bashir Ahmad Afridi

                                 Party  Votes  TotalValidVotes  \
0                   Mustaqbil Pakistan     77           145924
1                          Independent    182           145924
2                  Awami National Party  24468           145924
3  Pakistan Peoples Party (Shaheed Bhutto)    454           145924
4        Muttahida Qaumi Movement Pakistan    117           145924

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout  Year
0                2103      146044                 320578    46.18  2013
1                2103      146044                 320578    46.18  2013
2                2103      146044                 320578    46.18  2013
3                2103      146044                 320578    46.18  2013
4                2103      146044                 320578    46.18  2013
```

```
[ ]: print("NA2", NA2.isnull().any(), "\nNA8: ", NA8.isnull().any(), "\nNA13:", NA13.
     ↪isnull().any())
```

```
NA2 District                       False
Seat                           False
Constituency_title             False
Candidate_Name                 False
Party                          False
Votes                          False
Total_Valid_Votes              False
Total_Rejected_Votes           False
Total_Votes                    False
Total_Registered_Voters        False
Turnout                        False
Year                           False
dtype: bool
NA8:  District                     False
Seat                         False
```

```
ConstituencyTitle        False
CandidateName            False
Party                    False
Votes                    False
TotalValidVotes          False
TotalRejectedVotes       False
TotalVotes               False
TotalRegisteredVoters    False
Turnout                  False
Year                     False
dtype: bool
NA13: District                 False
Seat                     False
ConstituencyTitle        False
CandidateName            False
Party                    False
Votes                    False
TotalValidVotes          False
TotalRejectedVotes       False
TotalVotes               False
TotalRegisteredVoters    False
Turnout                  False
Year                     False
dtype: bool
```

```
[ ]: ####
     NA2.isnull().any()
     ####
```

```
[ ]: District                   False
     Seat                       False
     Constituency_title         False
     Candidate_Name             False
     Party                      False
     Votes                      False
     Total_Valid_Votes          False
     Total_Rejected_Votes       False
     Total_Votes                False
     Total_Registered_Voters    False
     Turnout                    False
     Year                       False
     dtype: bool
```

```
[ ]: print("\n NA2", NA2.columns, "\n NA8", NA8.columns, "\n NA13", NA13.columns)
```

```
 NA2 Index(['District', 'Seat', 'Constituency_title', 'Candidate_Name', 'Party',
       'Votes', 'Total_Valid_Votes', 'Total_Rejected_Votes', 'Total_Votes',
       'Total_Registered_Voters', 'Turnout', 'Year'],
      dtype='object')
 NA8 Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout', 'Year'],
      dtype='object')
 NA13 Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout', 'Year'],
      dtype='object')
```

```
[ ]: NA2.rename(columns={'Constituency_title':'ConstituencyTitle',
                         'Candidate_Name':'CandidateName',
                         'Total_Valid_Votes':'TotalValidVotes',
                         'Total_Rejected_Votes': 'TotalRejectedVotes',
                         'Total_Votes':'TotalVotes',
                         'Total_Registered_Voters':'TotalRegisteredVoters',},␣
      ↪inplace=True)
     NA2.columns, NA8.columns
```

```
[ ]: (Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
             'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
             'TotalRegisteredVoters', 'Turnout', 'Year'],
            dtype='object'),
      Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
             'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
             'TotalRegisteredVoters', 'Turnout', 'Year'],
            dtype='object'))
```

```
[ ]: ####
     NA2.shape, NA8.shape, NA13.shape
     ####
```

```
[ ]: ((1793, 12), (2316, 12), (4510, 12))
```

```
[ ]: # Concatenating All 3 Datasets

     df = pd.concat([NA2, NA8, NA13])
     print(df.shape)
     df.head()
```

```
(8619, 12)
```

```
[ ]:    District        Seat ConstituencyTitle                 CandidateName  \
     0  PESHAWAR  PESHAWAR-I              NA-1              Mr Sajid Abdullah
     1  PESHAWAR  PESHAWAR-I              NA-1               Mr Shabir Ahmad
     2  PESHAWAR  PESHAWAR-I              NA-1          Mr Usman Bashir Bilour
```

```
3  PESHAWAR  PESHAWAR-I             NA-1  Mr Muhammad Khurshid Khan Advocate
4  PESHAWAR  PESHAWAR-I             NA-1            Mr Muhammad Muazzam Butt

                              Party  Votes  TotalValidVotes  \
0           Pakistan Tehreek-e-Insaf   2029            65642
1  Muttahidda Majlis-e-Amal Pakistan  37179            65642
2                Awami National Party  23002            65642
3                         Independent   1537            65642
4         Pakistan Muslim League(QA)   1417            65642

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout  Year
0                1552       67194                 233907    28.73  2002
1                1552       67194                 233907    28.73  2002
2                1552       67194                 233907    28.73  2002
3                1552       67194                 233907    28.73  2002
4                1552       67194                 233907    28.73  2002
```

[ ]:
```
df.isnull().any()
```

[ ]:
```
District                 False
Seat                     False
ConstituencyTitle        False
CandidateName            False
Party                    False
Votes                    False
TotalValidVotes          False
TotalRejectedVotes       False
TotalVotes               False
TotalRegisteredVoters    False
Turnout                  False
Year                     False
dtype: bool
```

[ ]:
```
dist = df['District'].unique()
dist
```

[ ]:
```
array(['PESHAWAR', 'NOWSHERA', 'Nowshera', 'Charsadda', 'Charsdda',
       'Mardan', 'Sawabi', 'Kohat', 'Karak', 'Hangu', 'Abbottabad',
       'Haripur', 'Mansehra', 'Batagram', 'Kohistan', 'Dera Ismail Khan',
       'Dera Ismail Khan cum Tank', 'Bannu', 'Lakki Marwat', 'Buner',
       'Sawat', 'Shangla', 'Chitral', 'Upper Dir', 'Lower Dir',
       'Malakand', 'Tribal Area', 'Islamabad', 'Rawalpindi', 'Attock',
       'Chakwal', 'Jhelum', 'Sargodha', 'Khushab', 'khushab', 'Mianwali',
       'Bhakkar', 'Faisalabad', 'Jhang', 'Toba Tek Singh', 'Gujranwala',
       'Hafizabad', 'Gujrat', 'Mandi Bahauddin', 'Sialkot', 'Narowal',
       'Lahore', 'Sheikhupura', 'Sheiklhupura', 'Kasur', 'Okara',
       'Multan', 'Lodhran', 'Khanewal', 'Sahiwal', 'Pakpattan', 'Vehari',
       'Dera Ghazi Khan', 'Rajanpur', 'Muzaffargarh', 'Layyah',
       'Bahawalpur', 'Bahawalnagar', 'Rahimyar Khan', 'SUKKUR', 'Ghotki',
       'Shikarpur', 'Shikarapur', 'Larkana', 'Jacobabad',
       'Nausheroferoze', 'Nawabshah', 'Khairpur', 'Hyderabad', 'Badin',
       'Mirpurkhas', 'Tharparkar', 'Dadu', 'Sanghar', 'Thatta', 'Karachi',
       'Quetta', 'Quetta Cum Chagai cum Mastung', 'Pishin Cum Ziarat',
       'Killa', 'Loralai', 'Zhob Cum Killa Saifullah',
       'Sibi Cum Kolhu Cum Dera Bugti', 'Nasirabad', 'Kachhi',
       'Kalat Cum Mastung', 'Khuzdar', 'Awaran Cum Lasbela',
       'Kharan Cum Panjgur', 'Kech Cum Gwadar', 'Peshawar', 'Swabi',
       'Battagram', 'DIKhan', 'Swat', 'Malakand PA', 'MBDin',
       'Nanka Sahib', 'Nankana Sahib', 'Sukkur cum ShikarpurI',
       'Sukkur cum ShikarpurII', 'Shikarpur Old Shikarpur',
       'Larkana Old Larkana', 'Larka cum',
       'Kamber Shahdadkot Old Larkana', 'Jacobabad  Old Jacobabad',
       'Kashmore Old Jacobabad', 'Naushero Feroze', 'Matiari',
       'Tando Muhammad Khan', 'Tando Allahyar', 'Umerkot Old Mirpurkhas',
       'Jamshoro Old Dadu', 'Pishin', 'Killa Abdullah', 'Zhob', 'Sibi',
       'Bolan', 'Kalat', 'Awaran', 'Kharan', 'Kech', 'CHARSADDA',
       'MARDAN', 'SWABI', 'KOHAT', 'KARAK', 'HANGU', 'ABBOTTABAD',
       'HARIPUR', 'MANSEHRA', 'Tor Ghar cum Mansehra', 'BATTAGRAM',
       'KOHISTAN', 'DERA ISMAIL KHAN', 'DIKHAN', 'BANNU', 'LAKKI MARWAT',
       'BUNER', 'SWAT', 'SHANGLA', 'CHITRAL', 'UPPER DIR', 'LOWER DIR',
       'MALAKAND P AREA', 'TRIBAL AREA', 'ISLAMABAD', 'RAWALPINDI',
       'ATTOCK', 'CHAKWAL', 'JHELUM', 'SARGODHA', 'KHUSHAB', 'MIANWALI',
       'BHAKKAR', 'FAISALABAD', 'Chiniot', 'TOBA TEK SINGH', 'GUJRANWALA',
       'HAFIZABAD', 'GUJRAT', 'MANDI BAHAUDDIN', 'SIALKOT', 'NAROWAL',
       'LAHORE', 'SHEIKHUPURA', 'KASUR', 'OKARA', 'MULTAN', 'LODHRAN',
       'KHANEWAL', 'SAHIWAL', 'PAKPATTAN', 'VEHARI', 'DERA GHAZI KHAN',
       'RAJANPUR', 'MUZAFFARGARH', 'LAYYAH', 'BAHAWALPUR', 'BAHAWALNAGAR',
       'RAHIM YAR KHAN', 'GHOTKI', 'SHIKARPUR OLD SHIKARPUR',
       'SHEIKHUPUR', 'LARKANA', 'KAMBER SHAHDADKOT OLD LARKANA',
       'JACOBABAD OLD JACOBABAD', 'JACOBABAD', 'KASHMORE OLD JACOBABAD',
       'NAUSHEHRO FEROZ', 'NAWABSHAH', 'KHAIRPUR', 'MATIARI', 'HYDERABAD',
       'TANDO MUHAMMAD KHAN', 'TANDO ALLAHYAR', 'BADIN', 'MIRPUR KHAS',
       'MIRPURKHAS', 'UMERKOTOLD MIRPURKHAS', 'THARPARKAR',
       'JAMSHOROOLD DADU', 'DADU', 'SANGHAR', 'THATTA', 'KARACHI',
       'QUETTA'], dtype=object)
```

[ ]:
```
####
df['District'] = df['District'].str.lower()
df.District.tail(1).item()
####
```

[ ]:
```
'kech'
```

```
[ ]: ####
     help(str.strip)
     ####
```

```
Help on method_descriptor:

strip(self, chars=None, /) unbound builtins.str method
    Return a copy of the string with leading and trailing whitespace removed.

