# Optimus

***Release 21.9***

**Argenis León and Luis Aguirre**

**Nov 04, 2021**

# CONTENTS:

CHAPTER

# ONE

# LOAD

class optimus.engines.base.io.load.**BaseLoad**(*op*)

**avro**(*filepath_or_buffer*, *n_rows=None*, *storage_options=None*, *conn=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType

Loads a dataframe from a avro file.

**Parameters**

- **filepath_or_buffer** – path or location of the file. Must be string dataType
- **n_rows** –
- **storage_options** –
- **conn** –
- **args** – custom argument to be passed to the spark avro function
- **kwargs** – custom keyword arguments to be passed to the spark avro function

**csv**(*filepath_or_buffer*, *sep=','*, *header=True*, *infer_schema=True*, *encoding='UTF-8'*, *n_rows=None*, *null_value='None'*, *quoting=3*, *lineterminator='\r\n'*, *on_bad_lines='warn'*, *cache=False*, *na_filter=False*, *storage_options=None*, *conn=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType

Loads a dataframe from a csv file. It is the same read.csv Spark function with some predefined params.

**Parameters**

- **encoding** –
- **storage_options** –
- **quoting** –
- **filepath_or_buffer** – path or location of the file.
- **sep** – usually delimiter mark are ‘,’ or ‘;’.
- **header** – tell the function whether dataset has a header row. True default.
- **infer_schema** – infers the input schema automatically from data.
- **n_rows** –
- **null_value** –
- **cache** –
- **na_filter** –
- **lineterminator** –

- **on_bad_lines** –
- **conn** –

It requires one extra pass over the data. True default.

:return dataFrame

**excel**(*filepath_or_buffer*, *header=0*, *sheet_name=0*, *merge_sheets=False*, *skip_rows=0*, *n_rows=None*, *storage_options=None*, *conn=None*, *n_partitions=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Loads a dataframe from a excel file.

**Parameters**

- **filepath_or_buffer** – Path or location of the file. Must be string dataType
- **header** –
- **sheet_name** – excel sheet name
- **merge_sheets** –
- **skip_rows** –
- **n_rows** –
- **storage_options** –
- **conn** –
- **n_partitions** –
- **args** – custom argument to be passed to the excel function
- **kwargs** – custom keyword arguments to be passed to the excel function

**Returns**

**file**(*path*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Try to infer the file data format and encoding and load the data into a dataframe.

**Parameters**

- **path** – Path to the file you want to load.
- **args** – custom argument to be passed to the spark avro function.
- **kwargs** – custom keyword arguments to be passed to the spark avro function.

**Returns**

**hdf5**(*path*, *columns=None*, *n_partitions=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Loads a dataframe from a HDF5 file.

**Parameters**

- **path** – path or location of the file. Must be string dataType.
- **columns** – Specific column names to be loaded from the file.
- **n_partitions** –
- **args** – custom argument to be passed to the spark avro function.
- **kwargs** – custom keyword arguments to be passed to the spark avro function.

**Returns**

**json**(*filepath_or_buffer*, *multiline=False*, *n_rows=False*, *storage_options=None*, *conn=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Loads a dataframe from a json file.

**Parameters**

- **filepath_or_buffer** – path or location of the file.
- **multiline** –
- **n_rows** –
- **storage_options** –
- **conn** –
- **args** –
- **kwargs** –

**Returns**

**static model**(*path*)
Load a machine learning model from a file.

**Parameters** **path** – Path to the file we want to load.

**Returns**

**orc**(*path*, *columns*, *storage_options=None*, *conn=None*, *n_partitions=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Loads a dataframe from a OCR file.

**Parameters**

- **path** – path or location of the file. Must be string dataType.
- **columns** – Specific column names to be loaded from the file.
- **storage_options** –
- **conn** –
- **args** – custom argument to be passed to the spark avro function.
- **kwargs** – custom keyword arguments to be passed to the spark avro function.

**parquet**(*filepath_or_buffer*, *columns=None*, *n_rows=None*, *storage_options=None*, *conn=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType
Loads a dataframe from a parquet file.

**Parameters**

- **filepath_or_buffer** – path or location of the file. Must be string dataType
- **columns** – select the columns that will be loaded. In this way you do not need to load all the dataframe
- **storage_options** –
- **conn** –
- **args** – custom argument to be passed to the spark parquet function
- **kwargs** – custom keyword arguments to be passed to the spark parquet function

**tsv**(*filepath_or_buffer*, *header=True*, *infer_schema=True*, **args*, ***kwargs*)
Loads a dataframe from a tsv(Tabular separated values) file.

**Parameters**

- **filepath_or_buffer** – Path or location of the file. Must be string dataType
- **header** –
- **infer_schema** –
- **args** – custom argument to be passed to the spark avro function.
- **kwargs** – custom keyword arguments to be passed to the spark avro function.

**Returns**

**xml**(*path*, *n_rows=None*, *storage_options=None*, *conn=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType

Loads a dataframe from a XML file.

**Parameters**

- **path** –
- **n_rows** –
- **storage_options** –
- **conn** –
- **args** –
- **kwargs** –

**Returns**

CHAPTER

# TWO

# SAVE

class optimus.engines.base.io.save.**BaseSave**(*root: DataFrameType*)

**csv**(*path*, **args*, ***kwargs*)
Save data frame to a CSV file. :param path: path where the spark will be saved. :param mode: 'rb', 'wt', etc it uses the default value. :return: Dataframe in a CSV format in the specified path.

**database_table**(*table*, *db*, **args*, ***kwargs*)

**Parameters**

- **table** –
- **db** –
- **args** –
- **kwargs** –

**Returns**

**excel**(*path*, *mode='w'*, **args*, ***kwargs*)
Save data frame to a CSV file. :param path: File path or object :param mode: Python write mode, default 'w'. it uses the default value. :return: Dataframe in a CSV format in the specified path.

**file**(*path: str*, **args*, ***kwargs*)

**Parameters**

- **path** –
- **args** –
- **kwargs** –

**Returns**

**hdf5**(*path*, **args*, ***kwargs*)

**Parameters**

- **path** –
- **args** –
- **kwargs** –

**Returns**

`json`(*path*, **args*, ***kwargs*)

Save data frame in a json file :param path: path where the spark will be saved. :param mode: Specifies the behavior of the save operation when data already exists.

"append": Append contents of this DataFrame to existing data. "overwrite" (default case): Overwrite existing data. "ignore": Silently ignore this operation if data already exists. "error": Throw an exception if data already exists.

**Parameters** `num_partitions` – the number of partitions of the DataFrame

**Returns**

`orc`(*path*, **args*, ***kwargs*)

**Parameters**

- `path` –
- `args` –
- `kwargs` –

**Returns**

`parquet`(*path*, *mode='w'*, *num_partitions=1*, **args*, ***kwargs*)

Save data frame to a parquet file :param path: File path or object :param mode: Specifies the behavior of the save operation when data already exists.

"append": Append contents of this DataFrame to existing data. "overwrite" (default case): Overwrite existing data. "ignore": Silently ignore this operation if data already exists. "error": Throw an exception if data already exists.

**Parameters** `num_partitions` – the number of partitions of the DataFrame

**Returns**

`xml`(*filename: Optional[str] = None*, *mode: str = 'w'*)

**Parameters**

- `filename` –
- `mode` –

**Returns**

CHAPTER

# THREE

# COLUMNS

class optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)

Base class for all Cols implementations

**abs**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return the absolute numeric value of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the absolute value of each element.

