pandas-td Compatibility

Compatible Functions

- pytd.pandas_td.connect
- pytd.pandas_td.create_engine
- pytd.pandas_td.read_td_query
- pytd.pandas_td.read_td_job
- pytd.pandas_td.read_td_table
- pytd.pandas_td.read_td
- pytd.pandas_td.to_td

IPython Magics

- pytd.pandas_td.ipyhton.MagicContext
- pytd.pandas_td.ipyhton.MagicTable
- pytd.pandas_td.ipyhton.get_td_magic_context()
- pytd.pandas_td.ipyhton.TDMagics
- pytd.pandas_td.ipython.DatabasesMagics
- pytd.pandas_td.ipython.TableMagics
- pytd.pandas_td.ipython.JobMagics
- pytd.pandas_td.ipython.UseMagics
- pytd.pandas_td.ipython.QueryMagics
- td_hive
- td_presto
- pytd.pandas_td.ipython.load_ipython_extension

pytd.pandas_td.connect

`pytd.pandas_td.connect`(*apikey=None*, *endpoint=None*, ***kwargs*)[source]

Create a connection to Treasure Data

Parameters

- **apikey** (*str, optional*) – Treasure Data API key. If not given, a value of environment variable `TD_API_KEY` is used by default.
- **endpoint** (*str, optional*) – Treasure Data API server. If not given, `https://api.treasuredata.com` is used by default. List of available endpoints is: Sites and Endpoints
- **kwargs** (*dict, optional*) – Optional arguments

Returns

Return type

`pytd.Client`

pytd.pandas_td.create_engine

`pytd.pandas_td.create_engine`
(*url*, *con=None*, *header=True*, *show_progress=5.0*, *clear_progress=True*)[source]

Create a handler for query engine based on a URL.

The following environment variables are used for default connection:

> *TD_API_KEY API key TD_API_SERVER API server (default: `https://api.treasuredata.com`)*

Parameters

- **url** (*str*) – Engine descriptor in the form “type://apikey@host/database?params…” Use shorthand notation “type:database?params…” for the default connection. pytd: “params” will be ignored since pytd.QueryEngine does not have any extra parameters.
- **con** (`pytd.Client`, optional) – Handler returned by `pytd.pandas_td.connect()`. If not given, default client is used.
- **header** (*str or bool, default: True*) – Prepend comment strings, in the form “– comment”, as a header of queries. Set False to disable header.
- **show_progress** (*double or bool, default: 5.0*) – Number of seconds to wait before printing progress. Set False to disable progress entirely. pytd: This argument will be ignored.
- **clear_progress** (*bool, default: True*) – If True, clear progress when query completed. pytd: This argument will be ignored.

Returns

Return type

`pytd.query_engine.QueryEngine`

Examples

```
>>> import pytd.pandas_td as td
>>> con = td.connect(apikey=apikey, endpoint="https://api.treasuredata.com")
>>> engine = td.create_engine("presto:sample_datasets")
```

pytd.pandas_td.read_td_query

`pytd.pandas_td.read_td_query`**(*query*, *engine*, *index_col=None*, *parse_dates=None*, *distributed_join=False*, *params=None*)**[source]

Read Treasure Data query into a DataFrame.

Returns a DataFrame corresponding to the result set of the query string. Optionally provide an index_col parameter to use one of the columns as the index, otherwise default integer index will be used.

While Presto in pytd has two options to issue a query, by either `tdclient` or `prestodb`, pytd.pandas_td#read_td_query always uses the former to be compatible with the original pandas-td. Use `pytd.Client` to take advantage of the latter option.

Parameters

- **query** (*str*) – Query string to be executed.
- **engine** (`pytd.query_engine.QueryEngine`) – Handler returned by create_engine.
- **index_col** (*str, optional*) – Column name to use as index for the returned DataFrame object.
- **parse_dates** (*list or dict, optional*) –
    - List of column names to parse as dates
    - Dict of {column_name: format string} where format string is strftime compatible in case of parsing string times or is one of (D, s, ns, ms, us) in case of parsing integer timestamps
- **distributed_join** (bool, default: False) – (Presto only) If True, distributed join is enabled. If False, broadcast join is used. See https://prestodb.io/docs/current/release/release-0.77.html
- **params** (*dict, optional*) –

  Parameters to pass to execute method. pytd does not support parameter `type` ('hive', 'presto'), and query type needs to be defined by `engine`.

