Capstone Project: Battle of Neighborhoods

Tourism Guide for a Singapore Visitor

Purpose: This is my detailed final peer reviewed assignment for the IBM Data Science Professional Certificate program – Coursera Capstone.

INTRODUCTION: Singapore is a city country and one of the most visited places in Asia. There are number of travellers who seek information about Singapore while planning a visit to the country. They look for information like places to visit, travel mode, shoping avenues and stay during their visit. This project is built to provide a data centric recommendation that can enhance the correctness of the recommendation based on available data.

The sample guide in this notebook will provide the following use case scenario:

```
A person planning to visit Singapore as a Tourist or an foreigner and looking for a re
asonable accommodation.
The user wants to receive venue recommendation where he can stay or rent an HDB apartm
ent with close proximity to places of interest or search category option.
The recommendation should not only provide the most viable choice, but should also pre
sent a comparison table of all possible venues.
```

DATA DOWNLOAD We will make use of following data sources: Singapore Towns and median residential rental prices.

```
Data is retrieved from Singapore open dataset from median rent by town and flattype fr
om https://data.gov.sg website.
```

The original data source contains median rental prices of Singapore HDB units from 2005 up to 2nd quarter of 2018. I will retrieve rental the most recent recorded rental prices from this data source (Q2 2018) being the most relevant price available at this time. For this demonstration, I will simplify the analysis by using the average rental prices of all available flat type. Singapore Towns location data retrieved using Google maps API.

Data coordinates of Neighborhood Venues will be retrieved using google API. I also make use of MRT stations coordinate as a more important center of for all towns included in venue recommendations.

We will be using the FourSquare API to explore neighborhoods in selected towns in Singapore. The Foursquare explore function will be used to get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters. The following information are retrieved on the first query:

```
Venue ID
Venue Name
Coordinates : Latitude and Longitude
Category Name
```

1. METHODOLOGY : Singapore Towns List with median residential rental prices.

The source data contains median rental prices of Singapore HDB units from 2005 up to 2nd quarter of 2018. I will retrive the most recent recorded rental prices from this data source (Q2 2018) being the most relevant price available at this time. For this demonstration, I will simplify the analysis by using the average rental prices of all available flat type.

Data Cleanup and re-grouping. The retrieved table contains some un-wanted entries and needs some cleanup.

The following tasks will be performed:

```
Drop/ignore cells with missing data.
Use most current data record.
Fix data types.
```

Processed Singapore towns list with and median residential rental prices Town median_rent 0 ANG MO KIO 2033.333333 1 BEDOK 2087.500000 2 BISHAN 2233.333333 3 BUKIT BATOK 1962.500000 4 BUKIT MERAH 2162.500000 5 BUKIT PANJANG 1737.500000 6 CENTRAL 2450.000000 7 CHOA CHU KANG 1933.333333 8 CLEMENTI 2263.333333 9 GEYLANG 2166.666667 10 HOUGANG 1962.500000 11 JURONG EAST 2150.000000 12 JURONG WEST 1975.000000 13 KALLANG/WHAMPOA 2300.000000 14 MARINE PARADE 1950.000000 15 PASIR RIS 2066.666667 16 PUNGGOL 1825.000000 17 QUEENSTOWN 2162.500000 18 SEMBAWANG 1883.333333 19 SENGKANG 1900.000000 20 SERANGOON 2187.500000 21 TAMPINES 2075.000000 22 TOA PAYOH 2210.000000 23 WOODLANDS 1762.500000 24 YISHUN 1900.000000

```
Adding geographical coordinates of each town location.
```

1. Retrieve town coordinates.

Google apiwas be used to retrive the coordinates (latitude and longitude of each town centers. For this exercise, I just used the MRT stations as the center points of each evaluated towns. The town coordinates will be used in retrieval of Foursquare API location data.

singapore_average_rental_prices_by_town['Latitude'] = 0.0 singapore_average_rental_prices_by_town['Longitude'] = 0.0

for idx,town in singapore_average_rental_prices_by_town['Town'].iteritems(): address = town + " MRT station, Singapore" ; # I prefer to use MRT stations as more important central location of each town url = 'https://maps.googleapis.com/maps/api/geocode/json?address={}&key={}'.format(address,google_key (https://maps.googleapis.com/maps/api/geocode/json?address={}&key={}'.format(address,google_key)) lat = requests.get(url).json()["results"][0]["geometry"]["location"]['lat'] lng = requests.get(url).json()["results"][0]["geometry"]["location"]['lng'] singapore_average_rental_prices_by_town.loc[idx,'Latitude'] = lat singapore_average_rental_prices_by_town.loc[idx,'Longitude'] = lng

Singapore Median Rental Prices per Town merged with Coordinate Data median_rent Latitude Longitude Town ANG MO KIO 2033.333333 1.369972 103.849588 BEDOK 2087.500000 1.324011 103.930172 BISHAN 2233.333333 1.351042 103.849930 BUKIT BATOK 1962.500000 1.348506 103.749222 BUKIT MERAH 2162.500000 1.289642 103.816798 BUKIT PANJANG 1737.500000 1.276068 103.791904 CENTRAL 2450.000000 1.288155 103.846718 CHOA CHU KANG 1933.333333 1.385385 103.744337 CLEMENTI 2263.333333 1.315070 103.765246 GEYLANG 2166.666667 1.316367 103.882772 HOUGANG 1962.500000 1.371331 103.892544 JURONG EAST 2150.000000 1.333143 103.742329 JURONG WEST 1975.000000 1.338556 103.705828 KALLANG/WHAMPOA 2300.000000 1.311478 103.871351 MARINE PARADE 1950.000000 1.308410 103.888814 PASIR RIS 2066.666667 1.373191 103.949353 PUNGGOL 1825.000000 1.405170 103.902356 QUEENSTOWN 2162.500000 1.294835 103.805902 SEMBAWANG 1883.333333 1.449080 103.820058 SENGKANG 1900.000000 1.391661 103.895453 SERANGOON 2187.500000 1.349787 103.873635 TAMPINES 2075.000000 1.354430 103.942760 TOA PAYOH 2210.000000 1.332330 103.847425 WOODLANDS 1762.500000 1.436945 103.786516 YISHUN 1900.000000 1.429548 103.835033 V. Segmenting and Clustering Towns in Singapore Retrieving FourSquare Places of interest.

Using the Foursquare API, the explore API function was be used to get the most common venue categories in each

Segmenting and Clustering Neighborhoods in Singapore

Retrieving FourSquare Places of interest.

Using the Foursquare API, the explore API function was be used to get the most common venue categories in each neighborhood, and then used this feature to group the neighborhoods into clusters. The k-means clustering algorithm was used for the analysis. Fnally, the Folium library is used to visualize the recommended neighborhoods and their emerging clusters.

