Querying SQLite3 database

What is SQLite3?

SQLite3 is an in-process Python library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a popular choice as an embedded database for local/client storage in application software.

How to connect to the SQLite3?

You can connect to SQLIte3 using the connect() function by passing the required database name as an argument.

```
import sqlite3
sql_connection = sqlite3.connect('database.db')
```

This makes the variable `sql_connection` an object of the SQL code engine. You can then use this to run the required queries on the database.

How to create a database table using SQLite3 and Pandas?

You can directly load a Pandas dataframe to a SQLite3 database object using the following syntax.

```
df.to_sql(table_name, sql_connection, if_exists = 'replace', index = False)
```

Here, you use the `to_sql()` function to convert the `pandas` dataframe to an SQL table.

The `table_name` and `sql_connection` arguments specify the name of the required table and the database to which you should load the dataframe.

The `if_exists` parameter can take any one of three possible values:
`'fail'` : This denies the creation of a table if one with the same name exists in the database already.
`'replace'` : This overwrites the existing table with the same name.
`'append'` : This adds information to the existing table with the same name.

Keep the `index` parameter set to `True` only if the index of the data being sent holds some informational value. Otherwise, keep it as `False` .

How to query a database table using SQLite3 and Pandas?

You can use the Pandas function `read_sql()` to query a database table.

The function returns a Pandas dataframe with the output to the query. Use the function with the following syntax:

```
df = pandas.read_sql(query_statement, sql_connection)
```

Here, the `query_statement` argument contains the complete query to the required table as a string.

Example Queries

Some typical queries with their meanings are shown in the table below.

Query statement	Purpose
SELECT * FROM table_name	Retrieve all entries of the table.
SELECT COUNT(*) FROM table_name	Retrieve total number of entries in the table.
SELECT Column_name FROM table_name	Retrieve all entries of a specific column in the table.
SELECT * FROM table_name WHERE <condition>	Retrieve all entries of the table that meet the specified condition.

Author(s)

Abhishek Gagneja