Important Keywords

**Connect To Database**

Parameters: *dbapiModuleName=None, dbName=None, dbUsername=None, dbPassword=None, dbHost=localhost, dbPort=5432, dbConfigFile=./resources/db.cfg*

Loads the DB API 2.0 module given *dbapiModuleName* then uses it to connect to the database using *dbName*, *dbUsername*, and *dbPassword*.

Optionally, you can specify a *dbConfigFile* wherein it will load the default property values for *dbapiModuleName*, *dbNamedbUsername* and *dbPassword*

pymssql - database interface

Port for SQL Server-1433

Host:localhost

Connect to Database: pymssql DBNAME DBUsername DBPassword localhost 1433

**Description**

Parameters:

Select Statement

Uses the input *selectStatement* to query a table in the db which will be used to determine the description.

Example:

|  |
| --- |
|  |
| @{queryResults} | Description | select \* from person |
| Log Many | @{queryResults} |  |

You will get the following: [Column(name='id', type\_code=1043, display\_size=None, internal\_size=255, precision=None, scale=None, null\_ok=None)] [Column(name='first\_name', type\_code=1043, display\_size=None, internal\_size=255, precision=None, scale=None, null\_ok=None)] [*Column(name='last\_name', type\_code=1043, display\_size=None, internal\_size*=255, precision=None, scale=None, null\_ok=None)]

**Query**

Parameter:**selectStatement**

Uses the input *selectStatement* to query for the values that will be returned as a list of tuples.

For example, given we have a table *person* with the following data:

|  |  |  |
| --- | --- | --- |
| id | first\_name | last\_name |
| 1 | Franz Allan | See |

When you do the following:

|  |  |  |
| --- | --- | --- |
| @{queryResults} | Query | select \* from person |
| Log Many | @{queryResults} |  |

**Row Count**

Parameter: selectStatement

Uses the input *selectStatement* to query the database and returns the number of rows from the query.

For example, given we have a table *person* with the following data:

|  |  |  |
| --- | --- | --- |
| id | first\_name | last\_name |
| 1 | Franz Allan | See |
| 2 | Jerry | Schneider |

When you do the following:

|  |  |  |
| --- | --- | --- |
| ${rowCount} | Row Count | select \* from person |
| Log | ${rowCount} |  |

**Row Count Is Equal To X**

Parameter: Select Statement, number of rows

Check if the number of rows returned from *selectStatement* is equal to the value submitted. If not, then this will throw an AssertionError.

For example, given we have a table *person* with the following data:

|  |  |  |
| --- | --- | --- |
| id | first\_name | last\_name |
| 1 | Franz Allan | See |
| 2 | Jerry | Schneider |

When you have the following assertions in your robot

|  |  |  |
| --- | --- | --- |
| Row Count Is Equal To X | select id from person | 1 |
| Row Count Is Equal To X | select id from person where first\_name = 'John' | 0 |

Then you will get the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Row Count Is Equal To X | select id from person | 1 | # FAIL |
| Row Count Is Equal To X | select id from person where first\_name = 'John' | 0 | # PASS |

**Table Must Exist**

*tableName*

Check if the table given exists in the database.  
  
For example, given we have a table *person* in a database  
  
When you do the following:

|  |  |
| --- | --- |
| Table Must Exist | person |

Then you will get the following:

|  |  |  |
| --- | --- | --- |
| Table Must Exist | person | # PASS |
| Table Must Exist | first\_name | # FAIL |

**Delete All Rows From Table**

*TableName*

Delete all the rows within a given table.  
  
For example, given we have a table *person* in a database  
  
When you do the following:

|  |  |
| --- | --- |
| Delete All Rows From Table | person |

If all the rows can be successfully deleted, then you will get:

|  |  |  |
| --- | --- | --- |
| Delete All Rows From Table | person | # PASS |

If the table doesn't exist or all the data can't be deleted, then you will get:

|  |  |  |
| --- | --- | --- |
| Delete All Rows From Table | first\_name | # FAIL |

**Check If Exists/Not Exist In Database**

Parameter: *selectStatement*

Check if any row would be returned by given the input *selectStatement*. If there are no results, then this will throw an AssertionError.  
  
For example, given we have a table *person* with the following data:

|  |  |  |
| --- | --- | --- |
| id | first\_name | last\_name |
| 1 | Franz Allan | See |

When you have the following assertions in your robot

|  |  |
| --- | --- |
| Check If Exists In Database | select id from person where first\_name = 'Franz Allan' |
| Check If Exists In Database | select id from person where first\_name = 'John' |

Then you will get the following:

|  |  |  |
| --- | --- | --- |
| Check If Exists In Database | select id from person where first\_name = 'Franz Allan' | # PASS |
| Check If Exists In Database | select id from person where first\_name = 'John' | # FAIL |