    If chars is given and not None, remove characters in chars instead.
```

```
[ ]: # convert to lower case
     df['District'] = df['District'].str.lower()
     # remove trailing white spaces
     df['District'] = df['District'].str.strip()
```

```
[ ]: dist = df['District'].unique()
     dist
```

```
[ ]: array(['peshawar', 'nowshera', 'charsadda', 'charsdda', 'mardan',
            'sawabi', 'kohat', 'karak', 'hangu', 'abbottabad', 'haripur',
            'mansehra', 'batagram', 'kohistan', 'dera ismail khan',
            'dera ismail khan cum tank', 'bannu', 'lakki marwat', 'buner',
            'sawat', 'shangla', 'chitral', 'upper dir', 'lower dir',
            'malakand', 'tribal area', 'islamabad', 'rawalpindi', 'attock',
            'chakwal', 'jhelum', 'sargodha', 'khushab', 'mianwali', 'bhakkar',
            'faisalabad', 'jhang', 'toba tek singh', 'gujranwala', 'hafizabad',
            'gujrat', 'mandi bahauddin', 'sialkot', 'narowal', 'lahore',
            'sheikhupura', 'sheiklhupura', 'kasur', 'okara', 'multan',
            'lodhran', 'khanewal', 'sahiwal', 'pakpattan', 'vehari',
            'dera ghazi khan', 'rajanpur', 'muzaffargarh', 'layyah',
            'bahawalpur', 'bahawalnagar', 'rahimyar khan', 'sukkur', 'ghotki',
            'shikarpur', 'shikarapur', 'larkana', 'jacobabad',
            'nausheroferoze', 'nawabshah', 'khairpur', 'hyderabad', 'badin',
            'mirpurkhas', 'tharparkar', 'dadu', 'sanghar', 'thatta', 'karachi',
            'quetta', 'quetta cum chagai cum mastung', 'pishin cum ziarat',
            'killa', 'loralai', 'zhob cum killa saifullah',
            'sibi cum kolhu cum dera bugti', 'nasirabad', 'kachhi',
            'kalat cum mastung', 'khuzdar', 'awaran cum lasbela',
            'kharan cum panjgur', 'kech cum gwadar', 'swabi', 'battagram',
            'dikhan', 'swat', 'malakand pa', 'mbdin', 'nanka sahib',
            'nankana sahib', 'sukkur cum shikarpuri', 'sukkur cum shikarpurii',
            'shikarpur old shikarpur', 'larkana old larkana', 'larka cum',
            'kamber shahdadkot old larkana', 'jacobabad  old jacobabad',
            'kashmore old jacobabad', 'naushero feroze', 'matiari',
            'tando muhammad khan', 'tando allahyar', 'umerkot old mirpurkhas',
            'jamshoro old dadu', 'pishin', 'killa abdullah', 'zhob', 'sibi',
            'bolan', 'kalat', 'awaran', 'kharan', 'kech',
            'tor ghar cum mansehra', 'malakand p area', 'chiniot',
            'rahim yar khan', 'sheikhupur', 'jacobabad old jacobabad',
            'naushehro feroz', 'mirpur khas', 'umerkotold mirpurkhas',
            'jamshoroold dadu'], dtype=object)
```

```
[ ]: # get the top 10 closest matches to "charsadda"
     matches = fuzzywuzzy.process.extract("charsadda", dist, limit=10,␣
      ↪scorer=fuzzywuzzy.fuzz.token_sort_ratio)
     matches
```

```
[ ]: [('charsadda', 100),
      ('charsdda', 94),
      ('mardan', 53),
      ('tharparkar', 53),
      ('kharan', 53),
      ('chakwal', 50),
      ('larkana', 50),
      ('rahimyar khan', 45),
      ('hafizabad', 44),
      ('jacobabad', 44)]
```

```
[ ]: # function to replace rows in the provided column of the provided dataframe
     # that match the provided string above the provided ratio with the provided␣
      ↪string
     def replace_matches_in_column(df, column, string_to_match, min_ratio = 90):
         # get a list of unique strings
         strings = df[column].unique()

         # get the top 10 closest matches to our input string
         matches = fuzzywuzzy.process.extract(string_to_match, strings,
                                              limit=10, scorer=fuzzywuzzy.fuzz.
      ↪token_sort_ratio)

         # only get matches with a ratio > 90
         close_matches = [matches[0] for matches in matches if matches[1] >=␣
      ↪min_ratio]

         # get the rows of all the close matches in our dataframe
         rows_with_matches = df[column].isin(close_matches)

         # replace all rows with close matches with the input matches
         df.loc[rows_with_matches, column] = string_to_match
```

```
[ ]: replace_matches_in_column(df=df, column='District', string_to_match="charsadda")
[ ]: dist = df['District'].unique()
     dist
```

```
[ ]: array(['peshawar', 'nowshera', 'charsadda', 'mardan', 'sawabi', 'kohat',
            'karak', 'hangu', 'abbottabad', 'haripur', 'mansehra', 'batagram',
            'kohistan', 'dera ismail khan', 'dera ismail khan cum tank',
            'bannu', 'lakki marwat', 'buner', 'sawat', 'shangla', 'chitral',
            'upper dir', 'lower dir', 'malakand', 'tribal area', 'islamabad',
            'rawalpindi', 'attock', 'chakwal', 'jhelum', 'sargodha', 'khushab',
            'mianwali', 'bhakkar', 'faisalabad', 'jhang', 'toba tek singh',
            'gujranwala', 'hafizabad', 'gujrat', 'mandi bahauddin', 'sialkot',
            'narowal', 'lahore', 'sheikhupura', 'sheiklhupura', 'kasur',
            'okara', 'multan', 'lodhran', 'khanewal', 'sahiwal', 'pakpattan',
            'vehari', 'dera ghazi khan', 'rajanpur', 'muzaffargarh', 'layyah',
            'bahawalpur', 'bahawalnagar', 'rahimyar khan', 'sukkur', 'ghotki',
            'shikarpur', 'shikarapur', 'larkana', 'jacobabad',
            'nausheroferoze', 'nawabshah', 'khairpur', 'hyderabad', 'badin',
            'mirpurkhas', 'tharparkar', 'dadu', 'sanghar', 'thatta', 'karachi',
            'quetta', 'quetta cum chagai cum mastung', 'pishin cum ziarat',
            'killa', 'loralai', 'zhob cum killa saifullah',
            'sibi cum kolhu cum dera bugti', 'nasirabad', 'kachhi',
            'kalat cum mastung', 'khuzdar', 'awaran cum lasbela',
            'kharan cum panjgur', 'kech cum gwadar', 'swabi', 'battagram',
            'dikhan', 'swat', 'malakand pa', 'mbdin', 'nanka sahib',
            'nankana sahib', 'sukkur cum shikarpuri', 'sukkur cum shikarpurii',
            'shikarpur old shikarpur', 'larkana old larkana', 'larka cum',
            'kamber shahdadkot old larkana', 'jacobabad  old jacobabad',
            'kashmore old jacobabad', 'naushero feroze', 'matiari',
            'tando muhammad khan', 'tando allahyar', 'umerkot old mirpurkhas',
            'jamshoro old dadu', 'pishin', 'killa abdullah', 'zhob', 'sibi',
            'bolan', 'kalat', 'awaran', 'kharan', 'kech',
            'tor ghar cum mansehra', 'malakand p area', 'chiniot',
            'rahim yar khan', 'sheikhupur', 'jacobabad old jacobabad',
            'naushehro feroz', 'mirpur khas', 'umerkotold mirpurkhas',
            'jamshoroold dadu'], dtype=object)
```

```
[ ]: replace_matches_in_column(df=df, column='District', string_to_match="nowshera")
     replace_matches_in_column(df=df, column='District',␣
      ↪string_to_match="rawalpindi")
     replace_matches_in_column(df=df, column='District',␣
      ↪string_to_match="sheikhupura")
     replace_matches_in_column(df=df, column='District', string_to_match="shikarpur")
     replace_matches_in_column(df=df, column='District', string_to_match="nankana␣
      ↪sahib")
```

