Ben Cacic, Stanley Chow, Puneet Gill, Andrew Phan

Data Exploration

The data we were provided came in csv format, inside this we have >20,000 entries. Manually exploring the data would be an inane and redundant task as we will be querying the data after it is loaded. Initially when inspecting the data in its raw format, we can see it is a comma delimited file, this will allow easy csv loading similar to what we've done before. When exploring the csv before loading, we utilized Google Sheets to view the header information in a more neat manner, alongside this, we can just view the first few rows of the dataset and notice anything unique. When reading the queries we will need to create, we inspected the queries we will report on and made a list of the fields we will require for pulling the data needed (host_is_superhost, host_name, neighbourhood_group_cleansed, etc). Inspecting these fields displayed important variable type information. For example, the host_is_superhost column does not consist of boolean values, instead it is string values of “t”or “f”, thus when we do a comparison to determine if a host is a superhost, we will have to do a string comparison with the letter. There were also many fields with missing values, inconsistent values, and broken strings but we will be loading data as is, and reporting on it as is so we will ensure to not force a data type during loading and maintain strings.

Data Loading

The loading process was relatively simple, we handled this in two methods, one was utilizing Mongo Compass and one utilizing Mongo Shell.

Mongo Compass Loading Process - Displayed in “**Compass Loading Process.txt”**

Mongo Shell Loading Process - This shell does not come default with the MongoDB windows install, the shell is downloaded afterwards and installed into the /Bin/ of MongoDB. Once the shell is installed we can utilize all of MongoDB’s loading and create functionality.

1. Begin with installing a MongoDB instance, I used Windows so I used the installer from the MongoDB download page.
2. Install Mongosh on top of MongoDB instance.
3. Start MongoDB Server Instance.
4. Open Mongosh to access and manage our database.
5. To load the data we begin with creating our database:
    - use <databasename>
6. Load the csv utilizing mongoimport, which is a library capable of importing content from a JSON/CSV into a table, which is referred to as a collection.
    - mongoimport --db <databasename> --collection <tableName> --type csv --headerline --file “/path/to/listings.csv”
7. Once we execute the import we can validate the load was done correctly with a simple query, which is also the same query expected in our report.
    - db.<tableName>.find().limit(3).
8. Inspecting the query, the data loaded correctly and was ready to be queried.

Ben Cacic, Stanley Chow, Puneet Gill, Andrew Phan

Querying

To execute queries we created a powershell file which will execute all the queries and output the result into a file to maintain the selection. This will allow us to separate each query report to allow for further analysis.

Run_all_queries.ps1:

```
# Set the base directory for saving the output files
$baseDir = "C:\Users\Path\To\Where\You\Want\Query\Output\Folder""

# Query 1: Find and print the first 3 listings
mongosh <databaseName> --eval "load('query1.js')" > "$baseDir\query1Report.json"

# Query 2: Find and print the first 10 listings
mongosh <databaseName> --eval "load('query2.js')" > "$baseDir\query2Report.json"

# Query 3: Directly find and print listings of the first two superhosts
mongosh <databaseName> --eval "load('query3.js')" > "$baseDir\query3Report.json"

# Query 4: Find all the unique host_name values
mongosh <databaseName> --eval "load('query4.js')" > "$baseDir\query4Report.json"

# Query 5: Find all places with more than two beds in a specified neighbourhood, ordered by
review_scores_rating in descending order
mongosh <databaseName> --eval "load('query5.js')" > "$baseDir\query5Report.json"

# Query 6: Show the number of listings per host
mongosh <databaseName> --eval "load('query6.js')" > "$baseDir\query6Report.json"

# Query 7: Find the average review_scores_rating per neighbourhood, only show those scoring 95 and
above, ordered by rating in descending order
mongosh <databaseName> --eval "load('query7.js')" > "$baseDir\query7Report.json"
```

Reporting

For reporting the queries, we output it into a .json file. Each query has its own json file that has the output from the queries. We believe that each query outputted what we wanted correctly. The results of each query are in the query output folder as putting them in the report would greatly increase the length and reduce clarity of the report.

Experience

While it was a relatively easy assignment, there were some things we needed to find through reading documentation. One of the problems was just trying to query right in MongoDB as it is

quite different from what we're usually used to. We're used to using SQL statements with the form of

```
Unset
Ex.
SELECT person_id
FROM table
GROUP BY person_id
HAVING COUNT(*) > (
  (SELECT COUNT(DISTINCT city_id) FROM table
  )-1);
```

Reading through the documentation helped us see how queries are written in MongoDB and implement them correctly. Another thing that took some time was setting up MongoDB on our own machines. This took some time to figure out as not all libraries and shells were initially installed. MongoDB's compass interface is a nice add on as we can see the visualization of data rather than just see it on the shell command line. It is much easier to use compass rather than the shell command line for those who only need to work with the data. We assume that reading through documentation is very helpful whenever working with unfamiliar software and is an important skill to have when working in industry as new technologies come out.