Available parameters:

- `db` (str): use the database
- `result_url` (str): result output URL
- `priority` (int or str): priority
    - -2: "VERY LOW"
    - -1: "LOW"
    - 0: "NORMAL"
    - 1: "HIGH"
    - 2: "VERY HIGH"
- `retry_limit` (int): max number of automatic retries
- `wait_interval` (int): sleep interval until job finish
- `wait_callback` (function): called every interval against job itself
- `engine_version` (str): run query with Hive 2 if this parameter is set to `"experimental"` in `HiveQueryEngine`. Using Hive 2 to Create Queries

Returns

Query result in a DataFrame

Return type

`pandas.DataFrame`

pytd.pandas_td.read_td_job

`pytd.pandas_td.read_td_job`(*job_id*, *engine*, *index_col=None*, *parse_dates=None*)[source]

Read Treasure Data job result into a DataFrame.

Returns a DataFrame corresponding to the result set of the job. This method waits for job completion if the specified job is still running. Optionally provide an index_col parameter to use one of the columns as the index, otherwise default integer index will be used.

Parameters

- **job_id** (*int*) – Job ID.
- **engine** (`pytd.query_engine.QueryEngine`) – Handler returned by create_engine.
- **index_col** (*str, optional*) – Column name to use as index for the returned DataFrame object.
- **parse_dates** (*list or dict, optional*) –
    - List of column names to parse as dates

- Dict of {column_name: format string} where format string is strftime compatible in case of parsing string times or is one of (D, s, ns, ms, us) in case of parsing integer timestamps

Returns

Job result in a dataframe

Return type

`pandas.DataFrame`

pytd.pandas_td.read_td_table

`pytd.pandas_td.read_td_table`(*table_name*, *engine*, *index_col=None*, *parse_dates=None*, *columns=None*, *time_range=None*, *limit=10000*)[source]

Read Treasure Data table into a DataFrame.

The number of returned rows is limited by “limit” (default 10,000). Setting limit=None means all rows. Be careful when you set limit=None because your table might be very large and the result does not fit into memory.

Parameters

- **table_name** (*str*) – Name of Treasure Data table in database.
- **engine** (`pytd.query_engine.QueryEngine`) – Handler returned by create_engine.
- **index_col** (*str, optional*) – Column name to use as index for the returned DataFrame object.
- **parse_dates** (*list or dict, optional*) –
  - List of column names to parse as dates
  - Dict of {column_name: format string} where format string is strftime compatible in case of parsing string times or is one of (D, s, ns, ms, us) in case of parsing integer timestamps
- **columns** (*list, optional*) – List of column names to select from table.
- **time_range** (*tuple (start, end), optional*) – Limit time range to select. “start” and “end” are one of None, integers, strings or datetime objects. “end” is exclusive, not included in the result.
- **limit** (*int, default: 10,000*) – Maximum number of rows to select.

Returns

Return type

`pandas.DataFrame`

pytd.pandas_td.read_td

`pytd.pandas_td.read_td`(*query*, *engine*, *index_col=None*, *parse_dates=None*, *distributed_join=False*, *params=None*)[source]

Read Treasure Data query into a DataFrame.

Returns a DataFrame corresponding to the result set of the query string. Optionally provide an index_col parameter to use one of the columns as the index, otherwise default integer index will be used.

While Presto in pytd has two options to issue a query, by either `tdclient` or `prestodb`, pytd.pandas_td#read_td_query always uses the former to be compatible with the original pandas-td. Use `pytd.Client` to take advantage of the latter option.

