How to Connect Python to SQL Server using pyodbc

Python / June 12, 2021

Need to connect Python to SQL Server using *pyodbc?*

If so, you'll see the full steps to establish this type of connection using a simple example.

To start, here is a template that you can use to connect Python to SQL Server:

```
import pyodbc
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=server_name;'
                      'Database=database_name;'
                      'Trusted_Connection=yes;')

cursor = conn.cursor()
cursor.execute('SELECT * FROM database_name.table')

for row in cursor:
    print(row)
```

The Example to be Used

Let's review an example, where:

- The Server Name is: **RON\SQLEXPRESS**
- The Database Name is: **TestDB**
- The Table Name (with a dbo schema) is: **dbo.Person**
- The dbo.Person table contains the following data:

Name	**Age**	**City**
Jade	20	London
Mary	119	NY
Martin	25	London
Rob	35	Geneva
Maria	42	Paris
Jon	28	Toronto

Steps to Connect Python to SQL Server using pyodbc

Step 1: Install pyodbc

First, you'll need to install the *pyodbc* package which will be used to connect Python to SQL Server.

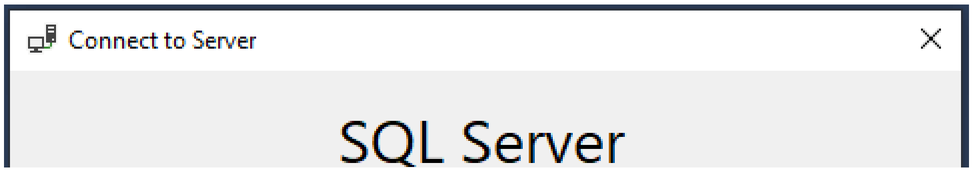
You can use PIP to install the pyodbc package:

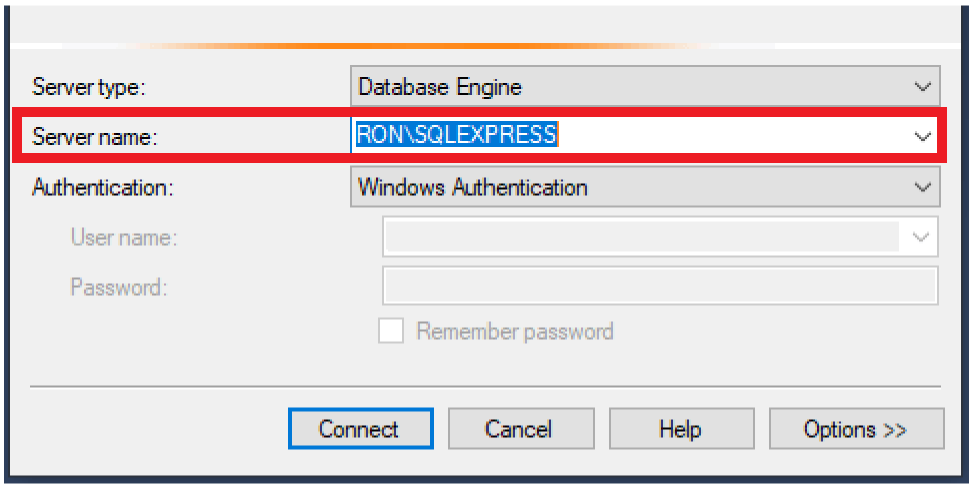
```
pip install pyodbc
```

Step 2: Retrieve the server name

Now retrieve your server name.

In the example below, the server name is: **RON\SQLEXPRESS**




Server type: Database Engine
Server name: RON\SQLEXPRESS
Authentication: Windows Authentication
User name:
Password:
Remember password

One way to find your current server name is by running the following query:

