Finding the best places to eat in a city

1. INTRODUCTION

1.1 Background

As a frequent traveler and a food lover, I have often found it hard to land myself in a good restaurant, especially in a new city. Almost everyone travels, and hence it would be beneficial to them if there were an automated way of finding a good restaurant to eat in, depending on the area they are in and what cuisine they would like.

1.2 Business Problem

The problem here is to know which restaurant out of so many to choose when traveling in a new place. This project aims to provide a wide variety of eatery options to choose from, as well as recommend the best one.

1.3 Interest

Obviously, the interest would be to anyone that travels that is interested in eating out.

2. DATA ACQUISITION AND CLEANING:

2.1 Data Sources

All the data for this project will be scraped using the Foursquare API. The data includes a list of restaurants and eateries in the areas, and tips for each of these restaurants, the number of tips, and the agree and disagree counts of these tips.

2.2 Data cleaning

Data was downloaded in three steps. Each step specified a different location in Vancouver so as to get as many results as possible, since the Foursquare API limits number of results for a particular query to 50 (for the basic account). The data was then loaded into a Dataframe, and the data frames were merged.

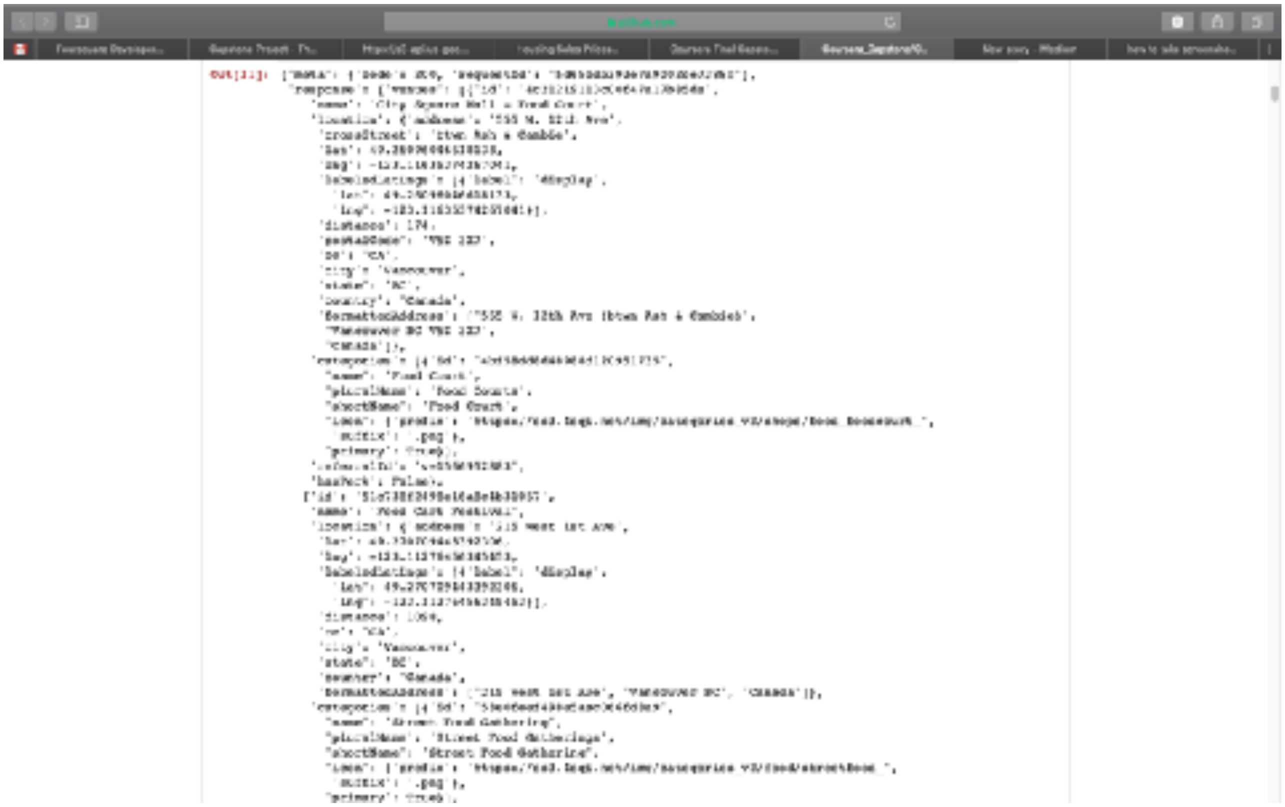
Each of the three initial data frames had a similar format in terms of number of columns and column names. For each data set, columns with too many missing values as well as irrelevant data were removed.

After clustering and creating maps with these data frames, tips for each venue were extracted from the Foursquare API and appended to a list. This was then cleaned and only required values were kept and converted into a Dataframe.

