

---

# **PyStratum MySQL MariaDB Documentation**

**P.R. Water**

**Jul 08, 2022**



---

## Contents:

---

<b>1</b>	<b>Licence</b>	<b>3</b>
<b>2</b>	<b>API</b>	<b>5</b>
2.1	pystratum_mysql package . . . . .	5
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



A stored procedure and function loader and wrapper generator for MySQL and MariaDB Python.



# CHAPTER 1

---

## Licence

---

This project is licensed under the terms of the [MIT-licentie](#).



# CHAPTER 2

---

## API

---

## 2.1 pystratum\_mysql package

### 2.1.1 Subpackages

#### pystratum\_mysql.backend package

##### Submodules

#### pystratum\_mysql.backend.MySqlBackend module

```
class pystratum_mysql.backend.MySqlBackend.MySqlBackend
    Bases: pystratum_backend.Backend
```

PyStratum Backend for MySQL & MariaDB.

```
create_constant_worker(config: configparser.ConfigParser, io: pystrat-
    tum_backend.StratumStyle.StratumStyle) → Op-
        tional[pystratum_backend.ConstantWorker.ConstantWorker]
```

Creates the object that does the actual execution of the constant command for the backend.

##### Parameters

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

##### Return type

ConstantWorker|None

```
create_routine_loader_worker(config: configparser.ConfigParser, io: pystrat-
    tum_backend.StratumStyle.StratumStyle) → Op-
        tional[pystratum_backend.RoutineLoaderWorker.RoutineLoaderWorker]
```

Creates the object that does the actual execution of the routine loader command for the backend.

##### Parameters

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

**Return type** RoutineLoaderWorker|None

```
create_routine_wrapper_generator_worker(config: configparser.ConfigParser, io: pystratum_backend.StratumStyle.StratumStyle) → Optional[pystratum_backend.RoutineWrapperGeneratorWorker.Rout
```

Creates the object that does the actual execution of the routine wrapper generator command for the back-end.

**Parameters**

- **config** (*ConfigParser*) – The settings from the PyStratum configuration file.
- **io** (*StratumStyle*) – The output object.

**Return type** RoutineWrapperGeneratorWorker|None

### pystratum\_mysql.backend.MySqlConstantWorker module

```
class pystratum_mysql.backend.MySqlConstantWorker(io: pystratum_backend.StratumStyle.StratumStyle, config: configparser.ConfigParser)
```

Bases: *pystratum\_mysql.backend.MySqlWorker.MySqlWorker*, *pystratum\_common.backend.CommonConstantWorker.CommonConstantWorker*

Class for creating constants based on column widths, and auto increment columns and labels for MySQL databases.

```
static derive_field_length(column: Dict[str, Any]) → Optional[int]
```

Returns the width of a field based on column.

**Parameters** **column** (*dict*) – The column of which the field is based.

**Return type** int|None

### pystratum\_mysql.backend.MySqlRoutineLoaderWorker module

```
class pystratum_mysql.backend.MySqlRoutineLoaderWorker(io: pystratum_backend.StratumStyle.StratumStyle, config: configparser.ConfigParser)
```

Bases: *pystratum\_mysql.backend.MySqlWorker.MySqlWorker*, *pystratum\_common.backend.CommonRoutineLoaderWorker.CommonRoutineLoaderWorker*

Class for loading stored routines into a MySQL instance from (pseudo) SQL files.

**MAX\_LENGTH\_BINARY = 255**

Maximum length of a varbinary.

**MAX\_LENGTH\_CHAR = 255**

Maximum length of a varchar.

**MAX\_LENGTH\_VARBINARY = 4096**

Maximum length of a varbinary.

**MAX\_LENGTH\_VARCHAR = 4096**

Maximum length of a varchar.

```
create_routine_loader_helper(routine_name: str, pystratum_old_metadata: Optional[Dict[KT, VT]], rdbms_old_metadata: Optional[Dict[KT, VT]]) → pystratum_mysql.helper.MySqlRoutineLoaderHelper.MySqlRoutineLoaderHelper
```

Creates a Routine Loader Helper object.

#### Parameters

- **routine\_name** (*str*) – The name of the routine.
- **pystratum\_old\_metadata** (*dict*) – The old metadata of the stored routine from PyStratum.
- **rdbms\_old\_metadata** (*dict*) – The old metadata of the stored routine from MySQL.