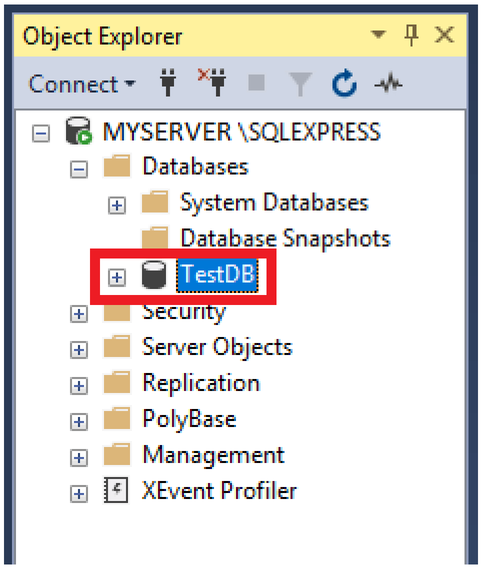
```
SELECT @@SERVERNAME
```

Step 3: Obtain the database name

Next, obtain the database name in which your desired table is stored.

You can find the database name under the *Object Explorer* menu (underneath the *Databases* section), which is located on the left side of your SQL Server.

In our example, the database name is: **TestDB**

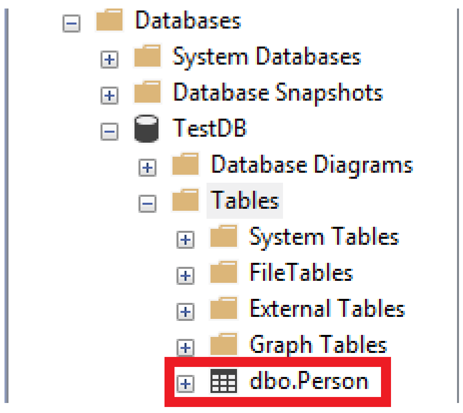


Step 4: Get the table name

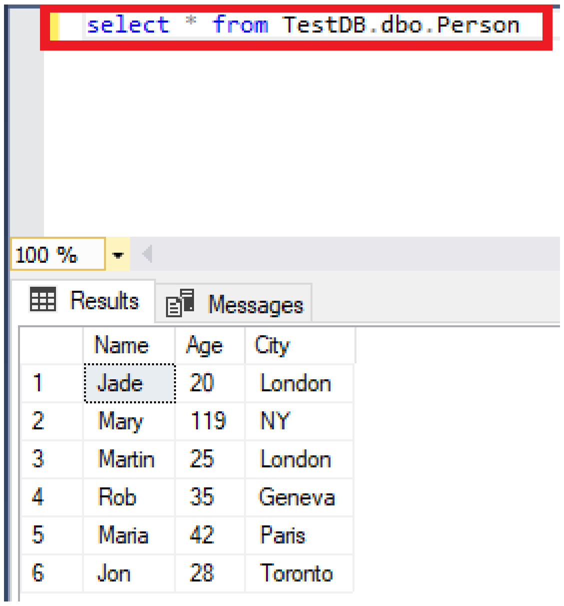
Now you'll need to get the name of your desired table.

The name of your table would also be located under the *Object Explorer* menu (underneath the *Tables* section).

Here, the name of the table is: **dbo.Person**



The following data will be displayed in SQL Server when running a simple SELECT query using the **dbo.Person** table. This is also the data that you'll get once you connect Python to SQL Server using pyodbc.



```
select * from TestDB.dbo.Person
```

	Name	Age	City
1	Jade	20	London
2	Mary	119	NY
3	Martin	25	London
4	Rob	35	Geneva
5	Maria	42	Paris
6	Jon	28	Toronto

