Jeremy Beard – MSDS 610 – 20220605 – Week 5 Assignment

What is an API, and what can we use them for?

An API is short for an Application Programming Interface, and it is a tool or package used to provide a way to access certain applications or capabilities. It is used to provide an interface between two technologies. Some ways an API can be used include: logging into a website account using Gmail, Facebook, etc., paying for a good or service using Paypal, accessing weather from an area, and posting to social media (ie bots). APIs are omnipresent in our everyday lives and are used all the time by companies or services which need to access specific parts of another service or application.

When should we consider putting API-fetched data in SQL vs a NoSQL database?

There are different use cases for when a sample of data coming from an API should be put into a SQL database vs. a NoSQL database. SQL databases are ideal for when the API data is organized and may even come with a pre-defined schema definition. This would make it easier to structure the data for the SQL format. The advantages of using SQL are that the queries can be very powerful compared to NoSQL, and the platforms are usually acid-compliant.

Data coming from an API should be put into a NoSQL database when the schema varies and changes, and is more unstructured. A NoSQL database can be scaled horizontally as compared to a SQL database which scales vertically. This makes NoSQL a bit less expensive when the need to scale comes into the picture. There can be different types of NoSQL such as document-based or key-value pair based.

What was challenging about using APIs?

Someone of the challenges with using APIs are the following: where exactly to start with the API, deciding the scope and the specific use of the API, choosing what data to pull and track from the API usage, and tying the API to specific business objectives. APIs can be used almost anywhere which makes choosing a starting point very difficult to precisely choose. IT groups can also have a difficult time tying the API development to that actual business strategy of the company which could make some parts of the API irrelevant and not used very often. Deciding on this scope can be very difficult.

<u>Technical Section</u>

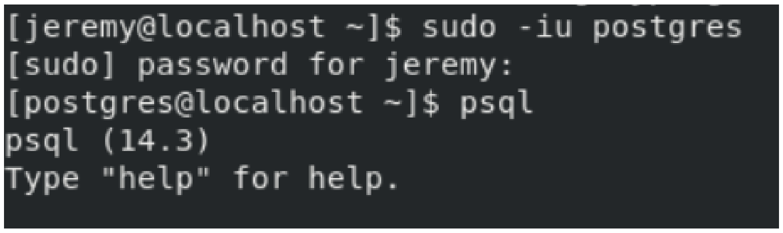
For the technical section of this assignment, I utilized Jupyter Notebooks and have uploaded the .ipynb and a pdf file of the Jupyter Notebook. In this pdf there is all code used as well as explanations for each of the steps. Please let me know if there are any questions!

In the section below I have provided screenshots of any steps performed outside of Jupyter Notebooks, which in this case is during the MISO + PostgreSQL part of the assignment.

Using the command line in the PostgreSQL server, I created a database called 'miso' and created a table within it called 'rt_solar' (I also created a table called 'rt_wind' to ensure my understanding of the example of the assignment).

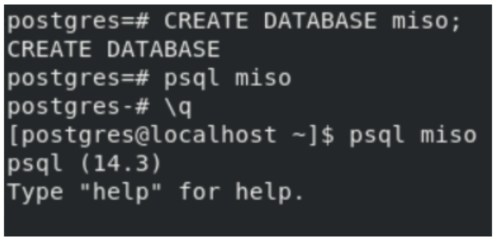
For the Python PostgreSQL work to function, I had to install the pip package sqlalchemy. I have noted this installation below.

After using Python to populate the rt_solar table, I then queried the table using the command line in the PostgreSQL server. The screenshot of that table is shown below.



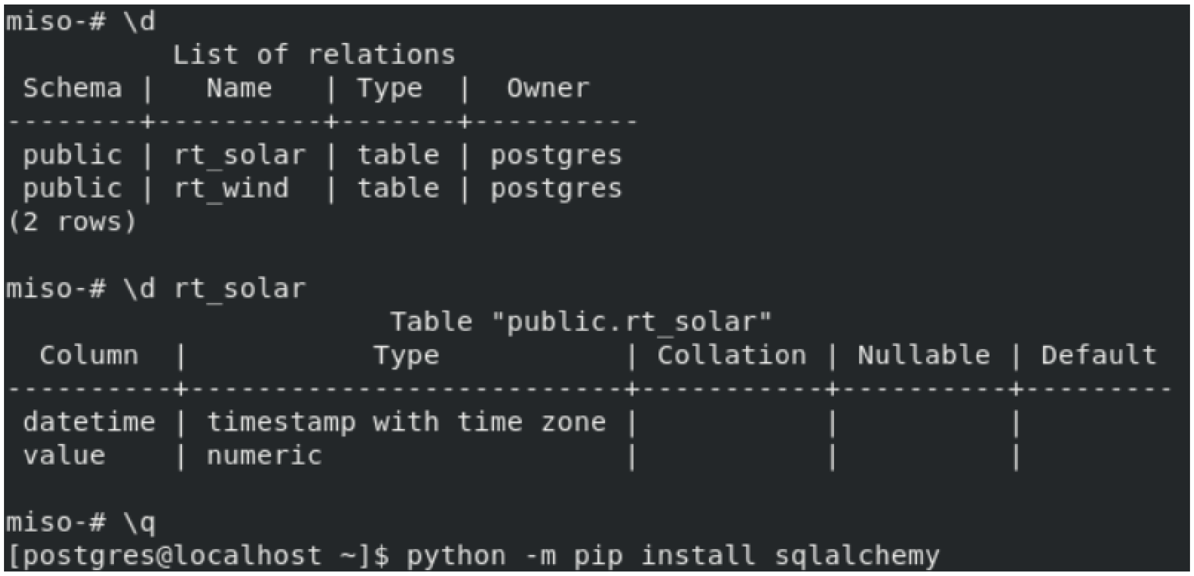
```
[jeremy@localhost ~]$ sudo -iu postgres
[sudo] password for jeremy:
[postgres@localhost ~]$ psql
psql (14.3)
Type "help" for help.
```

Logging into the PostgreSQL server



```
postgres=# CREATE DATABASE miso;
CREATE DATABASE
postgres=# psql miso
postgres-# \q
[postgres@localhost ~]$ psql miso
psql (14.3)
Type "help" for help.
```

Creating the database ‘miso’ in the PostgreSQL server

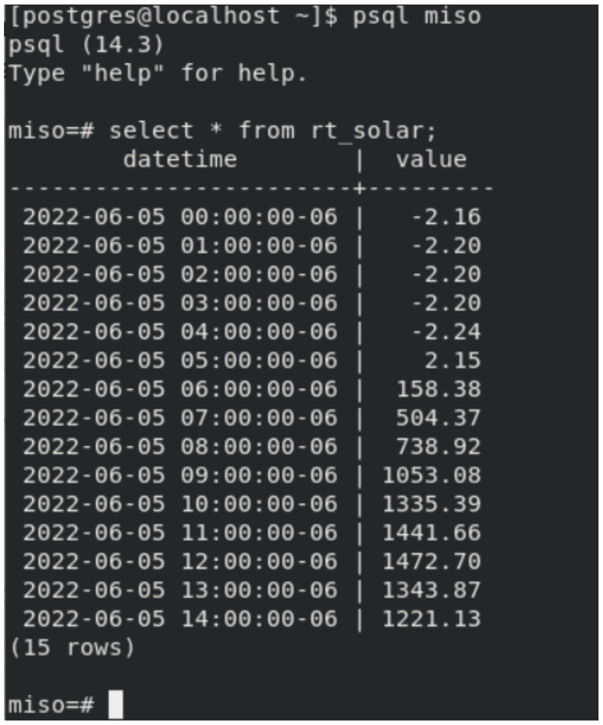


```
miso-# \d
         List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | rt_solar | table | postgres
 public | rt_wind  | table | postgres
(2 rows)

miso-# \d rt_solar
                      Table "public.rt_solar"
  Column  |           Type           | Collation | Nullable | Default
----------+--------------------------+-----------+----------+---------
 datetime | timestamp with time zone |           |          |
 value    | numeric                  |           |          |

miso-# \q
[postgres@localhost ~]$ python -m pip install sqlalchemy
```

Querying the rt_solar table after performing the “CREATE TABLE IF NOT EXISTS re_solar” command in the Jupyter Notebook (see Jupyter Notebook documentation)



```
[postgres@localhost ~]$ psql miso
psql (14.3)
Type "help" for help.

miso=# select * from rt_solar;
        datetime        |  value
------------------------+---------
 2022-06-05 00:00:00-06 |   -2.16
 2022-06-05 01:00:00-06 |   -2.20
 2022-06-05 02:00:00-06 |   -2.20
 2022-06-05 03:00:00-06 |   -2.20
 2022-06-05 04:00:00-06 |   -2.24
 2022-06-05 05:00:00-06 |    2.15
 2022-06-05 06:00:00-06 |  158.38
 2022-06-05 07:00:00-06 |  504.37
 2022-06-05 08:00:00-06 |  738.92
 2022-06-05 09:00:00-06 | 1053.08
 2022-06-05 10:00:00-06 | 1335.39
 2022-06-05 11:00:00-06 | 1441.66
 2022-06-05 12:00:00-06 | 1472.70
 2022-06-05 13:00:00-06 | 1343.87
 2022-06-05 14:00:00-06 | 1221.13
(15 rows)

miso=# 
```

After populating the rt_solar table in the Jupyter Notebook (see the Jupyter Notebook submitted), querying the rt_solar table using the command line in the PostgreSQL server

Thank you!

Jeremy Beard

References

1) https://www.misoenergy.org/markets-and-operations/RTDataAPIs/
2) Bush, T. (2019, December 10). Smarter Tech Decisions Using APIs. Nordic APIs. Retrieved June 5, 2022, from https://nordicapis.com/5-examples-of-apis-we-use-in-our-everyday-lives/
3) Joshi, V. (2019, September 25). Relational vs. NoSQL Databases for API Traffic. Dzone.Com. Retrieved June 5, 2022, from https://dzone.com/articles/relational-vs-nosql-databases-for-api-traffic-1
4) Iyengar, K., Lau, L., Ramadath, S., & Sohoni, V. (2020, February 13). The seven make-or-break API challenges CIOs need to address. McKinsey & Company. Retrieved June 5, 2022, from https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-seven-make-or-break-api-challenges-cios-need-to-address