**Return type** *MySqlRoutineLoaderHelper*

## pystratum\_mysql.backend.MySqlRoutineWrapperGeneratorWorker module

```
class pystratum_mysql.backend.MySqlRoutineWrapperGeneratorWorker.MySqlRoutineWrapperGenerat
```

Bases: *pystratum\_mysql.backend.MySqlWorker.MySqlWorker*,  
*pystratum\_common.backend.CommonRoutineWrapperGeneratorWorker*.  
*CommonRoutineWrapperGeneratorWorker*

Class for generating a class with wrapper methods for calling stored routines in a MySQL database.

## pystratum\_mysql.backend.MySqlWorker module

```
class pystratum_mysql.backend.MySqlWorker.MySqlWorker(io: pystratum_backend.StratumStyle.StratumStyle, config: configparser.ConfigParser)
```

Bases: *object*

**connect()** → None

Connects to the database.

**disconnect ()** → None  
Disconnects from the database.

## Module contents

### pystatum\_mysql.helper package

#### Submodules

##### pystatum\_mysql.helper.MySqlDataTypeHelper module

**class** pystatum\_mysql.helper.MySqlDataTypeHelper.**MySqlDataTypeHelper**

Bases: pystatum\_common.helper.DataTypeHelper.DataTypeHelper

Utility class for deriving information based on a MySQL data type.

**column\_type\_to\_python\_type** (*data\_type\_info*: Dict[str, Any]) → str

Returns the corresponding Python data type of a MySQL data type.

**Parameters** *data\_type\_info* (*dict*) – The MySQL data type metadata.

**Return type** str

**column\_type\_to\_python\_type\_hint** (*data\_type\_info*: Dict[str, Any]) → str

Returns the corresponding Python data type hinting of a MySQL data type.

**Parameters** *data\_type\_info* (*dict*) – The MySQL data type metadata.

**Return type** str

## [pystratum\\_mysql.helper.MySqlRoutineLoaderHelper module](#)

```
class pystratum_mysql.helper.MySqlRoutineLoaderHelper(io:  
                                                    py-  
                                                    s-  
                                                    tra-  
                                                    tum_backend.Str-  
                                                    dl:  
                                                    py-  
                                                    s-  
                                                    tra-  
                                                    tum_mysql.MySq-  
                                                    rou-  
                                                    tine_filename:  
                                                    str;  
                                                    rou-  
                                                    tine_file_encoding:  
                                                    str;  
                                                    py-  
                                                    s-  
                                                    tra-  
                                                    tum_old_metadata:  
                                                    Op-  
                                                    tional[Dict[KT,  
                                                    VT]],  
                                                    re-  
                                                    place_pairs:  
                                                    Dict[str,  
                                                    Any],  
                                                    rdbms_old_metadata:  
                                                    Op-  
                                                    tional[Dict[KT,  
                                                    VT]],  
                                                    sql_mode:  
                                                    str,  
                                                    char-  
                                                    ac-  
                                                    ter_set:  
                                                    str,  
                                                    col-  
                                                    late:  
                                                    str)
```

Bases: `pystratum_common.helper.RoutineLoaderHelper.RoutineLoaderHelper`

Class for loading a single stored routine into a MySQL instance from a (pseudo) SQL file.

### Module contents

## [pystratum\\_mysql.wrapper package](#)

### Submodules

### **pystratum\_mysql.wrapper.MySqlBulkWrapper module**

```
class pystratum_mysql.wrapper.MySqlBulkWrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.BulkWrapper.BulkWrapper  
Wrapper method generator for stored procedures with large result sets.
```

### **pystratum\_mysql.wrapper.MySqlFunctionsWrapper module**

```
class pystratum_mysql.wrapper.MySqlFunctionsWrapper(routine:  
    Dict[str,  
    Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.FunctionsWrapper.FunctionsWrapper  
Wrapper method generator for stored functions.
```

### **pystratum\_mysql.wrapper.MySqlLogWrapper module**

```
class pystratum_mysql.wrapper.MySqlLogWrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.LogWrapper.LogWrapper  
Wrapper method generator for stored procedures with designation type log.
```

### **pystratum\_mysql.wrapper.MySqlMultiWrapper module**

```
class pystratum_mysql.wrapper.MySqlMultiWrapper(routine:  
    Dict[str,  
    Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.MultiWrapper.MultiWrapper  
Wrapper method generator for stored procedures with designation type multi.
```

### **pystratum\_mysql.wrapper.MySqlNoneWrapper module**

```
class pystratum_mysql.wrapper.MySqlNoneWrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.
```

wrapper.NoneWrapper.NoneWrapper  
Wrapper method generator for stored procedures without any result set.

