

Chipotle Location Data Management

Nick Videtti

IST-769 Advanced Big Data Management

Final Project

Fall 2022

**Data**

The data used for this project were the data found on Kaggle for Chipotle locations in the United States. The key to this is having the user download the data file to a file on their computer. The data file can be found attached to this assignment or at [https://www.kaggle.com/datasets/jeffreybraun/chipotle-locations](https://www.kaggle.com/datasets/jeffreybraun/chipotle-locations?select=us-states.json).

Two files are included in these data. One of the files is a CSV file with Chipotle store locations, and the other is a JSON file containing locations of state borders for states with Chipotle locations. Both will be used in this project.

**Project Goals**

This project will cover the following topics. Each of these steps will be discussed and brief descriptions/reflections will be included. Full Python code can be found in the Jupyter Notebook attached to this assignment.

1. Configure PySpark session for both MongoDB and Elasticsearch
2. Use PySpark to read in raw data from downloaded files
3. Write raw data to MongoDB, then read it back into PySpark
4. Use PySpark to clean the raw data
5. Write cleaned data to MongoDB, then read it back into PySpark
6. Use PySpark to combine the two cleaned data sets
7. Create PySpark data frames for only coordinate data from the combined cleaned data, the cleaned location data, and the cleaned state borders data
8. Write the data frames created in steps 6 and 7 to MongoDB, then read them back into PySpark
9. Write all 8 data frames to Elasticsearch, then read them back into PySpark
10. Use Drill to query the cleaned location data using a few queries
11. Create a Kibana Map visualization using the coordinate-only location data

**1 - Configure PySpark session for both MongoDB and Elasticsearch**

Configuring a PySpark session for both MongoDB and Elasticsearch involved the following configuration options:

* MongoDB
  + *.config("spark.mongodb.input.uri", "mongodb://admin:mongopw@mongo:27017/admin?authSource=admin") \*
  + *.config("spark.mongodb.output.uri", "mongodb://admin:mongopw@mongo:27017/admin?authSource=admin") \*
  + *.config("spark.jars.packages","org.mongodb.spark:mongo-spark-connector\_2.12:3.0.1")\*
* Elasticsearch
  + *.config("spark.es.nodes", "elasticsearch") \*
  + *.config("spark.es.port","9200") \*

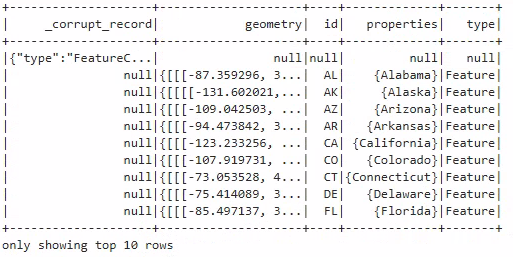
This was different from other assignments during the course due to needing to configure for multiple databases. I first attempted separate PySpark sessions for each of the two databases, then attempted to consolidate to one PySpark session, and ultimately took that latter approach.

(See attached Jupyter Notebook for full code)

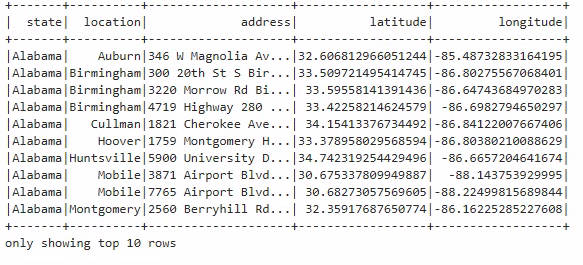
**2 - Use PySpark to read in raw data from downloaded files**

Both of the files were read into PySpark via local system files downloaded from Kaggle. The CSV file needed the “header” option set to “True”. After the data were read in, the first 10 rows of each resulting data frame were displayed to prove that the data were successfully read into PySpark. The JSON file with state borders was named “RAW\_CHIPOTLE\_STATE\_BORDERS” and the CSV file with locations was named “RAW\_CHIPOTLE\_LOCATIONS”.

