pandas-to-sql — Moving code from SQL to Python and Pandas

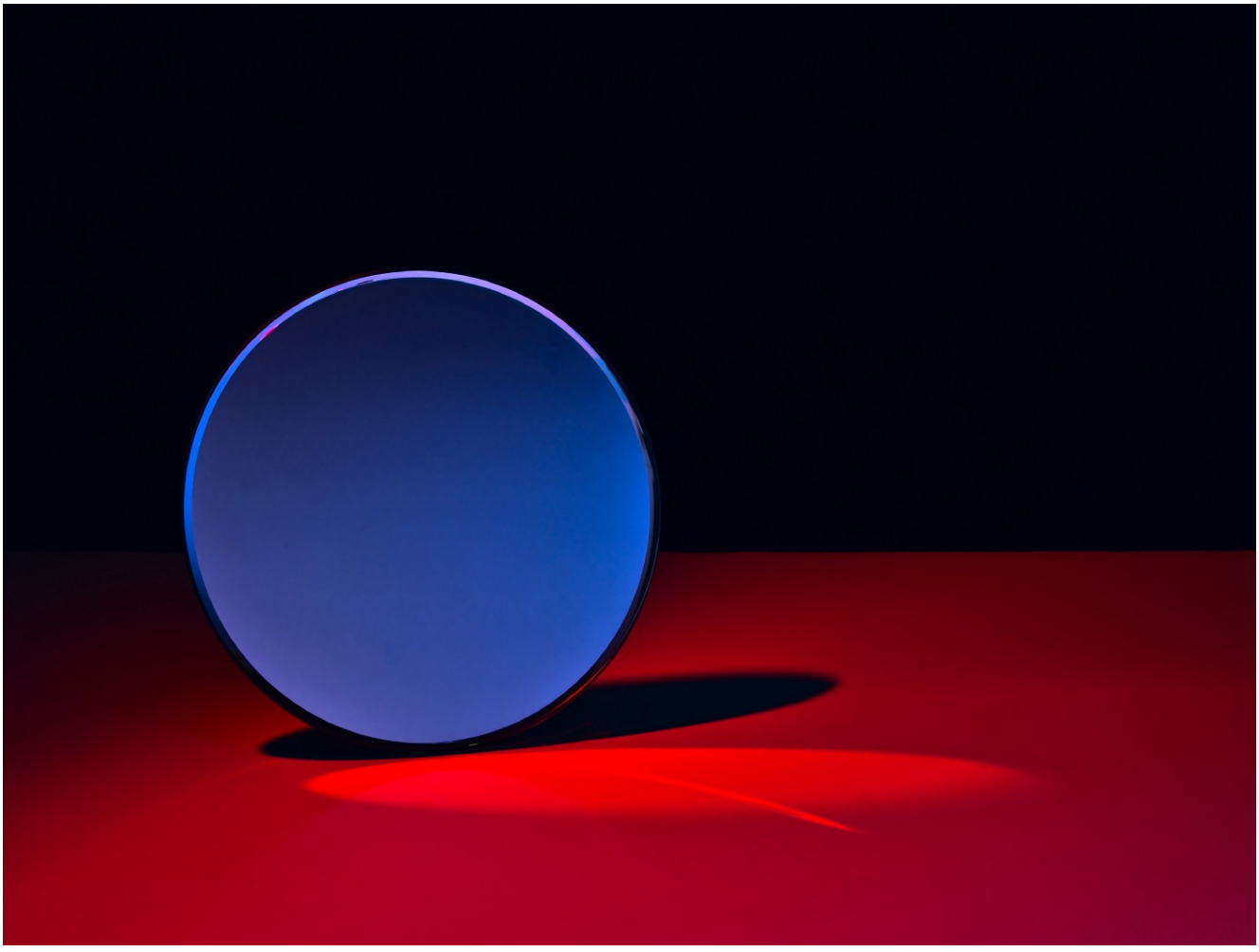

Photo by Michael Dziedzic on Unsplash

pandas-to-sql is a python library, allowing the use of python's Pandas DataFrames for creating SQL strings that can later be used to query a database. Here are some examples you can run.

A common structure of a data science project will start from the data fetching and preparation phase. In case the database that holds the data supports SQL, the data

fetching or even some of the data preparation may be executed using SQL (supported by the DB, why not?)

Issues with that:

- Maintaining code in 2 different languages (SQL and Python)
- The project becomes a 2-steps project (SQL part, then the Python part). It may be a choice of design, but it can also be a side effect of the 2 coding languages in the project
- Harder to use coding principles and best practices (e.g SOLID) on SQL code
- Usually, SQL code is not tested

How do pandas-to-sql try to solve those issues?

pandas-to-sql is a python library allowing users to use Pandas DataFrames, create different manipulations, and eventually use the **get_sql_string()** method to get a SQL string describing the manipulations. At this point, one can use this string to query the database.

How to install:

```
pip install pandas-to-sql
```

Simple example:

Output:

```
‘SELECT (sepal_length) AS sepal_length, (sepal_w
```

In the above example, we load the iris dataset into a Pandas DataFrame. We then convert the DataFrame using **pandas_to_sql.wrap_df**. From this point on all manipulations will be saved as a SQL string. At any point, one can get this SQL string using **df.get_sql_string()**.

Here are some more examples...

Filtering example:

Output:

```
‘SELECT (sepal_length) AS sepal_length, (sepal_w
```

Grouping example:

Output:

```
‘SELECT (avg(petal_length)) AS petal_length_mean,
```

Note the use of **conventions.flatten_grouped_dataframe()**. Since Pandas groupby returns a multi-index DataFrame, which is not applicable for SQL, this method edits the column names in the SQL query to match the default that **reset_index()** returns (and performs a **reset_index()** on the DataFrame returned from the groupby).

Concatenation example:

Output:

```
‘SELECT (sepal_length) AS sepal_length, (sepal_w
```

Note here the wrapping of pd **pandas_to_sql.wrap_pd(pd)**, in order to override **pd.concat()**

Here are all examples in a single Gist:

This library is not production-ready yet. My main goal is to check the amount of attraction for this idea and whether more people find this useful. If so, more support will be added. Current support is for SQLite only, hoping to add support for Presto soon. Feel free to open issues for any bug or new feature request. Issues

Thanks for reading :-)