3. METHODOLOGY:

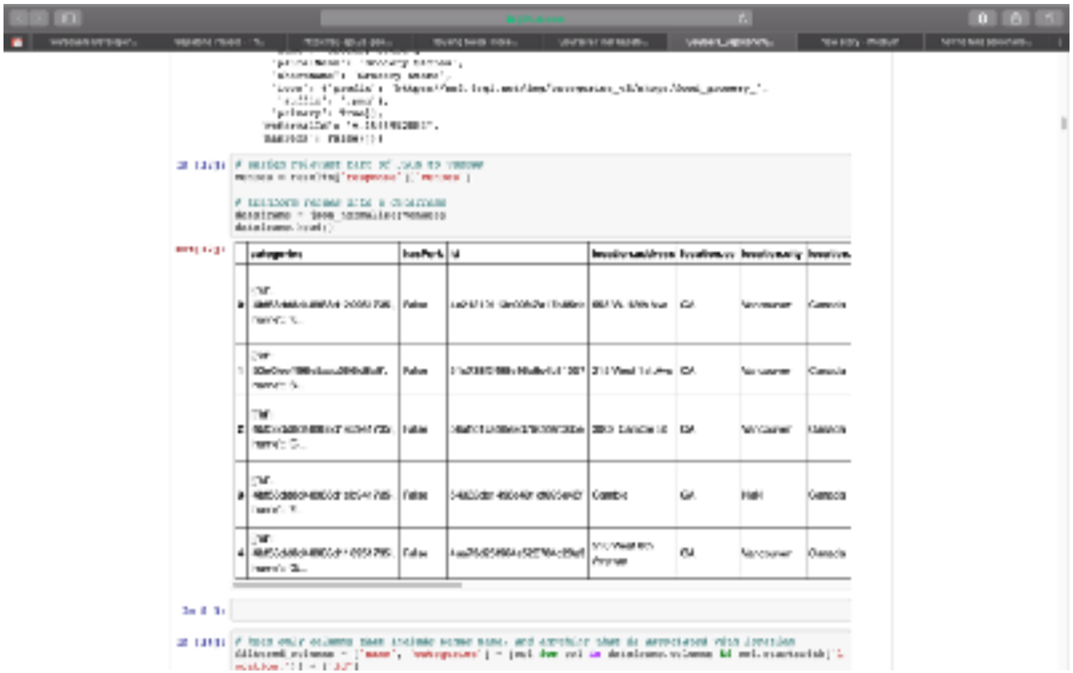
3.1 Exploratory Data Analysis

Using latitude and longitude values, a limit of 100, and a radius of 1000m for each location, data returned from the Foursquare API was converted to a json object and prettified using Beautiful Soup.



```
Out[11]: {'meta': {'code': 200, 'requestId': '[illegible]'},
          'response': {'venues': [{'id': '[illegible]',
            'name': 'City Square Mall & Food Court',
            'location': {'address': '555 W. 12th Ave',
             'crossStreet': 'btwn Ash & Cambie',
             'lat': 49.26090066538538,
             'lng': -123.11838274367341,
             'labeledLatLngs': [{'label': 'display',
               'lat': 49.26090066538173,
               'lng': -123.11635782578481}],
             'distance': 174,
             'postalCode': 'V5Z 3X7',
             'cc': 'CA',
             'city': 'Vancouver',
             'state': 'BC',
             'country': 'Canada',
             'formattedAddress': ['555 W. 12th Ave (btwn Ash & Cambie)',
              'Vancouver BC V5Z 3X7',
              'Canada']},
            'categories': [{'id': '4bf58dd8d48988d120951735',
              'name': 'Food Court',
              'pluralName': 'Food Courts',
              'shortName': 'Food Court',
              'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/shops/food_foodcourt_',
               'suffix': '.png'},
              'primary': True}],
            'referralId': 'v-[illegible]',
            'hasPerk': False},
           {'id': '51c738f2498e16a8e4b30957',
            'name': 'Food Cart Festival',
            'location': {'address': '215 West 1st Ave',
             'lat': 49.[illegible],
             'lng': -123.[illegible],
             'labeledLatLngs': [{'label': 'display',
               'lat': 49.[illegible],
               'lng': -123.[illegible]}],
             'distance': 1064,
             'cc': 'CA',
             'city': 'Vancouver',
             'state': 'BC',
             'country': 'Canada',
             'formattedAddress': ['215 West 1st Ave', 'Vancouver BC', 'Canada']},
            'categories': [{'id': '[illegible]',
              'name': 'Street Food Gathering',
              'pluralName': 'Street Food Gatherings',
              'shortName': 'Street Food Gathering',
              'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/food/streetfood_',
               'suffix': '.png'},
              'primary': True}],
```

Then, as previously mentioned, data frames were created and cleaned up.



```
     'pluralName': 'Grocery Stores',
     'shortName': 'Grocery Store',
     'icon': {'prefix': 'https://ss3.4sqi.net/img/categories_v2/shops/food_grocery_',
      'suffix': '.png'},
     'primary': True}],
   'referralId': 'v-[illegible]',
   'hasPerk': False}]}}
```

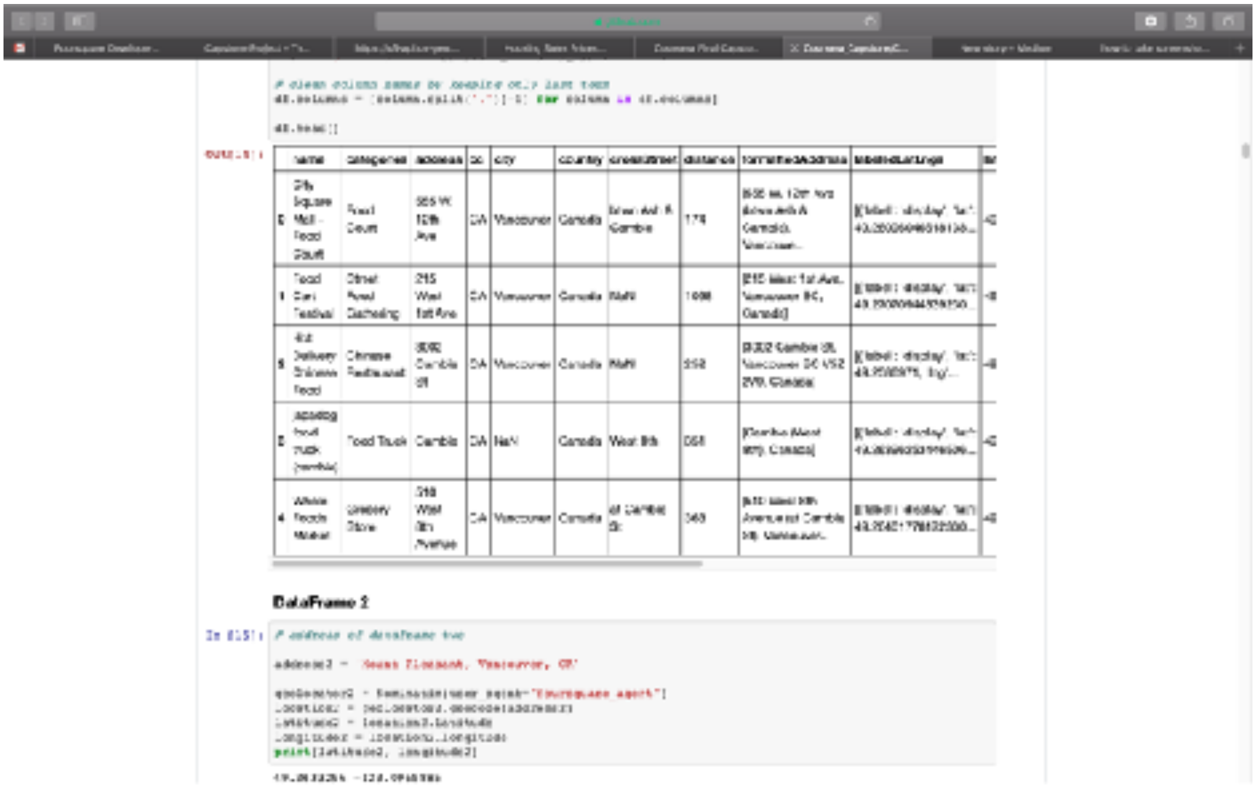
```
In [12]: # assign relevant part of JSON to venues
         venues = results['response']['venues']

         # tranform venues into a dataframe
         dataframe = json_normalize(venues)
         dataframe.head()
```