First 10 rows of RAW\_CHIPOTLE\_STATE\_BORDERS read in from downloaded file



First 10 rows of RAW\_CHIPOTLE\_LOCATIONS read in from downloaded file

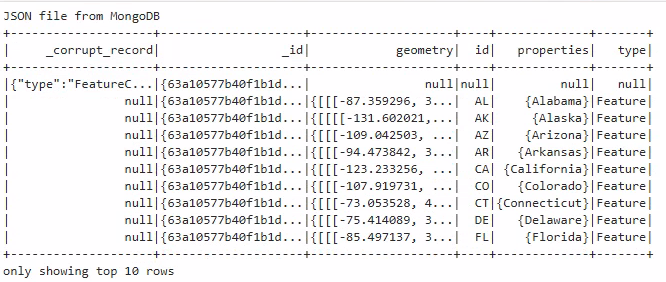


(See attached Jupyter Notebook for full code)

**3 - Write raw data to MongoDB, then read it back into PySpark**

Both RAW\_CHIPOTLE\_STATE\_BORDERS and RAW\_CHIPOTLE\_LOCATIONS were written to MongoDB into a database named “project” under the collections “raw\_chipotle\_state\_borders” and “raw\_chipotle\_locations”, respectively. Then, the data were read back into PySpark from MongoDB and the first 10 rows of each data frame were displayed to prove that the data were successfully read into PySpark.

First 10 rows of RAW\_CHIPOTLE\_STATE\_BORDERS read in from MongoDB



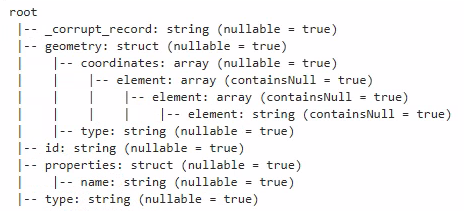
(See attached Jupyter Notebook for full code)

**4 - Use PySpark to clean the raw data**

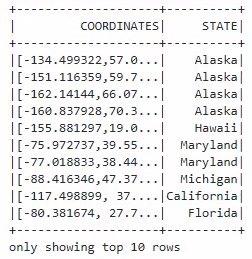
Both data frames were duplicated as temporary views and then Spark SQL was used to generate cleaned datasets, named CHIPOTLE\_STATE\_BORDERS and CHIPOTLE\_LOCATIONS. The following transformations were made.

* RAW\_CHPOTLE\_STATE\_BORDERS -> CHIPOTLE\_STATE\_BORDERS
  + Select only “geometry” and “properties” columns
  + Explode columns until no rows are nested
  + Rename fields “COORDINATES” and “STATE”
* RAW\_CHIPOTLE\_LOCATIONS -> CHIPOTLE\_LOCATIONS
  + Select “state”, “location”, “location” and “, “ and “state” concatenated, “address”, “longitude” and “ - “ and “latitude concatenated, “longitude”, and “latitude”
  + Rename columns listed above to “STATE”, “CITY”, “CITY\_STATE”, “ADDRESS”, “LON”, “LAT”, “LON\_LAT”, respectively

