SQL to Pandas DataFrame (with examples)

Python / May 28, 2019

In this tutorial, I'll show you how to get from SQL to pandas DataFrame using an example.

For illustration purposes, I created a simple database using MS Access, but the same principles would apply if you're using other platforms, such as MySQL, SQL Server, or Oracle.

Steps to get from SQL to Pandas DataFrame

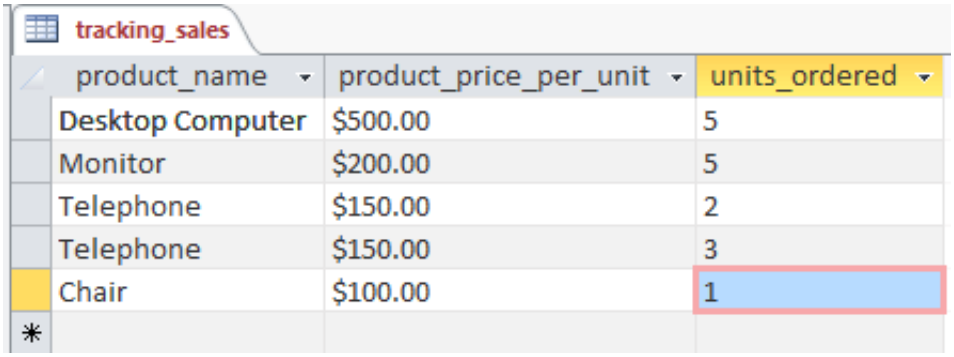
Step 1: Create a database

Initially, I created a database in MS Access, where:

- The database name is: **testdb**
- The database contains a single table called: **tracking_sales**
- The tracking_sales table has 3 fields with the following information:

product_name	product_price_per_unit	units_ordered
Desktop Computer	$500.00	5
Monitor	$200.00	5
Telephone	$150.00	2
Telephone	$150.00	3
Chair	$100.00	1

This is how the table would look like in MS Access:



tracking_sales

product_name	product_price_per_unit	units_ordered
Desktop Computer	$500.00	5
Monitor	$200.00	5
Telephone	$150.00	2
Telephone	$150.00	3
Chair	$100.00	1

Step 2: Connect Python to MS Access

Next, I established a connection between Python and MS Access using the pyodbc package.

Below is the Python code that can be used to connect Python to MS Access. Note that you'll need to modify the connection string to reflect the location where your Access database is stored on *your* computer.

In my case, the Access database is stored under this path:

C:\Users\Ron\Desktop\testdb.accdb

And here is the connection string that I used to connect Python to MS Access:

```
import pyodbc
conn = pyodbc.connect(r'Driver={Microsoft Access Driver (*.m
db, *.accdb)};DBQ=C:\Users\Ron\Desktop\testdb.accdb;')
```

Where **conn** is the connection string variable. We will need to use it in the next step.

Still have questions about establishing a connection between Python and MS Access?

If so, check the following source that explains the full steps to connect Python to MS Access.

Step 3: Write the SQL query

Now for the artistic part.

At this point, I wrote the SQL query, where I added the original 3 fields of:

- product_name
- product_price_per_unit
- units_ordered

I also created a new field called the *revenue* field, which is a function of the units_ordered multiplied by the product_price_per_unit:

*((units_ordered) * (product_price_per_unit)) AS revenue*

Here is the SQL query in Python:

```
SQL_Query = pd.read_sql_query(
'''select
product_name,
product_price_per_unit,
units_ordered,
((units_ordered) * (product_price_per_unit)) AS revenue
from tracking_sales''', conn)
```

Don't forget to put the connection string variable at the end. In our case, the connection string variable is **conn**.

So far, the code would look like this:

```
import pyodbc
import pandas as pd

conn = pyodbc.connect(r'Driver={Microsoft Access Driver (*.m
db, *.accdb)};DBQ=C:\Users\Ron\Desktop\testdb.accdb;')

SQL_Query = pd.read_sql_query(
'''select
product_name,
product_price_per_unit,
units_ordered,
((units_ordered) * (product_price_per_unit)) AS revenue
from tracking_sales''', conn)
```

Step 4: Assign the fields into the DataFrame

In the final step, I assigned the fields from the MS Access database into the DataFrame using this template:

```
df = pd.DataFrame(SQL_Query, columns=['field1','field2',...]
)
```

And for our example:

```
df = pd.DataFrame(SQL_Query, columns=['product_name','produc
t_price_per_unit','units_ordered','revenue'])
```

Here is the complete Python code to get from SQL to Pandas DataFrame:

```
import pyodbc
import pandas as pd

conn = pyodbc.connect(r'Driver={Microsoft Access Driver (*.m
db, *.accdb)};DBQ=C:\Users\Ron\Desktop\testdb.accdb;')

SQL_Query = pd.read_sql_query(
'''select
product_name,
product_price_per_unit,
units_ordered,
((units_ordered) * (product_price_per_unit)) AS revenue
from tracking_sales''', conn)

df = pd.DataFrame(SQL_Query, columns=['product_name','produc
t_price_per_unit','units_ordered','revenue'])
print (df)
```

Run the code (after adjusting the connection string based on your needs), and you would get the following DataFrame in Python:

	product_name	product_price_per_unit	units_ordered	revenue
0	Desktop Computer	$500.00	5	2500.0
1	Monitor	$200.00	5	1000.0
2	Telephone	$150.00	2	300.0
3	Telephone	$150.00	3	450.0
4	Chair	$100.00	1	100.0

Once you assign the fields from your database to the DataFrame, you'll be able to utilize the

true power of pandas.

For example, you may execute statistical analysis, create charts, apply machine learning and so on.

To demonstrate a simple concept, let's see how to find the maximum revenue using pandas.

Finding the maximum value using pandas

To find the maximum revenue, simply add the following syntax:

```
max1 = df['revenue'].max()
print (max1)
```

Putting everything together:

```
import pyodbc
import pandas as pd

conn = pyodbc.connect(r'Driver={Microsoft Access Driver (*.m
db, *.accdb)};DBQ=C:\Users\Ron\Desktop\testdb.accdb;')

SQL_Query = pd.read_sql_query(
'''select
product_name,
product_price_per_unit,
units_ordered,
((units_ordered) * (product_price_per_unit)) AS revenue
from tracking_sales''', conn)
```

```
df = pd.DataFrame(SQL_Query, columns=['product_name','produc
t_price_per_unit','units_ordered','revenue'])

max1 = df['revenue'].max()
print (max1)
```

And once you run the code (adjusted to your connection string), you'll get the maximum revenue of *2500*.

Previous Post

Next Post

Tutorials

Python Tutorials

R Tutorials

Julia Tutorials

Batch Scripts Tutorials

MS Access Tutorials

Excel Tutorials