**Connecting SQL datasets with Pandas**

You must have seen connecting csv and excel files with pandas to convert them to dataframes. Here we are going to see how can we connect databases with pandas and convert them to dataframes.

We are going to use various types of SQL like SQLite, MySQL, Microsoft SQL Server, Oracle and PostgreSQL.

**SQLite**

We have two ways to connect SQLite files. First, with sqlite library and the second with SQLAlchemy.

First we will see ***sqlite3*** library. Let’s import the library.

import pandas as pd  
import sqlite3

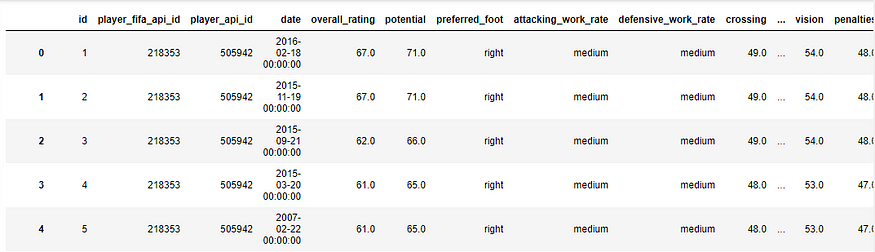
Now, connect the sqlite to the database file. And read the SQL query to read the table.

# Read in SQLite databases  
con = sqlite3.connect("database.sqlite")  
#Read the Table  
player = pd.read\_sql\_query("SELECT \* from Player\_Attributes", con)  
#close the connection  
con.close()

Here database.sqlite is file name and Player\_Attributes is table name. player is the dataframe.

Let’s move on to the ***SQLAlchemy*** method.

import sqlalchemy as sal  
from sqlalchemy import create\_enginedb\_name = "database.sqlite"  
table\_name = "Player\_Attributes"engine = sal.create\_engine("sqlite:///%s" % db\_name)  
df = pd.read\_sql\_query("SELECT \* from Player", engine)



**MySQL**

MySQL database can be connected through *msql.connector*module*.*

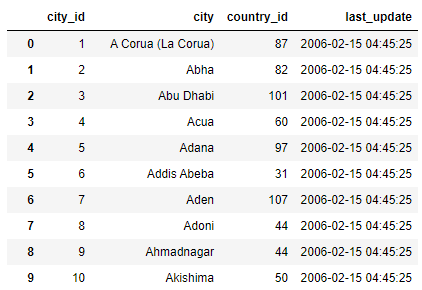
import mysql.connector as sqldb\_connection = sql.connect(host='localhost', port= '3306', database='testdb', user='root', password='1234567')db\_cursor = db\_connection.cursor()  
db\_cursor.execute('SELECT \* FROM testtable')table\_rows = db\_cursor.fetchall()df = pd.DataFrame(table\_rows)

We are using ***SQLAlchemy*** to connect to the database. We are using the default way to connect it.

db\_connection = create\_engine('mysql://root:1234567@localhost:3306/testdb')df = pd.read\_sql('SELECT \* FROM testtable', con=db\_connection)

There are other DBAPIs available- *mysqldb* and *pymysql*. Here we will see only *pymysql*.

import pymysqldb\_connection = create\_engine('mysql+pymysql://root:1234567@localhost:3306/testdb')df = pd.read\_sql('SELECT \* FROM testtable', con=db\_connection)



**Microsoft SQL Server**

For MSSQL dialect, we can connect to database into pandas dataframe by *pyodbc* module.

import pyodbc  
DB = {'servername': 'localhost\SQLExpress',  
 'database': 'testDB'}# create the connection  
conn = pyodbc.connect('DRIVER={SQL Server};SERVER=' + DB['servername'] + ';DATABASE=' + DB['database'] + ';Trusted\_Connection=yes')  
df = pd.read\_sql\_query('SELECT \* FROM TestTable',conn)

Now, let’s connect the database from ***SQLAlchemy***. Here, in MSSQL we take *pyodbc* as default DBAPI.

# in order to connect, we need server name, database name   
username='sa'  
password='1234567'  
host='localhost'  
port='1433'  
database= 'BIKE'url = 'mssql+pyodbc://{user}:{passwd}@{host}:{port}/{db}?driver=SQL+Server'.format(user=username, passwd=password, host=host, port=port, db=database)# establishing the connection to the database using engine as an interface  
engine = create\_engine(url)  
df = pd.read\_sql('SELECT \* FROM orders', engine)



*Note:* To can get the hostname, follow the below steps.

1. Open Command Prompt.
2. Type ***hostname***

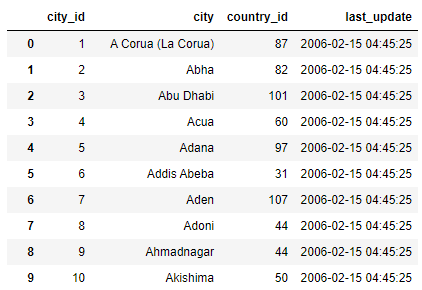
**Oracle**

We can connect oracle database through ***cx\_Oracle***. Let’s check how can we do this.

import cx\_Oracle'''if needed, place an 'r' before any parameter in order to address special characters such as '\'.'''  
dsn\_tns = cx\_Oracle.makedsn('Host Name', 'Port Number', service\_name='Service Name')'''if needed, place an 'r' before any parameter in order to address special characters such as '\'. For example, if your user name contains '\', you'll need to place 'r' before the user name: user=r'User Name'  
'''  
conn = cx\_Oracle.connect(user=r'User Name', password='Personal Password', dsn=dsn\_tns) c = conn.cursor()  
c.execute('select \* from database.table')conn.close()

Second method, connect through ***SQLAlchemy***. There are two DBAPI- default and *cx\_oracle*.

engine = create\_engine('oracle://root:1234567@127.0.0.1:1521/testdb')  
  
engine = create\_engine('oracle+cx\_oracle://root:1234567@testdb')



**PostgreSQL**

For postgreSQL dialect, connect through ***SQLAlchemy***, we have three DBAPI- *default, psycopg2* and *pg8000*.

*# default*  
engine = create\_engine('postgresql://root:1234567@localhost/mydatabase')  
  
*# psycopg2*  
engine = create\_engine('postgresql+psycopg2://root:1234567@localhost/mydatabase')  
  
*# pg8000*  
engine = create\_engine('postgresql+pg8000://root:1234567@localhost/mydatabase')