Assignment 2

Logistics

You must create an IPython notebook for the problem, and upload the notebook to Canvas. The assignment is due on **July 29, before the beginning of class**.

Deliverables. The notebook should be split by sections, one for each question. Each section should contain the following parts:

1. A heading for each question (use a Markdown cell).
2. (Optional) Write a short description of your approach in the Markdown cell.
3. Write the code, and show the results or plots. Please try to comment your code, and use informative variable names.

1 NYC Restaurants

We will analyze restaurant inspections in New York City, starting November 1, 2014 and ending January 31, 2015. The data is in the file `NYC_Restaurants.csv`.

Setup First, read in this data using:

- `df = pd.read_csv('NYC_Restaurants.csv', dtype=str)`.

This ensures that all fields are read in as strings, and loading the data is relatively fast.

[Q1, 6 points] Create a unique name for each restaurant. On the DataFrame created above, add a new column to your DataFrame, called 'RESTAURANT', that combines DBA, BUILDING, STREET, ZIPCODE, and BORO fields. For example, "WENDY'S 469 FLATBUSH AVENUE 11225 BROOKLYN". Print the first 10 values of the RESTAURANT column of your DataFrame.

[Q2, 6 points] How many restaurants are included in the data? Careful now:

- A "Subway" in one particular address (i.e., building, street, zipcode, and boro) counts as one restaurant; don't combine all Subways into one restaurant!
- The data can have multiple violations for the same restaurant!

[Q3, 6 points] How many chains are there? Let us define a chain to be the same restaurant name occurring in at least two different (building, street, zipcode, boro) addresses (i.e., one DBA with multiple restaurant locations).

You'll see multiple versions of the name "DUNKIN DONUTS". Just act as if they are different chains.

[Q4, 6 points] Plot a bar graph of the top 20 most popular chains. We already have the chains from the previous problem. Count the number of restaurants for each chain as a measure of its popularity.

[Q5, 6 points] What fraction of all restaurants are chain restaurants?

[Q6, 6 points] Plot the number of non-chain restaurants in each boro. First, we need to figure out all the non-chain restaurants, then select out only those restaurants, and finally plot the number of such restaurants by boro. Make sure to look at the plot; we don't want to see... oh... the "missing" boro.

[Q7, 8 points] Plot the *fraction* of non-chain restaurants in each boro. The boro with the most non-chain restaurants might just be the boro with the most restaurants in general. If we want to find the boro that

attracts the most "independent" restauranteurs, we must divide the number of non-chain restaurants by the total number of restaurants in the boro. Plot this.

Is the boro with the most independent restaurants also the one with the highest ratio of independent restaurants?

[Q8, 6 points] Plot the popularity of cuisines. Which cuisines are the most well-represented among all restaurants? Define the popularity of a cuisine as the number of restaurants serving that cuisine. Plot the popularity of the top 20 cuisines.

[Q9, 9 points] Plot the cuisines among restaurants which never got cited for violations. Ideally, you should explore and see what happens when there is no violation, but here I will just tell you: the 'VIOLATION CODE' field is missing.

First, find the restaurants that were *never* cited for a code violation. The compute the popularity of each cuisine among these "clean" restaurants. Plot the popularity of the top-20 among these cuisines.

[Q10, 6 points] What cuisines tend to be the "cleanest"?

- Select all cuisines for which there were at least 20 restaurants representing that cuisine.
- For each such cuisine, compute the ratio of the counts in Q9 to Q8. This is the ratio of restaurants that never got cited, versus total number of restaurants, for each cuisine.
- Find the top-10 cuisines with the highest ratios; these are that cuisines whose restaurants are "most likely to be clean."

[Q11, 8 points] What are the most common violations in each borough? Create a table of the number of times each violation description was observed in each borough, and figure out the most common violation description for each borough.

To create the table, check out the `crosstab` function. We will see a more general version of this when we discuss `groupby` in class.

Once you do have the table, you will still need to find the most common violation description for each borough.

[Q12, 9 points] What are the most common violations per borough, *after normalizing for the relative abundance of each violation*? Hopefully, the answer to the previous question left you unsatisfied, because some violations are just very common, irrespective of borough. A better approach would be to *normalize* the violation counts, as follows.

- *Get overall frequencies:* Figure out how common each violation is, over the entire dataset; let's call this `violationFrequency`.
- *Normalize:* Consider the table of number of violations by boro that you created for the previous question. For each borough, divide the number of violations of each type by the total number of violations for that type; i.e., divide the series of violations by `violationFrequency`. We want to do this *for each borough*.
- *Find the biggest violations:* Now, after this normalization, for each borough, figure out the most common violation description.

[Q13, 8 points] How many phone area codes correspond to a single zipcode? The first three digits of the restaurant phone numbers are their area codes. The area codes do not generally align with zip codes, but some area codes are only for a single zip code. You must figure out how many area codes have this property.

- To extract the first 3 characters of the phone number, recall that strings are pretty similar to lists.

[Q14, 10 points] Find common misspellings of street names Sometimes, it's *Avenue*, and sometimes, it's *Ave*. We will try to come up with an automated way to find common misspellings. The idea is the following: if *Ave* and *Avenue* are the same, they should show up often in similar-sounding street names, e.g., *Lexington Ave* and *Lexington Avenue*.

- Create a new column, called `STREET TYPE`, which is the the last word in the name of the street. For example, if the street is *"Astoria Boulevard"*, the street type should be *"Boulevard"*.
- Create another column, called `STREET BASE`, which contains everything *but* the last word in the name of the street. For example, if the street is *"Astoria Boulevard"*, the street base should be *"Astoria"*.
- Create a third column, called `STREET BASE & ZIP`, that combines the street base and the zipcode.

- Create a table containing just these three columns, and remove any duplicates or instances where street base is empty. This table now contains unique street names, along with the street type.
- Merge this table with itself, on the `STREET BASE & ZIP` column. Thus, in the new merged tabled, we will have two `STREET TYPE` fields for each street base and zipcode. For example, if both *Lexington Ave* and *Lexington Avenue* exist in the same zipcode, we will get a row for the street base *Lexington* and the two street types *Ave* and *Avenue*.
- From the merged table, select only the rows where the street types are different.
- Now, do a *cross-tabulation* of the two distinct street types (check out the `crosstab` function in Pandas). This gives us the number of times *Ave* and *Avenue* were used with the same `STREET BASE & ZIP`.
- From this cross-tabulation table, find the most commonly street type that occurs with each of the following: *AVE*, *ST*, *RD*, *PL*, *BOULEARD*, and *BULEVARD*.