```
[ ]: #del dist
```

```
pty = df['Party'].unique()
pty.sort()
pty
```

```
[ ]: array(['Aap Janab Sarkar Party', 'Afghan Qumi Movement (Pakistan)',
            'All Pakistan Bayrozgar Party', 'All Pakistan Muslim League',
            'All Pakistan Youth Working Party',
            'Awami Himayat Tehreek Pakistan', 'Awami Jamhuri Ittehad Pakistan',
            'Awami Justice Party', 'Awami Muslim League Pakistan',
            'Awami National Party', 'Awami Qaidat Party', 'Awami Qiadat Party',
            'Awami Workers Party', 'Azad Pakistan Party',
            'Bahawalpur National Awami Party', 'Balochistan National Congress',
            'Balochistan National Democratic Party',
            'Balochistan National Movement', 'Balochistan National Party',
            'Balochistan National Party (Awami)',
            'Balohistan National Movement', 'Christian Progressive Movement',
            'Communist Party of Pakistan', 'Ghareeb Awam Party',
            'Hazara Awami Ittehad Pakistan', 'Hazara Democratic Party',
            'Hazara Quami Mahaz', 'IndeIndependentdent',
            'IndeIndependentdentE', 'Independent', 'Independent (RETIRED)',
            'Independentt', 'Indepndent', 'Islami Inqalab Party',
            'Islami Tehreek Pakistan', 'Islamic Republican Party',
            'Istehkaam-e-Pakistan Movement Party', 'Istiqlal Party',
            'Istiqlil Party', 'Ittehad Milli Hazara',
            'Jamaat-e-Islami Pakistan',
            'Jamait Ahle-Hadith Pakistan(Elahi Zaheer)',
            'Jamhoori Wattan Party', 'Jamiat Ulama-e-Islam (F)',
            'Jamiat Ulama-e-Islam (S)', 'Jamiat Ulama-e-Pakistan  (Noorani)',
            'Jamiat Ulema-e-Pakistan (Niazi)',
            'JamiatUlema-e-pakistan(Nifaz-e-Shariat)', 'Jamote Qami Movement',
            'Jamote Qaumi Movement', 'Jannat Pakistan Party',
            'Jumiat Ulma-e-Islam(Nazryati)', 'Kakar Jamhoori Party Pakistan',
            'Karwan-i-Millat Pakistan', 'Labour Party Pakistan',
            'Labour Party Pakistan(Krandi)', 'Laboyr Party Pakistan',
            'MUTAHIDA DEENI MAHAZ', 'MUTTHIDA\xa0MAJLIS-E-AMAL\xa0PAKISTAN',
            'Majlis-e-Wahdat-e-Muslimeen Pakistan', 'Markazi Jamat-al-Hadais',
            'Markazi Jamiat Mushaikh Pakistan', 'Menecracy action Party',
            'Millat Party', 'Mohajar Qaumi Movement Pakistan',
            'Mohajir Ittehad Tehrik', 'Mohajir Kashmir Movement',
            'Mohajir Qaumi Movement Pakistan',
            'Mohib-e-Wattan Nowjawan Inqilabion Ki Anjuman (MNAKA)',
            'Mohib-e-Wattan Nowjawan Inqilabion Ki Anjuman(MNAKA)',
            'Mustaqbil Pakistan', 'Mutahida Baloch Movement Pakistan',
            'Mutahidda Qabail Party', 'Muttahida Qaumi Moement',
            'Muttahida Qaumi Moment', 'Muttahida Qaumi Movement',
            'Muttahida Qaumi Movement Pakistan', 'Muttahidda Majlis-e-Amal',
            'Muttahidda Majlis-e-Amal Pakistan', 'National Alliance',
            'National Party', 'National People Party Worker Group',
            'National Peoples Party', 'Nazim-e-Mistafa',
            'Nizam-e-Mustafa Party', 'PAK MUSLIM ALLIANCE',
            'Pak Justic Party (Haqiqi)', 'Pak Muslim Alliance',
            'Pak Wattan Party', 'Pakista Muslim League(J)',
            'Pakistan Amn Party', 'Pakistan Awami Inqalab',
            'Pakistan Awami Party', 'Pakistan Awami Quwat Party',
            'Pakistan Awami Tehreek', 'Pakistan Awami Tehrik-e-Inqilab',
            'Pakistan Awami party', 'Pakistan Bachao Party',
            'Pakistan Brohi Party', 'Pakistan Citizen Movement',
            'Pakistan Conservative Party', 'Pakistan Democratic Party',
            'Pakistan Democratic party', 'Pakistan Falah Party',
            'Pakistan Freedom Party', 'Pakistan Gharib Party',
            'Pakistan Insani Haqook Party (Pakistan Human Rights Party)',
            'Pakistan Islami Justice Party', 'Pakistan Ittehad Tehreek',
            'Pakistan Justice Party', 'Pakistan Kissan Ittehad',
            'Pakistan Mazdoor Kissan Party', 'Pakistan Motherland Party',
            'Pakistan Muhafiz Party', 'Pakistan Muhafiz Watan Party',
            'Pakistan Muhajir League', 'Pakistan Muhammadi Party',
            'Pakistan Mulim League(QA)', 'Pakistan Muslim League',
            'Pakistan Muslim League (F)', 'Pakistan Muslim League (J)',
            'Pakistan Muslim League (N)', 'Pakistan Muslim League (Safdar)',
            'Pakistan Muslim League (Sher-e-Bangal)',
            'Pakistan Muslim League (Zehri Group)',
            'Pakistan Muslim League Council',
            'Pakistan Muslim League Humkhiyal (Like Minded)',
            'Pakistan Muslim League \x93H\x94 Haqiqi',
            'Pakistan Muslim League(F)', 'Pakistan Muslim League(J)',
            'Pakistan Muslim League(N)', 'Pakistan Muslim League(QA)',
            'Pakistan Muslim League(Z)', 'Pakistan National Democratic Party',
            'Pakistan National Muslim League',
            'Pakistan Pakhtoonkhawa Milli Awami Party',
            'Pakistan Patriotic Movement',
            'Pakistan Peoples Party (Shaheed Bhutto)',
            'Pakistan Peoples Party Parliamentarian',
            'Pakistan Peoples Party Parliamentarians',
            'Pakistan Peoples Party Parlimentarians',
            'Pakistan Peoples Party(Sheed Bhutto)',
            'Pakistan Peoples Party(Sherpao)',
            'Pakistan Peoples party(Shaheed Bhutto)',
            'Pakistan Peoples party(Sherpao)', 'Pakistan Qaumi League',
            'Pakistan Qaumi Party', 'Pakistan Sariaki Party',
            'Pakistan Shia Political Party', 'Pakistan Sunni Tehreek',
            'Pakistan Tahreek-e-Insaf', 'Pakistan Tehreek-e-Insaf',
            'Pakistan Tehrek-e-Inqalab', 'Pakistan Workers Party',
            'Pakistan mazdoor Kissan Party',
            'Pakistan peoples Party Parlimentarians',
            'Pakistan peoples party(Sherpao)', 'Pasban',
            'Pashtoonkhwa Milli Awami Party', 'Punjab National Party',
            'Qaumi Inqilab Party', 'Qaumi Jamhoori Party',
            'Qaumi Tahaffaz party', 'Qaumi Watan Party (Sherpao)',
            'Qomi Awami Tehreek', 'Sairkistan Qaumi Ittehad',
            'Saraiki Sooba Movement Pakistan',
            'Seraiki Sooba Movement Pakistan', 'Shan-e-Pakistan Party',
            'Sindh Dost Ittehad', 'Sindh Taraqi Passand Party (STP)',
            'Sindh United Party', 'Sindh Urban-Rurel Alliance',
            'Sunni Ittehad Council', 'Sunni Tehreek',
            'Tameer-e-Pakistan Party', 'Tehreek Hussainia Pakistan',
            'Tehreek Tabdili Nizam Pakistan', 'Tehreek-e-Insaniat Pakistan',
            'Tehreek-e-Istehkaam Pakistan', 'Tehreek-e-Itehad Pakistan.',
            'Tehreek-e-Ittehad Ummat Pakistan',
            'Tehreek-e-Pasmanada Awam Pakistan', 'Tehreek-e-Suba Hazara',
            'Tehreek-e-Wafaq Pakistan', 'Tehrik-e-Istaqlal',
            'Tehrik-e-Masawaat', 'pakistan Social Democratic party'],
           dtype=object)
```

```
[ ]: df['Party'] = df['Party'].replace(['MUTTHIDA\xa0MAJLIS-E-AMAL\xa0PAKISTAN'],␣
      ↪'Muttahidda Majlis-e-Amal Pakistan')
     df['Party'] = df['Party'].replace(['Pakistan Muslim League'], 'Pakistan Muslim␣
      ↪League (QA)')
     # converting text to lower case and removing white spaces
     df['Party'] = df['Party'].str.lower()
     df['Party'] = df['Party'].str.strip()
```

```
[ ]: replace_matches_in_column(df=df, column='Party', string_to_match="Balochistan␣
      ↪National Movement")
     replace_matches_in_column(df=df, column='Party', string_to_match="Independent")
     replace_matches_in_column(df=df, column='Party', string_to_match="Istiqlal␣
      ↪Party")
     replace_matches_in_column(df=df, column='Party', string_to_match="Jamote Qaumi␣
      ↪Movement")
     replace_matches_in_column(df=df, column='Party', string_to_match="Labour Party␣
      ↪Pakistan")
     replace_matches_in_column(df=df, column='Party',␣
      ↪string_to_match="Mohib-e-Wattan Nowjawan Inqilabion Ki Anjuman (MNAKA)")
     replace_matches_in_column(df=df, column='Party', string_to_match="Muttahida␣
      ↪Qaumi Movement") # Muttahida Qaumi Movement Pakistan
     replace_matches_in_column(df=df, column='Party', string_to_match="Muttahidda␣
      ↪Majlis-e-Amal") # Muttahidda Majlis-e-Amal Pakistan
     replace_matches_in_column(df=df, column='Party', string_to_match="National␣
      ↪Peoples Party")
     replace_matches_in_column(df=df, column='Party',␣
      ↪string_to_match="Nizam-e-Mustafa Party")
```

```
replace_matches_in_column(df=df, column='Party', string_to_match="Pak Muslim␣
  ↪Alliance")
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Awami Party")
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Democratic Party")
# After analyzing each of the below strings.
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Muslim League (QA)", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Muslim League (N)", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Muslim League (J)", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Muslim League (F)", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Peoples Party Parliamentarians", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Peoples Party(Shaheed Bhutto)", min_ratio =95)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Peoples Party(Sherpao)", min_ratio =97)
replace_matches_in_column(df=df, column='Party', string_to_match="Pakistan␣
  ↪Tehreek-e-Insaf", min_ratio =95)
replace_matches_in_column(df=df, column='Party', string_to_match="Saraiki Sooba␣
  ↪Movement Pakistan", min_ratio =95)
```

[ ]:
```
df['Party'] = df['Party'].str.lower()
df['Party'].replace(['muttahida qaumi movement pakistan'], 'muttahida qaumi␣
  ↪movement', inplace=True)
df['Party'].replace(['indeindependentdente', 'independent (retired)',␣
  ↪'indepndent'], 'independent', inplace=True)
df['Party'].replace(['indeindependentdente','independent␣
  ↪(retired)','indepndent'], 'independent',inplace = True)
df['Party'].replace(['muttahidda majlis-e-amal␣
  ↪pakistan','mutthida\xa0majlis-e-amal\xa0pakistan'
                     ,'mutthidaï¿½majlis-e-amalï¿½pakistan']
                     ,'muttahidda majlis-e-amal' ,inplace = True)
df['Party'].replace(['nazim-e-mistafa'], 'nizam-e-mustafa party' ,inplace =␣
  ↪True)
df['Party'].replace(['pakistan muslim league (qa)'], 'pakistan muslim league␣
  ↪(q)' ,inplace = True)
df['Party'].replace(['pakistan muslim league council'], 'pakistan muslim league␣
  ↪(c)' ,inplace = True)
df['Party'].replace(['pakistan muslim league \x93h\x94 haqiqi'], 'pakistan␣
  ↪muslim league haqiqi' ,inplace = True)
```