### pystratum\_mysql.wrapper.MySqlRow0Wrapper module

```
class pystratum_mysql.wrapper.MySqlRow0Wrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
        bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.Row0Wrapper.Row0Wrapper
```

Wrapper method generator for stored procedures that are selecting 0 or 1 row.

### pystratum\_mysql.wrapper.MySqlRow1Wrapper module

```
class pystratum_mysql.wrapper.MySqlRow1Wrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
        bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.Row1Wrapper.Row1Wrapper
```

Wrapper method generator for stored procedures that are selecting 1 row.

### pystratum\_mysql.wrapper.MySqlRowsWithIndexWrapper module

```
class pystratum_mysql.wrapper.MySqlRowsWithIndexWrapper(routine:  
    Dict[str,  
        Any],  
    lob_as_string_flag:  
        bool)  
Bases: pystratum_common.wrapper.RowsWithIndexWrapper.RowsWithIndexWrapper,  
pystratum_mysql.wrapper.MySqlWrapper.MySqlWrapper
```

Wrapper method generator for stored procedures whose result set must be returned using tree structure using a combination of non-unique columns.

### pystratum\_mysql.wrapper.MySqlRowsWithKeyWrapper module

```
class pystratum_mysql.wrapper.MySqlRowsWithKeyWrapper(routine:  
    Dict[str,  
        Any],  
    lob_as_string_flag:  
        bool)  
Bases: pystratum_common.wrapper.RowsWithKeyWrapper.RowsWithKeyWrapper,  
pystratum_mysql.wrapper.MySqlWrapper.MySqlWrapper
```

Wrapper method generator for stored procedures whose result set must be returned using tree structure using a combination of unique columns.

## pystratum\_mysql.wrapper.MySqlRowsWrapper module

```
class pystratum_mysql.wrapper.MySqlRowsWrapper(routine:  
    Dict[str, Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.RowsWrapper.RowsWrapper  
Wrapper method generator for stored procedures that are selecting 0, 1, or more rows.
```

## pystratum\_mysql.wrapper.MySqlSingleton0Wrapper module

```
class pystratum_mysql.wrapper.MySqlSingleton0Wrapper(routine:  
    Dict[str,  
    Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.Singleton0Wrapper.Singleton0Wrapper  
Wrapper method generator for stored procedures that are selecting 0 or 1 row with one column only.
```

## pystratum\_mysql.wrapper.MySqlSingleton1Wrapper module

```
class pystratum_mysql.wrapper.MySqlSingleton1Wrapper(routine:  
    Dict[str,  
    Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_common.wrapper.Singleton1Wrapper.Singleton1Wrapper,  
pystratum_mysql.wrapper.MySqlWrapper.MySqlWrapper  
Wrapper method generator for stored procedures that are selecting 1 row with one column only.
```

## pystratum\_mysql.wrapper.MySqlTableWrapper module

```
class pystratum_mysql.wrapper.MySqlTableWrapper(routine:  
    Dict[str,  
    Any],  
    lob_as_string_flag:  
    bool)  
Bases: pystratum_mysql.wrapper.MySqlWrapper, pystratum_common.  
wrapper.TableWrapper.TableWrapper  
Wrapper method generator for printing the result set of stored procedures in a table format.
```

## pystratum\_mysql.wrapper.MySqlWrapper module

```
class pystratum_mysql.wrapper.MySqlWrapper(routine: Dict[str, Any],  
                                             lob_as_string_flag: bool)  
Bases: pystratum_common.wrapper.Wrapper, abc.ABC  
Parent class for wrapper method generators for stored procedures and functions.
```

**is\_lob\_parameter** (*parameters*: *List[Dict[str, Any]]*) → *bool*

Returns True if one of the parameters is a BLOB or CLOB. Otherwise, returns False.

**Parameters** **parameters** – The parameters of a stored routine.

**Return type** *bool*:

## Module contents

```
pystratum_mysql.wrapper.create_routine_wrapper(routine: Dict[str, Any],  
                                              lob_as_string_flag: bool) → pystratum_mysql.wrapper.MySqlWrapper.MySqlWrapper
```

A factory for creating the appropriate object for generating a wrapper method for a stored routine.

**Parameters**

- **routine** (*dict [str, str]*) – The metadata of the stored routine.
- **lob\_as\_string\_flag** (*bool*) – If True BLOBS and CLOBS must be treated as strings.

**Return type** *MySqlWrapper*

## 2.1.2 Submodules

### 2.1.3 pystratum\_mysql.MySqlConnector module

**class** `pystratum_mysql.MySqlConnector.MySqlConnector`

Bases: `object`

Interface for classes for connecting to a MySql instances.