Out[12]:

	categories	hasPerk	id	location.address	location.cc	location.city	location...
0	[{'id': '[illegible]', 'name': '...</br>	False	[illegible]	[illegible] W. 12th Ave	CA	Vancouver	Canada
1	[{'id': '[illegible]', 'name': 'S...	False	[illegible]	215 West 1st Ave	CA	Vancouver	Canada
2	[{'id': '[illegible]', 'name': 'G...	False	[illegible]	[illegible] Cambie St	CA	Vancouver	Canada
3	[{'id': '[illegible]', 'name': 'F...	False	[illegible]	Cambie	CA	Hall	Canada
4	[{'id': '[illegible]', 'name': 'S...	False	[illegible]	[illegible] West 6th Avenue	CA	Vancouver	Canada

```
In [ ]:
```

```
In [13]: # keep only columns that include venue name, and anything that is associated with location
         filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] + ['id']
```



```
# clean column names by keeping only last term
df.columns = [column.split('.')[-1] for column in df.columns]

df.head()
```

Out[13]:

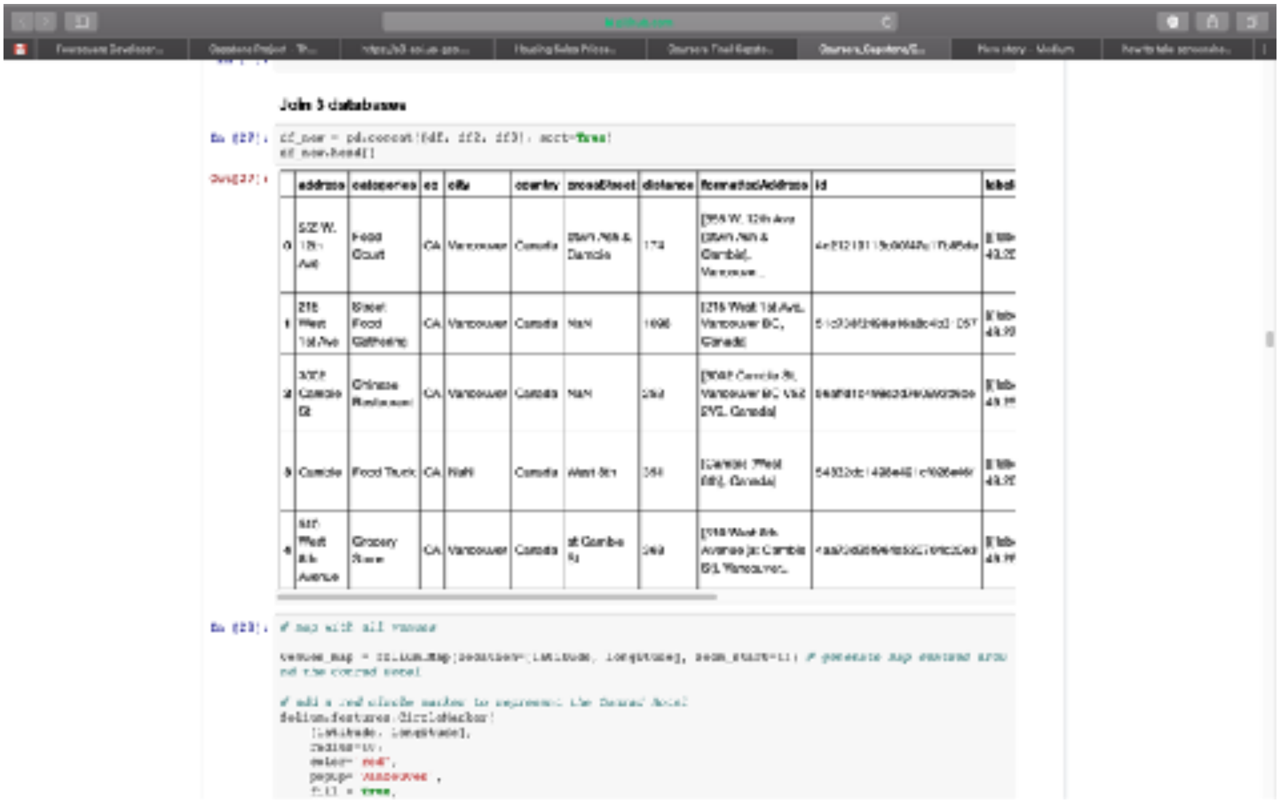
	name	categories	address	cc	city	country	crossStreet	distance	formattedAddress	labeledLatLngs	la
0	City Square Mall - Food Court	Food Court	555 W. 12th Ave	CA	Vancouver	Canada	btwn Ash & Cambie	174	[555 W. 12th Ave (btwn Ash & Cambie), Vancouver...	[{'label': 'display', 'lat': 49.26020464016136...	49
1	Food Cart Festival	Street Food Gathering	215 West 1st Ave	CA	Vancouver	Canada	NaN	1098	[215 West 1st Ave, Vancouver BC, Canada]	[{'label': 'display', 'lat': 49.27000944326250...	49
2	Hot Delivery Chinese Food	Chinese Restaurant	3092 Cambie St	CA	Vancouver	Canada	NaN	252	[3092 Cambie St, Vancouver BC V5Z 2W9, Canada]	[{'label': 'display', 'lat': 49.258875, 'lng'...	49
3	Japadog food truck (cambie)	Food Truck	Cambie	CA	NaN	Canada	West 8th	358	[Cambie (West 8th), Canada]	[{'label': 'display', 'lat': 49.26396053976506...	49
4	Whole Foods Market	Grocery Store	510 West 8th Avenue	CA	Vancouver	Canada	at Cambie St	363	[510 West 8th Avenue (at Cambie St), Vancouver...	[{'label': 'display', 'lat': 49.264017798122000...	49

DataFrame 2

In [15]:
```
# address of dataframe two

address2 = 'Queen Elizabeth, Vancouver, CA'

geolocator2 = Nominatim(user_agent="foursquare_agent")
location2 = geolocator2.geocode(address2)
latitude2 = location2.latitude
longitude2 = location2.longitude
print(latitude2, longitude2)
```

```
49.[illegible] -123.[illegible]
```

The data frames were merged and a total of 85 venues were left over.