**acos**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Apply arccosine function to a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return: Column containing the arccosine of each element.

**acosh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Apply the arcus hyperbolic cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic cosine of each element.

**add**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType

Apply a plus operation to two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_col** – Single output column in case no value is passed.

**Returns** Dataframe with the result of the arithmetic operation appended.

**agg_exprs**(*cols='*'*, *funcs=None*, **args*, *compute=True*, *tidy=True*, *parallel=False*)

Run a list of aggregation functions.

**Parameters**

- **cols** – Column over with to apply the aggregations functions.
- **funcs** – List of aggregation functions.

- **args** –
- **compute** – Compute the result or return a delayed function.
- **tidy** – Compact the dict output.
- **parallel** – Execute the function in every column or apply it over the whole dataframe.

**Returns** Return the calculates values from a list of aggregations functions.

**any_greater_than**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** –
- **value** –
- **inverse** –
- **tidy** –
- **compute** –

**Returns**

**abstract append**(*dfs: optimus.helpers.types.DataFrameTypeList*) → optimus.helpers.types.DataFrameType
Appends one or more columns or dataframes.

**Parameters** **dfs** – DataFrame, list of dataframes or list of columns to append to the dataframe

**Returns** DataFrame

**apply**(*cols='*'*, *func=None*, *func_return_type=None*, *args=None*, *func_type=None*, *where=None*, *filter_col_by_data_types=None*, *output_cols=None*, *skip_output_cols_processing=False*, *meta_action='apply_cols'*, *mode='vectorized'*, *set_index=False*, *default=None*, ***kwargs*) → optimus.helpers.types.DataFrameType

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **func** –
- **func_return_type** –
- **args** –
- **func_type** –
- **where** –
- **filter_col_by_data_types** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **skip_output_cols_processing** –
- **meta_action** –
- **mode** –
- **set_index** –
- **default** –

- **kwargs** –

**Returns**

**apply_by_data_types**(*cols='*'*, *func=None*, *args=None*, *data_type=None*) → optimus.helpers.types.DataFrameType

Apply a function using pandas udf or udf if apache arrow is not available.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **func** – Functions to be applied to a columns
- **args** –
- **func** – pandas_udf or udf. If 'None' try to use pandas udf (Pyarrow needed)
- **data_type** –

**Returns**

**asin**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Apply the arcsine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcsine of each element.

**asinh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Apply the arcus hyperbolic sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic sin of each element.

**assign**(*cols: Optional[Union[str, list, dict]] = None*, *values=None*, ***kwargs*)

Assign new columns to a Dataframe.

Returns a DataFrame with all original columns in addition to new ones. Existing columns that are re-assigned will be overwritten.

**Parameters**

- **cols** – A dict with the form {"col_name": "value"}, a list of columns or a single column
- **values** – When no dict is passed to 'cols', uses this parameter to get the values.
- **kwargs** –

**Returns**

**abstract static astype**(**args*, ***kwargs*)

Alias from cast function for compatibility with the pandas API.

**Parameters**

- **args** –

- **kwargs** –

**Returns**

**atan**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arctangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arctangent of each element.

**atanh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arcus hyperbolic tangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic tangent of each element.

**bag_of_words**(*features*, *analyzer='word'*, *ngram_range=2*) → optimus.helpers.types.DataFrameType

**Parameters**

- **analyzer** –
- **features** –
- **ngram_range** –

**Returns**

**boxplot**(*cols='*'*) → dict
Return the boxplot data in python dict format.

**Parameters** **cols** – "*", column name or list of column names to be processed.

**Returns** dict with box plot data.

**calculate_pattern_counts**(*cols='*'*, *n=10*, *mode=0*, *flush=False*) →
optimus.helpers.types.DataFrameType
Counts how many equal patterns there are in a column. Uses a cache to trigger the operation only if necessary. Saves the result to meta and returns the same dataframe.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – Return the Top n matches.
- **mode** – mode use to calculate the patterns.
- **flush** – Flushes the cache to process again

**Returns**

**capitalize**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Capitalize every word in a sentence.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**cast**(*cols=None*, *data_type=None*, *output_cols=None*, **args*, ***kwargs*) → optimus.helpers.types.DataFrameType

NOTE: We have two ways to cast the data. Use the use the native .astype() this is faster but can not handle some transformation like string to number in which should output nan.

Cast the elements inside a column or a list of columns to a specific data type. Unlike 'cast' this not change the columns data type

**Parameters**

- **cols** – Columns names to be casted or, dictionary or list of tuples of column names and types to be casted with the following structure: cols = [('columnName1', 'integer'), ('columnName2', 'float'), ('columnName3', 'string')] The first parameter in each tuple is the column name, the second is the final datatype of column after the transformation is made.
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **data_type** – final data type
- **args** – passed to cast function (df.cols.to_integer(..., -1)).
- **kwargs** – passed to cast function (df.cols.to_integer(..., default=-1)).

**Returns** Return the casted columns.

**ceil**(*cols='**', *output_cols=None*) → optimus.helpers.types.DataFrameType

Round each number in a column up to the nearest integer.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the ceil of each element.

**clip**(*cols='**', *lower_bound=None*, *upper_bound=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Assigns values outside boundary to boundary values.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **lower_bound** – Minimum threshold value. All values below this threshold will be set to it. A missing threshold (e.g NA) will not clip the value.
- **upper_bound** – Maximum threshold value. All values above this threshold will be set to it. A missing threshold (e.g NA) will not clip the value.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

concat(*dfs: optimus.helpers.types.DataFrameTypeList*) → optimus.helpers.types.DataFrameType
Same as append.

**Parameters** **dfs** – DataFrame, list of dataframes or list of columns to append to the dataframe

**Returns** DataFrame

copy(*cols='*'*, *output_cols=None*, *columns=None*) → optimus.helpers.types.DataFrameType
Copy one or multiple columns.

**Parameters**

- **cols** – Source column to be copied
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **columns** – tuple of column [('column1','column_copy')('column1','column1_copy')()]

**Returns**

correlation(*cols='*'*, *method='pearson'*, *compute=True*, *tidy=True*)
Compute pairwise correlation of columns, excluding NA/null values.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **method** –

**Method of correlation:** pearson : standard correlation coefficient kendall : Kendall Tau correlation coefficient spearman : Spearman rank correlation

callable: callable with input two 1d ndarrays and returning a float. Note that the returned matrix from corr will have 1 along the diagonals and will be symmetric regardless of the callable's behavior.

**Parameters**

- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return:

cos(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cosine of each element.

cosh(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the hyperbolic cosine of each element.

**count**() → int
Returns the number of columns in the dataframe.

**Returns** Returns the number of columns in the dataframe.

**count_array**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of lists in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_between**(*cols='*'*, *lower_bound=None*, *upper_bound=None*, *equal=True*, *bounds=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements between and lower and upper bound in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **lower_bound** – Lower bound.
- **upper_bound** – Upper bound.
- **equal** –
- **bounds** –
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_boolean**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number booleans in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_containing**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_data_type**(*cols='*'*, *data_type=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of mismatch values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **data_type** –
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_datetime**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** –
- **inverse** –
- **tidy** –
- **compute** –

**Returns**

**count_duplicated**(*cols='*'*, *keep='first'*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** –
- **keep** –
- **inverse** –
- **tidy** –
- **compute** –

**Returns**

**count_email**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like an email address in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.

- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_empty**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of empty values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_ending_with**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of elements that ends with the given string.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_equal**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements equal to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_expression**(*value=None*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.

- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_float**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of floats in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_gender**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like a gender in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_greater_than**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements greater or equal to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_greater_than_equal**(*cols='*'*, *value=None*, *inverse=False*, *compute=True*, *tidy=True*)
Count the number of elements greater than or equal to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.

- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_http_code**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like http code in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_int**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of integers in a column. :param cols: '*', list of columns names or a single column name. :param inverse: Inverse the function selection. :param compute: Compute the result or return a delayed function. :param tidy: The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return:

**count_ip**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like an ip address in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_less_than**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements smaller than to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_less_than_equal**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements smaller than or equal to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_match**(*cols='*'*, *regex=None*, *data_type=None*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of match values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **data_type** –
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_match_pattern**(*cols='*'*, *pattern=None*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** –
- **pattern** –
- **inverse** –
- **tidy** –
- **compute** –

**Returns**

**count_mismatch**(*cols='*'*, *data_type=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of mismatch values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **data_type** –
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_missings**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of missing values in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_nan**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of 'nan' values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_none**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of 'None' values in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_not_equal**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of elements not equal to a value in given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_nulls**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Count the number of 'nulls' values in a given column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.

- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_numeric**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the numeric elements in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_object**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts python object in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_phone_number**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like phone number in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_regex**(*cols='*'*, *regex=None*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of elements that match a regular expression.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **regex** – regular expression.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.

- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_social_security_number**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like social security number in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_starting_with**(*cols='*'*, *value=None*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of elements that start with the given string.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_str**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_uniques**(*cols='*'*, *estimate=False*, *compute=True*, *tidy=True*) → int
Count the number of uniques values in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **estimate** –
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return:

**count_url**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like an url address in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_values_in**(*cols='*'*, *values=None*, *inverse=False*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – Value used to evaluate the function.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**count_zeros**(*cols='*'*, *tidy=True*, *compute=True*)
Return the count of zeros by column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** –
- **compute** –

**Returns**

**count_zip_code**(*cols='*'*, *inverse=False*, *tidy=True*, *compute=True*)
Counts the number of strings that look like a zip code s in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **inverse** – Inverse the function selection.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: The number of elements that match the function.

**cross_tab**(*col_x*, *col_y*, *output='dict'*, *compute=True*) → dict

**Parameters**

- **col_x** –
- **col_y** –
- **output** –
- **compute** – Compute the result or return a delayed function.

**Returns**

**cummax**(*cols='*'*, *output_cols=None*)
Return cumulative maximum over a DataFrame or column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cumulative maximum.

**cummin**(*cols='*'*, *output_cols=None*)
Return cumulative minimum over a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cumulative minimum.

**cumprod**(*cols='*'*, *output_cols=None*)
Return cumulative product over a DataFrame or column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cumulative product.

**cumsum**(*cols='*'*, *output_cols=None*)
Return cumulative sum over a DataFrame or column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cumulative sum.

**cut**(*cols='*'*, *bins=None*, *labels=None*, *default=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Use cut when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable. For example, cut could convert ages to groups of age ranges. Supports binning into an equal number of bins, or a pre-specified array of bins.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **bins** –
- **labels** –
- **default** –
- **output_cols** –

**Returns**

**data_type**(*cols='*'*, *names=False*, *tidy=True*) → dict
Return the column(s) data type as string.

**Parameters**

- **cols** – Columns to be processed
- **names** – Returns aliases for every type instead of its internal name

**Returns** Return a dict of column and its respective data type.

**date_format**(*cols='*'*, *tidy=True*, *compute=True*, *cached=None*, ***kwargs*)
Get the date format from a column, compatible with 'format_date'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.
- **cached** – {None, True, False}, Gets cached date_formats (True), calculates them (False) or a combination of both (None).
- **kwargs** –

**Returns**

**date_formats**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the date format for every value in specified columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**day**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the day from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** – String format
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**days_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return the number of days between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**div**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType

Divide two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**domain**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Returns the domain string from a url. From https://www.hi-optimus.com it returns hi-optimus.com.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**double_metaphone**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

The Double Metaphone phonetic encoding algorithm is the second generation of this algorithm. It is called "Double" because it can return both a primary and a secondary code for a string; this accounts for some ambiguous cases as well as for multiple variants of surnames with common ancestry

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**drop**(*cols=None*, *regex=None*, *data_type=None*) → optimus.helpers.types.DataFrameType

Drop a list of columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **regex** – Regex expression to select the columns
- **data_type** –

**Returns**

**duplicate**(*cols='*'*, *output_cols=None*, *columns=None*) → optimus.helpers.types.DataFrameType
Alias of copy function.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **columns** – tuple of column [('column1','column_copy')('column1','column1_copy')()]

**Returns**

**email_domain**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the domain from an email address. From optimus@mail.col it will return 'mail'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**email_username**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the username from an email address. From optimus@mail.col it will return 'optimus'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**exec_agg**(*exprs*, *compute=True*)
Execute one or multiple aggregations functions.

**Parameters**

- **exprs** –
- **compute** – Compute the result or return a delayed function.

**Returns**

**exp**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return Euler's number, e (~2.718) raised to the power of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the absolute value of each element.

**expand_contracted_words**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Expand contracted words.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**extract**(*cols='*'*, *regex=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Extract a string that match a regular expression.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **regex** – Regular expression
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**fill_na**(*cols='*'*, *value=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Replace null data with a specified value.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **value** – value to replace the nan/None values
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Returns the column filled with given value.

**fingerprint**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Create the fingerprint for a column

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**floor**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Round each number in a column down to the nearest integer.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the floor of each element.

**static format_agg**(*exprs*)

**Parameters** **exprs** –

**Returns**

**format_date**(*cols='*'*, *current_format=None*, *output_format=None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
TODO: missing description

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **current_format** –
- **output_format** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**frequency**(*cols='*'*, *n=32*, *percentage=False*, *total_rows=None*, *count_uniques=False*, *compute=True*, *tidy=False*) → dict

Return the count of every element in the column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – numbers of bins to be returned.
- **percentage** – if True calculate the
- **total_rows** – If True returned the total count.
- **count_uniques** – If True returned the number of uniques elements.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: dict with the count of every element in the column.

**get**(*cols='*'*, *keys=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return items from a dict over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **keys** – The value of the dict key that will be returned.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the value of the key selected.

**groupby**(*by*, *agg*) → optimus.helpers.types.DataFrameType

This helper function aims to help managing columns name in the aggregation output. Also how to handle ordering columns because dask can order columns.

**Parameters**

- **by** – Column name.
- **agg** – List of tuples with the form [("agg", "col")]

**Returns**

**heatmap**(*col_x*, *col_y*, *bins_x=10*, *bins_y=10*, *compute=True*) → dict

**Parameters**

- **col_x** –

- **col_y** –
- **bins_x** –
- **bins_y** –
- **compute** –

**Returns**

**hist**(*cols='*'*, *buckets=32*, *compute=True*) → dict
Return the histogram representation of the distribution of the data.

**Parameters** **cols** – "*", column name or list of column names to be processed.

:param buckets:Number of histogram bins to be used. :param compute: :return:

**host**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the host string from a url.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**hour**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the hour from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** – String format
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**hours_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Return the number of hours between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**impute**(*cols='*'*, *data_type='auto'*, *strategy='auto'*, *fill_value=None*, *output_cols=None*)
Fill null values using a constant or any of the strategy available.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **data_type** –
  - If "auto", detect if it's continuous or categorical using the data type of the column.
  - If "continuous", sets the data as continuous and if no 'strategy' is passed then the mean is used.
  - If "categorical", sets the data as categorical and if no 'strategy' is passed then the most frequent value is used.
- **strategy** –
  - If "auto", automatically selects a strategy depending on the data type passed or inferred on 'data_type'.
  - If "mean", then replace missing values using the mean along each column. Can only be used with numeric data.
  - If "median", then replace missing values using the median along each column. Can only be used with numeric data.
  - If "most_frequent", then replace missing using the most frequent value along each column. Can be used with strings or numeric data.
  - If "constant", then replace missing values with fill_value. Can be used with strings or numeric data.
- **fill_value** – constant to be used to fill null values
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Return the Column filled with the imputed values.