**connect** () → `mysql.connector.connection.MySQLConnection`

Connects to the MySql instance.

**disconnect** () → `None`

Disconnects from the MySql instance.

**is\_alive** () → *bool*

Returns whether Python is (still) connected to a MySQL or MariaDB instance.

**Return type** *bool*

### 2.1.4 pystratum\_mysql.MySqlDataLayer module

```
class pystratum_mysql.MySqlDataLayer.MySqlDataLayer (connector: pystratum_mysql.MySqlConnector.MySqlConnector)
```

Bases: `object`

Class for connecting to a MySQL instance and executing SQL statements. Also, a parent class for classes with static wrapper methods for executing stored procedures and functions.

**commit** () → `None`

Commits the current transaction. See <https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection-commit.html>

**connect** () → `None`

Connects to a MySQL instance. See <https://dev.mysql.com/doc/connector-python/en/connector-python-connectargs.html> for a complete overview of all possible keys in config.

**connect\_if\_not\_alive()** → None

Connects or reconnects to the MySQL or MariaDB instance when Python is not (longer) connected to a MySQL or MariaDB instance. See <https://dev.mysql.com/doc/connector-python/en/connector-python-connectargs.html> for a complete overview of all possible keys in config.

**disconnect()** → None

Disconnects from the MySQL instance. See <https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection-disconnect.html>.

**execute\_multi(sql: str)** → None

Executes a multi query that does not select any rows.

**Parameters** **sql** (*str*) – The SQL statements.

**execute\_none(sql: str, \*params)** → int

Executes a query that does not select any rows. Returns the number of affected rows.

**Parameters**

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The values for the statement.

**Return type** **int**

**execute\_rows(sql: str, \*params)** → List[Dict[str, Any]]

Executes a query that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

**Parameters**

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The arguments for the statement.

**Return type** **list[dict[str, \*]]**

**execute\_singleton1(sql: str, \*params)** → Any

Executes SQL statement that selects 1 row with 1 column. Returns the value of the selected column.

**Parameters**

- **sql** (*str*) – The SQL calling the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type** **\***:

**execute\_sp\_bulk(bulk\_handler: pystratum\_middle.BulkHandler.BulkHandler, sql: str, \*params)**

→ int

Executes a stored routine with designation type “bulk”. Returns the number of rows processed.

**Parameters**

- **bulk\_handler** (*BulkHandler*) – The bulk handler for processing the selected rows.
- **sql** (*str*) – The SQL statement for calling the stored routine.
- **params** (*iterable*) – The arguments for calling the stored routine.

**Return type** **int**

**execute\_sp\_log(sql: str, \*params)** → int

Executes a stored routine with designation type “log”. Returns the number of log messages.

**Parameters**

- **sql** (*str*) – The SQL statement for calling the stored routine.

- **params** (*iterable*) – The arguments for calling the stored routine.

**Return type** `int`

**execute\_sp\_multi** (*sql: str, \*params*) → `List[Dict[str, Any]]`

Executes a stored routine with designation type “multi”. Returns a list of the result sets.

**Parameters**

- **sql** (*str*) – The SQL statement for calling the stored routine.
- **params** (*iterable*) – The arguments for calling the stored routine.

**Return type** `list[dict[str,*]]`

**execute\_sp\_none** (*sql: str, \*params*) → `int`

Executes a stored routine that does not select any rows. Returns the number of affected rows.

**Parameters**

- **sql** (*str*) – The SQL calling the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type** `int`

**execute\_sp\_row0** (*sql: str, \*params*) → `Optional[Dict[str, Any]]`

Executes a stored procedure that selects 0 or 1 row. Returns the selected row or None.

**Parameters**

- **sql** (*str*) – The SQL call the the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type** `Nonedict[str,*]`

**execute\_sp\_row1** (*sql: str, \*params*) → `Dict[str, Any]`

Executes a stored procedure that selects 1 row. Returns the selected row.

**Parameters**

- **sql** (*str*) – The SQL calling the the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type** `dict[str,*]`

**execute\_sp\_rows** (*sql: str, \*params*) → `List[Dict[str, Any]]`

Executes a stored procedure that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

**Parameters**

- **sql** (*str*) – The SQL statement.
- **params** (*iterable*) – The arguments for the statement.

**Return type** `list[dict[str,*]]`

**execute\_sp\_singleton0** (*sql: str, \*params*) → `Any`

Executes a stored procedure that selects 0 or 1 row with 1 column. Returns the value of selected column or None.