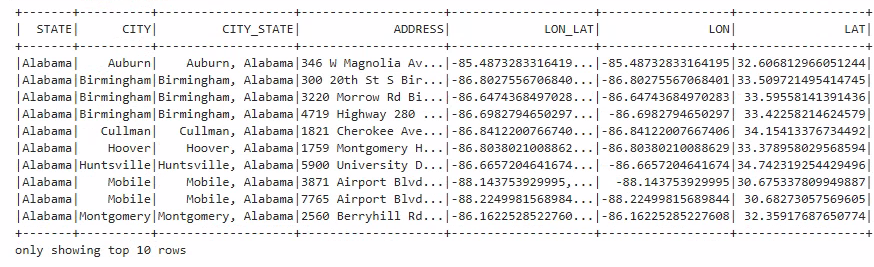
Schema of RAW\_CHIPOTLE\_STATE\_BORDERS



First 10 rows of CHIPOTLE\_STATE\_BORDERS



First 10 rows of CHIPOTLE\_LOCATIONS

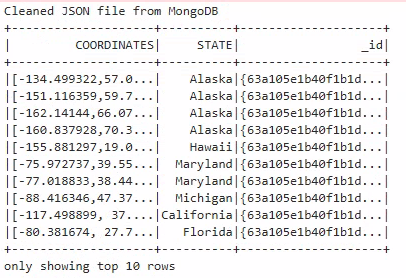


(See attached Jupyter Notebook for full code)

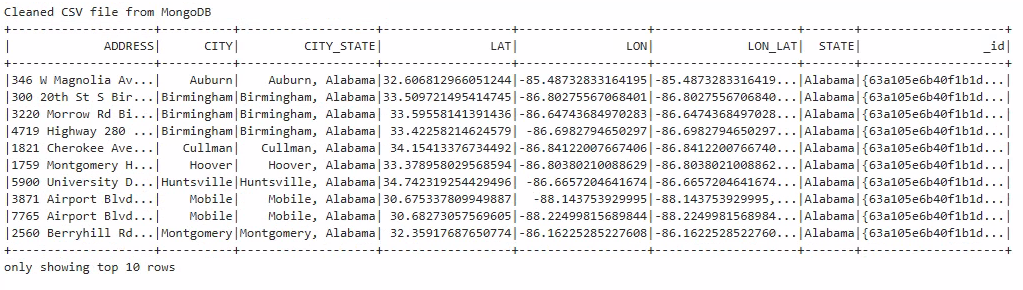
**5 - Write cleaned data to MongoDB, then read it back into PySpark**

Both CHIPOTLE\_STATE\_BORDERS and CHIPOTLE\_LOCATIONS were written to MongoDB into a database named “project” under the collections “chipotle\_state\_borders” and “chipotle\_locations”, respectively. Then, the data were read back into PySpark from MongoDB and the first 10 rows of each data frame were displayed to prove that the data were successfully read into PySpark.

First 10 rows of CHIPOTLE\_STATE\_BORDERS read in from MongoDB



First 10 rows of CHIPOTLE\_LOCATIONS read in from MongoDB

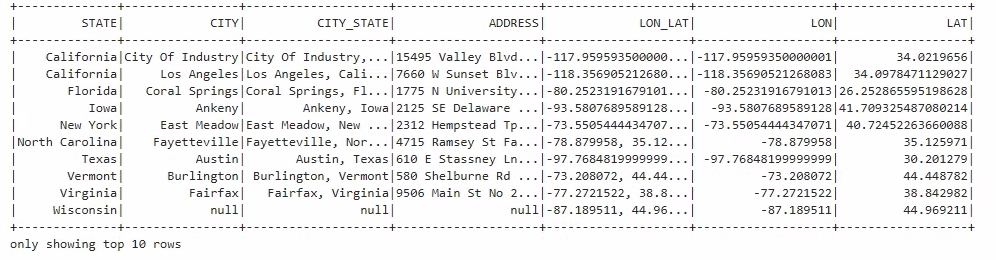


(See attached Jupyter Notebook for full code)

**6 - Use PySpark to combine the two cleaned data sets**

Both CHIPOTLE\_STATE\_BORDERS and CHIPOTLE\_LOCATIONS were duplicated as temporary views. Then, Spark SQL was used to UNION both data frames. Columns that did not exist in both data frames needed to be added as NULL columns in the data where they did not exist, and some columns needed to be renamed and/or transformed in one data frame to match the other. The resulting data frame was named CHIPOTLE\_COMBINED\_DATA. This resulting data frame was validated by looking at the three columns that should not be null in any records, “LON”, “LAT”, and “LON\_LAT” and returning the number of rows with null values in any of those 3 columns. The result was indeed 0, as was expected.