Parameters

- **query** (*str*) – Query string to be executed.
- **engine** (`pytd.query_engine.QueryEngine`) – Handler returned by create_engine.
- **index_col** (*str, optional*) – Column name to use as index for the returned DataFrame object.
- **parse_dates** (*list or dict, optional*) –
  - List of column names to parse as dates
  - Dict of {column_name: format string} where format string is strftime compatible in case of parsing string times or is one of (D, s, ns, ms, us) in case of parsing integer timestamps
- **distributed_join** (bool, default: False) – (Presto only) If True, distributed join is enabled. If False, broadcast join is used. See https://prestodb.io/docs/current/release/release-0.77.html
- **params** (*dict, optional*) –

  Parameters to pass to execute method. pytd does not support parameter `type` (‘hive’, ‘presto’), and query type needs to be defined by `engine`.

  Available parameters:

  - `db` (str): use the database
  - `result_url` (str): result output URL
  - `priority` (int or str): priority

- -2: “VERY LOW”
- -1: “LOW”
- 0: “NORMAL”
- 1: “HIGH”
- 2: “VERY HIGH”

- `retry_limit` (int): max number of automatic retries
- `wait_interval` (int): sleep interval until job finish
- `wait_callback` (function): called every interval against job itself
- `engine_version` (str): run query with Hive 2 if this parameter is set to `"experimental"` in `HiveQueryEngine`. Using Hive 2 to Create Queries

Returns

Query result in a DataFrame

Return type

`pandas.DataFrame`

pytd.pandas_td.to_td

`pytd.pandas_td.to_td`(*frame*, *name*, *con*, *if_exists='fail'*, *time_col=None*, *time_index=None*, *index=True*, *index_label=None*, *chunksize=10000*, *date_format=None*, *writer='bulk_import'*, ***kwargs*)[source]

Write a DataFrame to a Treasure Data table.

This method converts the dataframe into a series of key-value pairs and sends them using the Treasure Data streaming API. The data is divided into chunks of rows (default 10,000) and uploaded separately. If upload failed, the client retries the process for a certain amount of time (max_cumul_retry_delay; default 600 secs). This method may fail and raise an exception when retries did not success, in which case the data may be partially inserted. Use the bulk import utility if you cannot accept partial inserts.

Parameters

- **frame** (`pandas.DataFrame`) – DataFrame to be written.
- **name** (*str*) – Name of table to be written, in the form ‘database.table’.
- **con** (`pytd.Client`) – A client for a Treasure Data account returned by `pytd.pandas_td.connect()`.
- **if_exists** (*str, {'error' ('fail'), 'overwrite' ('replace'), 'append', 'ignore'}, default: 'error'*) –

  What happens when a target table already exists. For pandas-td compatibility, ‘error’, ‘overwrite’, ‘append’, ‘ignore’ can respectively be:

  - fail: If table exists, raise an exception.
  - replace: If table exists, drop it, recreate it, and insert data.
  - append: If table exists, insert data. Create if does not exist.
  - ignore: If table exists, do nothing.
- **time_col** (*str, optional*) – Column name to use as “time” column for the table. Column type must be integer (unixtime), datetime, or string. If None is given (default), then the current time is used as time values.
- **time_index** (*int, optional*) – Level of index to use as “time” column for the table. Set 0 for a single index. This parameter implies index=False.
- **index** (*bool, default: True*) – Write DataFrame index as a column.
- **index_label** (*str or sequence, default: None*) – Column label for index column(s). If None is given (default) and index is True, then the index names are used. A sequence should be given if the DataFrame uses MultiIndex.
- **chunksize** (*int, default: 10,000*) – Number of rows to be inserted in each chunk from the dataframe. pytd: This argument will be ignored.
- **date_format** (*str, default: None*) – Format string for datetime objects
- **writer** (str, {‘bulk_import’, ‘insert_into’, ‘spark’}, or `pytd.writer.Writer`, default: ‘bulk_import’) – A Writer to choose writing method to Treasure Data. If not given or string value, a temporal Writer instance will be created.
- **fmt** (*str, {'csv', 'msgpack'}, default: 'csv'*) –

  Format for bulk_import.