**Parameters**

- **sql** (*str*) – The SQL calling the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type**

•

**execute\_sp\_singleton1** (*sql: str, \*params*) → Any

Executes a stored routine with designation type “table”, i.e a stored routine that is expected to select 1 row with 1 column.

**Parameters**

- **sql** (*str*) – The SQL calling the the stored procedure.
- **params** (*iterable*) – The arguments for the stored procedure.

**Return type**

- The value of the selected column.

**is\_alive()** → bool

Returns whether Python is (still) connected to a MySQL or MariaDB instance.

**Return type** bool

**last\_sql()** → str

Returns the last execute SQL statement.

**line\_buffered = None**

If True log messages from stored procedures with designation type ‘log’ are line buffered (Note: In python sys.stdout is buffered by default).

**rollback()** → None

Rolls back the current transaction. See <https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection-rollback.html>

**start\_transaction** (*consistent\_snapshot: bool = False, isolation\_level: str = 'READ-COMMITTED', readonly: Optional[bool] = None*) → None

Starts a transaction. See <https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection-start-transaction.html>

**Parameters**

- **consistent\_snapshot** (*bool*) –
- **isolation\_level** (*str*) –
- **readonly** (*bool*) –

## 2.1.5 pystratum\_mysql.MySqlDefaultConnector module

**class** pystratum\_mysql.MySqlDefaultConnector.**MySqlDefaultConnector** (*params: Dict[str, Union[str, int]]*)

Bases: *pystratum\_mysql.MySqlConnector.MySqlConnector*

Connects to a MySQL instance using username and password.

**connect()** → mysql.connector.connection.MySQLConnection

Connects to the MySQL instance.

**disconnect()** → None

Disconnects from the MySQL instance.

**is\_alive()** → bool

Returns whether Python is (still) connected to a MySQL or MariaDB instance.

**Return type** bool

## 2.1.6 pystratum\_mysql.MySqlMetadataDataLayer module

**class** pystratum\_mysql.MySqlMetadataDataLayer.**MySqlMetadataDataLayer**(*io:*

*pystratum\_backend.StratumStyle.StratumConnector:*  
*pystratum\_mysql.MySqlConnector:MySQL*

Bases: pystratum\_common.MetadataDataLayer.MetadataDataLayer

Data layer for retrieving metadata and loading stored routines.

**call\_stored\_routine(*routine\_name*: str)** → int

Class a stored procedure without arguments.

**Parameters** **routine\_name** (*str*) – The name of the procedure.

**Return type** int

**check\_table\_exists(*table\_name*: str)** → int

Checks if a table exists in the current schema.

**Parameters** **table\_name** (*str*) – The name of the table.

**Return type** int

**connect()** → None

Connects to a MySQL instance.

**describe\_table(*table\_name*: str)** → List[Dict[str, Any]]

Describes a table.

**Parameters** **table\_name** (*str*) – The name of the table.

**Return type** list[dict[str,\*]]

**disconnect()** → None

Disconnects from the MySQL instance.

**drop\_stored\_routine(*routine\_type*: str, *routine\_name*: str)** → None

Drops a stored routine if it exists.

**Parameters**

- **routine\_type** (*str*) – The type of the routine (i.e. PROCEDURE or FUNCTION).
- **routine\_name** (*str*) – The name of the routine.

**drop\_temporary\_table(*table\_name*: str)** → None

Drops a temporary table.

**Parameters** **table\_name** (*str*) – The name of the table.

**execute\_none(*query*: str)** → int

Executes a query that does not select any rows.

**Parameters** **query** (*str*) – The query.

**Return type** int

**execute\_rows** (query: str) → List[Dict[str, Any]]

Executes a query that selects 0 or more rows. Returns the selected rows (an empty list if no rows are selected).

**Parameters** query (str) – The query.

**Return type** list[dict[str,\*]]

**execute\_singleton1** (query: str) → Any

Executes SQL statement that selects 1 row with 1 column. Returns the value of the selected column.

**Parameters** query (str) – The query.

**Return type**

•

**get\_all\_table\_columns** () → List[Dict[str, Union[str, int, None]]]

Selects metadata of all columns of all tables.

**Return type** list[dict[str,\*]]

**get\_correct\_sql\_mode** (sql\_mode: str) → str

Selects the SQL mode in the order as preferred by MySQL.

**Parameters** sql\_mode (str) – The SQL mode.

**Return type** str

**get\_label\_tables** (regex: str) → List[Dict[str, Any]]

Selects metadata of tables with a label column.

**Parameters** regex (str) – The regular expression for columns which we want to use.

**Return type** list[dict[str,\*]]

**get\_labels\_from\_table** (table\_name: str, id\_column\_name: str, label\_column\_name: str) → List[Dict[str, Any]]

Selects all labels from a table with labels.

