Capstone Project : Battle of the Neighborhoods: Location - Pharmacy

1. Data Acquisition

Import required python libraries

In [1]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

#!conda install -c conda-forge geopy --yes # uncomment this line if you
from geopy.geocoders import Nominatim # convert an address into latitud

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a p

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line
import folium # map rendering library

#!conda install -c conda-forge beautifulsoup4 --yes
import bs4
from bs4 import BeautifulSoup

print('Libraries imported.')
```

```
Libraries imported.
```

Clean data

Map and segment and cluster the neighborhoods in one borough. Create a new dataframe of the boroughs data.

In [2]:

```
request_data = requests.get('https://en.wikipedia.org/wiki/List_of_post
soup = BeautifulSoup(request_data, "html.parser")
table=soup.find('table')
#dataframe will consist of three columns: PostalCode, Borough, and Neig
df = pd.DataFrame(columns = ['PostalCode','Borough','Neighborhood'])
df.shape

# Search all the postcode, borough, neighborhood
for trcelldata in table.find_all('tr'):
    trdata=[]
    for tdcelldata in trcelldata.find_all('td'):
        trdata.append(tdcelldata.text.strip())
    if len(trdata)==3:
        df.loc[len(df)] = trdata
df.dropna()
# drop Borough not assigned
df.drop(df[ df['Borough'] == 'Not assigned' ].index, inplace = True)

# assign Borough to Neighborhood if latter not assigned
df.Neighborhood[df.Neighborhood == 'Not assigned'] = df.Borough

# Create a second frame grouping by postal code with neighborhoods tran
# In the dataset this is already done so no need to do this, but doing
df1=df.groupby('PostalCode')['Neighborhood'].apply(lambda x: "%s" % ',
# Merge , drop the extra column not comma separated and rename to corre
df2 = pd.merge(df, df1, on='PostalCode').drop_duplicates().drop(['Neigh

df2.shape
# Rename column to match that of df
geospatial_df = pd.read_csv('http://cocl.us/Geospatial_data')
geospatial_df.rename(columns={'Postal Code' : 'PostalCode'}, inplace=Tr
geospatial_df.head()
#Merge to get the required dataframe
df_neighborhood_geo = pd.merge(df2, geospatial_df, on='PostalCode')
df_neighborhood_geo.head()

```

Out[2]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M5A	Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636
3	M6A	North York	Lawrence Manor, Lawrence Heights	43.718518	-79.464763
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494

Visualize Functions for boroughs

In [3]:

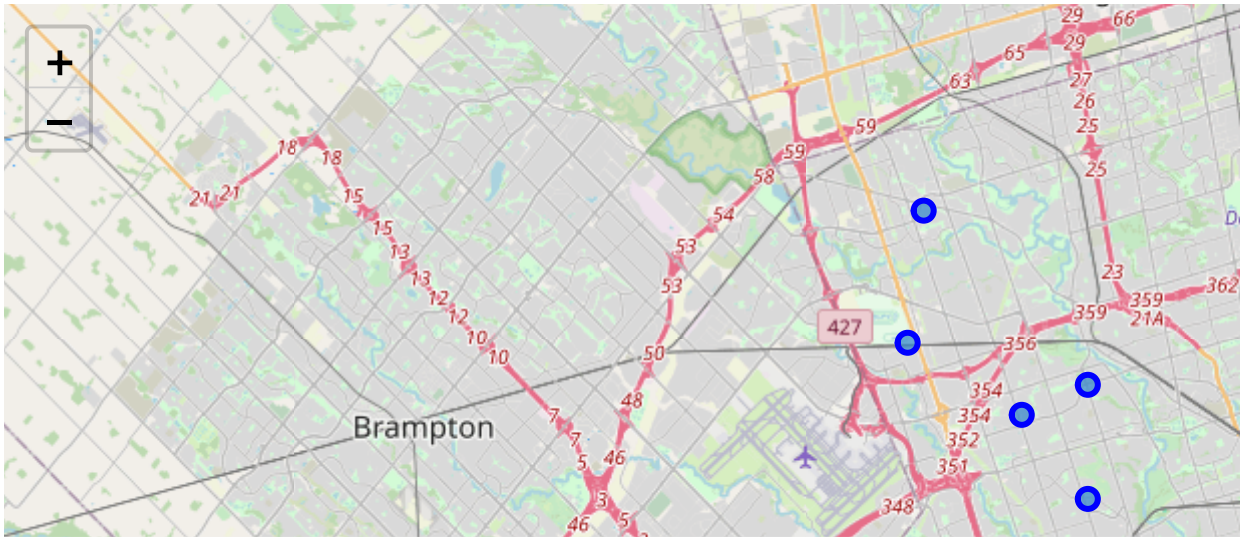
```
def get_lat_long_borough_data(borough_name, borough_address):
    bor_data = neighborhoods[neighborhoods['Borough'].str.contains(borc