In the ipynb notebook, the function getNearbyVenues extracts the following information for the dataframe it generates:

```
Venue ID
Venue Name
Coordinates : Latitude and Longitude
Category Name
```

The function getVenuesByCategory performs the following:

```
category based venue search to simulate user venue searches based on certain places of
interest. This search extracts the following information:
    Venue ID
    Venue Name
    Coordinates : Latitude and Longitude
    Category Name
For each retrieved venueID, retrive the venues category rating.
```

In [ ]:

```
Discussion and Conclusion

On this notebook, Analysis of best neighborhood venue based on Food venue category
has been presented. Recommendations based on other user searches
like available outdoor and recreation areas can also be done. As singapore is a cit
y state with a whole host of interesting venues spread around
the neighborhood, the information extracted in this document, will be a good additi
on to web based recommendations for visitors to find out venues
of interest and be a useful aid in deciding a place to stay or where to go during t
heir visits.

    Using Foursquare API, we have collected a good amount of venue recommnedations
in Singapore Neighborhood. Sourcing from the venue recommendations
    from FourSquare has some limitation, The list of venues is not exhaustive list
of all the available venues is the area. Also, not every venue
    found in the the area has a stored ratings. For this reason, the number of anal
yzed venues are only half of all the available venues.

The generated clusters from our results show that there are interesting and excelle
nt places in areas where the median rents are cheaper.
This kind of results may be very interesting for visitors who also look for budget
options. Our results also presented some interesting findings,
like the initial assumption among websites providing recommendations is that the Ce
ntral Area that have the highest median rent also have better
food venues. The results however shows that while Marine Parade, a cheaper location
has better rated food courts. Result shows that most popular
food venue among Singaporeans, residents and visitors are Food Courts, Coffee Shops
and Fast Food Restaurants.
```

In [ ]:

```
Appendix
```

Purpose: This is my detailed final peer reviewed assignment for the IBM Data Science Professional Certificate program – Coursera Capstone.

INTRODUCTION: Singapore is a city country and one of the most visited places in Asia. There are number of travellers who seek information about Singapore while planning a visit to the country. They look for information like places to visit, travel mode, shoping avenues and stay during their visit. This project is built to provide a data centric recommendation that can enhance the correctness of the recommendation based on available data.

First we will import libraries required for the task

Note: We can import additional libraries wherever required

In [1]:
```
!conda install -c conda-forge folium=0.5.0 --yes # comment/uncomment if not yet ins
talled.
!conda install -c conda-forge geopy --yes       # comment/uncomment if not yet ins
talled

import numpy as np # library to handle data in a vectorized manner
import pandas as pd # library for data analsysis

# Numpy and Pandas libraries were already imported at the beginning of this noteboo
k.
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files
from geopy.geocoders import Nominatim # convert an address into latitude and longit
ude values
from pandas.io.json import json_normalize # tranform JSON file into a pandas datafr
ame
# 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
import folium # map rendering library

import requests # library to handle requests
import lxml.html as lh
import bs4 as bs
import urllib.request
print('Libraries imported.')
```

```
Fetching package metadata .............
Solving package specifications: .

# All requested packages already installed.
# packages in environment at /opt/conda/envs/DSX-Python35:
#
folium                    0.5.0                      py_0    conda-forge
Fetching package metadata .............
Solving package specifications: .

# All requested packages already installed.
# packages in environment at /opt/conda/envs/DSX-Python35:
#
geopy                     1.19.0                     py_0    conda-forge
Libraries imported.
```

In [2]:
```
from IPython.display import HTML
import base64

# Extra Helper scripts to generate download links for saved dataframes in csv forma
t.
def create_download_link( df, title = "Download CSV file", filename = "data.csv"):
    csv = df.to_csv()
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="
_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)
```

1. Downloading Singapore towns list with and median residential rental prices

In [3]:
```
import zipfile
import os
!wget -q -O 'median-rent-by-town-and-flat-type.zip' "https://data.gov.sg/dataset/b3
5046dc-7428-4cff-968d-ef4c3e9e6c99/download"
zf = zipfile.ZipFile('./median-rent-by-town-and-flat-type.zip')
sgp_median_rent_by_town_data = pd.read_csv(zf.open("median-rent-by-town-and-flat-ty
pe.csv"))
sgp_median_rent_by_town_data.rename(columns = {'town':'Town'}, inplace = True)
sgp_median_rent_by_town_data.head()
```

Out[3]:

	quarter	Town	flat_type	median_rent
0	2005-Q2	ANG MO KIO	1-RM	na
1	2005-Q2	ANG MO KIO	2-RM	na
2	2005-Q2	ANG MO KIO	3-RM	800
3	2005-Q2	ANG MO KIO	4-RM	950
4	2005-Q2	ANG MO KIO	5-RM	-

Data Cleanup and re-grouping.

The retrieved table contains some un-wanted entries and needs some cleanup. The following tasks will be performed:

```
Drop/ignore cells with missing data.
Use most recent data record.
```

In [4]:
```
# Drop rows with rental price == 'na'.
sgp_median_rent_by_town_data_filter=sgp_median_rent_by_town_data[~sgp_median_rent_b
y_town_data['median_rent'].isin(['-','na'])]

# Take the most recent report which is "2018-Q2"
sgp_median_rent_by_town_data_filter=sgp_median_rent_by_town_data_filter[sgp_median_
rent_by_town_data_filter['quarter'] == "2018-Q2"]

# Now that all rows reports are "2018-Q2", we dont need this column anymore.
sgp_median_rent_by_town_data_filter=sgp_median_rent_by_town_data_filter.drop(['quar
ter'], axis=1)

# Ensure that median_rent column is float64.
sgp_median_rent_by_town_data_filter['median_rent']=sgp_median_rent_by_town_data_fil
ter['median_rent'].astype(np.float64)
```

In [5]:
```
singapore_average_rental_prices_by_town = sgp_median_rent_by_town_data_filter.group
by(['Town'])['median_rent'].mean().reset_index()
singapore_average_rental_prices_by_town
```

Out[5]:

	Town	median_rent
0	ANG MO KIO	2033.333333
1	BEDOK	2087.500000
2	BISHAN	2233.333333
3	BUKIT BATOK	1962.500000
4	BUKIT MERAH	2162.500000
5	BUKIT PANJANG	1737.500000
6	CENTRAL	2450.000000
7	CHOA CHU KANG	1933.333333
8	CLEMENTI	2263.333333
9	GEYLANG	2166.666667
10	HOUGANG	1962.500000
11	JURONG EAST	2150.000000
12	JURONG WEST	1975.000000
13	KALLANG/WHAMPOA	2300.000000
14	MARINE PARADE	1950.000000
15	PASIR RIS	2066.666667
16	PUNGGOL	1825.000000
17	QUEENSTOWN	2162.500000
18	SEMBAWANG	1883.333333
19	SENGKANG	1900.000000
20	SERANGOON	2187.500000
21	TAMPINES	2075.000000
22	TOA PAYOH	2210.000000
23	WOODLANDS	1762.500000
24	YISHUN	1900.000000

Adding geographical coordinates of each town location.