```
df['Party'].replace(['pakistan muslim league(z)'], 'pakistan muslim league (z)'␣
↪,inplace = True)
df['Party'].replace(['pakistan peoples party(shaheed bhutto)'], 'pakistan␣
↪peoples party (shaheed bhutto)' ,inplace = True)
df['Party'].replace(['pakistan peoples party parliamentarians'], 'pakistan␣
↪peoples party parliamentarians' ,inplace = True)
df['Party'].replace(['pakistan sariaki party'], 'Pakistan Siraiki Party (T)'␣
↪,inplace = True)
df['Party'].replace(['pasban'], 'pasban pakistan' ,inplace = True)
df['Party'].replace(['qaumi watan party (sherpao)'], 'qaumi watan party'␣
↪,inplace = True)
df['Party'].replace(['tehreek-e-suba hazara'], 'tehreek-e-suba hazara pakistan'␣
↪,inplace = True)
#...
df['Party'].replace(['pashtoonkhwa milli awami party'], 'pakhtoonkhwa milli␣
↪Awami party' ,inplace = True)
df['Party'].replace(['pakistan amn party'], 'pakistan aman party' ,inplace =␣
↪True)
df['Party'].replace(['pakistan awami inqelabi'], 'Pakistan Awami Inqelabi␣
↪League' ,inplace = True)
df['Party'].replace(['pakistan freedom party'], 'pakistan freedom movement'␣
↪,inplace = True)
df['Party'].replace(['pakistan insani haqook party (pakistan human rights␣
↪party)'], 'pakistan human rights party' ,inplace = True)
df['Party'].replace(['awami justice party'], 'awami justice party pakistan'␣
↪,inplace = True)
df['Party'].replace(['indeindependentdent'], 'independent' ,inplace = True)
df['Party'].replace(['jamiat ulama-e-pakistan  (noorani)'], 'jamiat␣
↪ulama-e-pakistan (noorani)' ,inplace = True)
df['Party'].replace(['jumiat ulma-e-islam(nazryati)'], 'jamiat ulma-e-islam␣
↪nazryati pakistan' ,inplace = True)
df['Party'].replace(['majlis-e-wahdat-e-muslimeen pakistan'], 'Majlis␣
↪Wahdat-e-Muslimeen Pakistan' ,inplace = True)
df['Party'].replace(['markazi jamat-al-hadais'], 'Markazi Jamiat Ahl-e-Hadith'␣
↪,inplace = True)
df['Party'].replace(['mohib-e-wattan nowjawan inqilabion ki anjuman (mnaka)'],␣
↪'Muhib-e-Watan Noujawan Anqlabion Ki Anjuman (MNAKA)' ,inplace = True)

pty = df['Party'].unique()
pty.sort()
pty
```

```
/tmp/ipython-input-1221807450.py:2: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
```

```
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Party'].replace(['muttahida qaumi movement pakistan'], 'muttahida qaumi
movement', inplace=True)
```

```
[ ]: array(['Majlis Wahdat-e-Muslimeen Pakistan',
            'Markazi Jamiat Ahl-e-Hadith',
            'Muhib-e-Watan Noujawan Anqlabion Ki Anjuman (MNAKA)',
            'Pakistan Siraiki Party (T)', 'aap janab sarkar party',
            'afghan qumi movement (pakistan)', 'all pakistan bayrozgar party',
            'all pakistan muslim league', 'all pakistan youth working party',
            'awami himayat tehreek pakistan', 'awami jamhuri ittehad pakistan',
            'awami justice party pakistan', 'awami muslim league pakistan',
            'awami national party', 'awami qaidat party', 'awami qiadat party',
            'awami workers party', 'azad pakistan party',
            'bahawalpur national awami party', 'balochistan national congress',
            'balochistan national democratic party',
            'balochistan national movement', 'balochistan national party',
            'balochistan national party (awami)',
            'christian progressive movement', 'communist party of pakistan',
            'ghareeb awam party', 'hazara awami ittehad pakistan',
            'hazara democratic party', 'hazara quami mahaz', 'independent',
            'islami inqalab party', 'islami tehreek pakistan',
            'islamic republican party', 'istehkaam-e-pakistan movement party',
            'istiqlal party', 'ittehad milli hazara',
            'jamaat-e-islami pakistan',
            'jamait ahle-hadith pakistan(elahi zaheer)',
            'jamhoori wattan party', 'jamiat ulama-e-islam (f)',
            'jamiat ulama-e-islam (s)', 'jamiat ulama-e-pakistan (noorani)',
            'jamiat ulema-e-pakistan (niazi)',
            'jamiat ulma-e-islam nazryati pakistan',
            'jamiatulema-e-pakistan(nifaz-e-shariat)', 'jamote qaumi movement',
            'jannat pakistan party', 'kakar jamhoori party pakistan',
            'karwan-i-millat pakistan', 'labour party pakistan',
            'labour party pakistan(krandi)',
            'markazi jamiat mushaikh pakistan', 'menecracy action party',
            'millat party', 'mohajar qaumi movement pakistan',
            'mohajir ittehad tehrik', 'mohajir kashmir movement',
            'mohajir qaumi movement pakistan', 'mustaqbil pakistan',
            'mutahida baloch movement pakistan', 'mutahida deeni mahaz',
            'mutahidda qabail party', 'muttahida qaumi movement',
            'muttahidda majlis-e-amal', 'national alliance', 'national party',
            'national people party worker group', 'national peoples party',
            'nizam-e-mustafa party', 'pak justic party (haqiqi)',
            'pak muslim alliance', 'pak wattan party',
            'pakhtoonkhwa milli Awami party', 'pakistan aman party',
            'pakistan awami inqalab', 'pakistan awami party',
            'pakistan awami quwat party', 'pakistan awami tehreek',
            'pakistan awami tehrik-e-inqilab', 'pakistan bachao party',
            'pakistan brohi party', 'pakistan citizen movement',
            'pakistan conservative party', 'pakistan democratic party',
            'pakistan falah party', 'pakistan freedom movement',
            'pakistan gharib party', 'pakistan human rights party',
            'pakistan islami justice party', 'pakistan ittehad tehreek',
            'pakistan justice party', 'pakistan kissan ittehad',
            'pakistan mazdoor kissan party', 'pakistan motherland party',
            'pakistan muhafiz party', 'pakistan muhafiz watan party',
            'pakistan muhajir league', 'pakistan muhammadi party',
            'pakistan muslim league (c)', 'pakistan muslim league (f)',
            'pakistan muslim league (j)', 'pakistan muslim league (n)',
            'pakistan muslim league (q)', 'pakistan muslim league (safdar)',
            'pakistan muslim league (sher-e-bangal)',
            'pakistan muslim league (z)',
            'pakistan muslim league (zehri group)',
            'pakistan muslim league haqiqi',
            'pakistan muslim league humkhiyal (like minded)',
            'pakistan national democratic party',
            'pakistan national muslim league',
            'pakistan pakhtoonkhawa milli awami party',
            'pakistan patriotic movement',
            'pakistan peoples party (shaheed bhutto)',
            'pakistan peoples party parliamentarians',
            'pakistan peoples party(sherpao)', 'pakistan qaumi league',
            'pakistan qaumi party', 'pakistan shia political party',
            'pakistan social democratic party', 'pakistan sunni tehreek',
            'pakistan tehreek-e-insaf', 'pakistan tehrek-e-inqalab',
            'pakistan workers party', 'pasban pakistan',
            'punjab national party', 'qaumi inqilab party',
            'qaumi jamhoori party', 'qaumi tahaffaz party',
            'qaumi watan party', 'qomi awami tehreek',
            'sairkistan qaumi ittehad', 'saraiki sooba movement pakistan',
            'shan-e-pakistan party', 'sindh dost ittehad',
            'sindh taraqi passand party (stp)', 'sindh united party',
            'sindh urban-rurel alliance', 'sunni ittehad council',
            'sunni tehreek', 'tameer-e-pakistan party',
            'tehreek hussainia pakistan', 'tehreek tabdili nizam pakistan',
            'tehreek-e-insaniat pakistan', 'tehreek-e-istehkaam pakistan',
            'tehreek-e-itehad pakistan.', 'tehreek-e-ittehad ummat pakistan',
            'tehreek-e-pasmanada awam pakistan',
            'tehreek-e-suba hazara pakistan', 'tehreek-e-wafaq pakistan',
            'tehrik-e-istaqlal', 'tehrik-e-masawaat'], dtype=object)
```

```
[ ]: df['CandidateName'] = df['CandidateName'].str.lower()
     df['CandidateName'] = df['CandidateName'].str.strip()
     df['CandidateName'].head(10)
```

```
[ ]: 0                      mr sajid abdullah
     1                        mr shabir ahmad
     2                 mr usman bashir bilour
     3    mr muhammad khurshid khan advocate
     4               mr muhammad muazzam butt
     5                   dr arbab alamgir khan
     6     mr abdul manan akhunzada advocate
     7                   maulana rehmat ullah
     8             mr arbab muhammad ayub jan
     9                   mr iqbal zafar jhagra
     Name: CandidateName, dtype: object
```

```
[ ]: # removing mr from the starting of the CandidateName but leaving dr
     df['CandidateName'] = df.loc[: , 'CandidateName'].replace(regex=True,␣
      ↪to_replace="mr " , value="")
     df['CandidateName'] = df.loc[: , 'CandidateName'].replace(regex=True,␣
      ↪to_replace="mrs ", value="")
     df['CandidateName'] = df.loc[: , 'CandidateName'].replace(regex=True,␣
      ↪to_replace="miss", value="")
     df['CandidateName'].head(10)
```

```
[ ]: 0                      sajid abdullah
     1                        shabir ahmad
     2                 usman bashir bilour
     3    muhammad khurshid khan advocate
     4               muhammad muazzam butt
     5               dr arbab alamgir khan
     6     abdul manan akhunzada advocate
     7                maulana rehmat ullah
     8             arbab muhammad ayub jan
     9                 iqbal zafar jhagra
     Name: CandidateName, dtype: object
```

```
[ ]: df.loc[:, 'CandidateName'], df['CandidateName']
```

```
[ ]: (0                      mr sajid abdullah
      1                        mr shabir ahmad
      2                 mr usman bashir bilour
      3        mr muhammad khurshid khan advocate
4                  mr muhammad muazzam butt
                        …
4505                          gulab baloch
4506                  muhammad yasin baloch
4507                            nazimuddin
4508                         sayed essa nori
4509                           zobaida jalal
Name: CandidateName, Length: 8619, dtype: object,
0                      mr sajid abdullah
1                        mr shabir ahmad
2                   mr usman bashir bilour
3       mr muhammad khurshid khan advocate
4                  mr muhammad muazzam butt
                        …
4505                          gulab baloch
4506                  muhammad yasin baloch
4507                            nazimuddin
4508                         sayed essa nori
4509                           zobaida jalal
Name: CandidateName, Length: 8619, dtype: object)
```

[ ]:
```
####
help(df.loc)
####
```