    geolocator = Nominatim(user_agent="t_explorer")
    location = geolocator.geocode(borough_address)
    latitude = location.latitude
    longitude = location.longitude
    print('The geograpical coordinate of Borough: ' + borough_address +
    return(bor_data, latitude,longitude)

# create map of borough using latitude and longitude values
def draw_folium_map(borr_data,latitude, longitude):
    map_bor = folium.Map(location=[latitude, longitude], zoom_start=11)

    # add markers to map
    for lat, lng, label in zip(bor_data['Latitude'], bor_data['Longituc
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            color='blue',
            fill=True,
            fill_color='#3186cc',
            fill_opacity=0.7,
            parse_html=False).add_to(map_bor)
    return map_bor
```

Visualize the borough the neighborhoods in it.

In [4]:
```
toronto_data = df_neighborhood_geo[df_neighborhood_geo.Borough.str.cont
toronto_data.reset_index(drop=True, inplace=True)
toronto_data.head()
neighborhoods = toronto_data
neighborhoods['Neighborhood']
# hoods = neighborhoods['Neighborhood'].unique()
# n = 0
# for hood in hoods:
#   nr = hood.split(",")
#   n = n + (len(nr))
#   print (n)


hoods = neighborhoods['Borough'].unique()
for hood in hoods:
  print ('    ' + hood)

(bor_data, latitude, longitude ) = get_lat_long_borough_data('Etobicoke

bor_data.head()
draw_folium_map(bor_data,latitude,longitude)
```

```
   Etobicoke
The geograpical coordinate of Borough: Etobicoke, Toronto are 43.6435559,
-79.5656326.
```

Out[4]:



Foursquare API Tool: To explore the neighborhoods and segment them.

Define Foursquare Credentials and Version

In [5]:
```
CLIENT_ID = 'xxxx' # your Foursquare ID
CLIENT_SECRET = 'xxxxx' # your Foursquare Secret for privacy xxxx
VERSION = '20180605' # Foursquare API version
```

Functions to help with processing venues

Functions to **get_category_type** and **get_med_nearby** to get medical nearby venues from the Foursquare lab.

In [6]:
```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

def get_med_nearby(lat, lon, category, client_id, client_secret, versic
    categories = category if isinstance(category, str) else ','.join(ca
    url = 'https://api.foursquare.com/v2/venues/search?client_id={}&cli
    url
    try:
        results = requests.get(url).json()['response']['venues']
        dataframe = json_normalize(venues)
        dataframe.head()
        # keep only columns that include venue name, and anything that
        filtered_columns = ['name', 'categories','location.lat','locati
        dataframe_filtered = dataframe.loc[:, filtered_columns]
        # filter the category for each row
        dataframe_filtered['categories'] = dataframe_filtered.apply(get
    except:
        dataframe_filtered = pd.DataFrame()
        print(f'\nError on {url}')
    return dataframe_filtered

```

2. Data Processing or Manipulation

Function to repeat the same process to all the neighborhoods in the borough for general and medical venues

In [7]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id=
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['it

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list fo
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)


def getNearbyMedVenues(names, latitudes, longitudes, radius=500):
    client_id = CLIENT_ID
    client_secret = CLIENT_SECRET
    version = VERSION
    limit = LIMIT
    categories = '4bf58dd8d48988d104941735,4bf58dd8d48988d10f951735'
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        url = 'https://api.foursquare.com/v2/venues/search?client_id={}
        results = requests.get(url).json()['response']['venues']
        venues_list.append([(
            name,
            lat,
            lng,
            v['name'],
```

```
            v['location']['lat'],
            v['location']['lng'],
            v['categories'][0]['name']) for v in results])


    nearby_med_venues = pd.DataFrame([item for venue_list in venues_lis
    nearby_med_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_med_venues)
```

Create a new dataframe called *bor_venues*.