**abstract static index_to_string**(*cols=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Maps a column of label indices back to a column containing the original labels as strings.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**infer_data_types**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** –

**Returns**

**infer_date_formats**(*cols='*'*, *sample=200*, *tidy=True*) → dict

Infer date formats in a dataframe from a sample. This function use Pandas no matter the engine you are using.

**Parameters** **cols** – Columns in which you want to infer the datatype.

**Returns** dict with the column and the inferred date format

**infer_type**(*cols='*'*, *sample=200*, *tidy=True*) → dict
Infer data types in a dataframe from a sample. First it identify the data type of every value in every cell. After that it takes all ghe values apply som heuristic to try to better identify the datatype. This function use Pandas no matter the engine you are using.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **sample** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name and the value.

**Returns** dict with the column and the inferred data type.

**inferred_data_type**(*cols='*'*, *use_internal=False*, *tidy=True*)
Get the inferred data types from the meta data.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **use_internal** – If no inferred data type is found, return a translated internal data type instead of None.
- **tidy** – The result format. If 'True' it will return a value if you 'False' will return the column name a value.

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: Python Dictionary with column names and its data types.

**iqr**(*cols='*'*, *more=None*, *relative_error=10000*, *estimate=True*)
Return the column Inter Quartile Range value.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **more** – Return info about q1 and q3
- **relative_error** –

**Returns** Return the column Inter Quartile Range value.

**item**(*cols='*'*, *n=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return items from a list over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – The position of the element that will be returned.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the value of the item selected.

**join**(*df_right: optimus.helpers.types.DataFrameType*, *how='left'*, *on=None*, *left_on=None*, *right_on=None*, *key_middle=False*) → optimus.helpers.types.DataFrameType
Join two dataframes using a column.

**Parameters**

- **df_right** – The dataframe that will be used to join the actual dataframe.

- **how** – {'left', 'right', 'outer', 'inner'}, default 'left'
- **on** – The column that will be used to join the two dataframes.
- **left_on** – The column in the actual dataframe that will be used to make to make the join.
- **right_on** – The column in the given dataframe that will be used to make to make the join.
- **key_middle** – Order the columns putting the left df columns before the key column and the right df columns

**Returns** Dataframe

**keep**(*cols=None*, *regex=None*) → optimus.helpers.types.DataFrameType
Drop a list of columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **regex** – Regex expression to select the columns

**Returns**

**kurtosis**(*cols='*'*, *tidy=True*, *compute=True*)
Returns the kurtosis of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Returns the kurtosis of the values over the requested columns.

**left**(*cols='*'*, *n=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the substring from the first character to the nth from right to left.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – Number of character to get starting from 0.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**lemmatize_verbs**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Finding the lemma of a word depending on its meaning and context.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**len**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the length of every string in a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return:

**levenshtein**(*cols='*'*, *other_cols=None*, *value=None*, *output_cols=None*)
Calculate the levenshtein distance to a specified column. The Levenshtein distance is a string metric for measuring the difference between two sequences.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **other_cols** –
- **value** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**ln**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the natural logarithm of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the natural logarithm of each element.

**log**(*cols='*'*, *base=10*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the logarithm base 10 of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **base** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the logarithm base 10 of each element.

**lower**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Lowercase the specified columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**mad**(*cols='*'*, *relative_error=10000*, *more=False*, *estimate=True*, *tidy=True*, *compute=True*)

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **relative_error** –
- **more** –
- **estimate** –

- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :param compute: Compute the result or return a delayed function.

**match_rating_codex**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
The match rating approach (MRA) is a phonetic algorithm developed by Western Airlines in 1977 for the indexation and comparison of homophonous names.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**max**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the maximum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**max_abs_scaler**(*cols='*'*, *output_cols=None*)
Scale each feature by its maximum absolute value.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**mean**(*cols='*'*, *tidy=True*, *compute=True*)
Return the mean of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the cumulative sum.

**median**(*cols='*'*, *relative_error=10000*, *tidy=True*, *compute=True*)
Returns the median of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **relative_error** –

- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** –

**Returns** Returns the median of the values over the requested columns

**metaphone**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the Metaphone algorithm to a specified column. Metaphone is a phonetic algorithm, published by Lawrence Philips in 1990, for indexing words by their English pronunciation.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**mid**(*cols='*'*, *start=0*, *n=1*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the substring from

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **start** –
- **n** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**min**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the minimum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and

value if not. If False it will return the functions name, the column name. and the value. :param compute: C :return:

**min_max_scaler**(*cols='*'*, *output_cols=None*)
Transform features by scaling each feature to a given range.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**minute**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the minutes from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** – String format
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**minutes_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return the number of minutes between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**mod**(*cols='*'*, *divisor=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return the Modulo of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **divisor** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing Molulo of each element.

**mode**(*cols='*'*, *tidy: bool = True*, *compute: bool = True*)

Return the mode value over.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**modified_z_score**(*cols='*'*, *estimate=True*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Returns the modified z-score of the given columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **estimate** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Returns the modified z-score of the given columns.

**month**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the month from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** – String format
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**months_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the number of months between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**move**(*column*, *position*, *ref_col=None*) → optimus.helpers.types.DataFrameType
Move a column to a specific position.

**Parameters**

- **column** – Column(s) to be moved
- **position** – Column new position. Accepts 'after', 'before', 'beginning', 'end' or a numeric value, relative to 'ref_col'.
- **ref_col** – Column taken as reference

**Returns** DataFrame

**mul**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType
Multiply two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**names**(*cols='*'*, *data_types=None*, *invert=False*, *is_regex=None*) → list
Return the names of the columns.

**Parameters**

- **cols** – Regex, "*" or columns to get.
- **data_types** – returns only columns with matching data types

- **invert** – invert column selection
- **is_regex** – if True, forces cols regex as a regex

**Returns**

**abstract static nest**(*cols*, *separator=''*, *output_col=None*, *drop=True*, *shape='string'*) → optimus.helpers.types.DataFrameType

Concatenate two or more columns into one.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **separator** –
- **output_col** – Column name or list of column names where the transformed data will be saved.
- **drop** –
- **shape** –

**Returns** Columns with all the specified columns concatenated.

**ngram_fingerprint**(*cols='*'*, *n_size=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Calculate the ngram for a fingerprinted string.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n_size** – The ngram size.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**ngrams**(*cols='*'*, *n_size=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Calculate the ngram for a fingerprinted string.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **n_size** – The ngram size.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**normalize_chars**(*cols='*'*, *output_cols=None*)

Remove diacritics from a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**normalize_spaces**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Remove extra whitespace between words and trim whitespace from the beginning and the end of each string.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**num_to_words**(*cols='*'*, *language='en'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Convert numbers to its string representation.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **language** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column with number converted to its string representation.

**nysiis**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the NYSIIS algorithm to a specified column. NYSIIS (New York State Identification and Intelligence System).