**Parameters**

• table\_name (str) – The name of the table.

• id\_column\_name (str) – The name of the auto increment column.

• label\_column\_name (str) – The name of the column with labels.

**Return type** list[dict[str,\*]]

**get\_routine\_parameters** (routine\_name: str) → List[Dict[str, Any]]

Selects metadata of the parameters of a stored routine.

**Parameters** routine\_name (str) – The name of the routine.

**Return type** list[dict[str,\*]]

**get\_routines** () → List[Dict[str, Any]]

Selects metadata of all routines in the current schema.

**Return type** list[dict[str,\*]]

**last\_sql** () → str

The last executed SQL statement.

**Return type** str

**set\_character\_set** (*character\_set: str, collate: str*) → None

Sets the default character set and collate.

#### Parameters

- **character\_set** (*str*) – The name of the character set.
- **collate** (*str*) – The name of the collate

**set\_sql\_mode** (*sql\_mode: str*) → None

Sets the SQL mode.

Parameters **sql\_mode** (*str*) – The SQL mode.

### 2.1.7 Module contents



---

## Python Module Index

---

**p**

pystratum\_mysql.wrapper.MySqlRowsWithKeyWrapper,  
    11  
pystratum\_mysql, 19  
pystratum\_mysql.backend, 8  
pystratum\_mysql.backend.MySqlBackend, 5  
pystratum\_mysql.backend.MySqlConstantWorker, 12  
pystratum\_mysql.backend.MySqlSingleton0Wrapper,  
    12  
pystratum\_mysql.backend.MySqlRoutineLoaderWorker, 12  
    6  
pystratum\_mysql.backend.MySqlRoutineWrapper, 12  
    7  
pystratum\_mysql.backend.MySqlWorker, 7  
pystratum\_mysql.helper, 9  
pystratum\_mysql.helper.MySqlDataTypeHelper,  
    8  
pystratum\_mysql.helper.MySqlRoutineLoaderHelper,  
    9  
pystratum\_mysql.MySqlConnector, 13  
pystratum\_mysql.MySqlDataLayer, 13  
pystratum\_mysql.MySqlDefaultConnector,  
    16  
pystratum\_mysql.MySqlMetadataDataLayer,  
    17  
pystratum\_mysql.wrapper, 13  
pystratum\_mysql.wrapper.MySqlBulkWrapper,  
    10  
pystratum\_mysql.wrapper.MySqlFunctionsWrapper,  
    10  
pystratum\_mysql.wrapper.MySqlLogWrapper,  
    10  
pystratum\_mysql.wrapper.MySqlMultiWrapper,  
    10  
pystratum\_mysql.wrapper.MySqlNoneWrapper,  
    10  
pystratum\_mysql.wrapper.MySqlRow0Wrapper,  
    11  
pystratum\_mysql.wrapper.MySqlRow1Wrapper,  
    11  
pystratum\_mysql.wrapper.MySqlRowsWithIndexWrapper,  
    11



---

## Index

---

### C

call\_stored\_routine() (pystratum\_mysql.backend.MySqlBackend.MySqlBackend method), 6

check\_table\_exists() (pystratum\_mysql.backend.MySqlBackend.MySqlBackend method), 17

column\_type\_to\_python\_type() (pystratum\_mysql.helper.MySqlDataTypeHelper.MySqlDataTypeHelper method), 8

column\_type\_to\_python\_type\_hint() (pystratum\_mysql.helper.MySqlDataTypeHelper.MySqlDataTypeHelper method), 8

commit() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 13

connect() (pystratum\_mysql.backend.MySqlWorker.MySqlWorker method), 7

connect() (pystratum\_mysql.MySqlConnector.MySqlConnector method), 13

connect() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 13

connect() (pystratum\_mysql.MySqlDefaultConnector.MySqlDefaultConnector.MySqlDefaultConnector method), 16

connect() (pystratum\_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer method), 17

connect\_if\_not\_alive() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 13

create\_constant\_worker() (pystratum\_mysql.backend.MySqlBackend.MySqlBackend method), 5

create\_routine\_loader\_helper() (pystratum\_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLoaderWorker method), 7

create\_routine\_loader\_worker() (pystratum\_mysql.backend.MySqlBackend.MySqlBackend method), 5

create\_routine\_wrapper() (in module pystratum\_mysql.wrapper), 13

### D

create\_routine\_wrapper\_generator\_worker() (pystratum\_mysql.backend.MySqlBackend.MySqlBackend method), 6

