In the Week 4 assignment for MSDS 610, we set out to answer various questions about NoSQL databases, including their comparisons to SQL. In the technical section, we used MongoDB, both inside and outside of Python, in order to examine a database that included information on Los Angeles parking tickets.

Why is NoSQL useful? What are use cases for NoSQL?

NoSQL, or Not-only Structured Query Language, is useful because it is an alternative to SQL. NoSQL provides higher performance, increased flexibility, and enhanced scalability relative to SQL. Most data solutions can utilize a SQL solution but when it comes to situations where users can enter millions of transactions per second, a NoSQL solution becomes much more effective as it is both horizontally and vertically scalable, as opposed to SQL which is only vertically scalable. Other use cases for NoSQL are big data solutions, unstructured data, semi-structured data.

What are some of the different NoSQL databases, and in what cases are they useful?

Some popular NoSQL databases are MongoDB, DocumentDB, Cassandra, Couchbase, HBase, Redis, and Neo4j.

MongoDB

Free/open-source, document-oriented databases.

DocumentDB

Amazon-owned, cloud database environment. JSON-based. Proprietary service / paid service.

Cassandra

Created by Apache, free/open-source, widely used, highly available.

Couchbase

NoSQL cloud database, used for enterprise applications. Low cost, can be well-suited for internet of things (IoT) data management.

HBase

Built on Hadoop. Open-source, based on Google's Big Table, written in Java.

What are pros and cons of NoSQL vs SQL?

The pros of NoSQL are that it is highly scalable and works well with unstructured and semi-structured data. Another pro is that NoSQL tools are often open source/free and don't necessarily need the use of a schema.

Some cons of NoSQL vs. SQL are that it has a weaker eventual consistency (BASE) vs. SQL, and because it can work so well with unstructured data, 'joins' are not always supported. There is also limited indexing and the integrity of the data is not built-in which can require frequent updates.

What did you find from the MongoDB queries? (e.g. what were the most expensive tickets)

From the MongoDB queries in this assignment, I found that the most expensive tickets were related to violations of handicapped-related parking regulations. Now I know in the future that if there is something to remember with regard to parking rules, definitely respect all handicapped signs!

There was one ticket which had an ambiguous reason, "8755**" which didn't make much sense to me. Other than this reason, all other expensive tickets were related to handicap violations.

Technical Section

To complete the assignment this week, I first installed MongoDB on my RedHat Linux virtual machine. I chose this method as it seemed like the best long-term solution for me personally. My VirtualBox VM's are something I have some experience in and it was easy enough to install using the official MongoDB installation reference.

After installing MongoDB on my Linux VM, I downloaded the LA Parking Citations database using the wget command:

```
wget https://www.dropbox.com/s/nh1h2r8ryimtm7h/los-angeles-parking-citations.zip
```

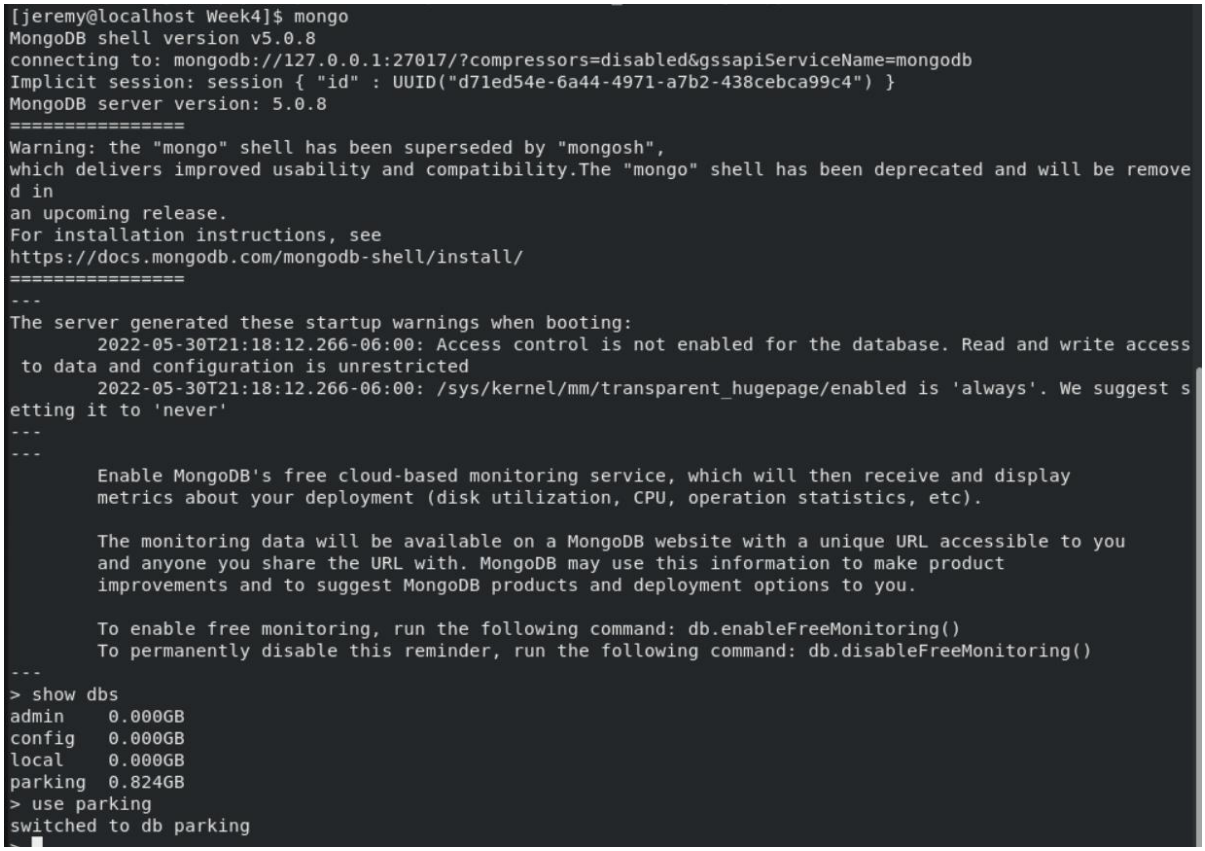
After unzipping the database file, I imported the database using the mongoimport command:

```
sudo mongoimport -d parking -c tickets --type csv --file parking-citations.csv –
                                    headerline
```