In [65]:
```
    geo = Nominatim(user_agent='Mypythonapi')
    for idx,town in singapore_average_rental_prices_by_town['Town'].iteritems():
        coord = geo.geocode(town + ' ' + "Singapore", timeout = 10)
        if coord:
            singapore_average_rental_prices_by_town.loc[idx,'Latitude'] = coord.lat
itude
            singapore_average_rental_prices_by_town.loc[idx,'Longitude'] = coord.lo
ngitude
        else:
            singapore_average_rental_prices_by_town.loc[idx,'Latitude'] = NULL
            singapore_average_rental_prices_by_town.loc[idx,'Longitude'] = NULL
```

In [66]:
```
singapore_average_rental_prices_by_town.set_index("Town")
```

Out[66]:

	median_rent	Latitude	Longitude
Town			
ANG MO KIO	2033.333333	1.369842	103.846609
BEDOK	2087.500000	1.323976	103.930216
BISHAN	2233.333333	1.351455	103.848263
BUKIT BATOK	1962.500000	1.349057	103.749591
BUKIT MERAH	2162.500000	1.280628	103.830591
BUKIT PANJANG	1737.500000	1.377921	103.771866
CENTRAL	2450.000000	1.290475	103.852036
CHOA CHU KANG	1933.333333	1.389260	103.743728
CLEMENTI	2263.333333	1.314026	103.762410
GEYLANG	2166.666667	1.318186	103.887056
HOUGANG	1962.500000	1.373360	103.886091
JURONG EAST	2150.000000	1.333115	103.742297
JURONG WEST	1975.000000	1.339636	103.707339
KALLANG/WHAMPOA	2300.000000	1.319116	103.866291
MARINE PARADE	1950.000000	1.302689	103.907395
PASIR RIS	2066.666667	1.375989	103.954360
PUNGGOL	1825.000000	1.405255	103.902354
QUEENSTOWN	2162.500000	1.294623	103.806045
SEMBAWANG	1883.333333	1.448065	103.820760
SENGKANG	1900.000000	1.390949	103.895175
SERANGOON	2187.500000	1.349807	103.873771
TAMPINES	2075.000000	1.354653	103.943571
TOA PAYOH	2210.000000	1.335391	103.849741
WOODLANDS	1762.500000	1.436897	103.786216
YISHUN	1900.000000	1.428136	103.833694

Now that we have latitude and longitude of Singapore we generate a basic map of Singapore

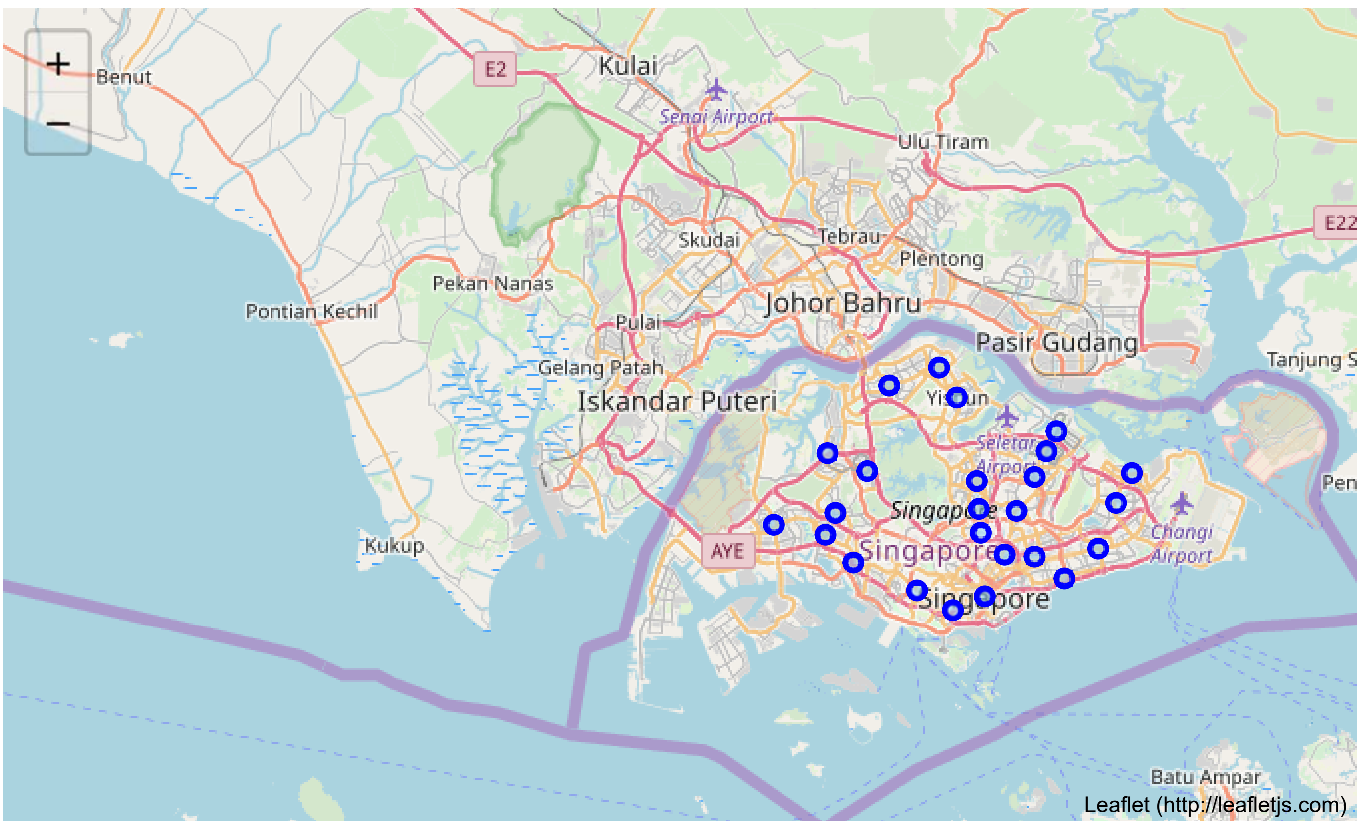
In [8]:
```
geo = Nominatim(user_agent='My-IBMNotebook')
address = 'Singapore'
location = geo.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geographical coordinate of Singapore {}, {}.'.format(latitude, longitude)
)

# create map of Singapore using latitude and longitude values
map_singapore = folium.Map(location=[latitude, longitude],tiles="OpenStreetMap", zo
om_start=10)

# add markers to map
for lat, lng, town in zip(
    singapore_average_rental_prices_by_town['Latitude'],
    singapore_average_rental_prices_by_town['Longitude'],
    singapore_average_rental_prices_by_town['Town']):
    label = town
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=4,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#87cefa',
        fill_opacity=0.5,
        parse_html=False).add_to(map_singapore)
map_singapore
```

```
The geographical coordinate of Singapore 1.2904753, 103.8520359.
```

Out[8]:



In [9]:
```
fileName = "singapore_average_rpbt.csv"
linkName = "Singapore Average Rental Prices"
create_download_link(singapore_average_rental_prices_by_town,linkName,fileName)
```

Out[9]: Singapore Average Rental Prices
(data:text/csv;base64,LFRvd24sbWVkaWFuX3JlbnQsTGF0aXR1ZGUsTG9uZ2l0dWRlCjAsQU5HIE1PIEtJTyw

Segmenting and Clustering Towns in Singapore

Retrieving FourSquare Places of interest.