First 10 rows of CHIPOTLE\_COMBNED\_DATA



Number of rows in CHIPOTLE\_COMBINED\_DATA with nulls for “LON”, “LAT”, or “LON\_LAT”



(See attached Jupyter Notebook for full code)

**7 - Create PySpark data frames for only coordinate data from the combined cleaned data, the cleaned location data, and the cleaned state borders data**

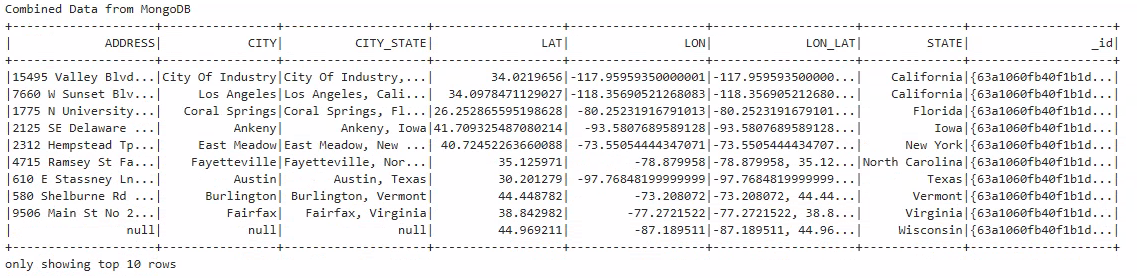
Three PySpark data frames were created by selecting only the “LON\_LAT” column (which was renamed from “COORDINATES” to “LON\_LAT” for CHIPOTLE\_STATE\_BORDERS) from CHIPOTLE\_STATE\_BORDERS, CHIPOTLE\_LOCATIONS, and CHIPOTLE\_COMBINED\_DATA. These were named CHIPOTLE\_STATE\_COORDINATES, CHIPOTLE\_LOCATION\_COORDINATES, and CHIPOTLE\_ALL\_COORDINATES, respectively.

(See attached Jupyter Notebook for full code)

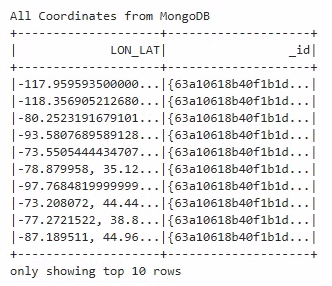
**8 - Write the data frames created in steps 6 and 7 to MongoDB, then read them back into PySpark**

CHIPOTLE\_COMBINED\_DATA, CHIPOTLE\_ALL\_COORDINATES, CHIPOTLE\_STATE\_COORDINATES and CHIPOTLE\_LOCATION\_COORDINATES were written to MongoDB into a database named “project” under the collections “chipotle\_combined\_data”, “chipotle\_all\_coordinates”, “chipotle\_state\_coordinates” and “chipotle\_location\_coordinates”, respectively. Then, the data were read back into PySpark from MongoDB and the first 10 rows of each data frame were displayed to prove that the data were successfully read into PySpark.

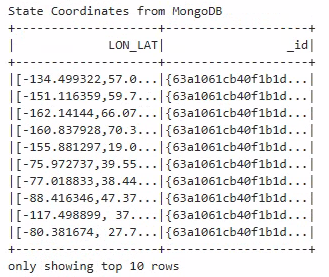
First 10 rows of CHIPOTLE\_COMBINED\_DATA read in from MongoDB



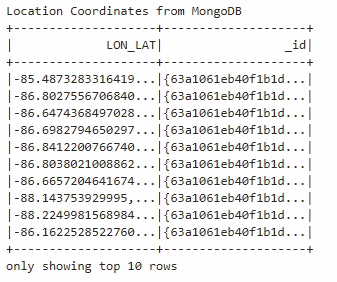
First 10 rows of CHIPOTLE\_ALL\_COORDINATES read in from MongoDB