derive\_field\_length() (pystratum\_mysql.backend.MySqlConstantWorker.MySqlConstantWorker static method), 6

describe\_table() (pystratum\_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer method), 17

disconnect() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 8

disconnect() (pystratum\_mysql.connector.MySqlConnector.MySqlConnector method), 13

disconnect() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 14

disconnect() (pystratum\_mysql.MySqlDefaultConnector.MySqlDefaultConnector.MySqlDefaultConnector method), 16

drop\_stored\_routine() (pystratum\_mysql.MetadataDataLayer.MetadataDataLayer method), 17

drop\_temporary\_table() (pystratum\_mysql.MetadataDataLayer.MetadataDataLayer method), 17

execute\_multi() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 14

execute\_none() (pystratum\_mysql.MySqlDataLayer.MySqlDataLayer method), 14

```

execute_none()          (pystra-      tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 17           method), 18
execute_rows()          (pystra-      get_routine_parameters()          (pystra-
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 14           method), 18
execute_rows()          (pystra-      get_routines()                  (pystra-
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 18           method), 18
execute_singleton1()    (pystra-      | 
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 14           is_alive()          (pystra-
execute_singleton1()    (pystra-      tum_mysql.MySqlConnector.MySqlConnector
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 18           method), 13
execute_sp_bulk()       (pystra-      | 
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 14           is_alive()          (pystra-
execute_sp_log()        (pystra-      tum_mysql.MySqlDataLayer.MySqlDataLayer
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 14           method), 16
execute_sp_multi()      (pystra-      is_lob_parameter()          (pystra-
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15           tum_mysql.wrapper.MySqlWrapper.MySqlWrapper
execute_sp_none()       (pystra-      method), 12
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15
execute_sp_row0()       (pystra-      | 
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15           last_sql()          (pystra-
execute_sp_row1()       (pystra-      tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15           method), 16
execute_sp_rows()       (pystra-      | 
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15           last_sql()          (pystra-
execute_sp_singleton0() (pystra-      tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 15           method), 18
execute_sp_singleton1() (pystra-      line_buffered()          (pystra-
    tum_mysql.MySqlDataLayer.MySqlDataLayer
    method), 16           tum_mysql.MySqlDataLayer.MySqlDataLayer
                           attribute), 16

```

**G**

```

get_all_table_columns() (pystra-      | 
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 18           MAX_LENGTH_BINARY          (pystra-
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
get_correct_sql_mode()  (pystra-      MAX_LENGTH_CHAR          (pystra-
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 18           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
get_label_tables()      (pystra-      MAX_LENGTH_VARBINARY     (pystra-
    tum_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer
    method), 18           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
get_labels_from_table() (pystra-      MAX_LENGTH_VARCHAR      (pystra-
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7

```

**L**

```

is_alive()              (pystra-      | 
    tum_mysql.MySqlConnector.MySqlConnector
    method), 13           last_sql()          (pystra-
                           tum_mysql.MySqlDataLayer.MySqlDataLayer
                           method), 16
is_lob_parameter()      (pystra-      line_buffered()          (pystra-
    tum_mysql.wrapper.MySqlWrapper.MySqlWrapper
    method), 12           tum_mysql.MySqlDataLayer.MySqlDataLayer
                           attribute), 16

```

**M**

```

MAX_LENGTH_BINARY        (pystra-      | 
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
MAX_LENGTH_CHAR          (pystra-      MAX_LENGTH_VARBINARY     (pystra-
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
MAX_LENGTH_VARBINARY     (pystra-      MAX_LENGTH_VARCHAR      (pystra-
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7
MAX_LENGTH_VARCHAR       (pystra-      MySqlBackend            (class      in      pystra-
                           tum_mysql.backend.MySqlRoutineLoaderWorker.MySqlRoutineLo
                           attribute), 7           MySqlBackend            (class      in      pystra-
                           tum_mysql.backend.MySqlBackend, 5
                           MySqlBulkWrapper        (class      in      pystra-
                           tum_mysql.wrapper.MySqlBulkWrapper),
                           MySqlConnector          (class      in      pystra-
                           tum_mysql.MySqlConnector), 13