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**one_hot_encode**(*cols='*'*, *prefix=None*, *drop=True*, ***kwargs*) → optimus.helpers.types.DataFrameType
Maps a categorical column to multiple binary columns, with at most a single one-value. :param cols: Columns to be encoded. :param prefix: Prefix of the columns where the output is going to be saved. :param drop: :return: Dataframe with encoded columns.

**pad**(*cols='*'*, *width=0*, *fill_char='0'*, *side='left'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Fill a string to match the given string length.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **width** – Total length of the string.
- **fill_char** – The char that will be used to fill the string.
- **side** – Fill the left or the right side.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**parse_inferred_types**(*col_data_type*)
Parse a engine column specific data type to a profiler data type.

**Parameters** **col_data_type** – Engine column specific data.

**Returns** Dict

**pattern**(*cols='*'*, *output_cols=None*, *mode=0*) → optimus.helpers.types.DataFrameType

**Replace alphanumeric and punctuation chars for canned chars. We aim to help to find string patterns**
c = Any alpha char in lower or upper case l = Any alpha char in lower case U = Any alpha char in upper case * = Any alphanumeric in lower or upper case. Used only in type 2 nd 3 # = Any numeric ! = Any punctuation

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **mode** – 0: Identify lower, upper, digits. Except spaces and special chars. 1: Identify chars, digits. Except spaces and special chars 2: Identify Any alphanumeric. Except spaces and special chars 3: Identify alphanumeric and special chars. Except white spaces

**pattern_counts**(*cols='*'*, *n=10*, *mode=0*, *flush=False*) → dict
Get how many equal patterns there are in a column. Triggers the operation only if necessary.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – Top n matches
- **mode** –
- **flush** – Flushes the cache to process again

**Returns**

**percentile**(*cols='*'*, *values=None*, *relative_error=10000*, *estimate=True*, *tidy=True*, *compute=True*)
Return values at the given percentile over requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **values** – Percentiles values you want to calculate. 0.25,0.5,0.75
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Return values at the given percentile over requested column.

**port**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the port string from a url.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**pos**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word .

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**pow**(*cols='*'*, *power=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the power of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **power** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the power of each element.

**profile**(*cols='*'*, *bins: int = 32*, *flush: bool = False*) → dict
Returns the profile of selected columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **bins** – Number of buckets.
- **flush** – Flushes the cache of the whole profile to process it again.

**Returns** Returns the profile of selected columns.

**qcut**(*cols='*'*, *quantiles=None*, *output_cols=None*)

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **quantiles** –
- **output_cols** –

**Returns**

**quality**(*cols='*'*, *flush=False*, *compute=True*) → dict
Return the data quality in the format {'col_name': {'mismatch': 0, 'missing': 9, 'match': 0, 'inferred_data_type': 'object'}}

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **flush** –
- **compute** –

**Returns** dict in the format {'col_name': {'mismatch': 0, 'missing': 9, 'match': 0, 'inferred_data_type': 'object'}}

**range**(*cols='*'*, *tidy: bool = True*, *compute: bool = True*)
Return the minimum and maximum of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**rdiv**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType
Divide two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**reciprocal**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the reciprocal(1/x) of of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the reciprocal of each element.

**remove**(*cols='*'*, *search=None*, *search_by='chars'*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Remove values from a string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **search** –
- **search_by** – Search by 'chars',
- **output_cols** – Column name or list of column names where the transformed data will be saved.:param search:

**Returns**

**remove_numbers**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove numbers from a string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_special_chars**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove special chars from a string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_stopwords**(*cols='*'*, *language='english'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Remove extra whitespace between words and trim whitespace from the beginning and the end of each string.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **language** – specify the stopwords language
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_urls**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Remove urls from the one or more columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_white_spaces**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Remove all white spaces from string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**rename**(*cols: Union[str, list, dict] = '*'*, *names: Optional[Union[str, list]] = None*, *func=None*) → optimus.helpers.types.DataFrameType

Changes the name of a column(s) dataFrame.

**Parameters**

- **cols** – string, dictionary or list of strings or tuples. Each tuple may have following form: (oldColumnName, newColumnName).
- **names** – string or list of strings with new names of columns. Ignored if a dictionary or list of tuples is passed to cols.
- **func** – can be lower, upper or any string transformation function.

**Returns** Dataframe with columns names replaced.

**replace**(*cols='*'*, *search=None*, *replace_by=None*, *search_by=None*, *ignore_case=False*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Replace a value, list of values by a specified string.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **search** – Values to look at to be replaced
- **replace_by** – New value to replace the old one. Supports an array when searching by characters.
- **search_by** – Can be "full","words","chars" or "values".
- **ignore_case** – Ignore case when searching for match
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** DataFrame

**replace_regex**(*cols='*'*, *search=None*, *replace_by=None*, *search_by=None*, *ignore_case=False*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Replace a value, list of values by a specified regex.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **search** – Values to look at to be replaced
- **replace_by** – New value to replace the old one. Supports an array when searching by characters.
- **search_by** – Can be "full","words","chars" or "values".
- **ignore_case** – Ignore case when searching for match
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**abstract static reverse**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Reverse the order of the characters strings in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**right**(*cols='*'*, *n=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Get the substring from the last character to n.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**round**(*cols='*'*, *decimals=0*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Round a DataFrame to a variable number of decimal places.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **decimals** – The number of decimals you want to
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the round of each element.

**schema_data_type**(*cols='*'*, *tidy=True*)
Return the column(s) data type as Type.

**Parameters**

- **cols** – Columns to be processed
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.

**Returns**

**second**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the seconds from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**seconds_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the number of seconds between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**select**(*cols='*'*, *regex=None*, *data_type=None*, *invert=False*, *accepts_missing_cols=False*) → optimus.helpers.types.DataFrameType
Select columns using index, column name, regex to data type.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **regex** – Regular expression to filter the columns
- **data_type** – Data type to be filtered for

- **invert** – Invert the selection
- **accepts_missing_cols** –

**Returns**

**set**(*cols='*'*, *value_func=None*, *where: Optional[Union[str, optimus.helpers.types.MaskDataFrameType]] = None*, *args=None*, *default=None*, *eval_value: bool = False*) → optimus.helpers.types.DataFrameType
Set a column value using a number, string or an expression.

**Parameters**

- **cols** – Columns to set or create.
- **value_func** – expression, function or value.
- **where** – When the condition in 'where' is True, replace with 'value_func'. Where False, replace with 'default' or keep the original value.
- **args** – Argument when 'value_func' param is a function.
- **default** – Entries where 'where' is False are replaced with corresponding value from other.
- **eval_value** – Parse 'value_func' param in case a string is passed.

**Returns**

**set_data_type**(*cols: Union[str, list, dict] = '*'*, *data_types: Optional[Union[str, list]] = None*, *inferred: bool = False*) → optimus.helpers.types.DataFrameType
Set profiler data type.

**Parameters**

- **cols** – A dict with the form {"col_name": profiler datatype}, a list of columns or a single column.
- **data_types** – If a string or a list passed to cols, uses this parameter to set the data types to those columns.
- **inferred** – Whether it was inferred or not.

**Returns** Dataframe with new data types in the meta data.

**set_date_format**(*cols: Union[str, list, dict] = '*'*, *date_formats: Optional[Union[str, list]] = None*, *inferred: bool = False*) → optimus.helpers.types.DataFrameType
Set date format.

**Parameters**

- **cols** – A dict with the form {"col_name": "date format"}, a list of columns or a single column
- **date_formats** – If a string or a list passed to cols, uses this parameter to set the date format to those columns.
- **inferred** – Whether it was inferred or not.