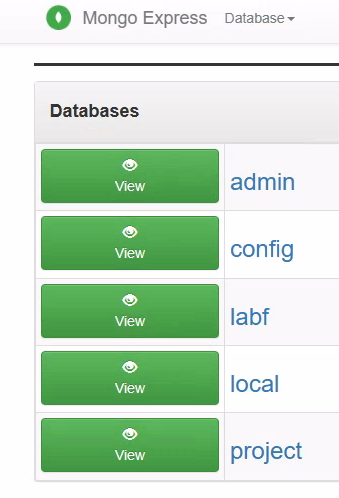
First 10 rows of CHIPOTLE\_STATE\_COORDINATES read in from MongoDB



First 10 rows of CHIPOTLE\_LOCATION\_COORDINATES read in from MongoDB

****

MongoDB Databases



MongoDB Collections in project Database



(See attached Jupyter Notebook for full code)

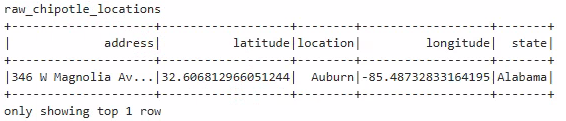
**9 - Write all 8 data frames to Elasticsearch, then read them back into PySpark**

RAW\_CHIPOTLE\_STATE\_BORDERS, RAW\_CHIPOTLE\_LOCATIONS, CHIPOTLE\_STATE\_BORDERS, CHIPOTLE\_LOCATIONS, CHIPOTLE\_COMBINED\_DATA, CHIPOTLE\_ALL\_COORDINATES, CHIPOTLE\_STATE\_COORDINATES and CHIPOTLE\_LOCATION\_COORDINATES were written to Elasticsearch under the indices “raw\_chipotle\_state\_borders”, “raw\_chipotle\_locations”, “chipolte\_state\_borders”, “chipotle\_locations”, “chipotle\_combined\_data”, “chipotle\_all\_coordinates”, “chipotle\_state\_coordinates” and “chipotle\_location\_coordinates”, respectively. Then, the data were read back into PySpark from Elasticsearch and the first row of each data frame was displayed to prove that the data were successfully read into PySpark.

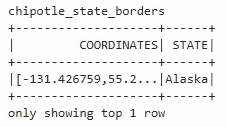
First row of RAW\_CHIPOTLE\_STATE\_BORDERS read in from Elasticsearch



First row of RAW\_CHIPOTLE\_LOCATIONS read in from Elasticsearch



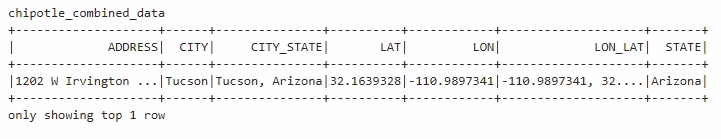
First row of CHIPOTLE\_STATE\_BORDERS read in from Elasticsearch



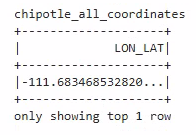
First row of CHIPOTLE\_LOCATIONS read in from Elasticsearch



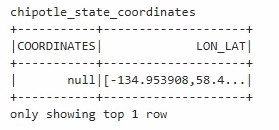
First row of CHIPOTLE\_COMBINED\_DATA read in from Elasticsearch



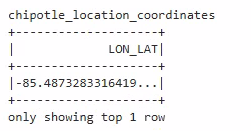
First row of CHIPOTLE\_ALL\_COORIDNATES read in from Elasticsearch