```
Help on _LocIndexer in module pandas.core.indexing object:

class _LocIndexer(_LocationIndexer)
 |  Access a group of rows and columns by label(s) or a boolean array.
 |
 |  ``.loc[]`` is primarily label based, but may also be used with a
 |  boolean array.
 |
 |  Allowed inputs are:
 |
 |  - A single label, e.g. ``5`` or ``'a'``, (note that ``5`` is
 |    interpreted as a *label* of the index, and **never** as an
 |    integer position along the index).
 |  - A list or array of labels, e.g. ``['a', 'b', 'c']``.
 |  - A slice object with labels, e.g. ``'a':'f'``.
 |
 |    .. warning:: Note that contrary to usual python slices, **both** the
 |        start and the stop are included
 |
 |  - A boolean array of the same length as the axis being sliced,
 |    e.g. ``[True, False, True]``.
 |  - An alignable boolean Series. The index of the key will be aligned before
 |    masking.
 |  - An alignable Index. The Index of the returned selection will be the input.
 |  - A ``callable`` function with one argument (the calling Series or
 |    DataFrame) and that returns valid output for indexing (one of the above)
 |
 |  See more at :ref:`Selection by Label <indexing.label>`.
 |
 |  Raises
 |  ------
 |  KeyError
 |      If any items are not found.
 |  IndexingError
 |      If an indexed key is passed and its index is unalignable to the frame
index.
 |
 |  See Also
 |  --------
 |  DataFrame.at : Access a single value for a row/column label pair.
 |  DataFrame.iloc : Access group of rows and columns by integer position(s).
 |  DataFrame.xs : Returns a cross-section (row(s) or column(s)) from the
 |                 Series/DataFrame.
 |  Series.loc : Access group of values using labels.
 |
 |  Examples
 |  --------
 |  **Getting values**
 |
 |  >>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
 |  …                   index=['cobra', 'viper', 'sidewinder'],
 |  …                   columns=['max_speed', 'shield'])
 |  >>> df
 |              max_speed  shield
 |  cobra               1       2
 |  viper               4       5
 |  sidewinder          7       8
 |
 |  Single label. Note this returns the row as a Series.
 |
 |  >>> df.loc['viper']
 |  max_speed    4
 |  shield       5
 |  Name: viper, dtype: int64
 |
 |  List of labels. Note using ``[[]]`` returns a DataFrame.
 |
 |  >>> df.loc[['viper', 'sidewinder']]
 |              max_speed  shield
 |  viper               4       5
 |  sidewinder          7       8
 |
 |  Single label for row and column
 |
 |  >>> df.loc['cobra', 'shield']
 |  2
 |
 |  Slice with labels for row and single label for column. As mentioned
 |  above, note that both the start and stop of the slice are included.
 |
 |  >>> df.loc['cobra':'viper', 'max_speed']
 |  cobra    1
 |  viper    4
 |  Name: max_speed, dtype: int64
 |
 |  Boolean list with the same length as the row axis
 |
 |  >>> df.loc[[False, False, True]]
 |              max_speed  shield
 |  sidewinder          7       8
 |
 |  Alignable boolean Series:
 |
 |  >>> df.loc[pd.Series([False, True, False],
 |  …                  index=['viper', 'sidewinder', 'cobra'])]
 |                       max_speed  shield
 |  sidewinder          7       8
 |
 |  Index (same behavior as ``df.reindex``)
 |
 |  >>> df.loc[pd.Index(["cobra", "viper"], name="foo")]
 |         max_speed  shield
 |  foo
 |  cobra          1       2
 |  viper          4       5
 |
 |  Conditional that returns a boolean Series
 |
 |  >>> df.loc[df['shield'] > 6]
 |              max_speed  shield
 |  sidewinder          7       8
 |
 |  Conditional that returns a boolean Series with column labels specified
 |
 |  >>> df.loc[df['shield'] > 6, ['max_speed']]
 |              max_speed
 |  sidewinder          7
 |
 |  Multiple conditional using ``&`` that returns a boolean Series
 |
 |  >>> df.loc[(df['max_speed'] > 1) & (df['shield'] < 8)]
 |              max_speed  shield
 |  viper          4       5
 |
 |  Multiple conditional using ``|`` that returns a boolean Series
 |
 |  >>> df.loc[(df['max_speed'] > 4) | (df['shield'] < 5)]
 |              max_speed  shield
 |  cobra               1       2
 |  sidewinder          7       8
 |
 |  Please ensure that each condition is wrapped in parentheses ``()``.
 |  See the :ref:`user guide<indexing.boolean>`
 |  for more details and explanations of Boolean indexing.
 |
 |  .. note::
 |      If you find yourself using 3 or more conditionals in ``.loc[]``,
 |      consider using :ref:`advanced indexing<advanced.advanced_hierarchical>`.
 |
 |      See below for using ``.loc[]`` on MultiIndex DataFrames.
 |
 |  Callable that returns a boolean Series
 |
 |  >>> df.loc[lambda df: df['shield'] == 8]
 |              max_speed  shield
 |  sidewinder          7       8
 |
 |  **Setting values**
 |
 |  Set value for all items matching the list of labels
 |
 |  >>> df.loc[['viper', 'sidewinder'], ['shield']] = 50
 |  >>> df
 |              max_speed  shield
 |  cobra               1       2
 |  viper               4      50
 |  sidewinder          7      50
 |
 |  Set value for an entire row
 |
 |  >>> df.loc['cobra'] = 10
 |  >>> df
 |              max_speed  shield
 |  cobra              10      10
 |  viper               4      50
 |  sidewinder          7      50
```

```
|
|  Set value for an entire column
|
|  >>> df.loc[:, 'max_speed'] = 30
|  >>> df
|              max_speed  shield
|  cobra              30      10
|  viper              30      50
|  sidewinder         30      50
|
|  Set value for rows matching callable condition
|
|  >>> df.loc[df['shield'] > 35] = 0
|  >>> df
|              max_speed  shield
|  cobra              30      10
|  viper               0       0
|  sidewinder          0       0
|
|  Add value matching location
|
|  >>> df.loc["viper", "shield"] += 5
|  >>> df
|              max_speed  shield
|  cobra              30      10
|  viper               0       5
|  sidewinder          0       0
|
|  Setting using a ``Series`` or a ``DataFrame`` sets the values matching the
|  index labels, not the index positions.
|
|  >>> shuffled_df = df.loc[["viper", "cobra", "sidewinder"]]
|  >>> df.loc[:] += shuffled_df
|  >>> df
|              max_speed  shield
|  cobra              60      20
|  viper               0      10
|  sidewinder          0       0
|
|  **Getting values on a DataFrame with an index that has integer labels**
|
|  Another example using integers for the index
|
|  >>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
|  …                     index=[7, 8, 9], columns=['max_speed', 'shield'])
|  >>> df
|     max_speed  shield
|  7          1       2
|  8          4       5
|  9          7       8
|
|  Slice with integer labels for rows. As mentioned above, note that both
|  the start and stop of the slice are included.
|
|  >>> df.loc[7:9]
|     max_speed  shield
|  7          1       2
|  8          4       5
|  9          7       8
|
|  **Getting values with a MultiIndex**
|
|  A number of examples using a DataFrame with a MultiIndex
|
|  >>> tuples = [
|  …     ('cobra', 'mark i'), ('cobra', 'mark ii'),
|  …     ('sidewinder', 'mark i'), ('sidewinder', 'mark ii'),
|  …     ('viper', 'mark ii'), ('viper', 'mark iii')
|  … ]
|  >>> index = pd.MultiIndex.from_tuples(tuples)
|  >>> values = [[12, 2], [0, 4], [10, 20],
|  …           [1, 4], [7, 1], [16, 36]]
|  >>> df = pd.DataFrame(values, columns=['max_speed', 'shield'], index=index)
|  >>> df
|                       max_speed  shield
|  cobra      mark i           12       2
|             mark ii           0       4
|  sidewinder mark i           10      20
|             mark ii           1       4
|  viper      mark ii           7       1
|             mark iii         16      36
|
|  Single label. Note this returns a DataFrame with a single index.
|
|  >>> df.loc['cobra']
|           max_speed  shield
|  mark i          12       2
|  mark ii          0       4
|
|  Single index tuple. Note this returns a Series.
|
|  >>> df.loc[('cobra', 'mark ii')]
|  max_speed    0
|  shield       4
|  Name: (cobra, mark ii), dtype: int64
|
|  Single label for row and column. Similar to passing in a tuple, this
|  returns a Series.
|
|  >>> df.loc['cobra', 'mark i']
|  max_speed    12
|  shield        2
|  Name: (cobra, mark i), dtype: int64
|
|  Single tuple. Note using ``[[]]`` returns a DataFrame.
|
|  >>> df.loc[[('cobra', 'mark ii')]]
|                 max_speed  shield
|  cobra mark ii          0       4
|
|  Single tuple for the index with a single label for the column
|
|  >>> df.loc[('cobra', 'mark i'), 'shield']
|  2
|
|  Slice from index tuple to single label
|
|  >>> df.loc[('cobra', 'mark i'):'viper']
|                       max_speed  shield
|  cobra      mark i           12       2
|             mark ii           0       4
|  sidewinder mark i           10      20
|             mark ii           1       4
|  viper      mark ii           7       1
|             mark iii         16      36
|
|  Slice from index tuple to index tuple
|
|  >>> df.loc[('cobra', 'mark i'):('viper', 'mark ii')]
|                      max_speed  shield
|  cobra      mark i          12       2
|             mark ii          0       4
|  sidewinder mark i          10      20
|             mark ii          1       4
|  viper      mark ii          7       1
|
|  Please see the :ref:`user guide<advanced.advanced_hierarchical>`
|  for more details and explanations of advanced indexing.
|
|  Method resolution order:
|      _LocIndexer
|      _LocationIndexer
|      pandas._libs.indexing.NDFrameIndexerBase
|      builtins.object
```

```
 |
 |  Data and other attributes defined here:
 |
 |  __annotations__ = {'_takeable': 'bool'}
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from _LocationIndexer:
 |
 |  __call__(self, axis: 'Axis | None' = None) -> 'Self'
 |      Call self as a function.
 |
 |  __getitem__(self, key)
 |
 |  __setitem__(self, key, value) -> 'None'
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from _LocationIndexer:
 |
 |  __dict__
 |      dictionary for instance variables
 |
 |  __weakref__
 |      list of weak references to the object
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from _LocationIndexer:
 |
 |  axis = None
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from pandas._libs.indexing.NDFrameIndexerBase:
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __reduce__ = __reduce_cython__(…)
 |
 |  __reduce_cython__(self)
 |
 |  __setstate__ = __setstate_cython__(…)
 |
 |  __setstate_cython__(self, __pyx_state)
 |
 |  ----------------------------------------------------------------------
 |  Static methods inherited from pandas._libs.indexing.NDFrameIndexerBase:
 |
 |  __new__(*args, **kwargs) class method of
pandas._libs.indexing.NDFrameIndexerBase
```

```
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from pandas._libs.indexing.NDFrameIndexerBase:
 |
 |  name
 |
 |  ndim
 |
 |  obj
```

```
[ ]: cn = df['CandidateName'].unique()
     cn.sort()
     print("cn size: ", cn.shape, "\nValues", cn)
```

```
cn size:  (7657,)
Values [' mehwish chaudhary' 'aadil altaf unar' 'aamanullah' … 'zulqurnain'
 'zumarad khan' 'zumurrad khan']
```

```
[ ]: df['CandidateName']
```

```
[ ]: 0                       sajid abdullah
     1                         shabir ahmad
     2                  usman bashir bilour
     3        muhammad khurshid khan advocate
     4                 muhammad muazzam butt
                             …
     4505                      gulab baloch
     4506              muhammad yasin baloch
     4507                        nazimuddin
     4508                   sayed essa nori
     4509                     zobaida jalal
     Name: CandidateName, Length: 8619, dtype: object
```

```
[ ]: print(fuzzywuzzy.process.extract("zumurad khan" , cn, limit=10,␣
      ↪scorer=fuzzywuzzy.fuzz.token_sort_ratio)) # acceptance value > 90
     print(fuzzywuzzy.process.extract("zobaida jalal", cn, limit=10,␣
      ↪scorer=fuzzywuzzy.fuzz.token_sort_ratio)) # acceptance value > 79
```