**Returns**

**sin**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the sine of each element.

**sinh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arctangent of each element.

**skew**(*cols='*'*, *tidy=True*, *compute=True*)
Return the skew of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Return the skew of the values over the requested columns.

**slice**(*cols='*'*, *start=None*, *stop=None*, *step=None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Slice substrings from each element in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **start** – Start position for slice operation.
- **stop** – Stop position for slice operation.
- **step** – Step size for slice operation.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**sort**(*order: Union[str, list] = 'asc'*, *cols=None*) → optimus.helpers.types.DataFrameType
Sort one or multiple columns in asc or desc order.

**Parameters**

- **order** – 'asc' or 'desc' accepted
- **cols** –

**Returns** Column containing the cumulative sum.

**soundex**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the Soundex algorithm to a specified column. Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English. The goal is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**sqrt**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the square root of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the square root of each element.

**standard_scaler**(*cols='*'*, *output_cols=None*)
Standardize features by removing the mean and scaling to unit variance.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**std**(*cols='*'*, *tidy=True*, *compute=True*)
Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**stem_verbs**(*cols='*'*, *stemmer: str = 'porter'*, *language: str = 'english'*, *output_cols=None*) →
optimus.helpers.types.DataFrameType

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **stemmer** – snowball, porter, lancaster
- **language** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**abstract static string_to_index**(*cols=None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Encodes a string column of labels to a column of label indices.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**strip_html**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove HTML tags.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**sub**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType
Subtract two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**sub_domain**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the subdomain string from a url. From https://www.hi-optimus.com it returns 'www'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**sum**(*cols='*'*, *tidy=True*, *compute=True*)
Return the sum of the values over the requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the sum of multiple columns.

**tan**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the tangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the tangent of each element.

**tanh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic tangent function to a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return: Column containing the hyperbolic tangent of each element.

**tf_idf**(*features*) → optimus.helpers.types.DataFrameType

**Parameters** **features** –

**Returns**

**time_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*, *func=None*) → optimus.helpers.types.DataFrameType
Returns a TimeDelta of the units between two datetimes.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **func** – Custom function to pass to the apply, like self.F.days_between

**Returns**

**title**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Capitalize the first word in a sentence.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**to_boolean**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Cast the elements inside a column or a list of columns to boolean. :param cols: "*", column name or list of column names to be processed. :param output_cols: :return:

**to_datetime**(*cols='*'*, *format=None*, *output_cols=None*, *transform_format=True*) → optimus.helpers.types.DataFrameType
TODO:?

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.
- **transform_format** –

**Returns**

**to_float**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Cast the elements inside a column or a list of columns to float. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return:

**to_integer**(*cols='*'*, *default=0*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Cast the elements inside a column or a list of columns to integer. :param cols: "*", column name or list of column names to be processed. :param default: :param output_cols: Column name or list of column names where the transformed data will be saved. :return:

**to_string**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Cast the elements inside a column or a list of columns to string. :param cols: "*", column name or list of column names to be processed. :param output_cols: :return:

**abstract static to_timestamp**(*cols*, *date_format=None*, *output_cols=None*)

**Parameters**

- **cols** –
- **date_format** –
- **output_cols** –

**Returns**

**top_domain**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the top domain string from a url. From 'https://www.hi-optimus.com' it returns 'hi-optimus.com'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**trim**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove leading and trailing characters.

Strip whitespaces (including newlines) or a set of specified characters from each string in the column from left and right sides. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return:

**unique_values**(*cols='*'*, *estimate=False*, *compute=True*, *tidy=True*) → list
Return a list of uniques values in a column.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **estimate** –
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value.

**unnest**(*cols='*'*, *separator=None*, *splits=2*, *index=None*, *output_cols=None*, *drop=False*, *mode='string'*) → optimus.helpers.types.DataFrameType
Split the columns values (array or string) in different columns.

**Parameters**

- **cols** – Columns to be un-nested
- **output_cols** – Resulted on or multiple columns after the unnest operation [(output_col_1_1,output_col_1_2),

(output_col_2_1, output_col_2] :param separator: char or regex :param splits: Number of columns splits. :param index: Return a specific index per columns. [1,2] :param drop: :param mode:

**unset_data_type**(*cols='*'*)
Unset user set data type.

**Parameters** **cols** – '*', list of columns names or a single column name.

**Returns**

**unset_date_format**(*cols='*'*)
Unset user defined date format.

**Parameters** **cols** – '*', list of columns names or a single column name.

**Returns**

**upper**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Uppercase the specified columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**url_file**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the file string from a url. From https://www.hi-optimus.com/index.html it returns 'index.html'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**url_fragment**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**url_path**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the top domain string from a url. From https://www.hi-optimus.com it returns 'hi-optimus.com'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**url_query**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the query string from a url. From https://www.hi-optimus.com/?rollout=true it returns 'roolout=true'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**url_scheme**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the top domain string from a url. From 'https://www.hi-optimus.com' it returns 'https'.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**var**(*cols='*'*, *tidy=True*, *compute=True*)
Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**weekday**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Get the hour from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**word_count**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Count the number of words in a paragraph.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**word_tokenize**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**year**(*cols='*'*, *format: Optional[str] = None*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Get the Year from a date in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **format** – String format
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**years_between**(*cols='*'*, *value=None*, *date_format=None*, *round=None*, *output_cols=None*) →
optimus.helpers.types.DataFrameType
Return the number of years between two dates.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **value** –
- **date_format** –
- **round** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**z_score**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Returns the z-score of the given columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Dataframe with the z-score of the given columns appended.

CHAPTER

# FOUR

# MATH

class optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)

Base class for all Cols implementations

**sum**(*cols='*'*, *tidy=True*, *compute=True*)

Return the sum of the values over the requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the sum of multiple columns.

**sub**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType

Subtract two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**mul**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType

Multiply two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**div**(*cols='*'*, *output_col=None*) → optimus.helpers.types.DataFrameType

Divide two or more columns.

**Parameters**

- **cols** – '*', list of columns names or a single column name
- **output_col** – Single output column in case no value is passed

**Returns** Dataframe with the result of the arithmetic operation appended.

**abs**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Return the absolute numeric value of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the absolute value of each element.

**exp**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return Euler's number, e (~2.718) raised to the power of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the absolute value of each element.

**mod**(*cols='*'*, *divisor=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the Modulo of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **divisor** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing Molulo of each element.

**log**(*cols='*'*, *base=10*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the logarithm base 10 of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **base** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the logarithm base 10 of each element.

**ln**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the natural logarithm of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the natural logarithm of each element.

**pow**(*cols='*'*, *power=2*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the power of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **power** –
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the power of each element.

**sqrt**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the square root of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the square root of each element.

**reciprocal**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the reciprocal(1/x) of of each value in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the reciprocal of each element.

**round**(*cols='*'*, *decimals=0*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Round a DataFrame to a variable number of decimal places.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **decimals** – The number of decimals you want to
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the round of each element.

**floor**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Round each number in a column down to the nearest integer.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the floor of each element.

**ceil**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Round each number in a column up to the nearest integer.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the ceil of each element.

CHAPTER

# FIVE

# TRIGONOMETRIC

**class** optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)
Base class for all Cols implementations

**sin**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the sine of each element.

**cos**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the cosine of each element.

**tan**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the tangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the tangent of each element.

**asin**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arcsine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcsine of each element.

**acos**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply arccosine function to a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return: Column containing the arccosine of each element.

**atan**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arctangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arctangent of each element.