  - **csv**

    Convert dataframe to temporary CSV file. Stable option but slower than msgpack option because pytd saves dataframe as temporary CSV file, then td-client converts it to msgpack. Types of columns are guessed by `pandas.read_csv` and it causes unintended type conversion e.g., 0-padded string `"00012"` into integer `12`.
  - **msgpack**

    Convert to temporary msgpack.gz file. Fast option but there is a slight difference on type conversion compared to csv.

IPython Magics

Use IPython magics to access to Treasure Data. Start by loading the magics.

```
In [1]: %load_ext pytd.pandas_td.ipython
```

pytd.pandas_td.ipyhton.MagicContext

class`pytd.pandas_td.ipython.MagicContext`[source]

`__init__`()[source]

Initialize self. See help(type(self)) for accurate signature.

`connect`()[source]

pytd.pandas_td.ipyhton.MagicTable

class`pytd.pandas_td.ipython.MagicTable`(*table*)[source]

`__init__`(*table*)[source]

Initialize self. See help(type(self)) for accurate signature.

pytd.pandas_td.ipyhton.get_td_magic_context()

pytd.pandas_td.ipyhton.TDMagics

class`pytd.pandas_td.ipython.TDMagics`(*shell*)[source]

`__init__`(*shell*)[source]

Create a configurable given a config config.

Parameters

- **config** (*Config*) – If this is empty, default values are used. If config is a `Config` instance, it will be used to configure the instance.
- **parent** (*Configurable instance, optional*) – The parent Configurable instance of this object.

Notes

Subclasses of Configurable must call the `__init__()` method of `Configurable` *before* doing anything else and using `super()`:

```
class MyConfigurable(Configurable):
    def __init__(self, config=None):
        super(MyConfigurable, self).__init__(config=config)
        # Then any other code you need to finish initialization.
```

This ensures that instances will be configured properly.

pytd.pandas_td.ipython.DatabasesMagics

class`pytd.pandas_td.ipython.DatabasesMagics`(*shell*)[source]

`td_databases`(*pattern*)

List databases in the form of pandas.DataFrame.

```
%td_databases [<database_name_pattern>]
```

Parameters

<database_name_pattern> (*string, optional*) – List databases matched to a given pattern. If not given, all existing databases will be listed.

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %td_databases sample
Out[2]:
                                    name        count     permission               created_at
updated_at
0    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx       348124  administrator 2019-01-23 05:48:11+00:00 2019-01-23 05:48:
11+00:00
1                             yyyyyyyyy            0  administrator 2017-12-14 07:52:34+00:00 2017-12-14 07:52:
34+00:00
2                         zzzzzzzzzzzz            0  administrator 2016-05-25 23:12:06+00:00 2016-05-25 23:12:
06+00:00
...

In [3]: %td_databases sample
Out[3]:
                         name     count     permission               created_at                updated_at
0                   sampledb         2  administrator 2014-04-11 22:29:38+00:00 2014-04-11 22:29:38+00:00
1             sample_xxxxxxxx         2  administrator 2017-06-02 23:37:41+00:00 2017-06-02 23:37:41+00:00
2             sample_datasets   8812278     query_only 2014-10-04 01:13:11+00:00 2018-03-16 04:59:06+00:00
...
```

`magics`= *{'cell': {}, 'line': {'td_databases': 'td_databases'}}*

`registered`= *True*

pytd.pandas_td.ipython.TableMagics

class`pytd.pandas_td.ipython.TablesMagics`(*shell*)[source]

`td_tables`(*pattern*)

List tables in databases.

```
%td_tables [<table_identifier_pattern>]
```

Parameters

<table_identifier_pattern> (*string, optional*) – List tables matched to a given pattern. Table identifier is represented as `database_name.table_name`. If not given, all existing tables will be listed.

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %td_tables
Out[2]:
            db_name                          name      count  estimated_storage_size
last_log_timestamp               created_at
0     xxxxx_demo_aa                customer_test         70                    1047 2018-02-05 06:20:32+00:00
2018-02-05 06:20:24+00:00
1     xxxxx_demo_aa                    email_log          0                       0 1970-01-01 00:00:00+00:00
2018-02-05 07:19:57+00:00
2             yy_wf           topk_similar_items      10598                  134208 2018-04-16 09:23:57+00:00
2018-04-16 09:59:48+00:00
...

In [3]: %td_tables sample
Out[3]:
                db_name                                  name    count  estimated_storage_size
last_log_timestamp               created_at
0                xx_test                      aaaaaaaa_sample        0                       0 1970-01-01 00:
00:00+00:00 2015-10-20 17:37:40+00:00
1               sampledb                             sampletbl        2                     843 1970-01-01 00:
00:00+00:00 2014-04-11 22:30:08+00:00
2            zzzz_test_db                   sample_output_tab        4                     889 2018-06-06 08:
26:20+00:00 2018-06-06 08:27:12+00:00
...
```

