Assessment 2

Assessment 2 assessed the strengths and weaknesses of Sqlite and Pandas for loading and calculating basic statistics on the uncorrupted parking citations data.

Uncorrupted data was separated from the original parking citations data prior to loading, resulting in a total of 4357535 rows.

Pandas and Sqlite were evaluated on the following tasks:

1. Loading data.
2. Calculate top 25 most common makes.
3. Calculate most common Color for each Make.
4. Find the first ticket issued for each Make.

Following performance and ease-of-use testing, Pandas was chosen to answer the following question:

> "Is an out-of-state license plate more likely to be expired at the time of receiving a ticket than an in-state license
> plate?"

Pandas was chosen mainly for its significant performance advantage over Sqlite.

The analysis was carried out in the following two notebooks:

1. notebooks/LoadIntoSqlite.ipynb
2. notebooks/LoadIntoPandas.ipynb

For the purposes of this experiment the Sqlite table for citations data was created in-memory; in a production setting it would be constructed on disk.

1. Loading Data

Data was loaded into Pandas and Sqlite five times, with the average case running time of the five trial runs recorded.

Sqlite

Loading data into Sqlite took an average of 37.2 seconds over the five trial runs.

Loading the data was a fairly straightforward task, with the column types specified ahead of time, reasonable assumptions made ahead of time, and consistent use of NULL to represent missing values.

Pandas

Loading data into Pandas took an average of 13.5 seconds over the five trial runs.

Loading the data was more complicated than in Sqlite. In particular, dates were not parsed gracefully, and the np.nan value caused Pandas' to_datetime() function to raise an exception when

parsing the Plate_Expiry_Date field. This required significantly more up-front data exploration and coding than loading into Sqlite.

2. Calculating Top 25 Makes

Sqlite

Querying for the top 25 most common car makes took an average of 9.79 seconds over the five trial runs.

The query was simple to write, with a single SELECT and GROUP BY statement.

Pandas

Querying for the top 25 most common car makes took an average of 202 milliseconds over the five trial runs.

The code was simple to write by leveraging a call to the size() method of the Pandas Series representing the citations Make column.

3. Calculating Most Common Color For Each Make

Sqlite

Querying for the top colors for each make took an average of 12.3 seconds over the five trial runs.

The query was somewhat more complicated to write, involving a sub-select statement from which the maximum count was extracted for each group. However, likely a basic knowledge of SQL would have been sufficient to construct the query.

Pandas

Querying for the top 25 most common car makes took an average of 791 milliseconds over the five trial runs.

The query was complicated to code. The data had to be grouped once by Make and Color, the index reset and grouped again by Make, and the idxmax() function used to obtain the most common color. This process was not particularly intuitive and required a significant amount of reference to documentation.

4. Finding First Ticket Isssued For Each Make

Sqlite

Querying for the top colors for each make took an average of 8.52 seconds over the five trial runs.

The query was simple to write, with a single SELECT statement augmented by a single GROUP BY and HAVING clause.

Pandas

Querying for the top 25 most common car makes took an average of 748 milliseconds over the five trial runs.

The query was complicated to code. It required using the idxmin() function to find the earliest date and then "merging" the row index back onto the original data using the loc[] function. This again was unintuitive and very different from task #3.

Out-Of-State Plates

The probability of an out-of-state plate being expired at the time of citation was calculated using the conditional probability formula; which in this case simplified to the number of plates out-of-state and expired divided by the number of plates out-of-state. A similar formula was used to calculate the probability for in-state plates.

For a more accurate comparison, citations where the plate expiry date was null were excluded from the analysis.

The probability for out-of-state plates was about 23.4%, whereas the probability for in-state plates was about 22.0%. In order to test whether or not the samples were significantly different, a hypothesis test was conducted (see notebook for details). Using a one-sided p-test, the null hypothesis that the two probabilities are equal was rejected.

Therefore, I conclude that out-of-state plates are more likely than in-state plates to be expired at the time of citation issuance, albeit by a small amount.

Summary

Pandas was selected as the "superior" tool in this case due to its significant performance benefit. However, Sqlite or another relational database would likely be the better tool in the following scenarios:

1. The data was larger than available memory.
2. The data was part of an ongoing batch load to a data warehouse.
3. The data needed to be available multiple times to different groups.
4. The target audience did not have a Python background and did not have the time to deal with incorrectly formatted data.
5. Access patterns were relatively well-known and did not require complex workflows (e.g. machine learning).