Storing my Foursquare credentials in variables

In [10]:
```
CLIENT_ID = '23NES53FFNN3HHE3UFXX2IJYTH3JGQU3YMHPEVKXVEBAEWB3' # your Foursquare ID
CLIENT_SECRET = '4A1Z0ENHPF4ZF5MG2KXZ4J3HELYTZNJ0SP42CNEROGZPJLHF' # your Foursquar
e Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: 23NES53FFNN3HHE3UFXX2IJYTH3JGQU3YMHPEVKXVEBAEWB3
CLIENT_SECRET:4A1Z0ENHPF4ZF5MG2KXZ4J3HELYTZNJ0SP42CNEROGZPJLHF
```

Getting coordinates of Singapore City

In [11]:
```
address = 'Singapore'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Singapore are {}, {}.'.format(latitude, longit
ude))
```

```
The geograpical coordinate of Singapore are 1.2904753, 103.8520359.
```

Creating foursquare url to get venues in the radius of 500 meters

In [118]:
```
LIMIT = 100

radius = 500 # define radius

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}
&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    location.latitude,
   location.longitude,
    radius,
    LIMIT)
url # display URL
```

Out[118]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=23NES53FFNN3HHE3UFXX2IJ
YTH3JGQU3YMHPEVKXVEBAEWB3&client_secret=4A1Z0ENHPF4ZF5MG2KXZ4J3HELYTZNJ0SP42CNER
OGZPJLHF&v=20180605&ll=1.2904753,103.8520359&radius=500&limit=100'
```

The following function retrieves the venues given the names and coordinates and stores it into dataframe.

In [119]:
```
import time

FOURSQUARE_EXPLORE_URL = 'https://api.foursquare.com/v2/venues/explore?'
FOURSQUARE_SEARCH_URL = 'https://api.foursquare.com/v2/venues/search?'

def getNearbyVenues(names, latitudes, longitudes, radius=500):
    global CLIENT_ID
    global CLIENT_SECRET
    global FOURSQUARE_EXPLORE_URL
    global FOURSQUARE_SEARCH_URL
    global VERSION
    global LIMIT
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print('getNearbyVenues',names)
        cyclefsk2()

        url = '{}&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.
format(
            FOURSQUARE_EXPLORE_URL,CLIENT_ID,CLIENT_SECRET,VERSION,
            lat,lng,radius,LIMIT)

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

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,lat,lng,
            v['venue']['id'],v['venue']['name'],
            v['venue']['location']['lat'],v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])
        time.sleep(2)

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in v
enue_list])
    nearby_venues.columns = ['Town','Town Latitude','Town Longitude','Venue','Venu
e Latitude','Venue Longitude','Venue Category']

    return(nearby_venues)
```

In [14]:
```
FOURSQUARE_SEARCH_URL = 'https://api.foursquare.com/v2/venues/search?'
# SEARCH VENUES BY CATEGORY

# Dataframe : venue_id_recover
# - store venue id to recover failed venues id score retrieval later if foursquare
limit is exceeded when getting score.
venue_id_rcols = ['VenueID']
venue_id_recover = pd.DataFrame(columns=venue_id_rcols)
def getVenuesByCategory(names, latitudes, longitudes, categoryID, radius=500):
    global CLIENT_ID
    global CLIENT_SECRET
    global FOURSQUARE_EXPLORE_URL
    global FOURSQUARE_SEARCH_URL
    global VERSION
    global LIMIT
    venue_columns = ['Town','Town Latitude','Town Longitude','VenueID','VenueName',
'score','category','catID','latitude','longitude']
    venue_DF = pd.DataFrame(columns=venue_columns)
    print("[#Start getVenuesByCategory]")
    for name, lat, lng in zip(names, latitudes, longitudes):
        cyclefsk2()
        print(name,",",end='')
        #print('getVenuesByCategory',categoryID,name) ; # DEBUG: be quiet
        # create the API request URL
        url = '{}client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&cat
egoryId={}'.format(
            FOURSQUARE_SEARCH_URL,CLIENT_ID,CLIENT_SECRET,VERSION,lat,lng,radius,LI
MIT,categoryID)
        # make the GET request
        results = requests.get(url).json()
        # Populate dataframe with the category venue results
        # Extracting JSON  data values

        for jsonSub in results['response']['venues']:
            #print(jsonSub)
            # JSON Results may not be in expected format or incomplete data, in tha
t case, skip!
            ven_id = 0
            try:
                # If there are any issue with a restaurant, retry or ignore and con
tinue
                # Get location details
                ven_id   = jsonSub['id']
                ven_cat  = jsonSub['categories'][0]['pluralName']
                ven_CID  = jsonSub['categories'][0]['id']
                ven_name = jsonSub['name']
                ven_lat  = jsonSub['location']['lat']
                ven_lng  = jsonSub['location']['lng']
                venue_DF = venue_DF.append({
                    'Town'       : name,
                    'Town Latitude' : lat,
                    'Town Longitude': lng,
                    'VenueID'    : ven_id,
                    'VenueName' : ven_name,
                    'score'     : 'nan',
                    'category'  : ven_cat,
```

Store venue id to recover failed venues id score retrieval later if foursquare limit is exceeded when getting score.

In [120]:
```
FOURSQUARE_SEARCH_URL = 'https://api.foursquare.com/v2/venues/search?'

venue_id_rcols = ['VenueID','Score']
venue_id_recover = pd.DataFrame(columns=venue_id_rcols)

def getVenuesIDScore(venueID):
    global CLIENT_ID
    global CLIENT_SECRET
    global FOURSQUARE_EXPLORE_URL
    global FOURSQUARE_SEARCH_URL
    global VERSION
    global LIMIT
    global venue_id_recover
    print("[#getVenuesIDScore]")
    venID_URL = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secre
t={}&v={}'.format(venueID,CLIENT_ID,CLIENT_SECRET,VERSION)
    print(venID_URL)
    venID_score = 0.00
    # Process results
    try:
        venID_result = requests.get(venID_URL).json()
        venID_score  = venID_result['response']['venue']['rating']
    except:
        venue_id_recover = venue_id_recover.append({'VenueID' : venueID, 'Score' :
0.0})
        cyclefsk2()
        return ["error",0.0]
    return ["success",venID_score]
```

In [121]:
```
singapore_average_rental_prices_by_town.dtypes
```

Out[121]:
```
Town             object
median_rent     float64
Latitude        float64
Longitude       float64
dtype: object
```

In [122]:
```
venue_columns = ['Town','Town Latitude','Town Longitude','VenueID','VenueName','sc
ore','category','catID','latitude','longitude']
singapore_town_venues = pd.DataFrame(columns=venue_columns)
```

Search Venues with recommendations on : Food Venues (Restaurants,Fastfoods, etc.)

To demonstrate user selection of places of interest, We will use this Food Venues category in our further analysis.