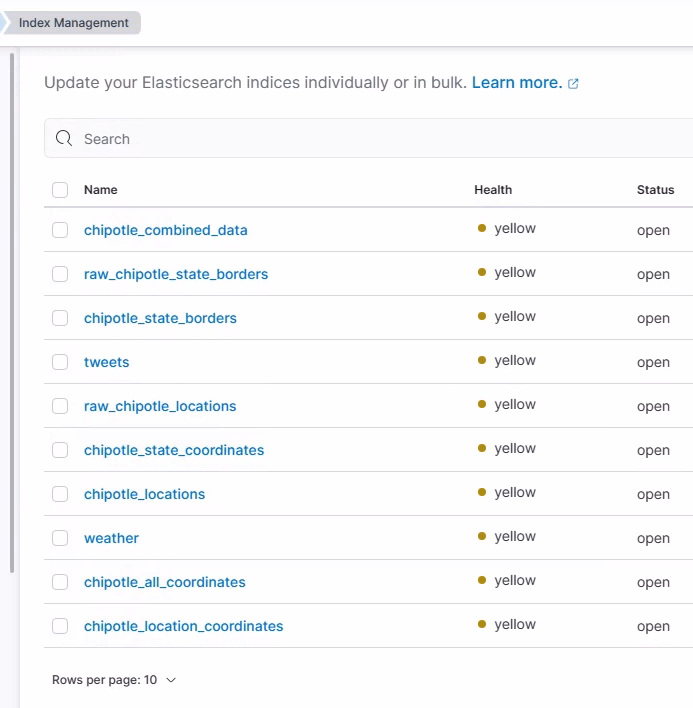
First row of CHIPOTLE\_STATE\_COORDINATES read in from Elasticsearch



First row of CHIPOTLE\_LOCATION\_COORIDNATES read in from Elasticsearch



Elasticsearch Indices



(See attached Jupyter Notebook for full code)

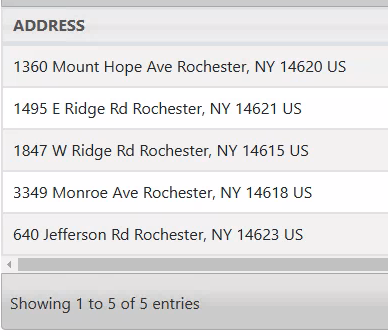
**10 - Use Drill to query the cleaned location data using a few queries**

Drill queries were run on the chipotle\_locations collection from MongoDB to answer the following 3 data questions.

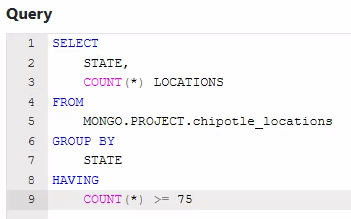
* What are the addresses of the Chipotle locations in Rochester, NY?
* How many Chipotle locations are there in each state, for states with at least 75 locations?
* What are the top 10 cities in the United States for most Chipotle locations, and how many Chipotle locations do those cities have?

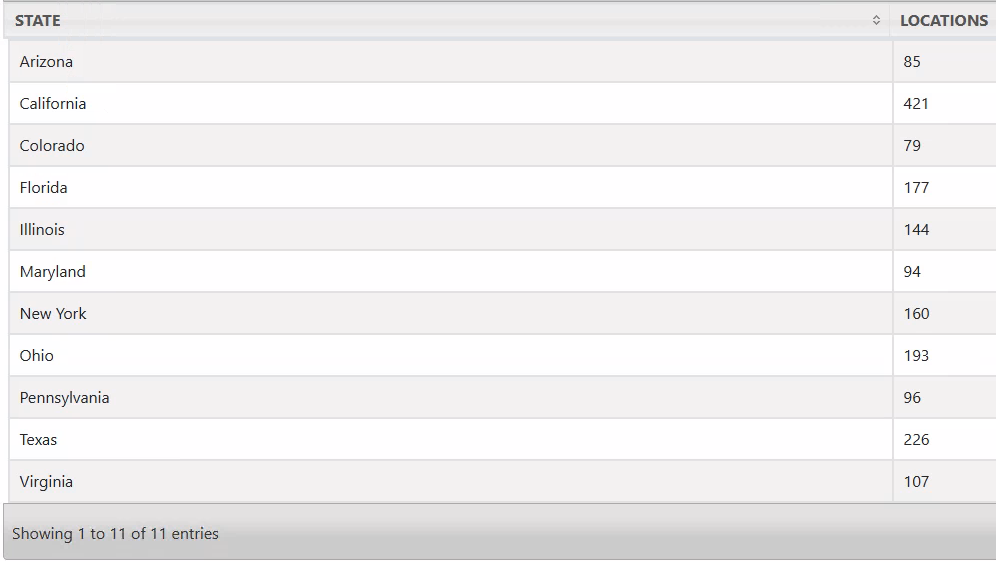
Drill Query for Locations in Rochester, NY