In [8]:
```
# type your answer here

bor_venues = getNearbyMedVenues(names=bor_data['Neighborhood'],
                                latitudes=bor_data['Latitude'],
                                longitudes=bor_data['Longitude']
                               )
bor_venues
```

Out[8]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Islington Avenue, Humber Valley Village	43.667856	-79.532242	Shoppers Drug Mart	43.663067	-79.531753	Pharmac
1	Eringate, Bloordale Gardens, Old Burnhamthorpe...	43.643515	-79.577201	Shoppers Drug Mart	43.641312	-79.576924	Pharmac
2	Eringate, Bloordale Gardens, Old Burnhamthorpe...	43.643515	-79.577201	Burnhamthorpe Health Centre	43.642328	-79.576959	Medica Cente
3	Eringate, Bloordale Gardens, Old Burnhamthorpe...	43.643515	-79.577201	Dr Henry Nirenberg Dental Office	43.641895	-79.578301	Dentist' Offic
	Eringate,						

Number of unique categories from all the returned venues

In [9]:
```
print('There are {} uniques categories.'.format(bor_venues['Venue Categ

```

```
There are ['Pharmacy' 'Medical Center' "Dentist's Office" "Doctor's Offic
e"
 'Veterinarian' 'Chiropractor' 'Medical Lab' 'Supplement Shop' 'Hospita
l'] uniques categories.
```

In [10]:
```
print('There are {} uniques categories.'.format(len(bor_venues['Venue C

```

```
There are 9 uniques categories.
```

3. Analyze Each Neighborhood

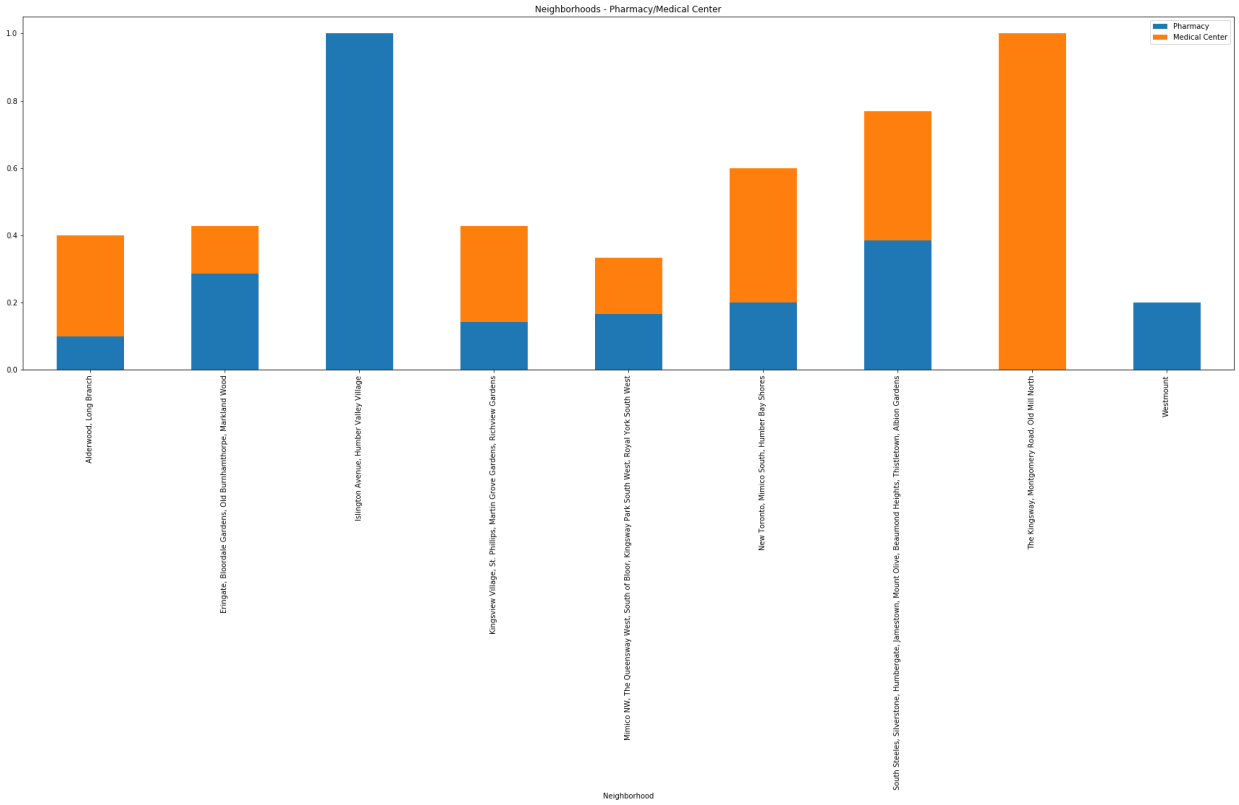
In [11]:
```
# one hot encoding
bor_allonehot = pd.get_dummies(bor_venues[['Venue Category']], prefix="

# add neighborhood column back to dataframe
bor_allonehot['Neighborhood'] = bor_venues['Neighborhood']
bor_allonehot.head()
```