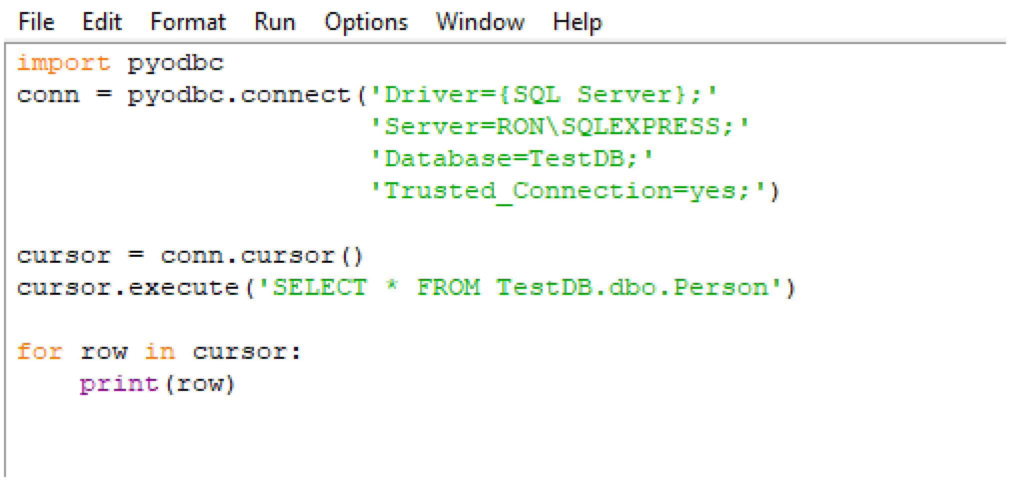
Step 5: Connect Python to SQL Server

And for the final part, open your Python IDLE and fill the server name, database and table information.

Here is the structure of the code that you may use in Python:

```
import pyodbc
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=server_name;'
                      'Database=database_name;'
                      'Trusted_Connection=yes;')

cursor = conn.cursor()
cursor.execute('SELECT * FROM database_name.table')

for row in cursor:
    print(row)
```

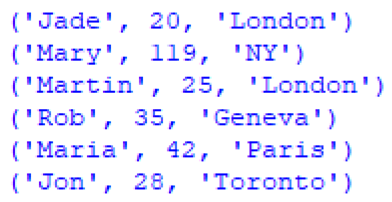
And this is how the code would look like in Python for our example:



```
import pyodbc
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=RON\SQLEXPRESS;'
                      'Database=TestDB;'
                      'Trusted_Connection=yes;')

cursor = conn.cursor()
cursor.execute('SELECT * FROM TestDB.dbo.Person')

for row in cursor:
    print(row)
```

Run the code in Python (adjusted to your server name, database and table information).

You'll notice that the results that were printed in Python match with the info that was displayed in SQL Server:



```
('Jade', 20, 'London')
('Mary', 119, 'NY')
('Martin', 25, 'London')
('Rob', 35, 'Geneva')
('Maria', 42, 'Paris')
('Jon', 28, 'Toronto')
```

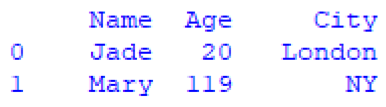
From SQL to Pandas DataFrame

You can take things further by going from SQL to Pandas DataFrame using **pd.read_sql_query**:

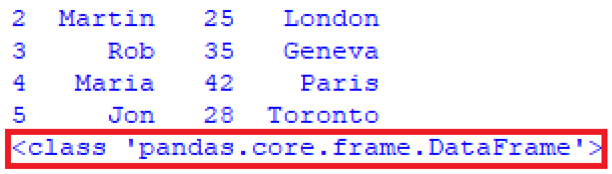
```
import pandas as pd
import pyodbc
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=RON\SQLEXPRESS;'
                      'Database=TestDB;'
                      'Trusted_Connection=yes;')

cursor = conn.cursor()

sql_query = pd.read_sql_query('SELECT * FROM TestDB.dbo.Perso
print(sql_query)
print(type(sql_query))
```

When applying **pd.read_sql_query,** don't forget to place the connection string variable at the end. In our case, the connection string variable is **conn**.

Once you run the code (adjusted to your database connection information), you'll get the following Pandas DataFrame:



```
     Name  Age    City
0    Jade   20  London
1    Mary  119      NY
```



```
2  Martin   25   London
3     Rob   35   Geneva
4   Maria   42    Paris
5     Jon   28  Toronto
<class 'pandas.core.frame.DataFrame'>
```

Note that the syntax of **print(type(sql_query))** was also added to the code to confirm that now we've got a DataFrame.

Conclusion and Additional Resources

You have seen how to connect Python to SQL Server. Once you established such a connection between Python and SQL Server, you can start *using SQL in Python* to manage your data.

You can also use Python to insert values into SQL Server table.

For further information about the *pyodbc* package, please visit the pyodbc documentation.

Convert Python List to a NumPy Array

How to Convert NumPy Array to a List in Python