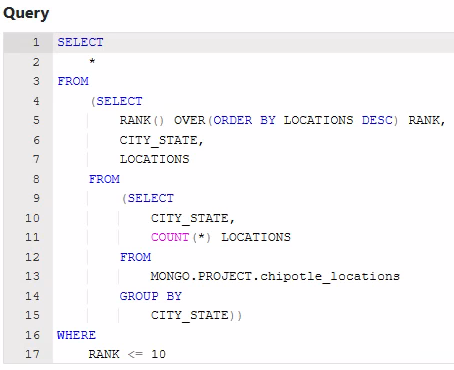


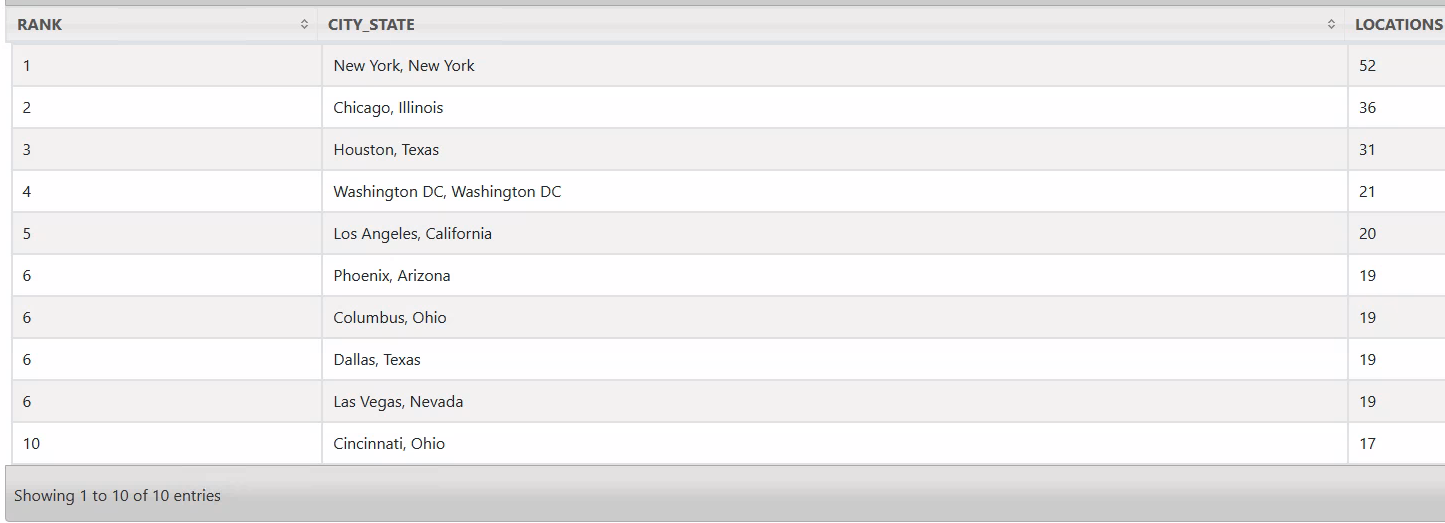
Drill Query for Number of Locations per STATE for states with 75 or more locations