Out[11]:

	Chiropractor	Dentist's Office	Doctor's Office	Hospital	Medical Center	Medical Lab	Pharmacy	Supplement Shop	Veterinaria
0	0	0	0	0	0	0	1	0	
1	0	0	0	0	0	0	1	0	
2	0	0	0	0	1	0	0	0	
3	0	1	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	

Group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [13]:
```
bor_onehot = bor_allonehot[['Neighborhood','Pharmacy']]
bor_grouped = bor_onehot.groupby('Neighborhood').mean().reset_index()
bor_grouped_count = bor_onehot.groupby('Neighborhood').count()

bor_dpeonehot = bor_allonehot[['Neighborhood','Pharmacy', 'Medical Cent
bor_dpegrouped = bor_dpeonehot.groupby('Neighborhood').mean().reset_inc
bor_dpegrouped_count = bor_dpeonehot.groupby('Neighborhood').count()

import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = [30,9]
plt.title('Neighborhoods - Pharmacy/Medical Center')
# # gca stands for 'get current axis'
ax = plt.gca()
bor_dpegrouped.plot(kind='bar',x='Neighborhood', stacked=True, ax=ax)
plt.show()

```



4. Cluster Neighborhoods

Run *k*-means to cluster the neighborhood into 3 clusters.

In [14]:
```
# neighborhoods_venues_sorted.drop('Cluster Labels',inplace=True)
neighborhoods_venues_sorted = bor_grouped
# neighborhoods_venues_sorted.head()
bor_grouped_clustering = bor_grouped.drop('Neighborhood', 1)

# set number of clusters
kclusters = 3
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(bor_grouped_c

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
print(kmeans.labels_)
```

```
[0 2 1 0 0 0 2 0 0]
```