```
[('zumurrad khan', 96), ('zumarad khan', 92), ('murad khan', 91), ('umara khan',
82), ('khan zada', 76), ('majid khan', 73), ('munir khan', 73), ('zahid khan',
73), ('muhammad khan', 72), ('musharaf khan', 72)]
[('zobaida jalal', 100), ('zubada jalal', 88), ('zubeda jalal', 80), ('jalil
zahid', 75), ('laiq zada', 64), ('qaisar jamal', 64), ('saida jan', 64), ('zahid
ali', 64), ('umaid ali dal', 62), ('aqal zaman', 61)]
```

```
[ ]: replace_matches_in_column(df=df, column='CandidateName',␣
      ↪string_to_match="zumurad khan", min_ratio=92)
     replace_matches_in_column(df=df, column='CandidateName',␣
      ↪string_to_match="zobaida jalal", min_ratio=80)
     replace_matches_in_column(df=df, column='CandidateName',␣
      ↪string_to_match="barkat ali", min_ratio=90)
     replace_matches_in_column(df=df, column='CandidateName',␣
      ↪string_to_match="muhammad yasin baloch", min_ratio=90)

     for candi in df['CandidateName'].unique(): # 7000
         replace_matches_in_column(df=df, column='CandidateName',␣
      ↪string_to_match=candi, min_ratio=90)

     # let us know the loop is completed
     print("All done!")
```

```
All done!
```

```
[ ]: df.to_csv('NA2002-18.csv', index=None)
```

```
[ ]: # Candidate List and Parties of 2018 Election

     cc   = pd.read_csv(os.path.join(path, "National Assembly Candidates List - 2018␣
      ↪Updated.csv"), encoding="ISO-8859-1")
     na18 = pd.read_csv(os.path.join(path, "2013-2018 Seat Changes in NA.csv"),␣
      ↪encoding="ISO-8859-1")
     pp   = pd.read_csv(os.path.join(path, "Political Parties in 2018 Elections -␣
      ↪Updated.csv"), encoding="ISO-8859-1")
     print(cc.shape, na18.shape, pp.shape)
```

```
(3438, 4) (288, 3) (119, 3)
```

```
[ ]: print(cc.columns, na18.columns)
```

```
Index(['NA#', 'Name', 'Party', 'Province'], dtype='object') Index(['2018 Seat
Number', 'Seat Name', '2013 Seat Number'], dtype='object')
```

```
[ ]: cc['NA#'] = 'NA-' + cc['NA#'].astype(str)
     cc['NA#']
```

```
[ ]: 0          NA-1
     1          NA-1
     2          NA-1
     3          NA-1
     4          NA-1
              …
     3433     NA-272
```

```
3434    NA-272
3435    NA-272
3436    NA-272
3437    NA-272
Name: NA#, Length: 3438, dtype: object
```

[ ]:
```
print(cc['NA#'].unique().shape) # 272
print(na18['2018 Seat Number'].unique().shape) # 273
na18.rename(columns={'2018 Seat Number': 'NA#'}, inplace=True)
na18.rename(columns={'Seat Name': 'Seat'}, inplace=True)
na18[na18['NA#'] == "Old Constituency Changed Considerably"]
```

```
(272,)
(273,)
```

[ ]:
```
                                       NA#  \
272  Old Constituency Changed Considerably
273  Old Constituency Changed Considerably
274  Old Constituency Changed Considerably
275  Old Constituency Changed Considerably
276  Old Constituency Changed Considerably
277  Old Constituency Changed Considerably
278  Old Constituency Changed Considerably
279  Old Constituency Changed Considerably
280  Old Constituency Changed Considerably
281  Old Constituency Changed Considerably
282  Old Constituency Changed Considerably
283  Old Constituency Changed Considerably
284  Old Constituency Changed Considerably
285  Old Constituency Changed Considerably
286  Old Constituency Changed Considerably
287  Old Constituency Changed Considerably

                                         Seat 2013 Seat Number
272                     Karachi 9 - Central 5           NA-247
273                     Karachi 12 - South 3           NA-250
274                               Attock III           NA-059
275                             Faisalabad 11           NA-085
276   Jhang- Cum-Chiniot (Old NA-87 Jhang-II)           NA-088
277                             Gujaranwala 7           NA-101
278                               Hafizabad 2           NA-103
279                                 Narowal 3           NA-117
280    Nankana Sahib-III (Old Sheikhupra-VII)           NA-137
281                                   Kasur 5           NA-142
282                                   Okara 5           NA-147
283                                 Sahiwal 4           NA-163
284                               Pakpattan 3           NA-166
```

```
285  JACOBABAD-CUM-KASHMORE(OLD JACOBABAD-II)          NA-209
286                                   Thatta 2          NA-238
287                       Kachhi-cum-Jhal Magsi          NA-267
```

```
[ ]: na18 = na18[na18['NA#'] != "Old Constituency Changed Considerably"]
     na18['NA#'] = na18.loc[:, 'NA#'].replace(regex=True, to_replace="NA-", value="")
     na18['NA#'] = pd.to_numeric(na18['NA#'])
     na18['NA#'] = na18['NA#'].astype(np.int64)
     na18['NA#'] = 'NA-' + na18['NA#'].astype(str)

     na18['NA#'].head()
```

```
[ ]: 0    NA-1
     1    NA-2
     2    NA-3
     3    NA-4
     4    NA-5
     Name: NA#, dtype: object
```

```
[ ]: # Add District column and its cleani
     na18['District'] = na18['Seat']
     # remove all those substring with ()
     na18['District'] = na18['District'].map(lambda x: (re.sub(r"\(*\)", "", x)))
     na18['District'] = na18['District'].map(lambda x: (re.sub(r"Cum.*", "", x)))
     na18['District'] = na18['District'].map(lambda x: (re.sub(r"-.*", "", x)))
     na18['District'] = na18['District'].map(lambda x: (re.sub(r"\(*\)", "", x)))
     na18['District'] = na18['District'].map(lambda x: (re.sub(r"(XX|IX|X?
      ↪I{0,3})(IX|IV|V?I{0,3})$", '', x)))
     na18['Distirct']  = na18['District'].map(lambda x: (re.sub(r" (XX|IX|X?
      ↪I{0,3})(IX|IV|V?I{0,3})$", "", x)))
     na18['District'].unique()
```

```
[ ]: array(['Chitral', 'Swat ', 'Upper Dir', 'Lower Dir', 'Lower Dir ',
            'Malakand', 'Boner', 'Shangla', 'Kohistan', 'Battagram',
            'Mansehra ', 'Abottabad ', 'Haripur', 'Swabi ', 'Mardan ',
            'Charsadda ', 'Nowshera ', 'Peshawar ', 'Kohat', 'Hangu', 'Karak',
            'Bannu', 'Lakki Marwat', 'Tank', 'D I Khan', 'D I Khan KUM Tank',
            'Tribal Area I ', 'Tribal Area II ', 'Tribal Area III ',
            'Tribal Area IV ', 'Tribal Area V ', 'Tribal Area VI ',
            'Tribal Area VII ', 'Tribal Area VIII ', 'Tribal Area IX ',
            'Tribal Area X ', 'Tribal Area XI ', 'Tribal Area XII ',
            'Islamabad 1', 'Islamabad 2', 'Islamabad 3', 'Attock ',
            'Rawalpindi ', 'Chakwal ', 'Jehlum ', 'Gujarat 1', 'Gujarat 2',
            'Gujarat 3', 'Gujarat 4', 'Sialkot 1', 'Sialkot 2', 'Sialkot 3',
            'Sialkot 4', 'Sialkot 5', 'Narowal 1', 'Narowal 2',
            'Gujaranwala 1', 'Gujaranwala 2', 'Gujaranwala 3', 'Gujaranwala 4',
            'Gujaranwala 5', 'Gujaranwala 6', 'MANDI BAHAUDDIN', 'Hafizabad 1',
            'Sargodha ', 'Khushab ', 'Mianwali ', 'Bhakkar ', 'Chiniot',
            'Faisalabad 1', 'Faisalabad 2', 'Faisalabad 3', 'Faisalabad 4',
            'Faisalabad 5', 'Faisalabad 6', 'Faisalabad 7', 'Faisalabad 8',
            'Faisalabad 9', 'Faisalabad 10', 'Toba Tek Singh 1',
            'Toba Tek Singh 2', 'Toba Tek Singh 3', 'Jhang', 'Nankana Sahib',
            'Sheikhupura 1', 'Sheikhupura 2', 'Sheikhupura 3', 'Sheikhupura 4',
            'Lahore 1', 'Lahore 2', 'Lahore 3', 'Lahore 4', 'Lahore 5',
            'Lahore 6', 'Lahore 7', 'Lahore 8', 'Lahore 9', 'Lahore 10',
            'Lahore 11', 'Lahore 12', 'Lahore 13', 'Lahore 14', 'Kasur 1',
            'Kasur 2', 'Kasur 3', 'Kasur 4', 'Okara 1', 'Okara 2', 'Okara 3',
            'Okara 4', 'Pakpattan 1', 'Pakpattan 2', 'Sahiwal 1', 'Sahiwal 2',
            'Sahiwal 3', 'Khanewal 1', 'Khanewal 2', 'Khanewal 3',
            'Khanewal 4', 'Multan 1', 'Multan 2', 'Multan 3', 'Multan 4',
            'Multan 5', 'Multan 6', 'Lodhran 1', 'Lodhran 2', 'Vehari 1',
            'Vehari 2', 'Vehari 3', 'Vehari 4', 'Bahawalnagar 1',
            'Bahawalnagar 2', 'Bahawalnagar 3', 'Bahawalnagar 4',
            'Bahawalpur 1', 'Bahawalpur 2', 'Bahawalpur 3', 'Bahawalpur 4',
            'Bahawalpur 5', 'Rahim Yar Khan 1', 'Rahim Yar Khan 2',
            'Rahim Yar Khan 3', 'Rahim Yar Khan 4', 'Rahim Yar Khan 5',
            'Rahim Yar Khan 6', 'Muzaffargarh 1', 'Muzaffargarh 2',
            'Muzaffargarh 3', 'Muzaffargarh 4', 'Muzaffargarh 5',
            'Muzaffargarh 6', 'Layyah 1', 'Layyah 2', 'Dera Ghazi Khan 1',
            'Dera Ghazi Khan 2', 'Dera Ghazi Khan 3', 'Dera Ghazi Khan',
            'Rajanpur 1', 'Rajanpur 2', 'Rajanpur 3',
            'JACOBABAD (OLD JACOBABAD', 'KASHMORE (OLD JACOBABAD',
            'SHIKARPUR (OLD SHIKARPUR', 'SHEIKHUPUR', 'Larkana (Old Larkana',
            'LARKANA', 'KAMBER SHAHDADKOT (OLD LARKANA', 'GHOTK', 'SUKKUR',
            'Khairpur 1', 'Khairpur 2', 'Khairpur 3', 'NAUSHEHRO FEROZ',
            'Nawabshah 1', 'Nawabshah 2', 'Sanghar 1', 'SANGHAR',
            'MIRPUR KHAS', 'MIRPURKHAS', 'UMERKOT(OLD MIRPURKHAS',
            'Tharparkar 1', 'Tharparkar 2', 'MATIAR', 'HYDERABAD',
            'TANDO MUHAMMAD KHAN', 'TANDO ALLAHYAR', 'BADIN', 'Sujawal',
            'Thatta 1', 'JAMSHORO(OLD DADU', 'DADU', 'Karachi 19 ',
            'Karachi 20 ', 'Karachi 21 ', 'Karachi 13 ', 'Karachi 18 ',
            'Karachi ', 'Karachi 14 ', 'Karachi 15 ', 'Karachi 16 ',
            'Karachi 17 ', 'Karachi 10 ', 'Karachi 11 ', 'Karachi 1 ',
            'Karachi 2 ', 'Karachi 3 ', 'Karachi 4 ', 'Karachi 5 ',
            'Karachi 6 ', 'Karachi 7 ', 'Karachi 8 ', 'Zhob', 'Loralai',
            'Sibi', 'Nasirabad', 'Jaffarabad', 'Pishin', 'Killa Abdullah',
            'Quetta', 'Quetta 3', 'Kalat', 'Chaghai', 'Khuzdar', 'Kharan',
            'Awaran', 'Kech'], dtype=object)
```

[ ]:
```
cc = cc.join(na18.set_index('NA#'), on='NA#')
cc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3438 entries, 0 to 3437
```