```
This Foursquare search is expected to collect venues in the following category:
    category
    Food Courts
    Coffee Shops
    Restaurants
    Cafés
    Other food venues
```

In [123]:
```
# Food Venues : Restaurants, Fastfoods, Etc
# For testing
if (0):
    categoryID = "4d4b7105d754a06377d81259"
    town_names = ['ANG MO KIO']
    lat_list   = [1.3699718]
    lng_list   = [103.8495876]
    tmp = getVenuesByCategory(names=town_names,latitudes=lat_list,longitudes=lng_l
ist,categoryID=categoryID)
    singapore_town_venues = pd.concat([singapore_town_venues,tmp], ignore_index=Tr
ue)
```

In [124]:
```
results = requests.get(url).json()
results
```

```
Out[124]: {'meta': {'code': 200, 'requestId': '5ca107de4434b961752ca80d'},
           'response': {'groups': [{'items': [{'reasons': {'count': 0,
                 'items': [{'reasonName': 'globalInteractionReason',
                   'summary': 'This spot is popular',
                   'type': 'general'}]},
                'referralId': 'e-0-4d438c6514aa8cfa743d5c3d-0',
                'venue': {'categories': [{'icon': {'prefix': 'https://ss3.4sqi.net/img/cat
          egories_v2/arts_entertainment/artgallery_',
                     'suffix': '.png'},
                    'id': '4bf58dd8d48988d1e2931735',
                    'name': 'Art Gallery',
                    'pluralName': 'Art Galleries',
                    'primary': True,
                    'shortName': 'Art Gallery'}],
                  'id': '4d438c6514aa8cfa743d5c3d',
                  'location': {'address': "1 St. Andrew's Road",
                   'cc': 'SG',
                   'city': 'Singapore',
                   'country': 'Singapore',
                   'distance': 61,
                   'formattedAddress': ["1 St. Andrew's Road", '178957', 'Singapore'],
                   'labeledLatLngs': [{'label': 'display',
                     'lat': 1.2907395913341984,
                     'lng': 103.85154786540198}],
                   'lat': 1.2907395913341984,
                   'lng': 103.85154786540198,
                   'postalCode': '178957'},
                  'name': 'National Gal\xadlery Singa\xadpore',
                  'photos': {'count': 0, 'groups': []}}},
               {'reasons': {'count': 0,
                 'items': [{'reasonName': 'globalInteractionReason',
                   'summary': 'This spot is popular',
                   'type': 'general'}]},
                'referralId': 'e-0-4b058810f964a52036af22e3-1',
                'venue': {'categories': [{'icon': {'prefix': 'https://ss3.4sqi.net/img/cat
          egories_v2/parks_outdoors/park_',
                     'suffix': '.png'},
                    'id': '4bf58dd8d48988d163941735',
                    'name': 'Park',
                    'pluralName': 'Parks',
                    'primary': True,
                    'shortName': 'Park'}],
                  'id': '4b058810f964a52036af22e3',
                  'location': {'address': 'Connaught Dr.',
                   'cc': 'SG',
                   'city': 'Singapore',
                   'country': 'Singapore',
                   'crossStreet': 'Opp Padang & City Hall',
                   'distance': 240,
                   'formattedAddress': ['Connaught Dr. (Opp Padang & City Hall)',
                    '179558',
                    'Singapore'],
                   'labeledLatLngs': [{'label': 'display',
                     'lat': 1.2889675708353954,
                     'lng': 103.85358044117562}],
                   'lat': 1.2889675708353954,
                   'lng': 103.85358044117562
```

In [125]:
```
# 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']
```

In [126]:
```
results = requests.get(url).json()
results
```

```
Out[126]: {'meta': {'code': 200, 'requestId': '5ca108199fb6b73b73218a7e'},
           'response': {'groups': [{'items': [{'reasons': {'count': 0,
                 'items': [{'reasonName': 'globalInteractionReason',
                   'summary': 'This spot is popular',
                   'type': 'general'}]},
                'referralId': 'e-0-4d438c6514aa8cfa743d5c3d-0',
                'venue': {'categories': [{'icon': {'prefix': 'https://ss3.4sqi.net/img/cat
          egories_v2/arts_entertainment/artgallery_',
                    'suffix': '.png'},
                   'id': '4bf58dd8d48988d1e2931735',
                   'name': 'Art Gallery',
                   'pluralName': 'Art Galleries',
                   'primary': True,
                   'shortName': 'Art Gallery'}],
                 'id': '4d438c6514aa8cfa743d5c3d',
                 'location': {'address': "1 St. Andrew's Road",
                  'cc': 'SG',
                  'city': 'Singapore',
                  'country': 'Singapore',
                  'distance': 61,
                  'formattedAddress': ["1 St. Andrew's Road", '178957', 'Singapore'],
                  'labeledLatLngs': [{'label': 'display',
                    'lat': 1.2907395913341984,
                    'lng': 103.85154786540198}],
                  'lat': 1.2907395913341984,
                  'lng': 103.85154786540198,
                  'postalCode': '178957'},
                 'name': 'National Gal\xadlery Singa\xadpore',
                 'photos': {'count': 0, 'groups': []}}},
               {'reasons': {'count': 0,
                 'items': [{'reasonName': 'globalInteractionReason',
                   'summary': 'This spot is popular',
                   'type': 'general'}]},
                'referralId': 'e-0-4b058810f964a52036af22e3-1',
                'venue': {'categories': [{'icon': {'prefix': 'https://ss3.4sqi.net/img/cat
          egories_v2/parks_outdoors/park_',
                    'suffix': '.png'},
                   'id': '4bf58dd8d48988d163941735',
                   'name': 'Park',
                   'pluralName': 'Parks',
                   'primary': True,
                   'shortName': 'Park'}],
                 'id': '4b058810f964a52036af22e3',
                 'location': {'address': 'Connaught Dr.',
                  'cc': 'SG',
                  'city': 'Singapore',
                  'country': 'Singapore',
                  'crossStreet': 'Opp Padang & City Hall',
                  'distance': 240,
                  'formattedAddress': ['Connaught Dr. (Opp Padang & City Hall)',
                   '179558',
                   'Singapore'],
                  'labeledLatLngs': [{'label': 'display',
                    'lat': 1.2889675708353954,
                    'lng': 103.85358044117562}],
                  'lat': 1.2889675708353954,
                  'lng': 103.85358044117562
```

Fetch the Venue details into dataframe

In [127]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue
.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

Out[127]:

	name	categories	lat	lng
0	National Gallery Singapore	Art Gallery	1.290740	103.851548
1	Esplanade Park	Park	1.288968	103.853580
2	The Oval @ Singapore Cricket Club Pavilion	Restaurant	1.289006	103.852438
3	Odette Restaurant	French Restaurant	1.289679	103.851691
4	Singapore F1 Padang Grandstand	Event Space	1.290656	103.852773

In [23]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
```

```
79 venues were returned by Foursquare.
```

For each retrieved venueID, retrive the venues category rating.