Join 3 databases

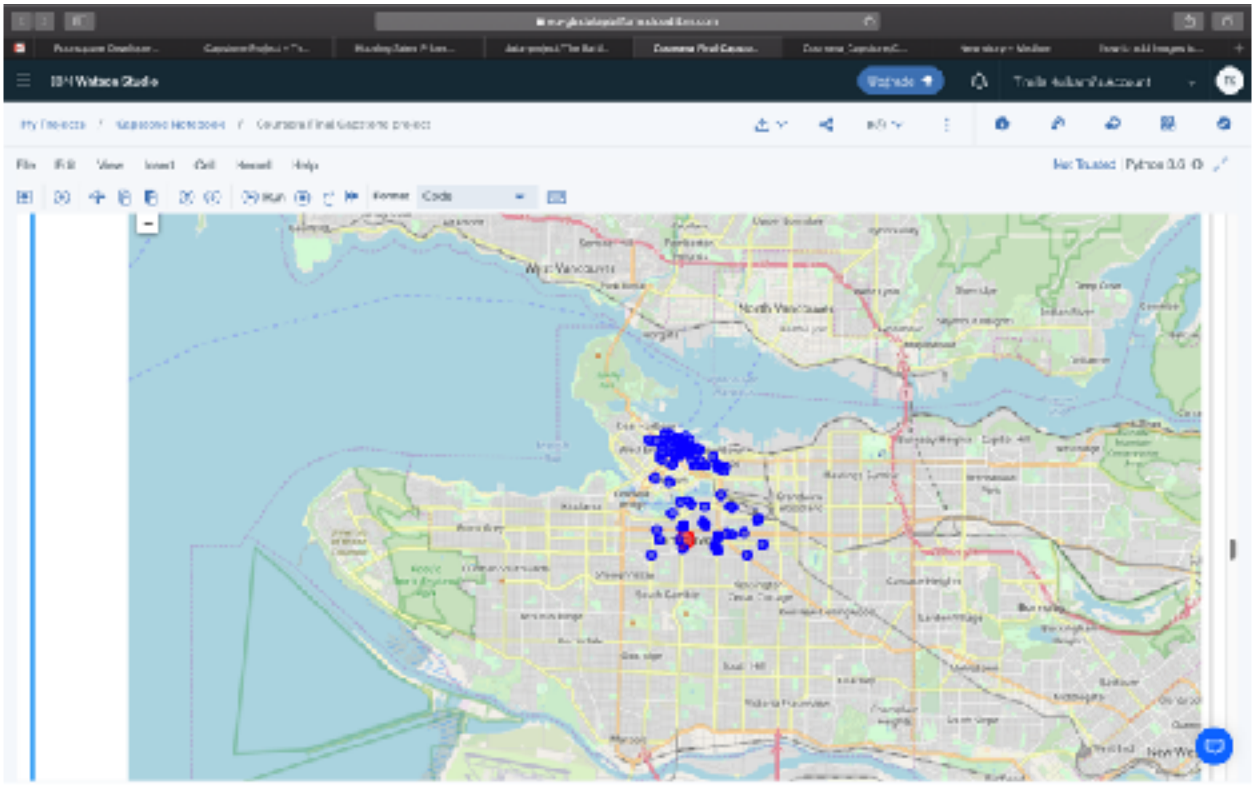
In [27]:
```
df_new = pd.concat([df, df2, df3], sort=True)
df_new.head()
```

Out[27]:

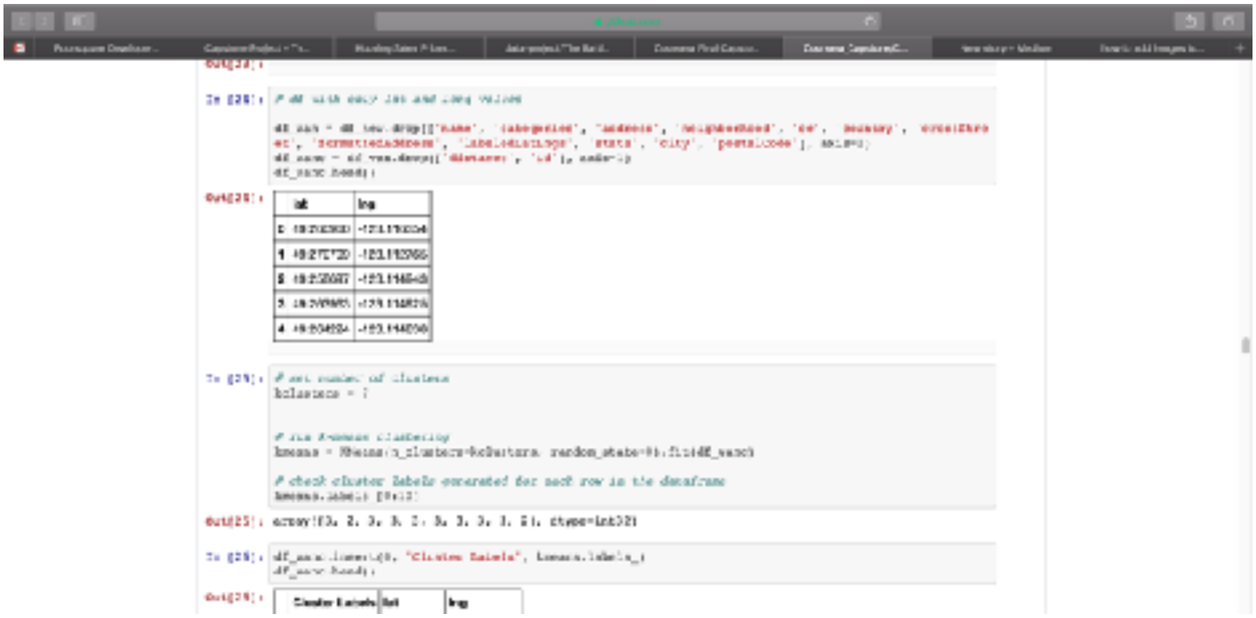
	address	categories	cc	city	country	crossStreet	distance	formattedAddress	id	label
0	555 W. 12th Ave	Food Court	CA	Vancouver	Canada	btwn Ash & Cambie	174	[555 W. 12th Ave (btwn Ash & Cambie), Vancouver...	4e212181[illegible]	[{'lab 49.2
1	215 West 1st Ave	Street Food Gathering	CA	Vancouver	Canada	NaN	1098	[215 West 1st Ave, Vancouver BC, Canada]	[illegible]	[{'lab 49.2
2	3092 Cambie St	Chinese Restaurant	CA	Vancouver	Canada	NaN	252	[3092 Cambie St, Vancouver BC V5Z 2W9, Canada]	[illegible]	[{'lab 49.2
3	Cambie	Food Truck	CA	NaN	Canada	West 8th	358	[Cambie (West 8th), Canada]	[illegible]	[{'lab 49.2
4	510 West 8th Avenue	Grocery Store	CA	Vancouver	Canada	at Cambie St	363	[510 West 8th Avenue (at Cambie St), Vancouver...	[illegible]	[{'lab 49.2