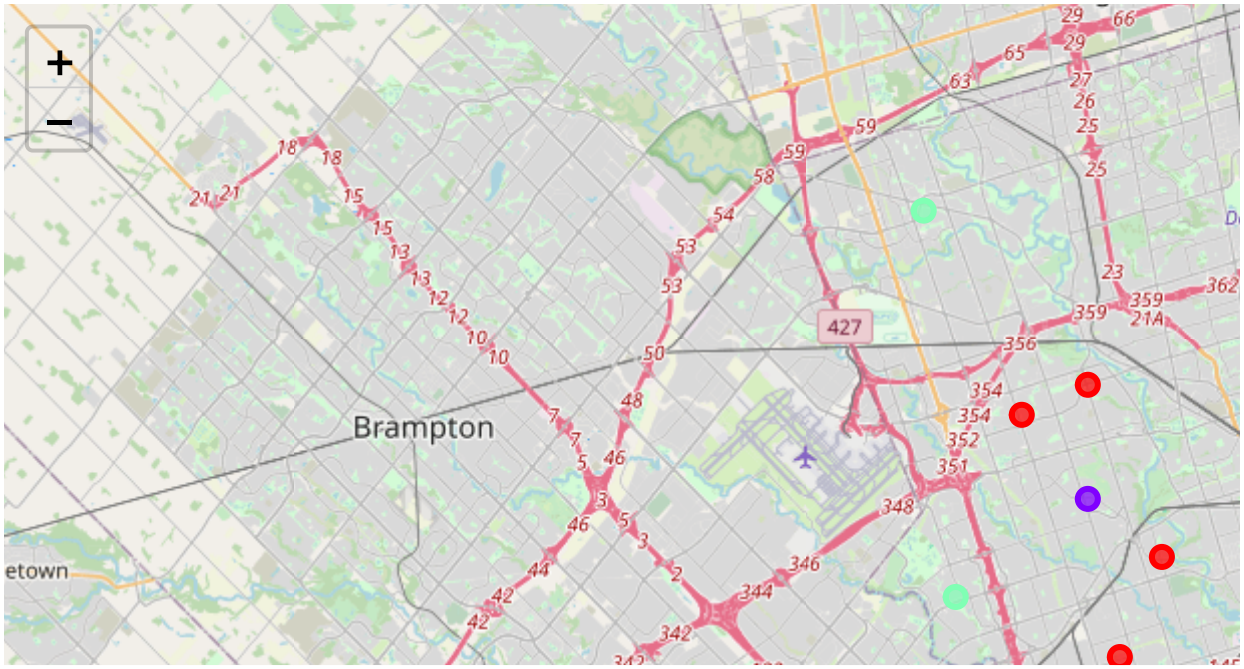
Create a new dataframe with cluster

In [15]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

bor_merged = bor_data
# merge toronto_grouped with toronto_data to add latitude/longitude for
bor_merged = bor_merged.join(neighborhoods_venues_sorted.set_index('Nei
bor_merged.dropna(subset = ["Cluster Labels"], inplace=True)
```

Visualize the resulting clusters

In [16]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, clusterf in zip(bor_merged['Latitude'], bor_merged['
    cluster = int(clusterf)
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_h
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[16]:



5. Examine Clusters

Now, you can examine each cluster and determine the discriminating venue categories that distinguish each cluster. Based on the defining categories, you can then assign a name to each cluster. I will leave this exercise to you.

Cluster 0

In [17]:
```
cluster0 = bor_merged.loc[bor_merged['Cluster Labels'] == 0.0, bor_merg

```

Cluster 1

In [18]:
```
cluster1 = bor_merged.loc[bor_merged['Cluster Labels'] == 1.0, bor_merg
cluster1
```

Out[18]:

	Borough	Cluster Labels	Pharmacy
0	Etobicoke	1.0	1.0

Cluster 2

In [19]:
```
cluster2 = bor_merged.loc[bor_merged['Cluster Labels'] == 2.0, bor_merg

```

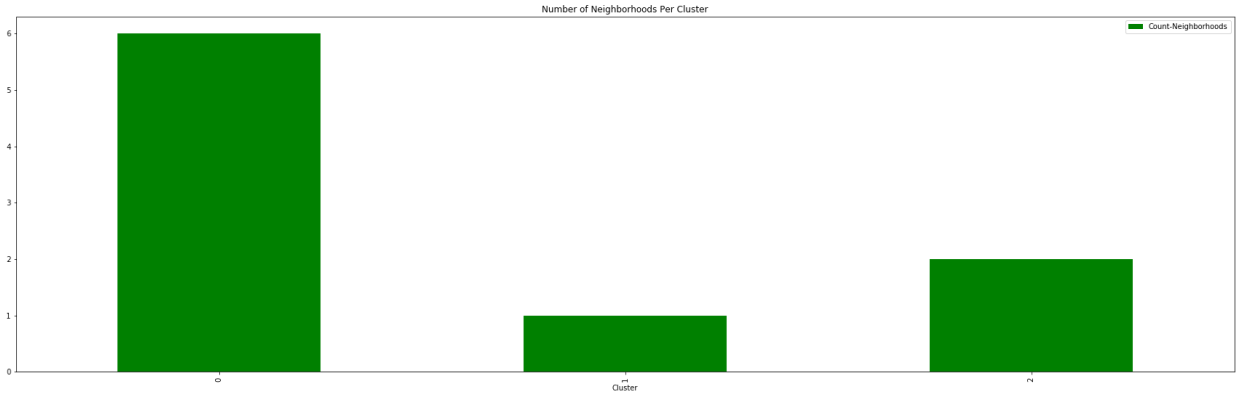
In [20]:
```
cluster0 = bor_merged.loc[bor_merged['Cluster Labels'] == 0.0, bor_merg