```

```

MySqlConstantWorker (class in pystrat-          12
    um_mysql.backend.MySqlConstantWorker),
    6
MySqlDataLayer (class in pystrat-          12
    um_mysql.MySqlDataLayer), 13
MySqlDataTypeHelper (class in pystrat-          12
    um_mysql.helper.MySqlDataTypeHelper),
    8
MySqlDefaultConnector (class in pystrat-          16
    um_mysql.MySqlDefaultConnector), 16
MySqlFunctionsWrapper (class in pystrat-          16
    um_mysql.wrapper.MySqlFunctionsWrapper),
    10
MySqlLogWrapper (class in pystrat-          16
    um_mysql.wrapper.MySqlLogWrapper),
    10
MySqlMetadataDataLayer (class in pystrat-          17
    um_mysql.MySqlMetadataDataLayer), 17
MySqlMultiWrapper (class in pystrat-          17
    um_mysql.wrapper.MySqlMultiWrapper),
    10
MySqlNoneWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlNoneWrapper),
    10
MySqlRoutineLoaderHelper (class in pystrat-          19
    um_mysql.helper.MySqlRoutineLoaderHelper),
    9
MySqlRoutineLoaderWorker (class in pystrat-          19
    um_mysql.backend.MySqlRoutineLoaderWorker),
    6
MySqlRoutineWrapperGeneratorWorker        19
    (class           in      pystrat-
     um_mysql.backend.MySqlRoutineWrapperGeneratorWorker)
    7
MySqlRow0Wrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlRow0Wrapper),
    11
MySqlRow1Wrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlRow1Wrapper),
    11
MySqlRowsWithIndexWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlRowsWithIndexWrapper),
    11
MySqlRowsWithKeyWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlRowsWithKeyWrapper),
    11
MySqlRowsWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlRowsWrapper),
    12
MySqlSingleton0Wrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlSingleton0Wrapper),
    12
MySqlSingleton1Wrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlSingleton1Wrapper),
    12
MySqlTableWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlTableWrapper),
    12
MySqlWorker (class in pystrat-          19
    um_mysql.backend.MySqlWorker), 7
MySqlWrapper (class in pystrat-          19
    um_mysql.wrapper.MySqlWrapper), 12

```

**P**

```

pystratum_mysql (module), 19
pystratum_mysql.backend (module), 8
pystratum_mysql.backend.MySqlBackend
    (module), 5
pystratum_mysql.backend.MySqlConstantWorker
    (module), 6
pystratum_mysql.backend.MySqlRoutineLoaderWorker
    (module), 6
pystratum_mysql.backend.MySqlRoutineWrapperGenerator
    (module), 7
pystratum_mysql.backend.MySqlWorker
    (module), 7
pystratum_mysql.helper (module), 9
pystratum_mysql.helper.MySqlDataTypeHelper
    (module), 8
pystratum_mysql.helper.MySqlRoutineLoaderHelper
    (module), 9
pystratum_mysql.MySqlConnector (module),
    13
pystratum_mysql.MySqlDataLayer (module),
    13
pystratum_mysql.MySqlDefaultConnector
    (module), 16
pystratum_mysql.MySqlMetadataDataLayer
    (module), 17
pystratum_mysql.wrapper (module), 13
pystratum_mysql.wrapper.MySqlBulkWrapper
    (module), 10
pystratum_mysql.wrapper.MySqlFunctionsWrapper
    (module), 10
pystratum_mysql.wrapper.MySqlLogWrapper
    (module), 10
pystratum_mysql.wrapper.MySqlMultiWrapper
    (module), 10
pystratum_mysql.wrapper.MySqlNoneWrapper
    (module), 10
pystratum_mysql.wrapper.MySqlRow0Wrapper
    (module), 11
pystratum_mysql.wrapper.MySqlRow1Wrapper
    (module), 11
pystratum_mysql.wrapper.MySqlRowsWithIndexWrapper
    (module), 11
pystratum_mysql.wrapper.MySqlRowsWithKeyWrapper
    (module), 11

```

`pystratum_mysql.wrapper.MySqlRowsWrapper  
(module), 12`  
`pystratum_mysql.wrapper.MySqlSingleton0Wrapper  
(module), 12`  
`pystratum_mysql.wrapper.MySqlSingleton1Wrapper  
(module), 12`  
`pystratum_mysql.wrapper.MySqlTableWrapper  
(module), 12`  
`pystratum_mysql.wrapper.MySqlWrapper  
(module), 12`

## R

`rollback ()` *(pystra-  
tum\_mysql.MySqlDataLayer.MySqlDataLayer  
method), 16*

## S

`set_character_set ()` *(pystra-  
tum\_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer  
method), 18*  
`set_sql_mode ()` *(pystra-  
tum\_mysql.MySqlMetadataDataLayer.MySqlMetadataDataLayer  
method), 19*  
`start_transaction ()` *(pystra-  
tum\_mysql.MySqlDataLayer.MySqlDataLayer  
method), 16*