In [28]:
```
# map with all venues

venues_map = folium.Map(location=[latitude, longitude], zoom_start=11) # generate map centred arou
nd the Conrad Hotel

# add a red circle marker to represent the Conrad Hotel
folium.features.CircleMarker(
    [latitude, longitude],
    radius=10,
    color='red',
    popup='Vancouver',
    fill = True,
```

The folium library was used to create a visualization (map) with all the venues superimposed on top.



After this, I ran a K-Means clustering algorithm on the data with number of clusters set to 7. Each cluster represents a different area, and depending on the location of the user of



```
In [24]: # df with only lat and long values

         df_van = df_ven.drop(['name', 'categories', 'address', 'neighborhood', 'cc', 'country', 'crossStreet', 'formattedAddress', 'labeledLatLngs', 'state', 'city', 'postalCode'], axis=1)
         df_van = df_van.drop(['distance', 'id'], axis=1)
         df_van.head()
```

Out[24]:

	lat	lng
0	49.282830	-123.118034
1	49.272720	-123.112966
2	49.252087	-123.114643
3	49.249985	-123.114826
4	49.204824	-123.114056

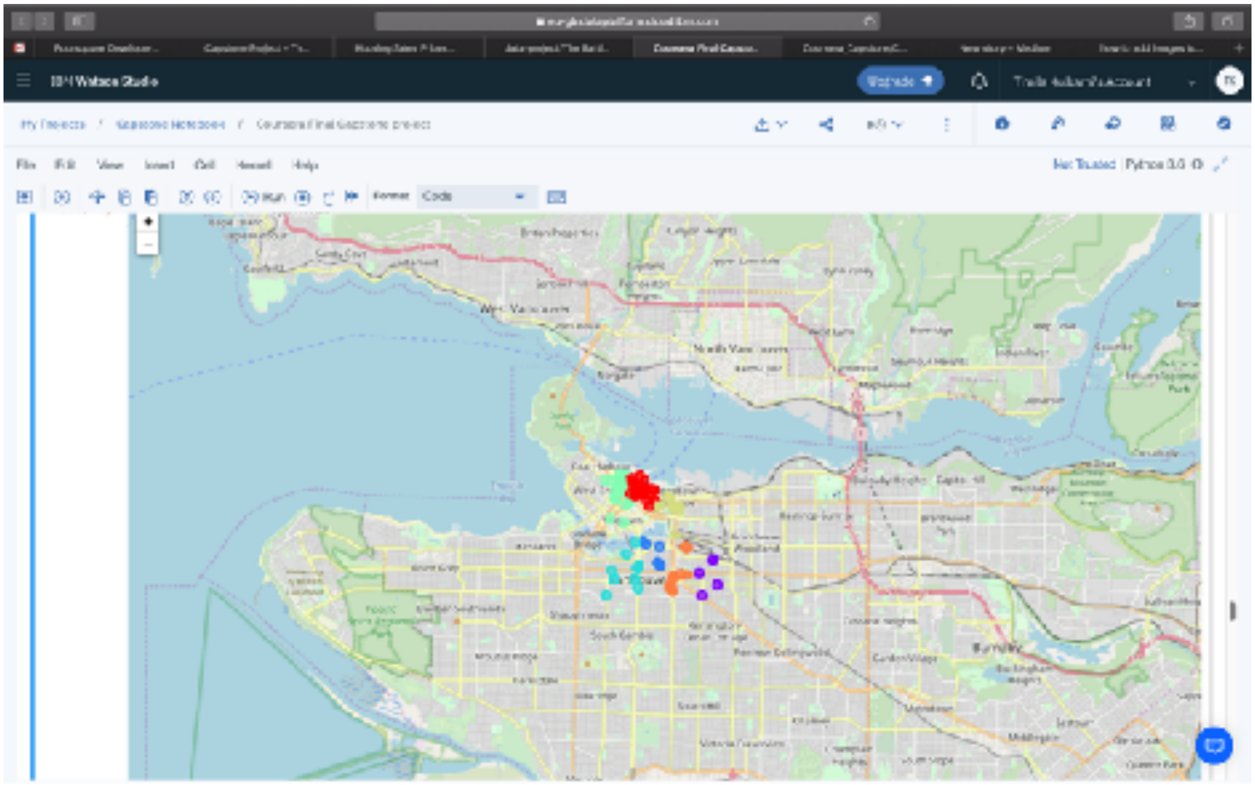
```
In [25]: # set number of clusters
         kclusters = 7


         # run k-means clustering
         kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(df_van)

         # check cluster labels generated for each row in the dataframe
         kmeans.labels_[0:10]
Out[25]: array([3, 2, 3, 3, 2, 3, 3, 3, 3, 5], dtype=int32)

In [26]: df_van.insert(0, 'Cluster Labels', kmeans.labels_)
         df_van.head()
```

Out[26]:

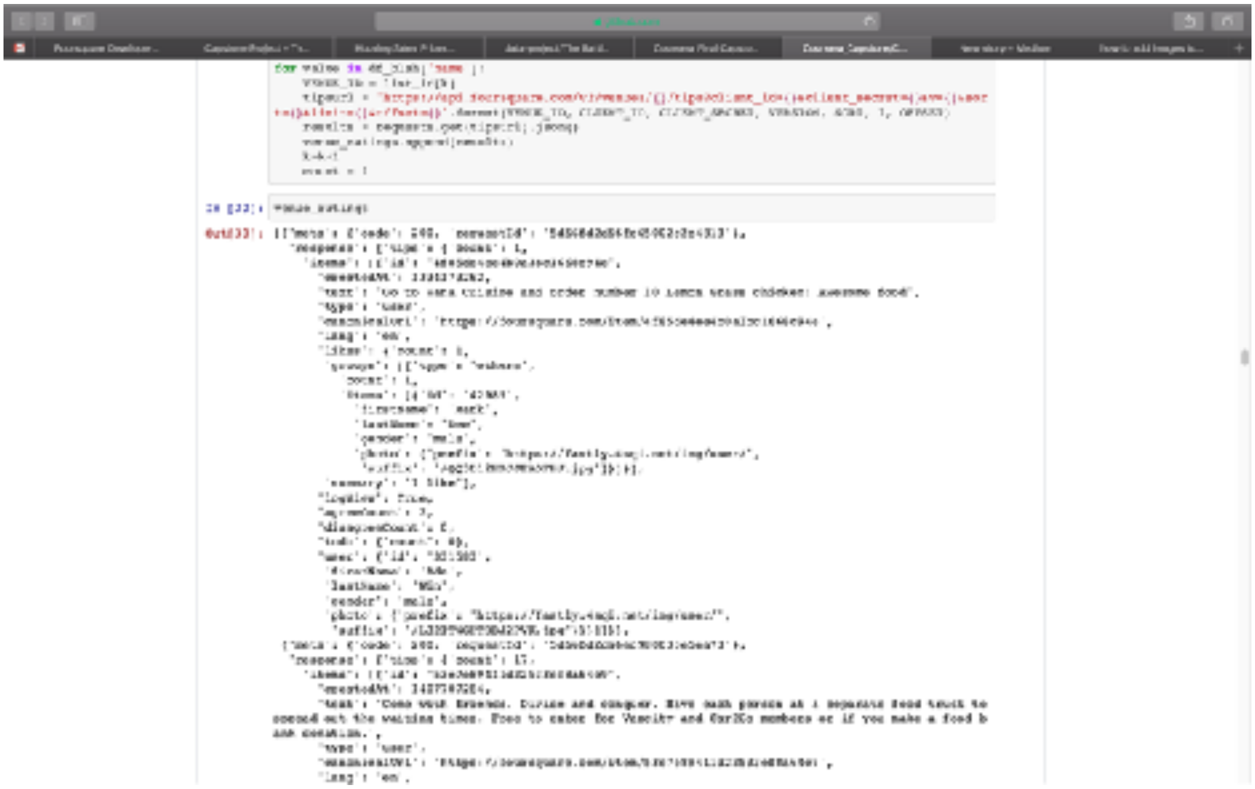
	Cluster Labels	lat	lng

this program, the restaurants can be chosen from the closest cluster.

These clusters were then visualized using folium with venues superimposed. On clicking, we can see the venue name as well as the cluster it belongs to. Each cluster has a different colour.



Next, I created a list of all the venue ids and looped through that list to get tips for each of the venues. I appended the results I got back from the API to a list after converting it to json.



```
for value in df_dish['name']:
    VENUE_ID = list_id[k]
    tipsurl = 'https://api.foursquare.com/v2/venues/{}/tips?client_id={}&client_secret={}&v={}&sort
=[illegible]&limit={}&offset={}'.format(VENUE_ID, CLIENT_ID, CLIENT_SECRET, VERSION, [illegible], 1, OFFSET)
    results = requests.get(tipsurl).json()
    venue_ratings.append(results)
    k=k+1
    count = 1
```