```
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   NA#               3438 non-null   object
 1   Name              3435 non-null   object
 2   Party             3438 non-null   object
 3   Province          3438 non-null   object
 4   Seat              3438 non-null   object
 5   2013 Seat Number  3438 non-null   object
 6   District          3438 non-null   object
 7   Distirct          3438 non-null   object
dtypes: object(8)
memory usage: 215.0+ KB
```

```
[ ]: print(pp.shape)
     pp['Name of Political Party'].unique()
```

```
(119, 3)
```

```
[ ]: array(['Aam Admi Tehreek Pakistan', 'Aam Awam Party',
            'Aam Log Party Pakistan',
            'All Pakistan Minority Movement Pakistan',
            'All Pakistan Muslim League',
            'All Pakistan Muslim League (Jinnah)', 'All Pakistan Tehreek',
            'Allah-O-Akbar Tehreek', 'Amun Taraqqi Party', 'Awam League',
            'Awami Justice Party Pakistan', 'Awami Muslim League Pakistan',
            'Awami National Party', 'Awami Party Pakistan-S',
            'Awami Workers Party', 'Balochistan Awami Party',
            'Balochistan National Movement', 'Balochistan National Party',
            'Balochistan National Party(Awami)', 'Barabri Party Pakistan',
            'Front National Pakistan', 'Grand Democratic Alliance',
            'Hazara Democratic Party', 'Humdardan-e-Watan Pakistan',
            'Islami Jamhoori Ittehad Pakistan', 'Islami Tehreek Pakistan',
            'Ittehad-e-Ummat Pakistan', 'Jamat-e-Islami Pakistan',
            'Jamhoori Watan Party', 'Jamiat Ulama-e-Islam Nazaryati Pakistan',
            'Jamiat Ulama-e-Islam(F)', 'Jamiat Ulema-e-Pakistan (Noorani)',
            'Jamiyat Ulma-e-Islam Pakistan(S)', 'Jamote Qaumi Movement',
            'Jannat Pakistan Party', 'Majlis Wahdat-e-Muslimeen Pakistan',
            'Markazi Jamiat Ahl-e-Hadith', 'Mohajir Qaumi Movement (Pakistan)',
            'Move on Pakistan',
            'Muhib-e-Watan Noujawan Anqlabion Ki Anjuman (MNAKA)',
            'Mustaqbil Pakistan', 'Mutahida Majlis-e-Amal Pakistan',
            'Mutahidda Ulema Mashaikh Council of Pakistan',
            'Mutahiddia Qabail Party', 'Mutihida League',
            'Muttahida Qaumi Movement Pakistan', 'National Party',
            'National Peace Council Party', 'Pak Sarzameen Party',
            'Pakhtoonkhwa Milli Awami Party', 'Pakistan Aman Party',
            'Pakistan Aman Tehreek', 'Pakistan Awami Inqelabi League',
            'Pakistan Awami League', 'Pakistan Awami Raj',
            'Pakistan Awami Tehreek', 'Pakistan Citizen Movement',
            'Pakistan Conservative Party', 'Pakistan Falah Party',
            'Pakistan Falahi Tehreek', 'Pakistan Freedom Movement',
            'Pakistan Human Party', 'Pakistan Human Rights Party',
            'Pakistan Islamic Republican Party',
            'Pakistan Justice and Democratic Party',
            'Pakistan Kissan Ittehad (Ch. Anwar)', 'Pakistan Muslim Alliance',
            'Pakistan Muslim League', 'Pakistan Muslim League (Council)',
            'Pakistan Muslim League (Functional)',
            'Pakistan Muslim League (Junejo)', 'Pakistan Muslim League (N)',
            'Pakistan Muslim League (Zia-ul-Haq Shaheed)',
            'Pakistan Muslim League Organization',
            'Pakistan Muslim League SHER-E-BANGAL A.K. Fazal-Ul-Haque',
            'Pakistan National Muslim League', 'Pakistan Peoples Party',
            'Pakistan Peoples Party (Shaheed Bhutto)',
            'Pakistan Peoples Party Parliamentarians',
            'Pakistan Peoples Party Workers', 'Pakistan Quami Yakjehti Party',
            'Pakistan Reh-e- Haq Party', 'Pakistan Siraiki Party (T)',
            'Pakistan Sunni Tehreek', 'Pakistan Supreme Democratic',
            'Pakistan Tehreek-e-Insaf', 'Pakistan Tehreek-e-Insaf (Nazriati)',
            'Pakistan Tehreek-e-Insaf-Gulalai', 'Pakistan Tehreek-e-Insaniat',
            'Pakistan Welfare Party', 'Pakistan Yaqeen Party',
            'Pasban Pakistan', 'Peoples Movement of Pakistan',
            'Peoples Muslim League (Pakistan)', 'Qaumi Watan Party',
            'Roshan Pakistan League', 'Saraikistan Democratic Party',
            'Sindh United Party', 'Sunni Ittehad Council', 'Sunni Tehreek',
            'Tabdeeli Pasand Party (Pakistan)', 'Tehreek Labbaik Pakistan',
            'Tehreek Tabdili Nizam Pakistan', 'Tehreek-e-Difa-e-Pakistan',
            'Tehreek-e-Labbaik Islam', 'Tehreek-e-Suba Hazara Pakistan',
            'Tehrik Jawanan Pakistan', 'Pakistan Rah-e-Haq Party',
            'Tehreek Jawanan Pakistan.', 'Allah-o-Akbar Tehreek',
            'Jamiat Ulma-e-Islam Nazryati Pakistan',
            'Pakistan Muslim League (Q)',
            'Pakistan Justice & Democratic Party',
            'Justice and Development Party Pakistan',
            'Muttahidda Qaumi Movement Pakistan', 'Muttahida Majlis-e-Amal',
            'Pakistan Muslim League-Nawaz ', 'Tehreek Jamhuriat Pakistan',
            'Pakistan Muhajir League'], dtype=object)
```

[ ]:
```python
pp.rename(columns={'Acronym':'PartyAcro'}              , inplace=True)
cc.rename(columns={'Party': 'PartyAcro'}               , inplace=True)
pp.rename(columns={'Name of Political Party': 'Party'}, inplace=True)
```

[ ]:
```python
# Clean Candidate file
```

```
pp['Party'].replace(['pakistan reh-e- haq party'], 'Pakistan Rah-e- Haq Party',␣
 ↪inplace=True)
pp['Party'].replace(['Pakistan Muslim League SHER-E-BANGAL A.K.␣
 ↪Fazal-Ul-Haque'], 'pakistan muslim league(sher-e-bangal)', inplace=True)
pp['Party'].replace(['Pakistan Muslim League (Zia-ul-Haq Shaheed)'], 'pakistan␣
 ↪muslim league (z)' ,inplace = True)
pp['Party'].replace(['Pakistan Muslim League (Junejo)'], 'pakistan muslim␣
 ↪league (j)' ,inplace = True)
pp['Party'].replace(['Pakistan Muslim League (Functional)'], 'pakistan muslim␣
 ↪league (f)' ,inplace = True)
pp['Party'].replace(['Pakistan Muslim League (Council)'], 'pakistan muslim␣
 ↪league (c)' ,inplace = True)
pp['Party'].replace(['Pakistan Muslim League-Nawaz'], 'pakistan muslim league␣
 ↪(n)' ,inplace = True)
pp['Party'].replace(['Pakistan Justice & Democratic Party'], 'Pakistan Justice␣
 ↪and Democratic Party' ,inplace = True)
pp['Party'].replace(['Pakistan Kissan Ittehad (Ch. Anwar)'], 'Pakistan Kissan␣
 ↪Ittehad' ,inplace = True)
pp['Party'].replace(['Jamiat Ulma-e-Islam Nazryati Pakistan'], 'Jamiat␣
 ↪Ulma-e-Islam Nazaryati Pakistan' ,inplace = True)
pp['Party'].replace(['Jamiat Ulma-e-Islam Nazryati Pakistan'], 'Jamiat␣
 ↪Ulma-e-Islam Nazaryati Pakistan' ,inplace = True)
pp['Party'].replace(['Jamiat Ulama-e-Islam(F)'], 'Jamiat Ulama-e-Islam (F)'␣
 ↪,inplace = True)
pp['Party'].replace(['Jamiat Ulama-e-Islam(S)'], 'Jamiat Ulama-e-Islam (S)'␣
 ↪,inplace = True)
pp['Party'].replace(['Mohajir Qaumi Movement (Pakistan)'], 'Mohajir Qaumi␣
 ↪Movement pakistan' ,inplace = True)
pp['Party'].replace(['Mutahida Majlis-e-Amal'], 'Muttahida Majlis-e-Amal'␣
 ↪,inplace = True)
pp['Party'].replace(['Muttahidda Qaumi Movement Pakistan'], 'Muttahida Qaumi␣
 ↪Movement Pakistan' ,inplace = True)
```

```
/tmp/ipython-input-3055495350.py:2: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  pp['Party'].replace(['pakistan reh-e- haq party'], 'Pakistan Rah-e- Haq
Party', inplace=True)
```