The generated data frame in the second function contains the following column:

In [24]:
```
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_se
cret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

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

        # 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 for item in ve
nue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [25]:
```
sgp_venues = getNearbyVenues(names=nearby_venues['name'],
                             latitudes=nearby_venues['lat'],
                             longitudes=nearby_venues['lng']
                            )
```

```
National Gallery Singapore
Esplanade Park
The Oval @ Singapore Cricket Club Pavilion
Odette Restaurant
Singapore F1 Padang Grandstand
Singapore F1 GP: Padang Stage
Aura
Esplanade Theatre
Esplanade Concourse
Victoria Theatre & Victoria Concert Hall
Smoke & Mirrors
Esplanade - Theatres On The Bay
Singapore F1 Circuit Gate 3
JAAN
Esplanade Concert Hall
Swissôtel The Stamford
The National Kitchen by Violet Oon Singapore
Victoria Concert Hall - Home of the SSO
Asian Civilisations Museum
Esplanade Riverside
Tokyo Milk Cheese Factory
Starbucks Reserve Store
Raffles City Shopping Centre
Sky Lounge @ Peninsula Excelsior
Duke Bakery
Royce
Hoshino Coffee
TAP Craft Beer Bar (One Raffles Link)
Din Tai Fung 鼎泰豐 (Din Tai Fung)
Esplanade Outdoor Theatre
Capitol Piazza
Cavenagh Bridge
Capitol Theatre
Southbridge
Barbershop By Timbre
Headquarters
The Fullerton Hotel
The Merlion
Esplanade Recital Studio
Singapore Cricket Club
Jumbo Seafood Gallery 珍宝海鮮樓
Sabaai Sabaai Traditional Thai Massage
Fairmont Singapore
Katanashi Japanese Tapas Bar
Tiong Bahru Bakery
The Sandwich Shop
Timbré
City Space
Crossfit Mobilus
Wooloomooloo Steakhouse
Braci
The Lighthouse Restaurant & Rooftop Bar
Shahi Maharani North Indian Restaurant
CityLink Mall
Lewin Terrace
Raffles City Market Place
Ah Sam Cold Drink Stall
```

In [26]:
```
print(sgp_venues.shape)
sgp_venues.head()
```

(6975, 7)

Out[26]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	National Gallery Singapore	1.29074	103.851548	National Gallery Singapore	1.290740	103.851548	Art Gallery
1	National Gallery Singapore	1.29074	103.851548	The Oval @ Singapore Cricket Club Pavilion	1.289006	103.852438	Restaurant
2	National Gallery Singapore	1.29074	103.851548	Odette Restaurant	1.289679	103.851691	French Restaurant
3	National Gallery Singapore	1.29074	103.851548	Singapore F1 Padang Grandstand	1.290656	103.852773	Event Space
4	National Gallery Singapore	1.29074	103.851548	Esplanade Park	1.288968	103.853580	Park

In [27]:
```
sgp_venues.groupby('Neighborhood').count()



#sgp_grouped = sgp_onehot.groupby('Neighborhood').mean().reset_index()
#sgp_grouped
```

Out[27]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
4Fingers Crispy Chicken	100	100	100	100	100	100
Ah Sam Cold Drink Stall	91	91	91	91	91	91
Anti:dote	100	100	100	100	100	100
Asian Civilisations Museum	81	81	81	81	81	81
Aura	77	77	77	77	77	77
Barbershop By Timbre	85	85	85	85	85	85
Braci	94	94	94	94	94	94
Capitol Piazza	71	71	71	71	71	71
Capitol Theatre	68	68	68	68	68	68
Cavenagh Bridge	81	81	81	81	81	81
City Space	95	95	95	95	95	95
CityLink Mall	99	99	99	99	99	99
Crossfit Mobilus	100	100	100	100	100	100
Din Tai Fung 鼎泰豐 (Din Tai Fung)	100	100	100	100	100	100
Duke Bakery	94	94	94	94	94	94
Empress	82	82	82	82	82	82
Esplanade - Theatres On The Bay	100	100	100	100	100	100
Esplanade Concert Hall	100	100	100	100	100	100
Esplanade Concourse	100	100	100	100	100	100
Esplanade Outdoor Theatre	90	90	90	90	90	90
Esplanade Park	93	93	93	93	93	93
Esplanade Recital Studio	85	85	85	85	85	85
Esplanade Riverside	86	86	86	86	86	86
Esplanade Theatre	100	100	100	100	100	100

one hot encoding

In [28]:
```
# one hot encoding
sgp_onehot = pd.get_dummies(sgp_venues[['Venue Category']], prefix="", prefix_sep="
")

# add neighborhood column back to dataframe
sgp_onehot['Neighborhood'] = sgp_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [sgp_onehot.columns[-1]] + list(sgp_onehot.columns[:-1])
sgp_onehot = sgp_onehot[fixed_columns]

sgp_onehot.head()
```

Out[28]:

	Neighborhood	Accessories Store	American Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	BBQ Joint	Bakery	Bar	E
0	National Gallery Singapore	0	0	1	0	0	0	0	0	0	C
1	National Gallery Singapore	0	0	0	0	0	0	0	0	0	C
2	National Gallery Singapore	0	0	0	0	0	0	0	0	0	C
3	National Gallery Singapore	0	0	0	0	0	0	0	0	0	C
4	National Gallery Singapore	0	0	0	0	0	0	0	0	0	C

In [29]:
```
sgp_grouped = sgp_onehot.groupby('Neighborhood').mean().reset_index()
sgp_grouped
```

Out[29]:

	Neighborhood	Accessories Store	American Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	BBQ Joint	Ba
0	4Fingers Crispy Chicken	0.000000	0.010000	0.000000	0.000000	0.000000	0.000000	0.000000	0.03
1	Ah Sam Cold Drink Stall	0.000000	0.000000	0.010989	0.000000	0.000000	0.021978	0.010989	0.01
2	Anti:dote	0.000000	0.000000	0.010000	0.010000	0.010000	0.030000	0.000000	0.03
3	Asian Civilisations Museum	0.000000	0.000000	0.012346	0.000000	0.000000	0.024691	0.012346	0.00
4	Aura	0.000000	0.000000	0.038961	0.000000	0.000000	0.025974	0.000000	0.02
5	Barbershop By Timbre	0.000000	0.000000	0.011765	0.000000	0.000000	0.023529	0.011765	0.01
6	Braci	0.000000	0.000000	0.010638	0.000000	0.000000	0.021277	0.010638	0.01
7	Capitol Piazza	0.000000	0.000000	0.042254	0.014085	0.014085	0.042254	0.000000	0.02
8	Capitol Theatre	0.000000	0.000000	0.029412	0.014706	0.014706	0.044118	0.000000	0.02
9	Cavenagh Bridge	0.000000	0.000000	0.012346	0.000000	0.000000	0.024691	0.012346	0.00
10	City Space	0.000000	0.000000	0.021053	0.000000	0.010526	0.031579	0.000000	0.02
11	CityLink Mall	0.000000	0.000000	0.010101	0.000000	0.010101	0.030303	0.000000	0.02
12	Crossfit Mobilus	0.000000	0.000000	0.020000	0.000000	0.000000	0.020000	0.010000	0.02
13	Din Tai Fung 鼎泰豐 (Din Tai Fung)	0.000000	0.000000	0.010000	0.000000	0.010000	0.030000	0.000000	0.02
14	Duke Bakery	0.000000	0.000000	0.021277	0.010638	0.010638	0.031915	0.000000	0.02
15	Empress	0.000000	0.000000	0.012195	0.000000	0.000000	0.036585	0.012195	0.01
16	Esplanade - Theatres On The Bay	0.010000	0.000000	0.010000	0.000000	0.000000	0.020000	0.000000	0.02
17	Esplanade Concert Hall	0.010000	0.000000	0.010000	0.000000	0.000000	0.020000	0.000000	0.02
18	Esplanade Concourse	0.010000	0.000000	0.010000	0.000000	0.000000	0.020000	0.000000	0.02
19	Esplanade Outd	0.000000	0.011111	0.000000	0.011111	0.000000	0.011111	0.000000	0.03

Let's print each neighborhood along with the top 5 most common venues

In [30]:

```
num_top_venues = 5

for hood in sgp_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = sgp_grouped[sgp_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num
_top_venues))
    print('\n')
```

```
----4Fingers Crispy Chicken----
           venue  freq
0          Hotel  0.10
1      Hotel Bar  0.05
2  Shopping Mall  0.05
3         Buffet  0.04
4    Event Space  0.04


----Ah Sam Cold Drink Stall----
                  venue  freq
0   Japanese Restaurant  0.07
1                  Café  0.05
2  Gym / Fitness Center  0.04
3    Italian Restaurant  0.04
4                 Hotel  0.04


----Anti:dote----
               venue  freq
0              Hotel  0.08
1               Café  0.05
2        Coffee Shop  0.04
3      Shopping Mall  0.04
4  French Restaurant  0.04


----Asian Civilisations Museum----
                  venue  freq
0  Gym / Fitness Center  0.05
1           Cocktail Bar  0.05
2                   Bar  0.05
3    Italian Restaurant  0.05
4           Yoga Studio  0.04


----Aura----
               venue  freq
0       Cocktail Bar  0.05
1      Shopping Mall  0.04
2  French Restaurant  0.04
3        Art Gallery  0.04
4       Concert Hall  0.04


----Barbershop By Timbre----
                  venue  freq
0   Japanese Restaurant  0.05
1    Italian Restaurant  0.05
2                   Bar  0.05
3  Gym / Fitness Center  0.05
4          Cocktail Bar  0.05


----Braci----
                  venue  freq
0   Japanese Restaurant  0.06
```

Let's put that into a pandas dataframe

First, let's write a function to sort the venues in descending order.

In [31]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

In [32]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = sgp_grouped['Neighborhood']

for ind in np.arange(sgp_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(sgp_group
ed.iloc[ind, :], num_top_venues)
neighborhoods_venues_sorted.head()
```

Out[32]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8t Cc
0	4Fingers Crispy Chicken	Hotel	Shopping Mall	Hotel Bar	Buffet	Event Space	Performing Arts Venue	Steakhouse	Jap Res
1	Ah Sam Cold Drink Stall	Japanese Restaurant	Café	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Hotel	Yog Stu
2	Anti:dote	Hotel	Café	French Restaurant	Chinese Restaurant	Cocktail Bar	Coffee Shop	Japanese Restaurant	Shc Mal
3	Asian Civilisations Museum	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Yoga Studio	Japanese Restaurant	Concert Hall	Chi Res
4	Aura	Cocktail Bar	French Restaurant	Hotel	Art Gallery	Coffee Shop	Concert Hall	Shopping Mall	Mus Ven

Adding Latitude and Longitude to each Neighborhood in the Dataframe

In [72]:
```
geo = Nominatim(user_agent='Mypythonapi')
for idx,town in neighborhoods_venues_sorted['Neighborhood'].iteritems():
    coord = geo.geocode(town + ' ' + "Singapore", timeout = 10)
    if coord:
        neighborhoods_venues_sorted.loc[idx,'Latitude'] = coord.latitude
        neighborhoods_venues_sorted.loc[idx,'Longitude'] = coord.longitude
   # else:
     #   neighborhoods_venues_sorted.loc[idx,'Latitude'] = NULL
     #   neighborhoods_venues_sorted.loc[idx,'Longitude'] = NULL
```

In [73]:
```
neighborhoods_venues_sorted.set_index("Neighborhood")
```

Out[73]:

	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8t Co
Neighborhood								
4Fingers Crispy Chicken	Hotel	Shopping Mall	Hotel Bar	Buffet	Event Space	Performing Arts Venue	Steakhouse	Japa Res
Ah Sam Cold Drink Stall	Japanese Restaurant	Café	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Hotel	Yoga Stuc
Anti:dote	Hotel	Café	French Restaurant	Chinese Restaurant	Cocktail Bar	Coffee Shop	Japanese Restaurant	Sho Mall
Asian Civilisations Museum	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Yoga Studio	Japanese Restaurant	Concert Hall	Chir Res
Aura	Cocktail Bar	French Restaurant	Hotel	Art Gallery	Coffee Shop	Concert Hall	Shopping Mall	Mus Ven
Barbershop By Timbre	Bar	Cocktail Bar	Japanese Restaurant	Italian Restaurant	Gym / Fitness Center	Yoga Studio	Salad Place	Con Hall
Braci	Japanese Restaurant	Café	Gym / Fitness Center	Cocktail Bar	Bar	Hotel	Yoga Studio	Lour
Capitol Piazza	French Restaurant	Hotel	Cocktail Bar	Chinese Restaurant	Art Gallery	Asian Restaurant	Shopping Mall	Eve Spa
Capitol Theatre	French Restaurant	Hotel	Cocktail Bar	Shopping Mall	Chinese Restaurant	Asian Restaurant	Dumpling Restaurant	Japa Res
Cavenagh Bridge	Gym / Fitness Center	Cocktail Bar	Bar	Italian Restaurant	Japanese Restaurant	Yoga Studio	Concert Hall	Sala Plac
City Space	Hotel	Shopping Mall	French Restaurant	Cocktail Bar	Café	Chinese Restaurant	Clothing Store	Asia Res
CityLink Mall	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Clothing Store	Café	Japanese Restaurant	Asia Res
Crossfit Mobilus	Bar	Nightclub	Hotel	Yoga Studio	Japanese Restaurant	Café	Cocktail Bar	Sea Res
Din Tai Fung 鼎泰豐 (Din Tai Fung)	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Chinese Restaurant	Japanese Restaurant	Café	Clot Stor
Duke Bakery	Hotel	French	Shopping	Cocktail	Coffee	Asian	Clothing	Chir

Cluster Neighborhoods

In [74]:
```
# set number of clusters
kclusters = 5

sgp_grouped_clustering = sgp_grouped.drop('Neighborhood', 1)

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

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

Out[74]:
```
array([2, 1, 0, 3, 4, 3, 1, 0, 0, 3], dtype=int32)
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [91]:
```
# add clustering labels
#neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

sgp_merged = singapore_average_rental_prices_by_town
sgp_merged = neighborhoods_venues_sorted
#neighborhoods_venues_sorted.head()
sgp_merged.shape # check the last columns!
```