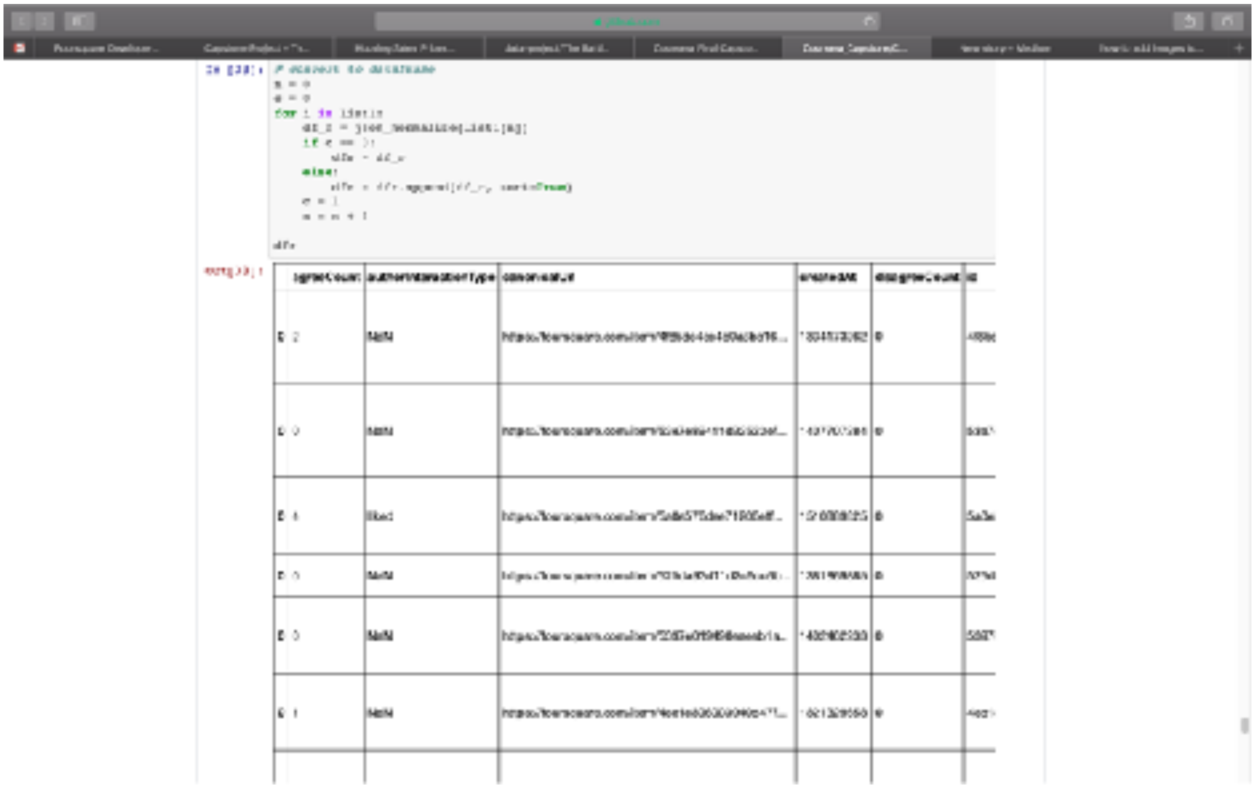
In [33]:

```
venue_ratings
```

Out[33]:

```
[{'meta': {'code': 200, 'requestId': '[illegible]'},
  'response': {'tips': {'count': 1,
    'items': [{'id': '[illegible]',
      'createdAt': [illegible],
      'text': 'Go to sara cuisine and order number 10 lemon grass chicken: awesome food',
      'type': 'user',
      'canonicalUrl': 'https://foursquare.com/item/[illegible]',
      'lang': 'en',
      'likes': {'count': 1,
       'groups': [{'type': 'others',
         'count': 1,
         'items': [{'id': '[illegible]',
           'firstName': 'mark',
           'lastName': '[illegible]',
           'gender': 'male',
           'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
            'suffix': '[illegible].jpg'}}]}],
       'summary': '1 like'},
      'logView': True,
      'agreeCount': 2,
      'disagreeCount': 0,
      'todo': {'count': 0},
      'user': {'id': '[illegible]',
       'firstName': '[illegible]',
       'lastName': '[illegible]',
       'gender': 'male',
       'photo': {'prefix': 'https://fastly.4sqi.net/img/user/',
        'suffix': '[illegible].jpg'}}}]}}},
 {'meta': {'code': 200, 'requestId': '[illegible]'},
  'response': {'tips': {'count': 17,
    'items': [{'id': '[illegible]',
      'createdAt': [illegible],
      'text': 'Come with friends. Divide and conquer. Give each person an [illegible] separate food truck to
spread out the waiting times. Free to enter for Vancity and Car2Go members or if you make a food b
ank donation.',
      'type': 'user',
      'canonicalUrl': 'https://foursquare.com/item/[illegible]',
      'lang': 'en',
```

From this list, I created another list with only the required values. Then I looped through this list and created a Dataframe out of it.



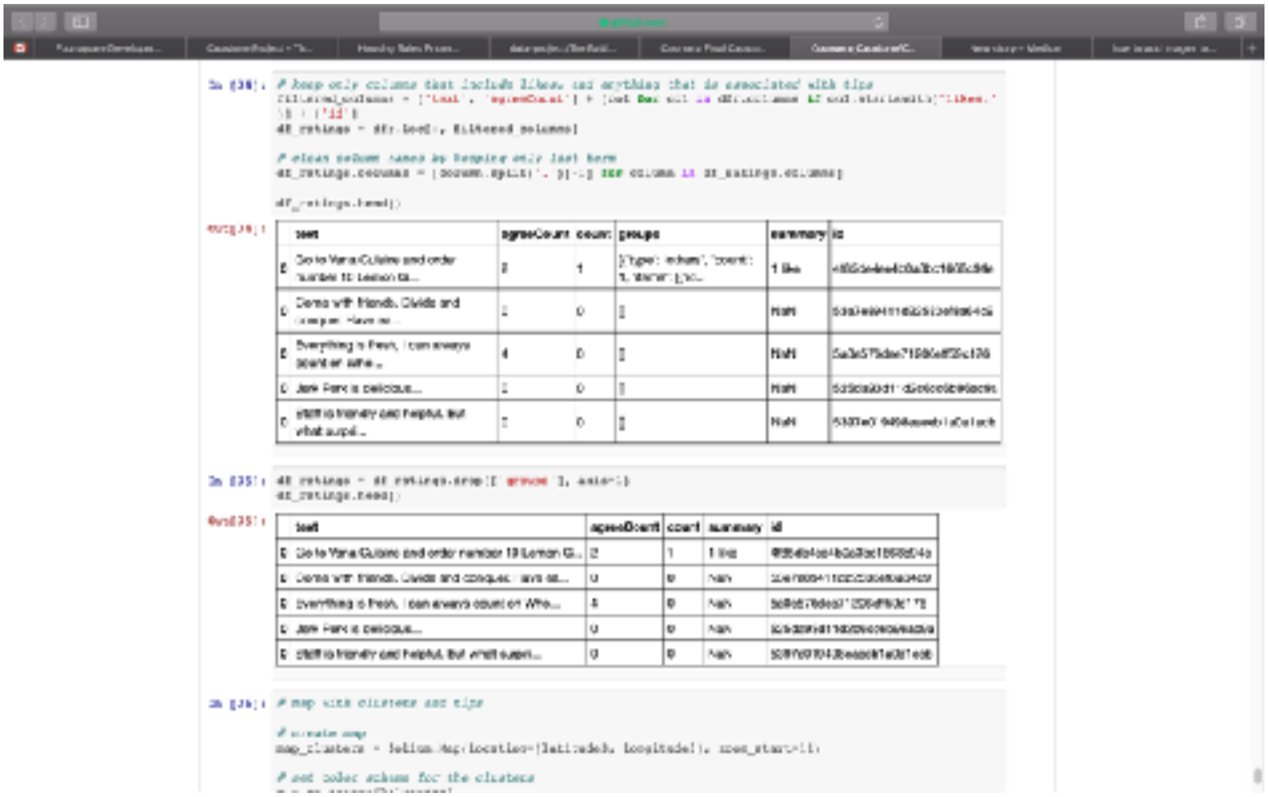
In [[illegible]]:

```
# convert to dataframe
n = 0
c = 0
for i in list[illegible]:
    df_r = json_normalize(list[illegible][n])
    if c == 0:
        dfr = df_r
    else:
        dfr = dfr.append(df_r, sort=True)
    c = 1
    n = n + 1

dfr
```

Out[[illegible]]:

	agreeCount	authorInteractionType	canonicalUrl	createdAt	disagreeCount	id
0	2	NaN	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]
0	0	NaN	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]
0	4	liked	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]
0	0	NaN	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]
0	0	NaN	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]
0	1	NaN	https://foursquare.com/item/[illegible]...	[illegible]	0	[illegible]

Then I applied the same cleaning methods to the Dataframe, i.e dropped unwanted columns, missing values, etc.



In [34]:

```
# keep only columns that include likes, and anything that is associated with tips
filtered_columns = ['text', 'agreeCount'] + [col for col in dfr.columns if col.startswith('likes.'
)] + ['id']
df_ratings = dfr.loc[:, filtered_columns]

# clean column names by keeping only last term
df_ratings.columns = [column.split('.')[-1] for column in df_ratings.columns]

df_ratings.head()
```

Out[34]:

	text	agreeCount	count	groups	summary	id
0	Go to Yama Cuisine and order number 10 Lemon Gr...	2	1	[{'type': 'others', 'count': 1, 'items': [{'c...	1 like	[illegible]
1	Come with friends. Divide and conquer. Have se...	0	0	[]	NaN	[illegible]
2	Everything is fresh, I can always count on Whe...	4	0	[]	NaN	[illegible]
3	Jerk Pork is delicious...	0	0	[]	NaN	[illegible]
4	staff is friendly and helpful, but what surpri...	0	0	[]	NaN	[illegible]

In [35]:

```
df_ratings = df_ratings.drop(['groups'], axis=1)
df_ratings.head()
```

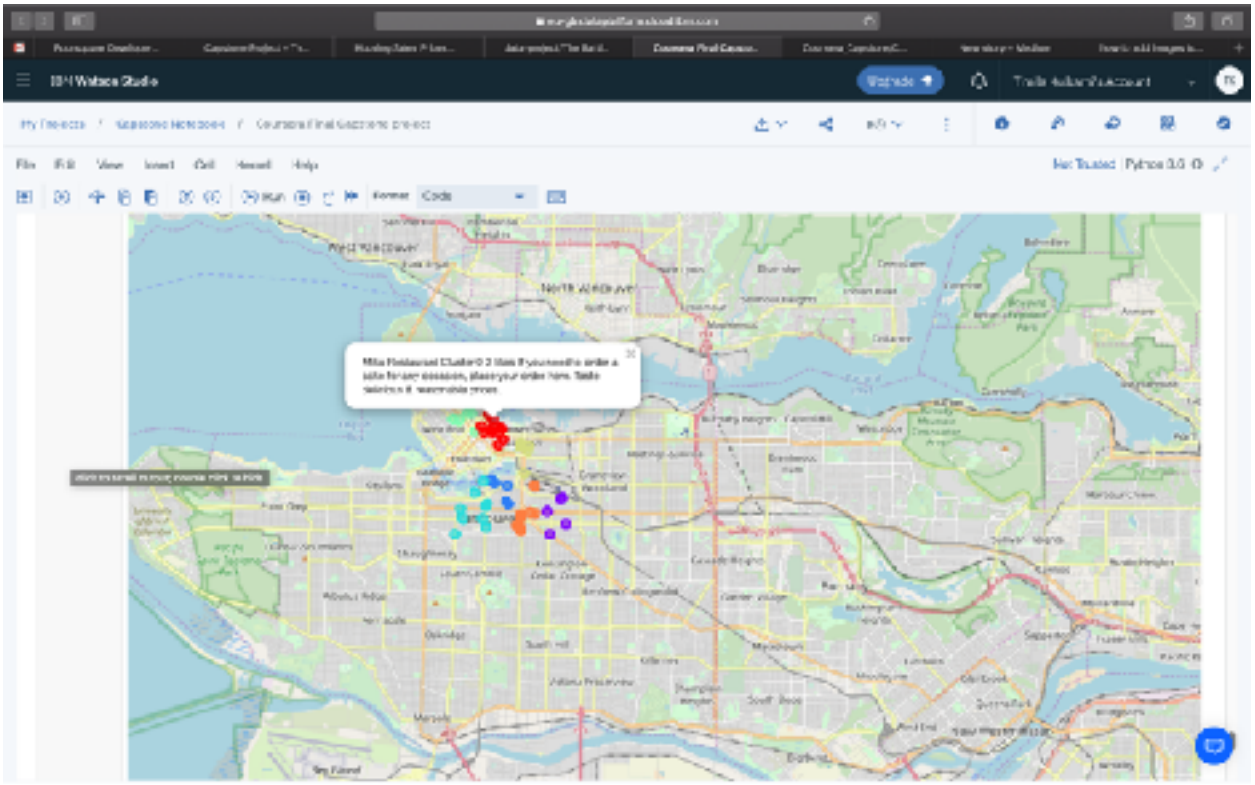
Out[35]:

	text	agreeCount	count	summary	id
0	Go to Yama Cuisine and order number 10 Lemon G...	2	1	1 like	[illegible]
1	Come with friends. Divide and conquer. Have se...	0	0	NaN	[illegible]
2	Everything is fresh, I can always count on Whe...	4	0	NaN	[illegible]
3	Jerk Pork is delicious...	0	0	NaN	[illegible]
4	staff is friendly and helpful, but what surpri...	0	0	NaN	[illegible]

In [36]:

```
# map with clusters and tips

# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
```

I then visualized this using a folium map with the same clusters, venues superimposed on top. When clicked, we can see name of the place, cluster it belongs to, number of likes, and one review.



4. RESULTS:

In the end, I was able to get 88 different venues with all the data in them and tips for 62 of the venues. From this, I was able to create a visualization in the form of the map to display the various restaurants all over the city, and make sure that when clicked on, name, cluster, likes, and tips were displayed. From the map we can also conclude the best restaurants for each cluster, based on tips and likes as follows:
Cluster 0: Yagger's Downtown Restaurant & Sports
Cluster 1: Harry's (Best Foods Grocery)
Cluster 2: Dougie Dog Truck
Cluster 3: City Square Mall - Food Court
Cluster 4: Chongqing Chinese Restaurant & Hawksworth Restaurant

Cluster 5: New Town Bakery & Restaurant
Cluster 6: One-O-One Bulk Food & Deli

5. DISCUSISONS:

There is much scope for improvement in this project. For example, the best restaurant for each cluster could be generated with a bit more code. Also, this project does not include all the venues in and around Vancouver city.

This project was mainly focused on features in the dataset returned by the API. It did not take into account the likes and dislikes of the person, as a perfect recommendation system would.

6. CONCLUSION:

This project is limited to Vancouver city, however, the same technique can be applied to any other city. The locations just need to be replaced with those of the required city. In this aspect, I feel the project has achieved what it was originally meant to do: improve the choice of restaurants for tourists in a new city.