cluster1 = bor_merged.loc[bor_merged['Cluster Labels'] == 1.0, bor_merg

cluster2 = bor_merged.loc[bor_merged['Cluster Labels'] == 2.0, bor_merg

def get_cluster_hoods(bor_merged,clusternum):
    return bor_merged.loc[bor_merged['Cluster Labels'] == clusternum]

cluster_stats1 = []
for x in range(kclusters):
    dfcluster = get_cluster_hoods(bor_merged,x)
    print('Cluster:', x, 'Avg # Pharmacies:', dfcluster['Pharmacy'].mea
    cluster_stats1.append([(
            x,
            dfcluster['Pharmacy'].mean(),
            dfcluster.shape[0])])
cluster_stats1
```

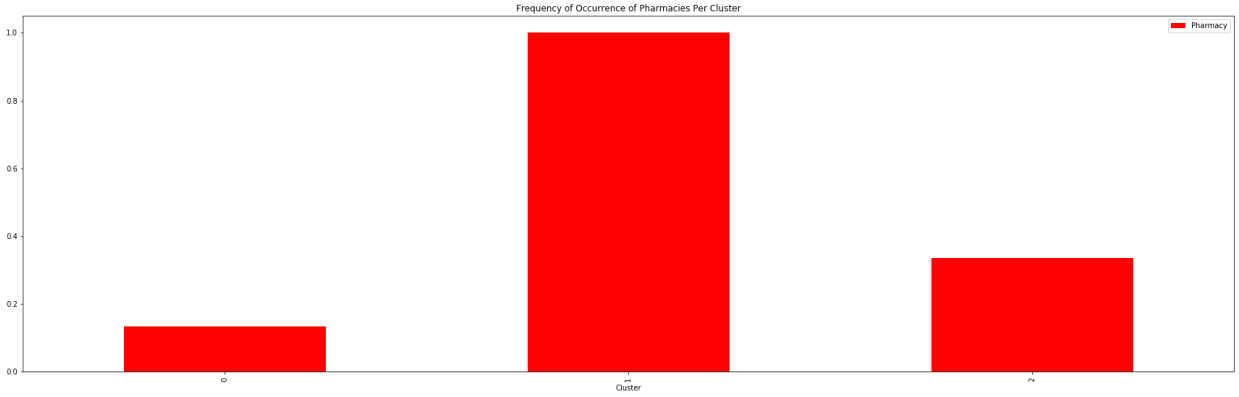
```
Cluster: 0 Avg # Pharmacies: 0.1349206349206349 #Neighborhoods: 6
Cluster: 1 Avg # Pharmacies: 1.0 #Neighborhoods: 1
Cluster: 2 Avg # Pharmacies: 0.3351648351648352 #Neighborhoods: 2
```

Out[20]:
```
[[(0, 0.1349206349206349, 6)], [(1, 1.0, 1)], [(2, 0.3351648351648352,
2)]]
```

In [21]:

```
df = pd.DataFrame([item for x in cluster_stats1 for item in x],columns=
df
# gca stands for 'get current axis'
ax = plt.gca()
df.plot(kind='bar',x='Cluster',y='Count-Neighborhoods', stacked=True,cc
plt.title('Number of Neighborhoods Per Cluster')

plt.show()

ax = plt.gca()

df.plot(kind='bar',x='Cluster',y='Pharmacy', stacked=True,color='red',
plt.title('Frequency of Occurrence of Pharmacies Per Cluster')

