

Using SQL with Pandas



<https://github.com/learn-co-curriculum/dsc-using-sql-with-pandas> <https://github.com/learn-co-curriculum/dsc-using-sql-with-pandas/issues/new/choose>

Introduction

Consider the structure of a **Pandas DataFrame**.

| | country | description | designation | points | price | province | region_1 | region_2 | taster_name | taster_twitter_handle | title | variety | winery |
|---|----------|---|------------------------------------|--------|-------|-------------------|---------------------|-------------------|--------------------|-----------------------|---|----------------|---------------------|
| 0 | Italy | Aromas include tropical fruit, broom, brimston... | Vulkà Bianco | 87 | NaN | Sicily & Sardinia | Etna | NaN | Kerin O'Keefe | @kerinokeefe | Nicosia 2013 Vulkà Bianco (Etna) | White Blend | Nicosia |
| 1 | Portugal | This is ripe and fruity, a wine that is smooth... | Avidagos | 87 | 15.0 | Douro | NaN | NaN | Roger Voss | @vossroger | Quinta dos Avidagos 2011 Avidagos Red (Douro) | Portuguese Red | Quinta dos Avidagos |
| 2 | US | Tart and snappy, the flavors of lime flesh and... | NaN | 87 | 14.0 | Oregon | Willamette Valley | Willamette Valley | Paul Gregutt | @paulgwine | Rainstorm 2013 Pinot Gris (Willamette Valley) | Pinot Gris | Rainstorm |
| 3 | US | Pineapple rind, lemon pith and orange blossom ... | Reserve Late Harvest | 87 | 13.0 | Michigan | Lake Michigan Shore | NaN | Alexander Peartree | NaN | St. Julian 2013 Reserve Late Harvest Riesling ... | Riesling | St. Julian |
| 4 | US | Much like the regular bottling from 2012, this... | Vintner's Reserve Wild Child Block | 87 | 65.0 | Oregon | Willamette Valley | Willamette Valley | Paul Gregutt | @paulgwine | Sweet Cheeks 2012 Vintner's Reserve Wild Child... | Pinot Noir | Sweet Cheeks |

Now, let's consider the structure of a table from a **SQL database**.

| IdNum | LName | FName | JobCode | Salary | Phone |
|-------|------------|---------|---------|--------|--------------|
| 1876 | CHIN | JACK | TA1 | 42400 | 212/588-5634 |
| 1114 | GREENWALD | JANICE | ME3 | 38000 | 212/588-1092 |
| 1556 | PENNINGTON | MICHAEL | ME1 | 29860 | 718/383-5681 |
| 1354 | PARKER | MARY | FA3 | 65800 | 914/455-2337 |
| 1130 | WOOD | DEBORAH | PT2 | 36514 | 212/587-0013 |

You've probably noticed by now that they're essentially the same--a table of values, with each row having a unique index and each column having a unique name. This allows us to quickly and easily access information when using SQL. In this section, we'll learn how we can use SQL-style queries to query pandas DataFrames!

You will be able to:

- Compare accessing data in a DataFrame using query methods and conditional logic
- Query DataFrames with SQL using the `pandasql` library

Using `.query()`

Pandas DataFrames come with a built-in query method, which allows you to get information from DataFrames quickly without using the cumbersome slicing syntax.

See the following examples:

```
# Getting Data using slicing syntax
foo_df = bar_df[bar_df[bar_df['Col_1'] > bar_df['Col_2']]]


# Using The query method
foo_df = bar_df.query("Col_1 > Col_2")

# These two lines are equivalent!
```

Note that if you want to use `and` and `or` statements with the `.query()` method, you'll need to use `"&"` and `"|"` instead.

```
foo_df = bar_df.query("Col_1 > Col_2 & Col_2 <= Col_3")
```

Using SQL syntax with `pandasql`

Since SQL is such a powerful, comfortable tool for Data Scientists, some people had the bright idea of creating a library that lets users query DataFrames using SQL-style syntax. This library is called [pandasql](https://pypi.org/project/pandasql/)  (<https://pypi.org/project/pandasql/>).

We can install `pandasql` using the bash command `pip install pandasql`.

Importing `pandasql`

In order to use `pandasql`, we need to start by importing a `sqldf` object from `pandasql`

```
from pandasql import sqldf
```

Next, it's helpful to write a lambda function that will make it quicker and easier to write queries. Normally, you would have to pass in the global variables every time we use an object. In order to avoid doing this every time, here's how to write a lambda that does this for you:

```
 Help sqldf(q, globals())
```

Now, when you pass a query into `pysqldf`, the lambda will also pass along the globals, saving you that repetitive task.

Writing Queries

To write a query, you just format it as a multi-line string!

```
q = """SELECT
    m.date, m.beef, b.births
FROM
    meats m
INNER JOIN
    births b
    ON m.date = b.date;"""
```

In order to query DataFrames, you can just pass in the query string you've created to our `sqldf` object that you stored in `pysqldf`. This will return a DataFrame.

```
results = pysqldf(q)
```

Summary

These advanced methods for querying DataFrames can make your life a lot easier by simplifying the syntax and allowing us to make use of SQL--use them to save yourself time and give keep your SQL skills strong!

How do you feel about this lesson?



Have specific feedback?

[Tell us here!](https://github.com/learn-co-curriculum/dsc-using-sql-with-pandas/issues/new/choose) (<https://github.com/learn-co-curriculum/dsc-using-sql-with-pandas/issues/new/choose>)