`magics=` *{'cell': {}, 'line': {'td_tables': 'td_tables'}}*

`registered=` *True*

pytd.pandas_td.ipython.JobMagics

class`pytd.pandas_td.ipython.JobsMagics`(*shell*)[source]

`td_jobs`(*line*)

List job activities in an account.

```
%td_jobs
```

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %td_jobs
Out[2]:
     status     job_id    type                 start_at                                                query
0     error  448650806    hive 2019-04-12 05:33:36+00:00  with null_samples as (\n  select\n    id,\n   ...
1   success  448646994  presto 2019-04-12 05:23:29+00:00  -- read_td_query\n-- set session distributed_j...
2   success  448646986  presto 2019-04-12 05:23:27+00:00  -- read_td_query\n-- set session distributed_j...
...
```

`magics=` *{'cell': {}, 'line': {'td_jobs': 'td_jobs'}}*

`registered=` *True*

pytd.pandas_td.ipython.UseMagics

class`pytd.pandas_td.ipython.UseMagics`(*shell*)[source]

`td_use`(*line*)

Use a specific database.

This magic pushes all table names in a specified database into the current namespace.

```
%td_use [<database_name>]
```

Parameters

<database_name> (*string*) – Database name.

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %td_use sample_datasets
INFO: import nasdaq
INFO: import www_access

In [3]: nasdaq  # describe table columns in the form of DataFrame
Out[3]: <pytd.pandas_td.ipython.MagicTable at 0x117651908>
```

`magics`= *{'cell': {}, 'line': {'td_use': 'td_use'}}*

`registered`= *True*

pytd.pandas_td.ipython.QueryMagics

class`pytd.pandas_td.ipython.QueryMagics`(*shell*)[source]

`create_job_parser`()[source]

`parse_job_args`(*line*)[source]

`create_query_parser`(*engine_type*)[source]

`parse_query_args`(*engine_type*, *line*)[source]

`push_code`(*code*, *end='\n'*)[source]

`display_code_block`()[source]

`build_query`(*cell*)[source]

`build_engine`(*engine_type*, *database*, *args*)[source]

`convert_time`(*d*)[source]

`set_index`(*d*, *index*, *args*)[source]

`pivot`(*d*, *args*)[source]

`post_process`(*d*, *args*)[source]

`run_job`(*line*)[source]

`run_query`(*engine_type*, *line*, *cell*)[source]

`td_job`(*line*)

Get job result.

```
%td_job [--pivot] [--plot] [--dry-run] [--verbose]
        [--connection <connection>] [--dropna] [--out <out>]
        [--out-file <out_file>] [--quiet] [--timezone <timezone>]
        job_id
```

Parameters

- **<job_id>** (*integer*) – Job ID.
- **--pivot** (*optional*) – Run pivot_table against dimensions.
- **--plot** (*optional*) – Plot the query result.
- **-n** (*--dry_run,*) – Output translated code without running query.
- **-v** (*--verbose,*) – Verbose output.
- **<connection>, -c <connection>** (*--connection*) – Use specified connection.
- d (*--dropna,*) – Drop columns if all values are NA.
- **<out>, -o <out>** (*--out*) – Store the result to variable.
- **<out_file>, -O <out_file>** (*--out-file*) – Store the result to file.
- q (*--quiet,*) – Disable progress output.
- **<timezone>, -T <timezone>** (*--timezone*) – Set timezone to time index.