**sinh**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arctangent of each element.

**cosh**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the hyperbolic cosine of each element.

**tanh**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the hyperbolic tangent function to a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return: Column containing the hyperbolic tangent of each element.

**asinh**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arcus hyperbolic sine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic sin of each element.

**acosh**(*cols='*', output_cols=None*) → optimus.helpers.types.DataFrameType
Apply the arcus hyperbolic cosine function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.

- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic cosine of each element.

**atanh**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType

Apply the arcus hyperbolic tangent function to a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** Column containing the arcus hyperbolic tangent of each element.

CHAPTER

# SIX

# STATISTICAL

class optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)

Base class for all Cols implementations

**median**(*cols='*'*, *relative_error=10000*, *tidy=True*, *compute=True*)

Returns the median of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** –

**Returns** Returns the median of the values over the requested columns

**mean**(*cols='*'*, *tidy=True*, *compute=True*)

Return the mean of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the cumulative sum.

**var**(*cols='*'*, *tidy=True*, *compute=True*)

Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**std**(*cols='*'*, *tidy=True*, *compute=True*)

Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**min**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the minimum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and

value if not. If False it will return the functions name, the column name. and the value. :param compute: C :return:

**max**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the maximum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**percentile**(*cols='*'*, *values=None*, *relative_error=10000*, *estimate=True*, *tidy=True*, *compute=True*)
Return values at the given percentile over requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **values** – Percentiles values you want to calculate. 0.25,0.5,0.75
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Return values at the given percentile over requested column.

**iqr**(*cols='*'*, *more=None*, *relative_error=10000*, *estimate=True*)
Return the column Inter Quartile Range value.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **more** – Return info about q1 and q3
- **relative_error** –

**Returns** Return the column Inter Quartile Range value.

**hist**(*cols='*'*, *buckets=32*, *compute=True*) → dict
Return the histogram representation of the distribution of the data.

**Parameters** **cols** – "*", column name or list of column names to be processed.

:param buckets:Number of histogram bins to be used. :param compute: :return:

**frequency**(*cols='*'*, *n=32*, *percentage=False*, *total_rows=None*, *count_uniques=False*, *compute=True*, *tidy=False*) → dict
Return the count of every element in the column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – numbers of bins to be returned.
- **percentage** – if True calculate the
- **total_rows** – If True returned the total count.
- **count_uniques** – If True returned the number of uniques elements.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: dict with the count of every element in the column.

**count**() → int
Returns the number of columns in the dataframe.

**Returns** Returns the number of columns in the dataframe.

CHAPTER

# SEVEN

# ROWS

class optimus.engines.base.rows.**BaseRows**(*root: DataFrameType*)

Base class for all Rows implementations

**append**(*dfs: DataFrameTypeList*, *names_map=None*) → DataFrameType

Appends 2 or more dataframes :param dfs: :param names_map:

**apply**(*func*, *args=None*, *output_cols=None*, *mode='vectorized'*) → DataFrameType

This will aimed to handle vectorized and not vectorized operations :param func: :param args: :param output_cols: :param mode: :return:

**approx_count**() → DataFrameType

Aprox count :return:

**array**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**between**(*cols='*'*, *lower_bound=None*, *upper_bound=None*, *equal=True*, *bounds=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **lower_bound** –
- **upper_bound** –
- **equal** –
- **bounds** –
- **drop** –
- **how** –

**Returns**

**between_index**(*lower_bound=None*, *upper_bound=None*, *cols='*'*)

**Parameters**

- **columns** –
- **lower_bound** –
- **upper_bound** –

**Returns**

**boolean**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**contains**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

**count**(*compute=True*) → *int*
Count dataframe rows

**credit_card_number**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**datetime**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**drop**(*where*) → DataFrameType
Drop rows depending on a mask or an expression :param where: Mask, expression or name of the column to be taken as mask :return: Optimus Dataframe

**drop_arrays**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_between**(*cols='*'*, *lower_bound=None*, *upper_bound=None*, *equal=True*, *bounds=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **lower_bound** –
- **upper_bound** –
- **equal** –
- **bounds** –
- **how** –

**Returns**

**drop_booleans**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_by_data_type**(*cols='*'*, *data_type=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **data_type** –
- **how** –

**Returns**

**drop_by_expression**(*where=None*, *cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **where** –
- **cols** –
- **how** –

**Returns**

**drop_by_regex**(*cols='*'*, *regex=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **regex** –
- **how** –

**Returns**

**drop_contains**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_credit_card_numbers**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_datetimes**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_duplicated**(*cols='*'*, *keep='first'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **keep** –
- **how** –

**Returns**

**drop_emails**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –

- **how** –

**Returns**

**drop_empty**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_ends_with**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_equal**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_find**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_float**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_genders**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_greater_than**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_greater_than_equal**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_http_codes**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_int**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_ips**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_less_than**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_less_than_equal**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_match**(*cols='*'*, *regex=None*, *data_type=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **regex** –
- **data_type** –
- **how** –

**Returns**

**drop_mismatch**(*cols='*'*, *data_type=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **data_type** –
- **how** –

**Returns**

**drop_missings**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_nan**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

drop_none(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

drop_not_equal(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

drop_nulls(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

drop_numeric(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

drop_objects(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

drop_pattern(*cols='*'*, *pattern=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **pattern** –
- **how** –

**Returns**

**drop_phone_numbers**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_social_security_numbers**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_starts_with**(*cols='*'*, *value=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **how** –

**Returns**

**drop_str**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_uniques**(*cols='*'*, *keep='first'*, *how='any'*) → DataFrameType
Drops first (passed to keep) matches of duplicates and unique values. :param cols: :param keep: :param how: :return: Dataframe

**drop_urls**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**drop_value_in**(*cols='*'*, *values=None*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **values** –

- **how** –

**Returns**

**drop_zip_codes**(*cols='*'*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **how** –

**Returns**

**duplicated**(*cols='*'*, *keep='first'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **keep** –
- **drop** –
- **how** –

**Returns**

**email**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**empty**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**ends_with**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

**equal**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

**expression**(*where=None*, *cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **where** –
- **cols** –
- **drop** –
- **how** –

**Returns**

**find**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

**float**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**gender**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

greater_than(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

greater_than_equal(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

http_code(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

int(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

ip(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

less_than(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

less_than_equal(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

limit(*count=10*) → DataFrameType
Limit the number of rows :param count: :return:

match(*cols='*'*, *regex=None*, *data_type=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **regex** –
- **data_type** –
- **drop** –
- **how** –

**Returns**

match_data_type(*cols='*'*, *data_type=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **data_type** –
- **drop** –
- **how** –

**Returns**

match_regex(*cols='*'*, *regex=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **regex** –
- **drop** –
- **how** –

**Returns**

**mismatch**(*cols='*'*, *data_type=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **data_type** –
- **drop** –
- **how** –

**Returns**

**missing**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**nan**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**none**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**not_equal**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –

- **value** –
- **drop** –
- **how** –

**Returns**

**null**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**numeric**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**object**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**pattern**(*cols='*'*, *pattern=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **pattern** –
- **drop** –
- **how** –

**Returns**

**phone_number**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –

- **how** –

**Returns**

**reverse**() → DataFrameType

**Returns**

**select**(*expr=None*, *contains=None*, *case=None*, *flags=0*, *na=False*, *regex=False*) → DataFrameType
Return selected rows using an expression :param expr: Expression used, For Ex: (df["A"] > 3) & (df["A"] <= 1000) or Column name "A" :param contains: List of string :param case: :param flags: :param na: :param regex: :return:

**social_security_number**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **drop** –
- **how** –

**Returns**

**sort**(*cols='*'*, *order='desc'*, *cast=True*) → DataFrameType
Sort rows taking into account multiple columns :param cols: :param order: :param cast: cast rows before sorting them.