plt.show()

```





In [22]:
```
cluster0 = bor_merged.loc[bor_merged['Cluster Labels'] == 0.0]
cluster1 = bor_merged.loc[bor_merged['Cluster Labels'] == 1.0]
cluster2 = bor_merged.loc[bor_merged['Cluster Labels'] == 2.0]
cluster0
```

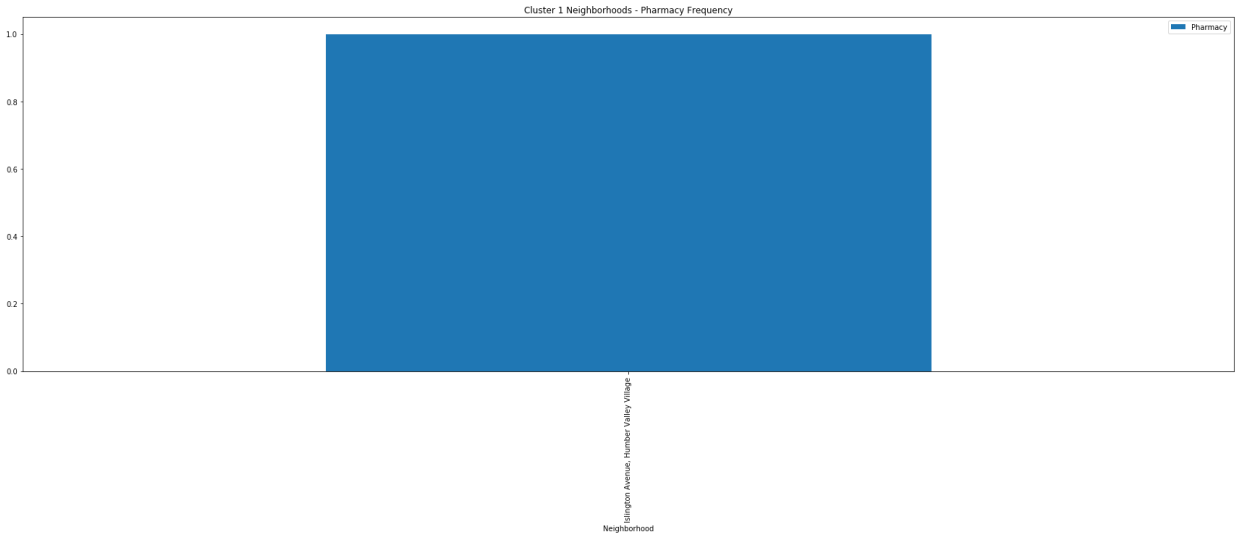
Out[22]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	Pharmacy
3	M9P	Etobicoke	Westmount	43.696319	-79.532242	0.0	0.200000
4	M9R	Etobicoke	Kingsview Village, St. Phillips, Martin Grove ...	43.688905	-79.554724	0.0	0.142857
5	M8V	Etobicoke	New Toronto, Mimico South, Humber Bay Shores	43.605647	-79.501321	0.0	0.200000
7	M8W	Etobicoke	Alderwood, Long Branch	43.602414	-79.543484	0.0	0.100000
9	M8X	Etobicoke	The Kingsway, Montgomery Road, Old Mill North	43.653654	-79.506944	0.0	0.000000
11	M8Z	Etobicoke	Mimico NW, The Queensway West, South of Bloor,...	43.628841	-79.520999	0.0	0.166667

Cluster 0 Neighborhoods (Low in Pharmacy from Modeling)

In [23]:
```
cluster1.plot(x ='Neighborhood', y='Pharmacy', kind = 'bar', title='Clu

```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20f4a290>



Create data for neighborhoods with pharmacy and medical center in cluster 0

In [24]:
```
bor_dpeonehot_merge = pd.merge(bor_dpegrouped, cluster0, on='Neighborhc
bor_dpeonehot_merge
```

Out[24]:

	Neighborhood	Pharmacy_x	Medical Center	PostalCode	Borough	Latitude	Longitude	Cluster Labels	Ph
0	Alderwood, Long Branch	0.100000	0.300000	M8W	Etobicoke	43.602414	-79.543484	0.0	
1	Kingsview Village, St. Phillips, Martin Grove ...	0.142857	0.285714	M9R	Etobicoke	43.688905	-79.554724	0.0	
2	Mimico NW, The Queensway West, South of Bloor,...	0.166667	0.166667	M8Z	Etobicoke	43.628841	-79.520999	0.0	
3	New Toronto, Mimico South, Humber Bay Shores	0.200000	0.400000	M8V	Etobicoke	43.605647	-79.501321	0.0	
4	The Kingsway, Montgomery Road, Old Mill North	0.000000	1.000000	M8X	Etobicoke	43.653654	-79.506944	0.0	
5	Westmount	0.200000	0.000000	M9P	Etobicoke	43.696319	-79.532242	0.0	

In [ ]:
```
```