After importing the database, I set out to answer the following prompts:

1. What are the amounts and violation descriptions for the top 10 most expensive tickets?
2. Use Python to connect to the MongoDB.
3. Within Python, extract the entries from the DB where the license plate is not CA.
4. Create a sorted bar chart of the states from #3.

Firstly, I had to start mongo and then switch to the imported database:



Then, I started on the prompts:

What are the amounts and violation descriptions for the top 10 most expensive tickets?

Using the following command, I was able to find the most expensive tickets and their descriptions:

```
db.tickets.find({'Fine amount': {'$exists': true, '$ne': ''}}, {'_id': -1, 'Ticket number': 1, 'Fine amount': 1, 'Violation Description': 1}).sort({'Fine amount': -1}).limit(10).pretty();
```

The most difficult part of this was filtering the fine amounts that were empty, or “”. I eventually figured out how to set the collection to only return what fine amount was not equal to “”.

After completing this part of the assignment, I realized I should change “ ‘_id’: -1” to “ ‘_id’: 0” and that would remove the id attribute as we didn’t need this.

```
> db.tickets.find({'Fine amount': {'$exists': true, '$ne': ''}}, {'_id':-1, 'Ticket number': 1, 'Fine amount':1, 'Violation De
scription':1}).sort({'Fine amount': -1}).limit(10).pretty();
{
        "_id" : ObjectId("62952839cd410252a45ab66e"),
        "Ticket number" : 1109907271,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("62952839cd410252a45a8ea1"),
        "Ticket number" : 1109908925,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("62952839cd410252a45a8ea8"),
        "Ticket number" : 1109908995,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("6295282ccd410252a45413d7"),
        "Ticket number" : 1109909220,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("6295282ccd410252a45413d5"),
        "Ticket number" : 1109909205,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("62952814cd410252a446a5ab"),
        "Ticket number" : 1112999020,
        "Violation Description" : "8755**",
        "Fine amount" : 505
}
{
        "_id" : ObjectId("6295280acd410252a440a5f8"),
        "Ticket number" : 1109139006,
        "Violation Description" : "HANDICAP/NO DP ID",
        "Fine amount" : 363
}
{
        "_id" : ObjectId("6295280acd410252a440a78c"),
        "Ticket number" : NumberLong("4272626536"),
        "Violation Description" : "DISABLED PARKING/NO",
        "Fine amount" : 363
}
{
        "_id" : ObjectId("6295280acd410252a440a60f"),
        "Ticket number" : 1112058835,
        "Violation Description" : "HANDICAP/CROSS HATCH",
        "Fine amount" : 363
}
{
        "_id" : ObjectId("6295280acd410252a440a78d"),
        "Ticket number" : NumberLong("4272626540"),
        "Violation Description" : "DISABLED PARKING/NO",
        "Fine amount" : 363
}
```

It appears that the most expensive tickets were $505 and $363 and were for the following reasons: 8755**, HANDICAP/NO DP ID, HANDICAP/CROSS HATCH, and DISABLED PARKING/NO. These descriptions are not the most helpful in determining exactly what the driver of the car did to deserve the ticket but they appear to be mostly related to violating handicapped regulations.

Use Python to connect to the MongoDB.

To connect to the MongoDB using Python, I simply installed pymongo using pip, and then used the following lines of code within Jupyter Notebooks. I didn't need to include the last line of code as by that point, I was already connected to the parking database, but I included it here anyway for good measure.

```
from pymongo import MongoClient
```

```
client = MongoClient()
```

```
db = client['parking']
```

```
tickets = db['tickets']
```

Within Python, extract the entries from the DB where the license plate is not CA.

Please see the Jupyter Notebook pdf section of this assignment which goes over the PyMongo data extraction of the parking tickets database.

Create a sorted bar chart of the states from #3.

Equivalently, please see the Jupyter Notebook pdf section of this assignment which goes over creating a sorted bar chart of the extracted states from the previous prompt.

Thank you!

Jeremy Beard

References

1) Install MongoDB Community Edition on Red Hat or CentOS — MongoDB Manual. (n.d.). MongoDB. Retrieved May 30, 2022, from https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-red-hat/
2) Foote, K. D. (2022, February 10). A Review of Different Database Types: Relational versus Non-Relational. DATAVERSITY. Retrieved May 30, 2022, from https://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/#