**starts_with**(*cols='*'*, *value=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **value** –
- **drop** –
- **how** –

**Returns**

**str**(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType
#TODO:? :param cols: :param drop: :param how: :return:

**to_list**(*input_cols*) → list

**Parameters** **input_cols** –

**Returns**

**unique**(*cols='*'*, *keep='first'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- **cols** –
- **keep** –
- **drop** –
- **how** –

**Returns**

`static unnest`(*cols*) → DataFrameType

**Parameters** `cols` –

**Returns**

`url`(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- `cols` –
- `drop` –
- `how` –

**Returns**

`value_in`(*cols='*'*, *values=None*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- `cols` –
- `values` –
- `drop` –
- `how` –

**Returns**

`zip_code`(*cols='*'*, *drop=False*, *how='any'*) → DataFrameType

**Parameters**

- `cols` –
- `drop` –
- `how` –

**Returns**

CHAPTER

# EIGHT

# DATES

class optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)

Base class for all Cols implementations

**median**(*cols='*'*, *relative_error=10000*, *tidy=True*, *compute=True*)

Returns the median of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** –

**Returns** Returns the median of the values over the requested columns

**mean**(*cols='*'*, *tidy=True*, *compute=True*)

Return the mean of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the cumulative sum.

**var**(*cols='*'*, *tidy=True*, *compute=True*)

Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**std**(*cols='*'*, *tidy=True*, *compute=True*)

Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**min**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the minimum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and

value if not. If False it will return the functions name, the column name. and the value. :param compute: C :return:

**max**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the maximum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**percentile**(*cols='*'*, *values=None*, *relative_error=10000*, *estimate=True*, *tidy=True*, *compute=True*)
Return values at the given percentile over requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **values** – Percentiles values you want to calculate. 0.25,0.5,0.75
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Return values at the given percentile over requested column.

**iqr**(*cols='*'*, *more=None*, *relative_error=10000*, *estimate=True*)
Return the column Inter Quartile Range value.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **more** – Return info about q1 and q3
- **relative_error** –

**Returns** Return the column Inter Quartile Range value.

**hist**(*cols='*'*, *buckets=32*, *compute=True*) → dict
Return the histogram representation of the distribution of the data.

**Parameters** **cols** – "*", column name or list of column names to be processed.

:param buckets:Number of histogram bins to be used. :param compute: :return:

**frequency**(*cols='*'*, *n=32*, *percentage=False*, *total_rows=None*, *count_uniques=False*, *compute=True*, *tidy=False*) → dict
Return the count of every element in the column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – numbers of bins to be returned.
- **percentage** – if True calculate the
- **total_rows** – If True returned the total count.
- **count_uniques** – If True returned the number of uniques elements.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: dict with the count of every element in the column.

**count**() → int
Returns the number of columns in the dataframe.

**Returns** Returns the number of columns in the dataframe.

CHAPTER

# NINE

# DATA QUALITY

class optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)
Base class for all Cols implementations

**median**(*cols='*'*, *relative_error=10000*, *tidy=True*, *compute=True*)
Returns the median of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** –

**Returns** Returns the median of the values over the requested columns

**mean**(*cols='*'*, *tidy=True*, *compute=True*)
Return the mean of the values over the requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Column containing the cumulative sum.

**var**(*cols='*'*, *tidy=True*, *compute=True*)
Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**std**(*cols='*'*, *tidy=True*, *compute=True*)
Return unbiased variance over requested columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **tidy** – The result format. If tidy it will return a value if you process a column or column name and value if not.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**min**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the minimum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and

value if not. If False it will return the functions name, the column name. and the value. :param compute: C :return:

**max**(*cols='*'*, *numeric=None*, *tidy: bool = True*, *compute: bool = True*)
Return the maximum value over one or one each column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **numeric** – if True, cast to numeric before processing.
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns**

**percentile**(*cols='*'*, *values=None*, *relative_error=10000*, *estimate=True*, *tidy=True*, *compute=True*)
Return values at the given percentile over requested column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **values** – Percentiles values you want to calculate. 0.25,0.5,0.75
- **relative_error** –
- **tidy** – The result format. If True it will return a value if you process a column or column name and value if not. If False it will return the functions name, the column name.
- **compute** – Compute the final result. False imply to return a delayed object.

**Returns** Return values at the given percentile over requested column.

**iqr**(*cols='*'*, *more=None*, *relative_error=10000*, *estimate=True*)
Return the column Inter Quartile Range value.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **more** – Return info about q1 and q3
- **relative_error** –

**Returns** Return the column Inter Quartile Range value.

**hist**(*cols='*'*, *buckets=32*, *compute=True*) → dict
Return the histogram representation of the distribution of the data.

**Parameters** **cols** – "*", column name or list of column names to be processed.

:param buckets:Number of histogram bins to be used. :param compute: :return:

**frequency**(*cols='*'*, *n=32*, *percentage=False*, *total_rows=None*, *count_uniques=False*, *compute=True*, *tidy=False*) → dict
Return the count of every element in the column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **n** – numbers of bins to be returned.
- **percentage** – if True calculate the
- **total_rows** – If True returned the total count.
- **count_uniques** – If True returned the number of uniques elements.
- **compute** – Compute the result or return a delayed function.
- **tidy** – The result format. If True it will return a value if you

process a column or column name and value if not. If False it will return the functions name, the column name and the value. :return: dict with the count of every element in the column.

**count**() → int
Returns the number of columns in the dataframe.

**Returns** Returns the number of columns in the dataframe.

CHAPTER

# TEN

# STRINGS

**class** optimus.engines.base.columns.**BaseColumns**(*root: optimus.helpers.types.DataFrameType*)
Base class for all Cols implementations

**lower**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Lowercase the specified columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**upper**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Uppercase the specified columns.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**title**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Capitalize the first word in a sentence.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** BaseDataFrame

**capitalize**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Capitalize every word in a sentence.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_special_chars**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove special chars from a string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**replace**(*cols='*'*, *search=None*, *replace_by=None*, *search_by=None*, *ignore_case=False*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Replace a value, list of values by a specified string.

**Parameters**

- **cols** – '*', list of columns names or a single column name.
- **search** – Values to look at to be replaced
- **replace_by** – New value to replace the old one. Supports an array when searching by characters.
- **search_by** – Can be "full","words","chars" or "values".
- **ignore_case** – Ignore case when searching for match
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns** DataFrame

**normalize_spaces**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove extra whitespace between words and trim whitespace from the beginning and the end of each string.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**len**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Return the length of every string in a column. :param cols: "*", column name or list of column names to be processed. :param output_cols: Column name or list of column names where the transformed data will be saved. :return:

**abstract static reverse**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Reverse the order of the characters strings in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

**remove_white_spaces**(*cols='*'*, *output_cols=None*) → optimus.helpers.types.DataFrameType
Remove all white spaces from string in a column.

**Parameters**

- **cols** – "*", column name or list of column names to be processed.
- **output_cols** – Column name or list of column names where the transformed data will be saved.

**Returns**

CHAPTER

ELEVEN

# INDICES AND TABLES

- genindex
- modindex
- search

# INDEX

## A



## B



## C

## D

## E



## F

## N



## O



## P

## Q



## R



## S

## T



## U



## V



## W



## X

## Y



## Z