```
[ ]: # Remove duplicates
     pp.drop_duplicates(subset=['PartyAcro'], keep="first", inplace=True)
     pp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 112 entries, 0 to 118
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Party      112 non-null    object
 1   Symbol     106 non-null    object
 2   PartyAcro  111 non-null    object
dtypes: object(3)
memory usage: 3.5+ KB
```

```
[ ]: pp
```

```
[ ]:                                      Party       Symbol PartyAcro
     0                 Aam Admi Tehreek Pakistan          Mug      AATP
     1                            Aam Awam Party  Wheat Bunch       AAP
     2                     Aam Log Party Pakistan          Hut      ALPP
     3    All Pakistan Minority Movement Pakistan      Giraffe     APMMP
     4                All Pakistan Muslim League        Eagle      APML
     ..                                       …            …         …
     111              Pakistan Muslim League (Q)        cycle      PMLQ
     112   Pakistan Justice and Democratic Party          NaN      PJDP
     113   Justice and Development Party Pakistan          NaN      JDPP
     117              Tehreek Jamhuriat Pakistan          NaN        TJ
     118                 Pakistan Muhajir League          NaN       NaN

     [112 rows x 3 columns]
```

```
[ ]: cc[cc['PartyAcro']=='PTI'].head()
```

```
[ ]:      NA#                    Name PartyAcro            Province       Seat  \
     10  NA-1             Abdul Latif       PTI  Khyber Pakhtunkhwa    Chitral
     20  NA-2       Dr. Haider Ali Khan       PTI  Khyber Pakhtunkhwa     Swat I
     28  NA-3           Saleem Rehman       PTI  Khyber Pakhtunkhwa    Swat II
     37  NA-4             Murad Saeed       PTI  Khyber Pakhtunkhwa   Swat III
     49  NA-5  Sahibzada Sibghat Ullah       PTI  Khyber Pakhtunkhwa  Upper Dir

        2013 Seat Number  District  Distirct
     10           NA-032   Chitral   Chitral
     20           NA-029      Swat      Swat
     28           NA-030      Swat      Swat
     37      Newly Added      Swat      Swat
```

```
49           NA-033  Upper Dir  Upper Dir
```

[ ]:
```
####
cc.shape, pp.shape
####
```

[ ]:
```
((3438, 8), (112, 3))
```

[ ]:
```
# del cnd
cnd = cc.join(pp.set_index('PartyAcro'), on='PartyAcro')
cnd.shape
```

[ ]:
```
(3438, 10)
```

[ ]:
```
cnd.info(), cnd.head()
```

[ ]:
```
####
cnd['Name']
####
```

[ ]:
```
0                 Eid Ul Hussain
1                 Mohammad Amjad
2                 Mohammad Yahya
3                Nisar Dastageer
4        Shahzada Muharamad Taim
                      …
3433     Muhammad Akhtar Mengal
3434     Mohammad Aslam Bhootani
3435              Mohammad Akram
3436          Nawab Khan Bizenjo
3437         Wazir Ahmed Noorani
Name: Name, Length: 3438, dtype: object
```

[ ]:
```
cnd[cnd['PartyAcro']=="PTI"].head()



cnd['Name'] = cnd['Name'].str.replace('[^a-zA-Z ]', '', regex=True )
#cnd['Name'] = cnd['Name'].map(lambda x: (re.sub('[^a-zA-Z ]', '', x)))
cnd['Name'] = cnd['Name'].str.lower()
cnd['Name'] = cnd['Name'].str.strip()

cnd['Party'] = cnd['Party'].str.lower()
cnd['Party'] = cnd['Party'].str.strip()

cnd[cnd['PartyAcro']=="PTI"].head()
```

```
[ ]:      NA#                      Name PartyAcro           Province       Seat  \
    10   NA-1              abdul latif       PTI  Khyber Pakhtunkhwa    Chitral
    20   NA-2        dr haider ali khan       PTI  Khyber Pakhtunkhwa     Swat I
    28   NA-3             saleem rehman       PTI  Khyber Pakhtunkhwa    Swat II
    37   NA-4               murad saeed       PTI  Khyber Pakhtunkhwa   Swat III
    49   NA-5  sahibzada sibghat ullah       PTI  Khyber Pakhtunkhwa  Upper Dir

       2013 Seat Number   District   Distirct                          Party Symbol
    10           NA-032    Chitral    Chitral  pakistan tehreek-e-insaf    Bat
    20           NA-029       Swat       Swat  pakistan tehreek-e-insaf    Bat
    28           NA-030       Swat       Swat  pakistan tehreek-e-insaf    Bat
    37      Newly Added       Swat       Swat  pakistan tehreek-e-insaf    Bat
    49           NA-033  Upper Dir  Upper Dir  pakistan tehreek-e-insaf    Bat
```

```
[ ]: print(df.columns, cnd.columns)
     df.info()
     cnd.info()
```

```
Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
       'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
       'TotalRegisteredVoters', 'Turnout', 'Year'],
      dtype='object') Index(['NA#', 'Name', 'PartyAcro', 'Province', 'Seat',
'2013 Seat Number',
       'District', 'Distirct', 'Party', 'Symbol'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
Index: 8619 entries, 0 to 4509
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   District               8619 non-null   object
 1   Seat                   8619 non-null   object
 2   ConstituencyTitle      8619 non-null   object
 3   CandidateName          8619 non-null   object
 4   Party                  8619 non-null   object
 5   Votes                  8619 non-null   int64
 6   TotalValidVotes        8619 non-null   int64
 7   TotalRejectedVotes     8619 non-null   int64
 8   TotalVotes             8619 non-null   int64
 9   TotalRegisteredVoters  8619 non-null   int64
 10  Turnout                8619 non-null   float64
 11  Year                   8619 non-null   object
dtypes: float64(1), int64(5), object(6)
memory usage: 1.1+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3438 entries, 0 to 3437
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   NA#               3438 non-null   object
 1   Name              3435 non-null   object
 2   PartyAcro         3438 non-null   object
 3   Province          3438 non-null   object
 4   Seat              3438 non-null   object
 5   2013 Seat Number  3438 non-null   object
 6   District          3438 non-null   object
 7   Distirct          3438 non-null   object
 8   Party             1706 non-null   object
 9   Symbol            1686 non-null   object
dtypes: object(10)
memory usage: 268.7+ KB
```

```
[ ]: cnd.rename(columns={'NA#':'ConstituencyTitle'}, inplace=True)
     cnd.rename(columns={'Name of Political Party':'Party'}, inplace=True)
     cnd.rename(columns={'Name':'CandidateName'}, inplace=True)
```

```
[ ]: cnd.to_csv('Candidates2018.csv', index=None)
     pp.to_csv('Parties_cleand.csv', index=None)
```

```
[ ]: # Reading 2018 Results Data
     NA18 = pd.read_csv(os.path.join(path, "NA-Results2018 Ver 2.csv"), encoding =␣
      ↪"ISO-8859-1")
     print("Data Dimensions of NA18 are: ", NA18.shape)

     print("\nNA 2018.csv")
     NA18.info()

     NA18 = NA18.drop('Unnamed: 0', axis=1)
     NA18.rename(columns={'district':'District'}, inplace=True)

     NA18.District = NA18.Seat
     NA18['District'] = NA18['District'].str.replace("."," ") # to deal with D.I.␣
      ↪Khan
     NA18['District'] = NA18['District'].str.replace(r"\(.*\)","")
     NA18['District']  = NA18['District'] .str.replace('[^a-zA-Z -]', '')
     NA18['District'] = NA18['District'].str.replace(r"-.*","")
     NA18['District']  = NA18['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA18['District']  = NA18['District'] .str.replace(r" (XX|IX|X?I{0,3})(IX|IV|V?
      ↪I{0,3})$", '')
     NA18['District'].unique()

     NA18['Turnout'] = NA18['Turnout'].str.rstrip('%').str.rstrip(' ')
     NA18['Turnout'] = pd.to_numeric(NA18['Turnout'], errors='coerce')
```

```
NA18.rename(columns={'Constituency_Title':'ConstituencyTitle', 'Candidate_Name':
↪'CandidateName', 'Total_Valid_Votes':'TotalValidVotes',␣
↪'Total_Rejected_Votes':'TotalRejectedVotes', 'Total_Votes':'TotalVotes',␣
↪'Total_Registered_Voters':'TotalRegisteredVoters', 'Part':'Party' },␣
↪inplace=True)
NA18.columns
```

```
Data Dimensions of NA18 are:  (3428, 12)

NA 2018.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3428 entries, 0 to 3427
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0               3428 non-null   int64
 1   district                 3428 non-null   object
 2   Seat                     3428 non-null   object
 3   Constituency_Title       3428 non-null   object
 4   Candidate_Name           3428 non-null   object
 5   Part                     3428 non-null   object
 6   Votes                    3428 non-null   int64
 7   Total_Valid_Votes        3428 non-null   int64
 8   Total_Rejected_Votes     3428 non-null   int64
 9   Total_Votes              3428 non-null   int64
 10  Total_Registered_Voters  3428 non-null   int64
 11  Turnout                  3428 non-null   object
dtypes: int64(6), object(6)
memory usage: 321.5+ KB
```

```
[ ]: Index(['District', 'Seat', 'ConstituencyTitle', 'CandidateName', 'Party',
            'Votes', 'TotalValidVotes', 'TotalRejectedVotes', 'TotalVotes',
            'TotalRegisteredVoters', 'Turnout'],
           dtype='object')
```

```
[ ]: # convert to lower case
     NA18['District'] = NA18['District'].str.lower()
     # remove trailing white spaces
     NA18['District'] = NA18['District'].str.strip()

     # convert to lower case
     NA18['CandidateName'] = NA18['CandidateName'].str.lower()
     # remove trailing white spaces
     NA18['CandidateName'] = NA18['CandidateName'].str.strip()

     # convert to lower case
     NA18['Party'] = NA18['Party'].str.lower()
```

```
# remove trailing white spaces
NA18['Party'] = NA18['Party'].str.strip()
```

[ ]:
```
NA18.head()
```

[ ]:
```
  District     Seat ConstituencyTitle                     CandidateName  \
0  chitral  Chitral              NA-1      moulana abdul akbar chitrali
1  chitral  Chitral              NA-1                   saeed ur rehman
2  chitral  Chitral              NA-1                    muhammad yahya
3  chitral  Chitral              NA-1  shahzada muhammad taimur khisrao
4  chitral  Chitral              NA-1                   eid ul hussain

                              Party  Votes  TotalValidVotes  \
0  muttahida majlis-e-amal pakistan  48616           158925
1          pakistan rah-e-haq party   3223                0
2                       independent    698                0
3                       independent   2414                0
4              awami national party   3613                0

   TotalRejectedVotes  TotalVotes  TotalRegisteredVoters  Turnout
0                5430      164355                 269579    60.97
1                   0           0                      0    60.97
2                   0           0                      0    60.97
3                   0           0                      0    60.97
4                   0           0                      0    60.97
```

[ ]:
```
NA18.to_csv('NA2018_Clean.csv', index=None)
```