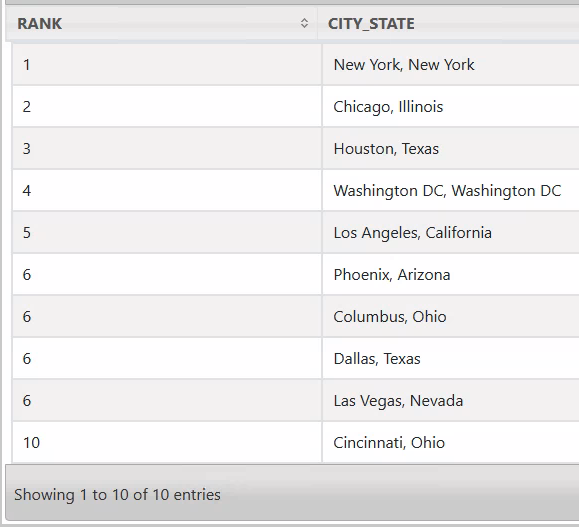


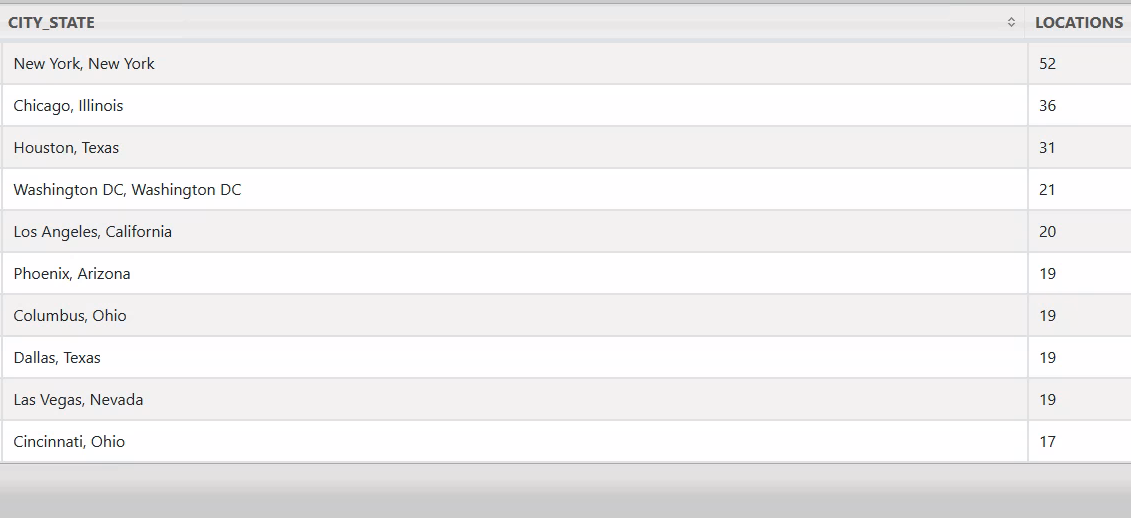


Drill Query for Top 10 CITY\_STATE by Number of locations









**11 - Create a Kibana Map visualization using the coordinate-only location data**

Using the chipotle\_location\_coordinates index, an index pattern was created in order to give a data source for a Kibana Map visualization. A geographic field was needed for the location coordinates, but “LON\_LAT” was a string field. Kibana requires an array of two doubles, and needs reversed order from the data source since latitude needs to be before longitude in Kibana. Once this was achieved, the map visualization was created.

Calculation script for geo-point field

*emit(*

*Double.parseDouble(*

*doc['LON\_LAT.keyword']*

*.value*

*.substring(*

*doc['LON\_LAT.keyword']*

*.value*

*.indexOf(',')+1))*

*,*

*Double.parseDouble(*

*doc['LON\_LAT.keyword']*

*.value*

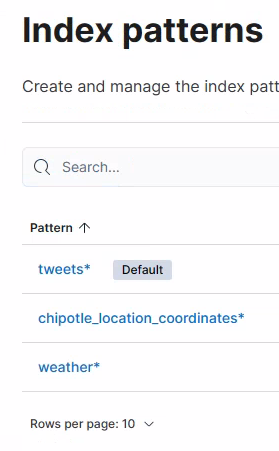
*.substring(0,*

*doc['LON\_LAT.keyword']*

*.value*

*.indexOf(','))))*

Kibana Index Patterns



Kibana geo\_point field for Map



Kibana Map