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %td_job 451709460  # select * from sample_datasets.nasdaq limit 5
Out[2]:
                     symbol  open  volume     high      low    close
time
1992-08-25 16:00:00   ATRO   0.0    3900   0.7076   0.7076   0.7076
1992-08-25 16:00:00   ALOG   0.0   11200  11.0000  10.6250  11.0000
1992-08-25 16:00:00   ATAX   0.0   11400  11.3750  11.0000  11.0000
1992-08-25 16:00:00   ATRI   0.0    5400  14.3405  14.0070  14.2571
1992-08-25 16:00:00   ABMD   0.0   38800   5.7500   5.2500   5.6875
```

td_hive

td_hive(line, cell)

Run a Hive query.

```
%%td_hive [<database>] [--pivot] [--plot] [--dry-run] [--verbose]
          [--connection <connection>] [--dropna] [--out <out>]
          [--out-file <out_file>] [--quiet] [--timezone <timezone>]

<query>
```

Parameters

- **<query>** (*string*) – Hive query.
- **<database>** (*string, optional*) – Database name.
- **--pivot** (*optional*) – Run pivot_table against dimensions.
- **--plot** (*optional*) – Plot the query result.
- **-n** (*--dry_run,*) – Output translated code without running query.
- **-v** (*--verbose,*) – Verbose output.
- **<connection>, -c <connection>** (*--connection*) – Use specified connection.
- **-d** (*--dropna,*) – Drop columns if all values are NA.
- **<out>, -o <out>** (*--out*) – Store the result to variable.
- **<out_file>, -O <out_file>** (*--out-file*) – Store the result to file.
- **-q** (*--quiet,*) – Disable progress output.
- **<timezone>, -T <timezone>** (*--timezone*) – Set timezone to time index.

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %%td_hive
   ...: select hivemall_version()
   ...:
Out[2]:
                         _c0
0  0.6.0-SNAPSHOT-201901-r01
```

td_presto

td_presto(line,cell)

Run a Presto query.

```
%%td_presto [<database>] [--pivot] [--plot] [--dry-run] [--verbose]
            [--connection <connection>] [--dropna] [--out <out>]
            [--out-file <out_file>] [--quiet] [--timezone <timezone>]

<query>
```

Parameters

- **<query>** (*string*) – Presto query.
- **<database>** (*string, optional*) – Database name.
- **--pivot** (*optional*) – Run pivot_table against dimensions.
- **--plot** (*optional*) – Plot the query result.
- **-n** (*--dry_run,*) – Output translated code without running query.
- **-v** (*--verbose,*) – Verbose output.
- **<connection>, -c <connection>** (*--connection*) – Use specified connection.
- **-d** (*--dropna,*) – Drop columns if all values are NA.
- **<out>, -o <out>** (*--out*) – Store the result to variable.
- **<out_file>, -O <out_file>** (*--out-file*) – Store the result to file.
- **-q** (*--quiet,*) – Disable progress output.
- **<timezone>, -T <timezone>** (*--timezone*) – Set timezone to time index.

Returns

Return type

`pandas.DataFrame`

Examples

```
In [1]: %load_ext pytd.pandas_td.ipython

In [2]: %%td_presto
   ...: select * from sample_datasets.nasdaq limit 5
   ...:
Out[2]:
                    symbol  open  volume     high      low    close
time
1989-01-26 16:00:00   SMTC   0.0    8000   0.4532   0.4532   0.4532
1989-01-26 16:00:00   SEIC   0.0  163200   0.7077   0.6921   0.7025
1989-01-26 16:00:00   SIGI   0.0    2800   3.9610   3.8750   3.9610
1989-01-26 16:00:00   NAVG   0.0    1800  14.6740  14.1738  14.6740
1989-01-26 16:00:00   MOCO   0.0   71101   3.6722   3.5609   3.5980
```

`magics`= *{'cell': {'td_hive': 'td_hive', 'td_presto': 'td_presto'}, 'line': {'td_job': 'td_job'}}*

`registered`= *True*

pytd.pandas_td.ipython.load_ipython_extension

`pytd.pandas_td.ipython.load_ipython_extension`(***ipython***)[source]