Out[91]:
```
(78, 13)
```

In [81]:
```
town_venues_sorted = pd.DataFrame(columns=columns)
town_venues_sorted['Neighborhood'] = sgp_grouped['Neighborhood']

for ind in np.arange(sgp_grouped.shape[0]):
    town_venues_sorted.iloc[ind, 1:] = return_most_common_venues(sgp_grouped.iloc[i
nd, :], num_top_venues)

print(town_venues_sorted.shape)
town_venues_sorted.head()
```

(78, 11)

Out[81]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8t Co
0	4Fingers Crispy Chicken	Hotel	Shopping Mall	Hotel Bar	Buffet	Event Space	Performing Arts Venue	Steakhouse	Jap Res
1	Ah Sam Cold Drink Stall	Japanese Restaurant	Café	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Hotel	Yog Stu
2	Anti:dote	Hotel	Café	French Restaurant	Chinese Restaurant	Cocktail Bar	Coffee Shop	Japanese Restaurant	Sho Mal
3	Asian Civilisations Museum	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Yoga Studio	Japanese Restaurant	Concert Hall	Chi Res
4	Aura	Cocktail Bar	French Restaurant	Hotel	Art Gallery	Coffee Shop	Concert Hall	Shopping Mall	Mus Ven

Run k-means to cluster the Towns into 5 clusters.

In [104]:
```
# set number of clusters
kclusters = 5
sgp_grouped_clustering = sgp_grouped.drop('Neighborhood', 1)
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=1).fit(sgp_grouped_clustering)

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

```
[0 2 3 1 4 1 2 3 3 1]
78
```

In [99]:
```
town_venues_sorted.head()
```

Out[99]:

	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8tl Co
Neighborhood								
4Fingers Crispy Chicken	Hotel	Shopping Mall	Hotel Bar	Buffet	Event Space	Performing Arts Venue	Steakhouse	Japa Res
Ah Sam Cold Drink Stall	Japanese Restaurant	Café	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Hotel	Yoga Stuc
Anti:dote	Hotel	Café	French Restaurant	Chinese Restaurant	Cocktail Bar	Coffee Shop	Japanese Restaurant	Sho Mall
Asian Civilisations Museum	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Yoga Studio	Japanese Restaurant	Concert Hall	Chir Res
Aura	Cocktail Bar	French Restaurant	Hotel	Art Gallery	Coffee Shop	Concert Hall	Shopping Mall	Mus Ven
Barbershop By Timbre	Bar	Cocktail Bar	Japanese Restaurant	Italian Restaurant	Gym / Fitness Center	Yoga Studio	Salad Place	Con Hall
Braci	Japanese Restaurant	Café	Gym / Fitness Center	Cocktail Bar	Bar	Hotel	Yoga Studio	Lour
Capitol Piazza	French Restaurant	Hotel	Cocktail Bar	Chinese Restaurant	Art Gallery	Asian Restaurant	Shopping Mall	Eve Spa
Capitol Theatre	French Restaurant	Hotel	Cocktail Bar	Shopping Mall	Chinese Restaurant	Asian Restaurant	Dumpling Restaurant	Japa Res
Cavenagh Bridge	Gym / Fitness Center	Cocktail Bar	Bar	Italian Restaurant	Japanese Restaurant	Yoga Studio	Concert Hall	Sala Plac
City Space	Hotel	Shopping Mall	French Restaurant	Cocktail Bar	Café	Chinese Restaurant	Clothing Store	Asia Res
CityLink Mall	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Clothing Store	Café	Japanese Restaurant	Asia Res
Crossfit Mobilus	Bar	Nightclub	Hotel	Yoga Studio	Japanese Restaurant	Café	Cocktail Bar	Sea Res
Din Tai Fung 鼎泰豐 (Din Tai Fung)	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Chinese Restaurant	Japanese Restaurant	Café	Clot Stor
Duke Bakery	Hotel	French	Shopping	Cocktail	Coffee	Asian	Clothing	Chir

In [113]:
```
#town_venues_sorted = town_venues_sorted.set_index('Neighborhood')
#sgp_merged = sgp_merged.set_index('Neighborhood')
# add clustering labels
sgp_merged['Cluster Labels'] = kmeans.labels_
# merge sg_grouped with singapore_average_rental_prices_by_town to add latitude/lo
ngitude for each neighborhood
#sgp_merged = sgp_merged.join(town_venues_sorted)
sgp_merged
```

Out[113]:

	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8tl Co
Neighborhood								
4Fingers Crispy Chicken	Hotel	Shopping Mall	Hotel Bar	Buffet	Event Space	Performing Arts Venue	Steakhouse	Japa Res
Ah Sam Cold Drink Stall	Japanese Restaurant	Café	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Hotel	Yoga Stuc
Anti:dote	Hotel	Café	French Restaurant	Chinese Restaurant	Cocktail Bar	Coffee Shop	Japanese Restaurant	Sho Mall
Asian Civilisations Museum	Gym / Fitness Center	Italian Restaurant	Cocktail Bar	Bar	Yoga Studio	Japanese Restaurant	Concert Hall	Chir Res
Aura	Cocktail Bar	French Restaurant	Hotel	Art Gallery	Coffee Shop	Concert Hall	Shopping Mall	Mus Ven
Barbershop By Timbre	Bar	Cocktail Bar	Japanese Restaurant	Italian Restaurant	Gym / Fitness Center	Yoga Studio	Salad Place	Con Hall
Braci	Japanese Restaurant	Café	Gym / Fitness Center	Cocktail Bar	Bar	Hotel	Yoga Studio	Lou
Capitol Piazza	French Restaurant	Hotel	Cocktail Bar	Chinese Restaurant	Art Gallery	Asian Restaurant	Shopping Mall	Eve Spa
Capitol Theatre	French Restaurant	Hotel	Cocktail Bar	Shopping Mall	Chinese Restaurant	Asian Restaurant	Dumpling Restaurant	Japa Res
Cavenagh Bridge	Gym / Fitness Center	Cocktail Bar	Bar	Italian Restaurant	Japanese Restaurant	Yoga Studio	Concert Hall	Sala Plac
City Space	Hotel	Shopping Mall	French Restaurant	Cocktail Bar	Café	Chinese Restaurant	Clothing Store	Asia Res
CityLink Mall	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Clothing Store	Café	Japanese Restaurant	Asia Res
Crossfit Mobilus	Bar	Nightclub	Hotel	Yoga Studio	Japanese Restaurant	Café	Cocktail Bar	Sea Res
Din Tai Fung 鼎泰豐 (Din Tai Fung)	Hotel	Shopping Mall	Coffee Shop	Cocktail Bar	Chinese Restaurant	Japanese Restaurant	Café	Clot Stor
Duke Bakery	Hotel	French	Shopping	Cocktail	Coffee	Asian	Clothing	Chir

Visualising the findings / clusters on the Map

In [114]:
```
map_clusters = folium.Map(location=[latitude, longitude], tiles="Openstreetmap", z
oom_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, cluster in zip(sgp_merged['Latitude'], sgp_merged['Longitude'],
sgp_merged.index.values,kmeans.labels_):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=1).add_to(map_clusters)

map_clusters
```

Out[114]: