# Table of Contents

# Introduction/Business Problem

## Background Discussion

Canada and the UK are two of the most popular places for immigration, particularly among Asians. Both places are culturally diversified, vibrant, full of opportunities and have good catchment areas. They are also the financial hubs of their respective countries.

According to the 'World Migration Report 2020' published by the United Nation, the United Kingdom and Canada ranked no. 5 and 8 respectively in the top 20 destinations of immigration, whilst India and China ranked no. 1 and 3 of the country of origin, followed by other Asian countries such as Bangladesh, Pakistan and Philippines.

In the same report, the migrant populations in Canada increased from 18% in 2000 to over 21% in 2019. In 2019, China and India were in the top 3 countries of migrants and contributed around 1.409 million of migrants in total.

Further to the above report, the 'Migration Statics Quarterly Report: August 2020' published by the Office of National Statistics of the UK Government reported that during the year of April 2019 to March 2020, there were around 715,000 and 403,000 people migrated to and left the UK respectively, which contributed to around 313,000 people of net migration with intention of staying 12 months or longer. In the same report, the statistician's comment mentioned that it was being driven by the increase in non-EU student arrivals, mainly from China and India. The 'Migrants in the UK: An Overview' published by the University of Oxford pointed out that there were 35% of foreign-born population living in London in 2019. The 'Where do migrants live in the UK?' published on the same website report showed that 43% of non-EU migrants were family migrants in 2017, albeit it's the lowest number compared with the highest number, which was 60% in Yorkshire & Humber.

## Problem

There is no right or wrong answer to migrate either to Canada or the UK – it's up to personal preferences, financial situation, influence by other factors such as other family members or friends' advice, social connection and so on. Some people would like to pursuit a more lay back lifestyle and some people may prefer to live in a city.

As the business owner of an immigration agency operating in Hong Kong, I received an inquiry from a family of four looking for migrating to London or Toronto. The family liked city lifestyle and preferred similar one when living in either city. They were looking for a better studying environment for the children. They also had a fair understanding of both cities. None of the family members held a foreign passport. They already researched some big cities and narrowed down to London and Toronto. However, they struggled with choosing from one of them, so they were welcome some professional advice to support the final decision.

# Data Acquisition

Public data from Wikipedia and Foursquare will be used in this project.

Postal codes of Toronto and List of areas of London were the 2 data sources being used in this project. Web scraping technique was used to get the postcodes and borough data for data cleansing, transformation and exploratory analysis.

## Toronto Dataset

I loaded the borough and neighbourhood for Toronto by using the web scraping technique:

```python
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
tbl = pd.read_html(url)
len(tbl)
```

```python
df_toronto = tbl[0]
df_toronto.head()
```

| | Postal Code | Borough | Neighbourhood |
|---|---|---|---|
| 0 | M1A | Not assigned | Not assigned |
| 1 | M2A | Not assigned | Not assigned |
| 2 | M3A | North York | Parkwoods |
| 3 | M4A | North York | Victoria Village |
| 4 | M5A | Downtown Toronto | Regent Park, Harbourfront |

## London Dataset

Same as getting the Toronto dataset, I used web scraping technique to get the borough and neighbourhood for London:

```python
url = 'https://en.wikipedia.org/wiki/List_of_areas_of_London'
tbl = pd.read_html(url)
len(tbl)
```

```
5
```

```python
df_london = tbl[1]
df_london.head()
```

| | Location | London borough | Post town | Postcode district | Dial code | OS grid ref |
|---|---|---|---|---|---|---|
| 0 | Abbey Wood | Bexley, Greenwich [7] | LONDON | SE2 | 020 | TQ465785 |
| 1 | Acton | Ealing, Hammersmith and Fulham[8] | LONDON | W3, W4 | 020 | TQ205805 |
| 2 | Addington | Croydon[8] | CROYDON | CR0 | 020 | TQ375645 |
| 3 | Addiscombe | Croydon[8] | CROYDON | CR0 | 020 | TQ345665 |
| 4 | Albany Park | Bexley | BEXLEY, SIDCUP | DA5, DA14 | 020 | TQ478728 |

Since I could not find any single website which provide postcode and coordinates for London, I looked for the coordinates for all postcodes for the post town London on Wikipedia and created a CSV file to store the data. Below is a snippet of the file content:

London_Coordinates.csv

```
Postcode,Latitude,Longitude
EC1,51.523,-0.092
EC2,51.518,-0.084
EC3,51.508,-0.076
EC4,51.511,-0.098
E1,51.5176,-0.0558
E1W,51.5058,-0.0572
E2,51.5281,-0.0617
E3,51.5281,-0.0205
E4,51.6307,-0.0118
E5,51.5597,-0.0500
```

Foursquare APIs were used to explore various neighbourhoods of the two cities. Assumption is made to explore selected boroughs only for downtown Toronto in Toronto and post town London in Greater London.

# Data Wrangling

## Data Cleaning and Transformation - Toronto Dataset

Before I cleansed the data, I would like to know the number of rows in the dataset for comparison later.

```
# Display number of rows and columns
df_toronto.shape

(180, 3)
```

The first step was to drop all rows of 'Not assigned' borough.

```
df_toronto.drop(df_toronto[df_toronto['Borough'] == 'Not assigned'].index, axis = 0, inplace = True)
```

The next step was to assign the coordinates to each of the postal codes in the dataframe. To simplify the process, I downloaded a CSV file which prepared in advance to my computer and loaded it to the Jupyter notebook project file.

```python
df_toronto1 = pd.read_csv('Geospatial_Coordinates.csv')
df_toronto1.head()
```

| | Postal Code | Latitude | Longitude |
|---|---|---|---|
| 0 | M1B | 43.806686 | -79.194353 |
| 1 | M1C | 43.784535 | -79.160497 |
| 2 | M1E | 43.763573 | -79.188711 |
| 3 | M1G | 43.770992 | -79.216917 |
| 4 | M1H | 43.773136 | -79.239476 |

Then I merged the two datasets into a new dataset. Finally, I checked the number of rows after cleansed the data.

```python
df_toronto2 = df_toronto.merge(df_toronto1)
df_toronto2.head(12)
```

| | Postal Code | Borough | Neighbourhood | 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 |
| 5 | M9A | Etobicoke | Islington Avenue, Humber Valley Village | 43.667856 | -79.532242 |

```python
df_toronto2.shape
```

```
(103, 5)
```

Since my client was looking for vibrant city lifestyle, I assumed to explore and analyse the neighbourhoods in downtown areas in both cities only. So, I filtered out all boroughs which contain 'Toronto' from the Toronto dataset and assign it to a new dataset:

```
toronto = ['Downtown Toronto', 'East Toronto', 'West Toronto', 'Central Toronto']
df_toronto3 = df_toronto2.loc[df_toronto2['Borough'].isin(toronto)].reset_index(drop = True)
```

```
df_toronto3.head()
```

| | Postal Code | Borough | Neighbourhood | Latitude | Longitude |
|---|---|---|---|---|---|
| 0 | M5A | Downtown Toronto | Regent Park, Harbourfront | 43.654260 | -79.360636 |
| 1 | M7A | Downtown Toronto | Queen's Park, Ontario Provincial Government | 43.662301 | -79.389494 |
| 2 | M5B | Downtown Toronto | Garden District, Ryerson | 43.657162 | -79.378937 |
| 3 | M5C | Downtown Toronto | St. James Town | 43.651494 | -79.375418 |
| 4 | M4E | East Toronto | The Beaches | 43.676357 | -79.293031 |

## Data Cleaning and Transformation - London Dataset

After I loaded the required table from the Wiki page mentioned in the **Data Acquisition** section above to a dataset, I renamed the column names and dropped unused columns to start with the data cleansing process:

```
# Rename column names
columns = ['Location', 'Borough', 'Post Town', 'Postcode', 'Dial Code', 'OS Grid Ref']
df_london.columns = columns
```

```
# Drop unused columns
df_london.drop(['Dial Code', 'OS Grid Ref'], axis = 1, inplace = True)
df_london.head()
```

| | Location | Borough | Post Town | Postcode |
|---|---|---|---|---|
| 0 | Abbey Wood | Bexley, Greenwich [7] | LONDON | SE2 |
| 1 | Acton | Ealing, Hammersmith and Fulham[8] | LONDON | W3, W4 |
| 2 | Addington | Croydon[8] | CROYDON | CR0 |
| 3 | Addiscombe | Croydon[8] | CROYDON | CR0 |
| 4 | Albany Park | Bexley | BEXLEY, SIDCUP | DA5, DA14 |

I noticed that the annotations of borough were loaded from the Wiki page to the dataset. So, I removed them to make sure that correct data will be used:

```
# Remove the reference number from Borough
df_london['Borough'] = df_london['Borough'].map(lambda x: x.rstrip(']').rstrip('0123456789').rstrip('['))
df_london.head()
```

| | Location | Borough | Post Town | Postcode |
|---|---|---|---|---|
| 0 | Abbey Wood | Bexley, Greenwich | LONDON | SE2 |
| 1 | Acton | Ealing, Hammersmith and Fulham | LONDON | W3, W4 |
| 2 | Addington | Croydon | CROYDON | CR0 |
| 3 | Addiscombe | Croydon | CROYDON | CR0 |
| 4 | Albany Park | Bexley | BEXLEY, SIDCUP | DA5, DA14 |

I also noticed that there're more than one postcode in a row. I split them into multiple rows and assigned same values from the other columns:

```
df_london1 = df_london.drop('Postcode', axis=1).join(df_london['Postcode'].str.split(',', expand = True).stack().reset_index(level = 1, drop = True).rename('Postcode'))
```

```
df_london1.head()
```

| | Location | Borough | Post Town | Postcode |
|---|---|---|---|---|
| 0 | Abbey Wood | Bexley, Greenwich | LONDON | SE2 |
| 1 | Acton | Ealing, Hammersmith and Fulham | LONDON | W3 |
| 1 | Acton | Ealing, Hammersmith and Fulham | LONDON | W4 |
| 2 | Addington | Croydon | CROYDON | CR0 |
| 3 | Addiscombe | Croydon | CROYDON | CR0 |

The next step was to filter out rows for post town 'London' only:

```
df_london3.drop(df_london3[df_london3['Post Town'] != 'LONDON'].index, axis = 0, inplace = True)
df_london3.reset_index(drop = True, inplace = True)
df_london3.tail()
```

| | Location | Borough | Postcode | Post Town |
|---|---|---|---|---|
| 560 | Woodford | Redbridge | E18 | LONDON |
| 561 | Woodford Green | Redbridge, Waltham Forest | IG8 | LONDON |
| 562 | Woodside Park | Barnet | N12 | LONDON |
| 563 | Woolwich | Greenwich | SE18 | LONDON |
| 564 | Wormwood Scrubs | Hammersmith and Fulham | W12 | LONDON |

Then I regrouped the dataset by Post Town, Postcode and Borough and saved the resultset to a new dataset.

```
df_london4 = df_london3.groupby(['Post Town', 'Postcode', 'Borough'])['Location'].apply(', '.join).reset_index()
df_london4.head(20)
```

| | Post Town | Postcode | Borough | Location |
|---|---|---|---|---|
| 0 | LONDON | DA14 | Bexley | Longlands |
| 1 | LONDON | DA15 | Bexley | Longlands |
| 2 | LONDON | DA16 | Bexley, Greenwich | Falconwood |
| 3 | LONDON | DA18 | Bexley, Greenwich | Thamesmead |
| 4 | LONDON | DA7 | Bexley | Bexleyheath (also Bexley New Town) |
| 5 | LONDON | E13 | Newham | Upton Park |

As I assumed that only the data for post town London will be used for comparison, also London is huge and can be described as Greater London and City of London (also referred to 'Square Mile' by local people as it's too small!) depending on the context of discussion, I decided to narrow down to analyse neighbourhoods for the postcodes that begin with 'E', 'EC', 'N', 'NW', 'SE', 'SW', 'W' and 'WC' only (ref: [London postal district]).

```python
# create a new dataframe and store all rows for City of London
city_of_london = ['E', 'EC', 'N', 'NW', 'SE', 'SW', 'W', 'WC']
df_london5 = df_london4[df_london4['Postcode'].str[:2].isin(city_of_london)]
```

I also decided to keep the first occurrence of rows which had duplicated postcode in the rows and removed the rest of them:

```python
df_london5.drop_duplicates(subset = ['Postcode'], inplace = True, ignore_index = True)
df_london5.head()
```

| | Post Town | Postcode | Borough | Location |
|---|---|---|---|---|
| 0 | LONDON | EC2 | City | Barbican |
| 1 | LONDON | NW10 | Brent | Neasden |
| 2 | LONDON | NW3 | Camden | Gospel Oak, Primrose Hill |
| 3 | LONDON | NW4 | Barnet | Brent Cross |
| 4 | LONDON | NW6 | Brent | Kensal Green |

```python
df_london5['Postcode'].is_unique
```

```
True
```

Then I loaded a CSV file which contained the coordinates for the postcodes into a dataframe, merged it to the dataframe cleansed above to a new dataframe:

```python
df_london_coord = pd.read_csv('London_Coordinates.csv')
```

```python
df_london6 = df_london5.merge(df_london_coord)
df_london6.head(12)
```

| | Post Town | Postcode | Borough | Location | Latitude | Longitude |
|---|---|---|---|---|---|---|
| 0 | LONDON | EC2 | City | Barbican | 51.5180 | -0.0840 |
| 1 | LONDON | NW10 | Brent | Neasden | 51.5410 | -0.2531 |
| 2 | LONDON | NW3 | Camden | Gospel Oak, Primrose Hill | 51.5517 | -0.1706 |
| 3 | LONDON | NW4 | Barnet | Brent Cross | 51.5937 | -0.2181 |
| 4 | LONDON | NW6 | Brent | Kensal Green | 51.5438 | -0.1971 |
| 5 | LONDON | NW7 | Barnet | Arkley | 51.6147 | -0.2301 |
| 6 | LONDON | NW8 | Camden | Primrose Hill | 51.5333 | -0.1734 |
| 7 | LONDON | SE11 | Lambeth | Oval | 51.4913 | -0.1085 |
| 8 | LONDON | SE12 | Greenwich | Blackheath Royal Standard | 51.4467 | -0.0176 |
| 9 | LONDON | SE13 | Lewisham | Ladywell | 51.4572 | -0.0059 |
| 10 | LONDON | SE17 | Southwark | Elephant and Castle, Newington | 51.4874 | -0.0924 |
| 11 | LONDON | SE2 | Bexley | Bexleyheath (also Bexley New Town) | 51.4860 | -0.1203 |

Finally, I renamed the column name from 'Location' to 'Neighbourhood' so that I can call a function to use Foursquare API to retrieve venue data.

```python
df_london6.rename(columns = {'Location': 'Neighbourhood'}, inplace = True)
df_london6.head()
```

| | Post Town | Postcode | Borough | Neighbourhood | Latitude | Longitude |
|---|---|---|---|---|---|---|
| 0 | LONDON | EC2 | City | Barbican | 51.5180 | -0.0840 |
| 1 | LONDON | NW10 | Brent | Neasden | 51.5410 | -0.2531 |
| 2 | LONDON | NW3 | Camden | Gospel Oak, Primrose Hill | 51.5517 | -0.1706 |
| 3 | LONDON | NW4 | Barnet | Brent Cross | 51.5937 | -0.2181 |
| 4 | LONDON | NW6 | Brent | Kensal Green | 51.5438 | -0.1971 |

## Foursquare API

The Foursquare API will be used to obtain venue data of the selected Toronto and London borough neighbourhoods. To use it, we also need to define the version of the API that will be using.

To protect my Foursquare API credential privacy and prevent it from malicious use, I saved the credentials in a JSON file. Please also note that the file did not save to my GitHub repository.

```python
json_file = '4squarecredential.json'
with open(json_file) as f:
    data = json.load(f)

CLIENT_ID = data['credential']['CLIENT_ID']
CLIENT_SECRET = data['credential']['CLIENT_SECRET']
VERSION = data['credential']['VERSION']

LIMIT = 100 # A default Foursquare API limit value
```

## Methodology

I used GitHub repository to store the project file and data files for coordinates for Toronto and London.

One of the key data exploratory tasks is to visualise the data at different stages to compare the difference before and after the data is massaged, modelled and/or analysed. As Clustering algorithm was used to cluster the neighbourhoods and compare them between the two cities, I visualised the selected borough neighbourhoods

### Visualisation – Toronto

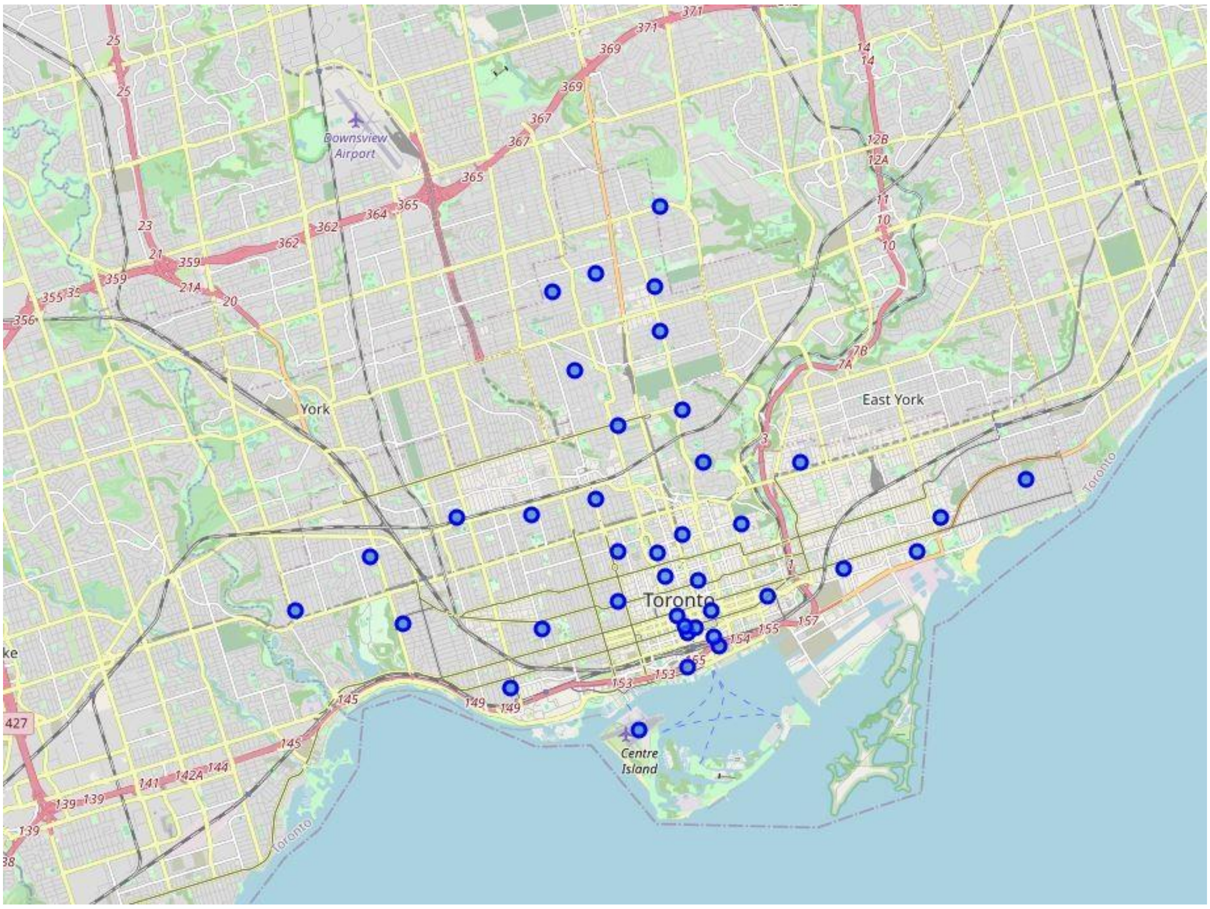
Firstly, I determined the coordinates of Toronto:

```python
address = 'Toronto, Ontario'

geolocator = Nominatim(user_agent="my_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Toronto, Ontraio are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Toronto, Ontraio are 43.6534817, -79.3839347.
```

Then, I highlighted and plotted the neighbourhoods of Toronto in a map by using Folium:

```python
# create map of Toronto using latitude and longitude values
map_toronto = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers to map
for lat, lng, label in zip(df_toronto3['Latitude'], df_toronto3['Longitude'], df_toronto3['Neighbourhood']):
    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_toronto)

map_toronto
```




## Visualisation – London

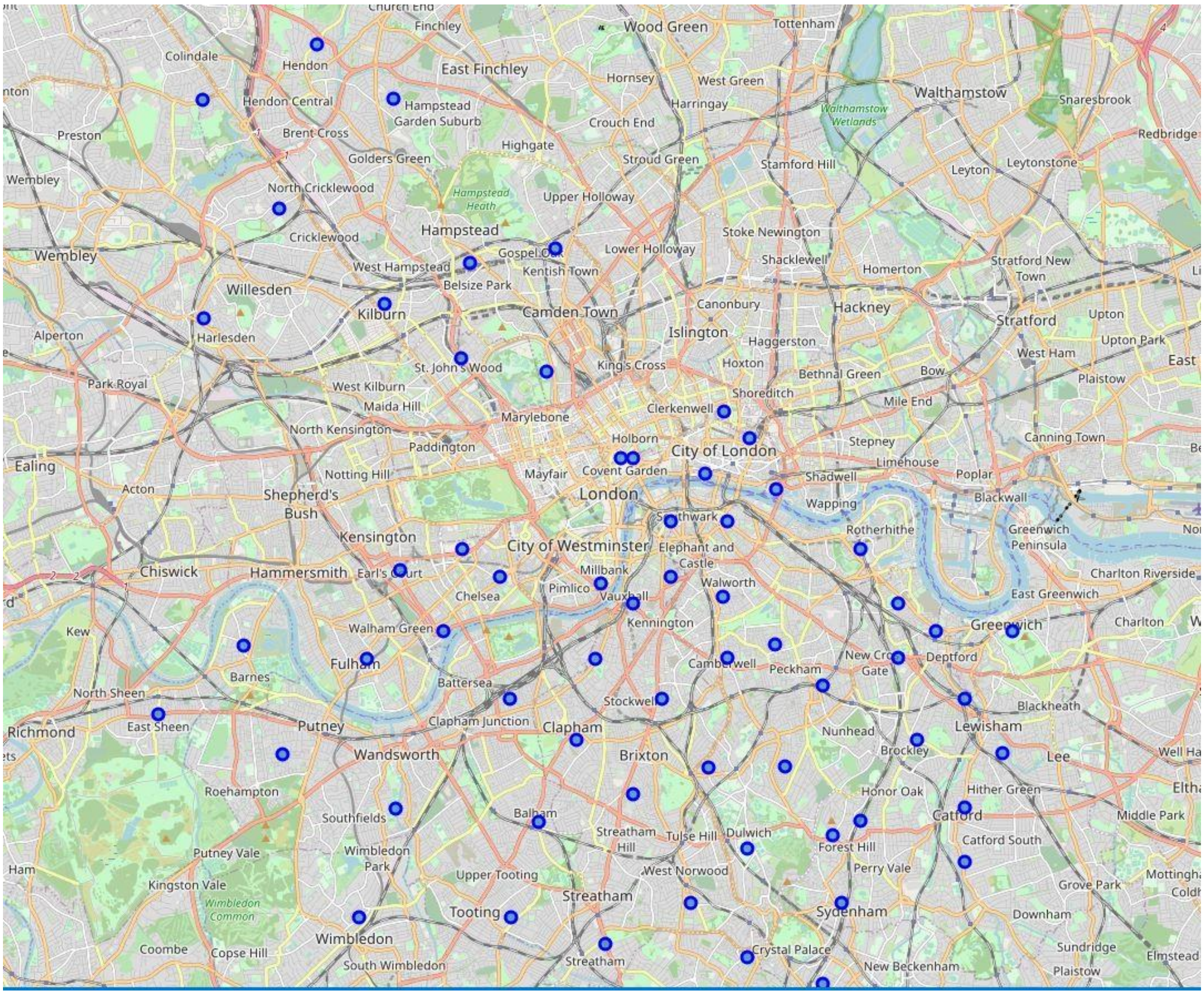
I determined the coordinates of London:

```python
address = 'London, England'

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

```
The geograpical coordinate of London, England are 51.5073219, -0.1276474.
```

Then, I highlighted and plotted the neighbourhoods of London in a map by using Folium:

```python
# create map of London using latitude and longitude values
map_london = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers to map
for lat, lng, label in zip(df_london6['Latitude'], df_london6['Longitude'], df_london6['Neighbourhood']):
    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_london)

map_london
```




## Exploratory Data Analysis – Toronto

The Toronto and London data cleansed. To test the workability of Foursquare API, I got the first neighbourhood of Toronto and did an initial exploratory to examine the top 100 venues within radius of 500 metres of the neighbourhood. The first neighbourhood was 'Regent Park, Harbourfront':

```
df_toronto3.loc[0, 'Neighbourhood']

'Regent Park, Harbourfront'
```

Then I got the coordinates of Regent Park, Harbourfront and used it to retrieve the venues by calling Foursquare API:

```python
nb_name = df_toronto3.loc[0, 'Neighbourhood']
nb_lat = df_toronto3.loc[0, 'Latitude']
nb_lng = df_toronto3.loc[0, 'Longitude']
print('The latitude and longitude of {} are {} and {}'.format(nb_name, nb_lat, nb_lng))
```

```
The latitude and longitude of Regent Park, Harbourfront are 43.6542599 and -79.3606359
```

```python
# build the url string
radius = 500
url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&ll={},{}&v={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, nb_lat, nb_lng, VERSION, radius,
LIMIT)
```

```python
results = requests.get(url).json()
results
```


```
ss3.4sqi.net/img/categories_v2/shops/gym_yogastudio_',
          'suffix': '.png'},
         'primary': True}],
       'photos': {'count': 0, 'groups': []}},
      'referralId': 'e-0-4b58dd55f964a5208f6f28e3-26'},
     {'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '4d84d98181fdb1f7d4a704c0',
       'name': 'Caffe Furbo',
       'location': {'address': '12 case goods lane',
        'lat': 43.649969882303814,
        'lng': -79.35884946388191,
        'labeledLatLngs': [{'label': 'display',
          'lat': 43.649969882303814,
          'lng': -79.35884946388191}],
        'distance': 498,
        'postalCode': 'M5A 3C4',
        'cc': 'CA',
        'city': 'Toronto',
        'state': 'ON',
        'country': 'Canada',
        'formattedAddress': ['12 case goods lane',
         'Toronto ON M5A 3C4',
         'Canada']},
```


To examine the result, the item keys were loaded into a structured pandas dataframe and retrieved the top 10 categories of venues:

```python
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(10)
```

| | name | categories | lat | lng |
|---|---|---|---|---|
| 0 | Roselle Desserts | Bakery | 43.653447 | -79.362017 |
| 1 | Tandem Coffee | Coffee Shop | 43.653559 | -79.361809 |
| 2 | Cooper Koo Family YMCA | Distribution Center | 43.653249 | -79.358008 |
| 3 | Body Blitz Spa East | Spa | 43.654735 | -79.359874 |
| 4 | Impact Kitchen | Restaurant | 43.656369 | -79.356980 |
| 5 | Morning Glory Cafe | Breakfast Spot | 43.653947 | -79.361149 |
| 6 | Corktown Common | Park | 43.655618 | -79.356211 |
| 7 | The Extension Room | Gym / Fitness Center | 43.653313 | -79.359725 |
| 8 | The Distillery Historic District | Historic Site | 43.650244 | -79.359323 |
| 9 | SOMA chocolatemaker | Chocolate Shop | 43.650622 | -79.358127 |

```
nearby_venues_by_categories = nearby_venues['categories'].value_counts().to_frame(name = 'Count')
nearby_venues_by_categories.head(10)
```

| | Count |
|---|---|
| Coffee Shop | 8 |
| Park | 3 |
| Bakery | 3 |
| Pub | 3 |
| Theater | 2 |
| Breakfast Spot | 2 |
| Café | 2 |
| Spa | 1 |
| Performing Arts Venue | 1 |
| Bank | 1 |

It was not surprised that the top 10 categories nearby Regent Park, Harbourfront were cafes and leisure places as the neighbourhood's name suggested that it's a leisure area. Then I explored all neighbourhoods in Toronto to get total number of venues, counted number of venues by categories and get the number of unique categories. There're 1,624 venues and 235 unique categories:

```
print(toronto_venues.shape)
toronto_venues.head()
```

(1624, 7)

| | Neighbourhood | Neighbourhood Latitude | Neighbourhood Longitude | Venue | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|---|---|---|---|
| 0 | Regent Park, Harbourfront | 43.65426 | -79.360636 | Roselle Desserts | 43.653447 | -79.362017 | Bakery |
| 1 | Regent Park, Harbourfront | 43.65426 | -79.360636 | Tandem Coffee | 43.653559 | -79.361809 | Coffee Shop |
| 2 | Regent Park, Harbourfront | 43.65426 | -79.360636 | Cooper Koo Family YMCA | 43.653249 | -79.358008 | Distribution Center |
| 3 | Regent Park, Harbourfront | 43.65426 | -79.360636 | Body Blitz Spa East | 43.654735 | -79.359874 | Spa |
| 4 | Regent Park, Harbourfront | 43.65426 | -79.360636 | Impact Kitchen | 43.656369 | -79.356980 | Restaurant |

```
toronto_venues.groupby('Neighbourhood').count()
```

| Neighbourhood | Neighbourhood Latitude | Neighbourhood Longitude | Venue | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|---|---|---|
| Berczy Park | 55 | 55 | 55 | 55 | 55 | 55 |
| Brockton, Parkdale Village, Exhibition Place | 23 | 23 | 23 | 23 | 23 | 23 |
| Business reply mail Processing Centre, South Central Letter Processing Plant Toronto | 16 | 16 | 16 | 16 | 16 | 16 |
| CN Tower, King and Spadina, Railway Lands, Harbourfront West, Bathurst Quay, South Niagara, Island airport | 16 | 16 | 16 | 16 | 16 | 16 |
| Central Bay Street | 68 | 68 | 68 | 68 | 68 | 68 |
| Christie | 16 | 16 | 16 | 16 | 16 | 16 |
| Church and Wellesley | 75 | 75 | 75 | 75 | 75 | 75 |
| Commerce Court, Victoria Hotel | 100 | 100 | 100 | 100 | 100 | 100 |
| Davisville | 33 | 33 | 33 | 33 | 33 | 33 |
| Davisville North | 9 | 9 | 9 | 9 | 9 | 9 |

```
print('There are {} uniques categories.'.format(len(toronto_venues['Venue Category'].unique())))
```

There are 235 uniques categories.

The next task was to analyse each neighbourhood. Before analysing them, I normalised the dataframe by applying the One-Hot Encoding technique to 'flatten' the dataframe by applying '1' to the venue category if it exists and '0' if it didn't:

```
# one hot encoding
toronto_onehot = pd.get_dummies(toronto_venues[['Venue Category']], prefix = "", prefix_sep = "")

# add neighbourhood column back to dataframe
toronto_onehot['Neighbourhood'] = toronto_venues['Neighbourhood']

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

toronto_onehot.head()
```

| | Neighbourhood | Afghan Restaurant | Airport | Airport Food Court | Airport Gate | Airport Lounge | Airport Service | Airport Terminal | American Restaurant | Antique Shop | ... | Theater | Res |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Regent Park, Harbourfront | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | |
| 1 | Regent Park, Harbourfront | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | |
| 2 | Regent Park, Harbourfront | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | |
| 3 | Regent Park, Harbourfront | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | |
| 4 | Regent Park, Harbourfront | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | |

5 rows × 236 columns

```
toronto_onehot.shape
```

```
(1624, 236)
```

After I explored and analysed the characteristics of the 'flatten' dataframe, I created a new dataframe to store the top 10 most common venues for each of the neighbourhoods. This will be used to cluster the neighbourhoods and visualise the results:

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighbourhood']
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
nbh_venues_sorted = pd.DataFrame(columns = columns)
nbh_venues_sorted['Neighbourhood'] = toronto_grouped['Neighbourhood']

for ind in np.arange(toronto_grouped.shape[0]):
    nbh_venues_sorted.iloc[ind, 1:] = most_common_venues(toronto_grouped.iloc[ind, :], num_top_venues)

nbh_venues_sorted.head()
```

| | Neighbourhood | 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 | 8th Most Common Venue | 9th Most Common Venue | 10th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Berczy Park | Coffee Shop | Bakery | Cocktail Bar | Beer Bar | Cheese Shop | Seafood Restaurant | Restaurant | Farmers Market | Sandwich Place | Breakfast Spot |
| 1 | Brockton, Parkdale Village, Exhibition Place | Café | Breakfast Spot | Nightclub | Coffee Shop | Climbing Gym | Burrito Place | Restaurant | Italian Restaurant | Intersection | Bar |
| 2 | Business reply mail Processing Centre, South C... | Skate Park | Pizza Place | Brewery | Burrito Place | Restaurant | Farmers Market | Fast Food Restaurant | Butcher | Recording Studio | Auto Workshop |
| 3 | CN Tower, King and Spadina, Railway Lands, Har... | Airport Lounge | Airport Service | Boutique | Harbor / Marina | Sculpture Garden | Boat or Ferry | Rental Car Location | Bar | Coffee Shop | Plane |
| 4 | Central Bay Street | Coffee Shop | Café | Italian Restaurant | Sandwich Place | Japanese Restaurant | Bubble Tea Shop | Salad Place | Burger Joint | Department Store | Thai Restaurant |

## Exploratory Data Analysis – London

I repeated the same steps used in exploring the data for Toronto. Key highlights were:

1. There're 2,203 venues returned by Foursquare API compared to 1,624 venues for Toronto.
2. There're 269 unique categories compared to 235 unique categories for Toronto.

```
print(london_venues.shape)
london_venues.head()
```

```
(2203, 7)
```

```
print('There are {} uniques categories.'.format(len(london_venues['Venue Category'].unique())))
```

```
There are 269 uniques categories.
```

## Clustering – Toronto

Unsupervised machine learning model is commonly used to cluster the data. Among all unsupervised machine learning algorithm, K-Means is widely used as it's simple and easy to implement. So, I selected and used this algorithm to cluster the neighbourhoods for both cities.

To simply the process, I clustered the data into 5 clusters:

```python
# set number of clusters
kclusters = 5

toronto_grouped_clustering = toronto_grouped.drop('Neighbourhood', 1)

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

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

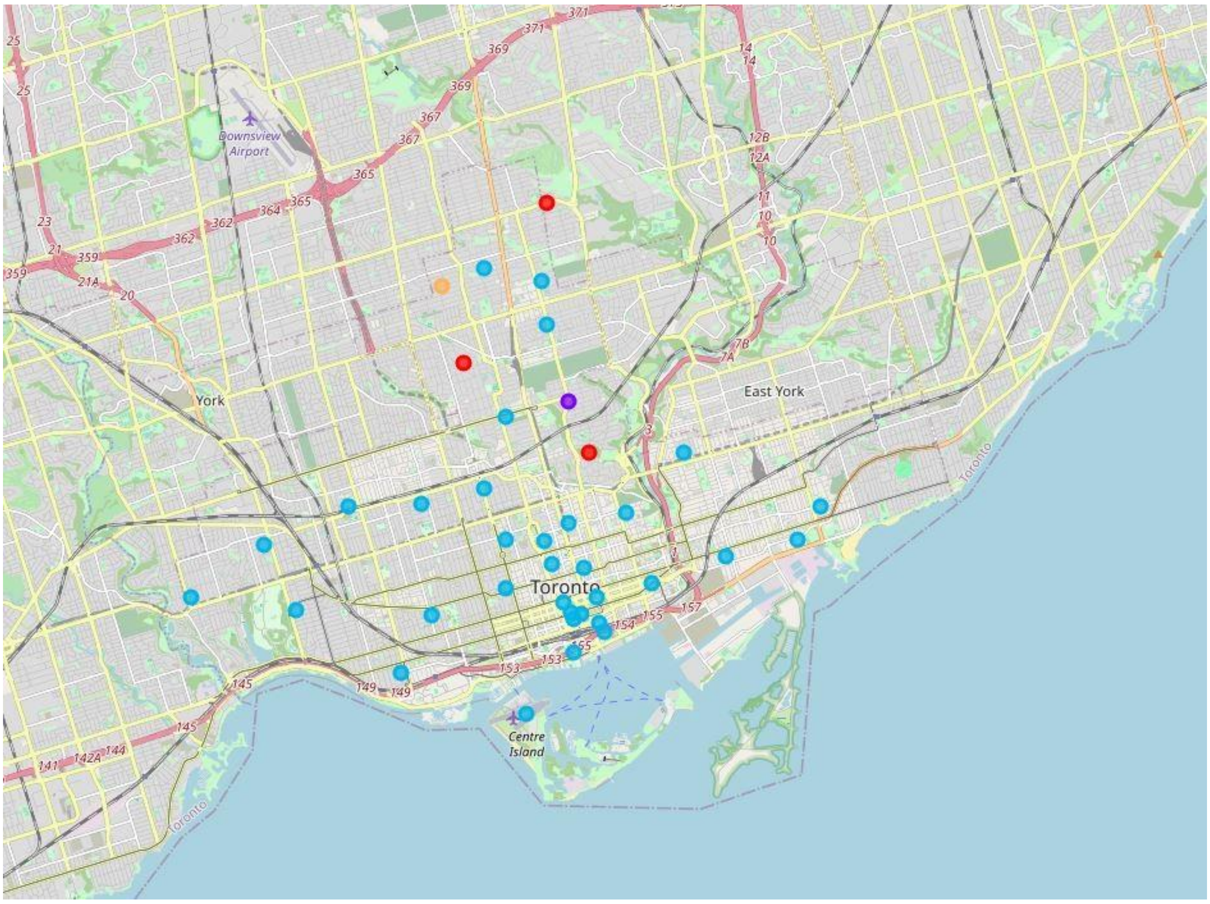
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```python
# add clustering labels
nbh_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

toronto_merged = df_toronto3

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighbourhood
toronto_merged = toronto_merged.join(nbh_venues_sorted.set_index('Neighbourhood'), on = 'Neighbourhood')

toronto_merged.head()
```

| | Borough | Neighbourhood | Latitude | Longitude | Cluster Labels | 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 | 8th Most Common Venue | 9th Most Common Venue | 10th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Downtown Toronto | Regent Park, Harbourfront | 43.654260 | -79.360636 | 2 | Coffee Shop | Bakery | Park | Pub | Café | Theater | Breakfast Spot | Event Space | Shoe Store | Hotel |
| 1 | Downtown Toronto | Queen's Park, Ontario Provincial Government | 43.662301 | -79.389494 | 2 | Coffee Shop | Yoga Studio | Diner | Restaurant | Portuguese Restaurant | Park | Music Venue | Mexican Restaurant | Italian Restaurant | Hobby Shop |
| 2 | Downtown Toronto | Garden District, Ryerson | 43.657162 | -79.378937 | 2 | Clothing Store | Coffee Shop | Café | Japanese Restaurant | Bubble Tea Shop | Cosmetics Shop | Diner | Lingerie Store | Ramen Restaurant | Italian Restaurant |
| 3 | Downtown Toronto | St. James Town | 43.651494 | -79.375418 | 2 | Coffee Shop | Café | Cocktail Bar | Restaurant | Beer Bar | Gastropub | American Restaurant | Farmers Market | Hotel | Japanese Restaurant |
| 4 | East Toronto | The Beaches | 43.676357 | -79.293031 | 3 | Pub | Trail | Health Food Store | Neighborhood | Yoga Studio | Dog Run | Dim Sum Restaurant | Diner | Discount Store | Distribution Center |

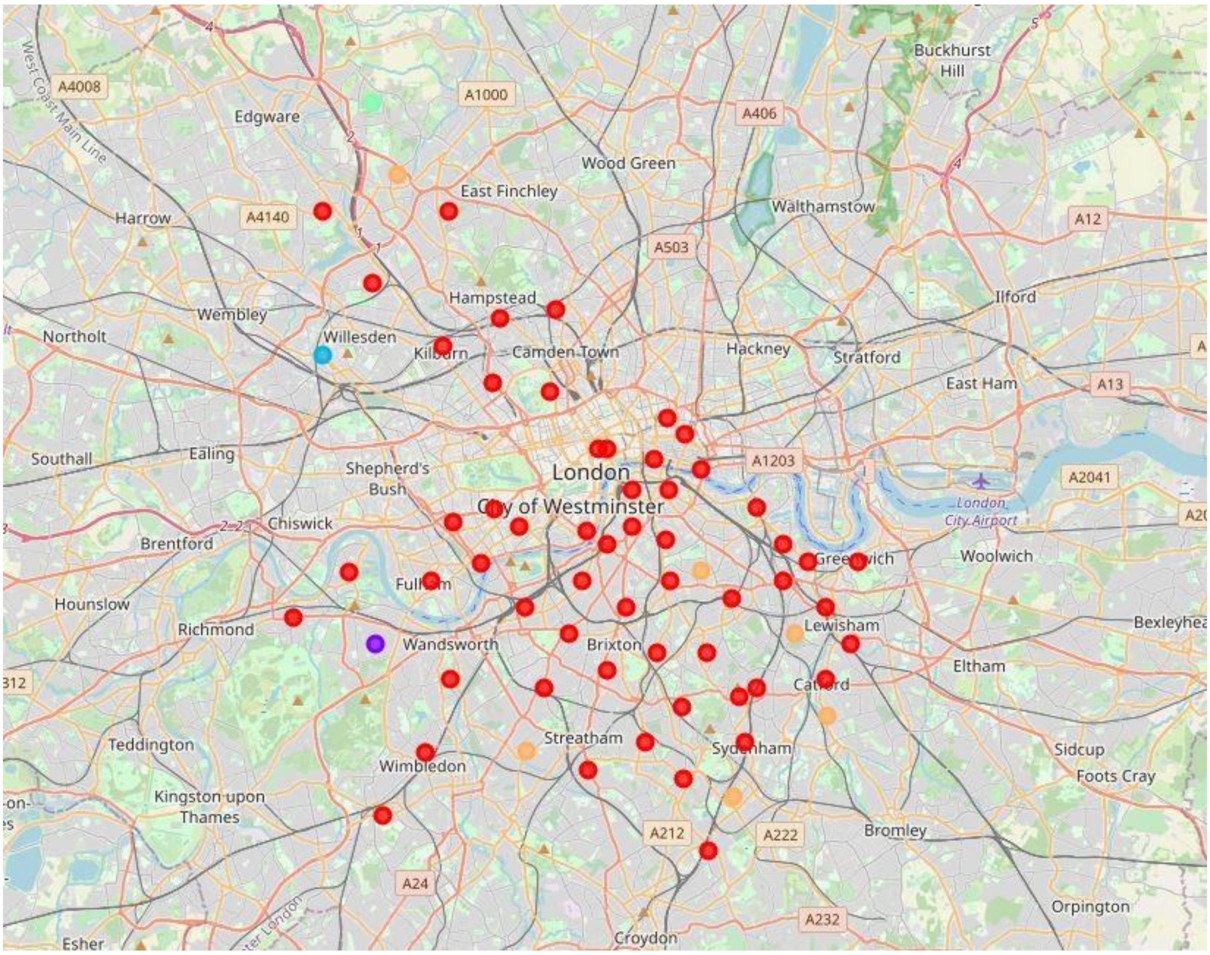
People are usually welcome visual presentation than reading article as it's more impactful and easier to digest and explain. So, I visualised the resulting clusters on a map plotted by Folium:

```python
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=12)

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

# add markers to the map
markers_colours = []
for lat, lon, poi, cluster in zip(toronto_merged['Latitude'], toronto_merged['Longitude'], toronto_merged['Neighbourhood'], toronto_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html = True)
    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
```




## Clustering – London

I also repeated the same process as Toronto. The resulting clusters displayed in the Folium map:




## Examine Clusters – Toronto

The following codes were used to reveal the top 10 categories for each cluster:

```
toronto_merged.loc[toronto_merged['Cluster Labels'] == 0, toronto_merged.co
lumns[[1] + list(range(4, toronto_merged.shape[1]))]]

toronto_merged.loc[toronto_merged['Cluster Labels'] == 1, toronto_merged.co
lumns[[1] + list(range(4, toronto_merged.shape[1]))]]

toronto_merged.loc[toronto_merged['Cluster Labels'] == 2, toronto_merged.co
lumns[[1] + list(range(4, toronto_merged.shape[1]))]]

toronto_merged.loc[toronto_merged['Cluster Labels'] == 3, toronto_merged.co
lumns[[1] + list(range(4, toronto_merged.shape[1]))]]

toronto_merged.loc[toronto_merged['Cluster Labels'] == 4, toronto_merged.co
lumns[[1] + list(range(4, toronto_merged.shape[1]))]]
```

A sample of the result as shown below:

| | Neighbourhood | Cluster Labels | 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 | 8th Most Common Venue | 9th Most Common Venue | 10th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Regent Park, Harbourfront | 2 | Coffee Shop | Bakery | Park | Pub | Café | Theater | Breakfast Spot | Event Space | Shoe Store | Hotel |
| 1 | Queen's Park, Ontario Provincial Government | 2 | Coffee Shop | Yoga Studio | Diner | Restaurant | Portuguese Restaurant | Park | Music Venue | Mexican Restaurant | Italian Restaurant | Hobby Shop |
| 2 | Garden District, Ryerson | 2 | Clothing Store | Coffee Shop | Café | Japanese Restaurant | Bubble Tea Shop | Cosmetics Shop | Diner | Lingerie Store | Ramen Restaurant | Italian Restaurant |
| 3 | St. James Town | 2 | Coffee Shop | Café | Cocktail Bar | Restaurant | Beer Bar | Gastropub | American Restaurant | Farmers Market | Hotel | Japanese Restaurant |
| 5 | Berczy Park | 2 | Coffee Shop | Bakery | Cocktail Bar | Beer Bar | Cheese Shop | Seafood Restaurant | Restaurant | Farmers Market | Sandwich Place | Breakfast Spot |
| 6 | Central Bay Street | 2 | Coffee Shop | Café | Italian Restaurant | Sandwich Place | Japanese Restaurant | Bubble Tea Shop | Salad Place | Burger Joint | Department Store | Thai Restaurant |
| 7 | Christie | 2 | Grocery Store | Café | Park | Candy Store | Italian Restaurant | Nightclub | Baby Store | Coffee Shop | Athletics & Sports | Restaurant |
| 8 | Richmond, Adelaide, King | 2 | Coffee Shop | Café | Gym | Restaurant | Hotel | Thai Restaurant | Bar | Clothing Store | Cosmetics Shop | Concert Hall |

## Examine Clusters – London

Same as examining the clusters in Toronto, I also used the same process but modified the code slightly to reveal the top 10 categories for each cluster:

```
london_merged.loc[london_merged['Cluster Labels'] == 0, london_merged.colum
ns[[1] + list(range(4, london_merged.shape[1]))]]

london_merged.loc[london_merged['Cluster Labels'] == 1, london_merged.colum
ns[[1] + list(range(4, london_merged.shape[1]))]]

london_merged.loc[london_merged['Cluster Labels'] == 2, london_merged.colum
ns[[1] + list(range(4, london_merged.shape[1]))]]

london_merged.loc[london_merged['Cluster Labels'] == 3, london_merged.colum
ns[[1] + list(range(4, london_merged.shape[1]))]]

london_merged.loc[london_merged['Cluster Labels'] == 4, london_merged.colum
ns[[1] + list(range(4, london_merged.shape[1]))]]
```

| | Postcode | Latitude | Longitude | Cluster Labels | 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 | 8th Most Common Venue | 9th Most Common Venue | 10th Most Common Venue |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | EC2 | 51.5180 | -0.0840 | 0 | Coffee Shop | Gym / Fitness Center | Food Truck | Italian Restaurant | Pub | Hotel | Bar | Cocktail Bar | English Restaurant | Sushi Restaurant |
| 2 | NW3 | 51.5517 | -0.1706 | 0 | Café | Bakery | Coffee Shop | Pub | Greek Restaurant | Bagel Shop | Pizza Place | Museum | Burger Joint | Convenience Store |
| 4 | NW6 | 51.5438 | -0.1971 | 0 | Pub | Indian Restaurant | Café | Middle Eastern Restaurant | Park | Coffee Shop | Brazilian Restaurant | Portuguese Restaurant | Thai Restaurant | Korean Restaurant |
| 6 | NW8 | 51.5333 | -0.1734 | 0 | Cricket Ground | Café | Coffee Shop | Deli / Bodega | French Restaurant | Fast Food Restaurant | Sandwich Place | Salad Place | Restaurant | Recording Studio |
| 7 | SE11 | 51.4913 | -0.1085 | 0 | Pub | Café | Fish & Chips Shop | Indian Restaurant | Pizza Place | Gastropub | Park | Italian Restaurant | Bar | Museum |
| 8 | SE12 | 51.4467 | -0.0176 | 0 | Grocery Store | Supermarket | Coffee Shop | Italian Restaurant | Shopping Mall | Furniture / Home Store | Sandwich Place | Theater | Cocktail Bar | Bakery |
| 9 | SE13 | 51.4572 | -0.0059 | 0 | Pub | Electronics Store | Supermarket | Park | Coffee Shop | Turkish Restaurant | Video Game Store | Café | Bus Stop | Fast Food Restaurant |

# Results and Discussion

The results of the Toronto clusters showed that:

1. Park is the most common venue, followed by trail and restaurant in Cluster 1. It tends to be a leisure and family welcome cluster.
2. Trial, playground and yoga studio are the most common venues in Cluster 2. It could be a leisure related cluster that closed to residential area.
3. Cafes, coffee shops and restaurants dominate the most common venues in Cluster 3, which suggests that it would be a tourist cluster.
4. Clusters 4 and 5 would likely be local community area as they have pub, music venue, garden, department store and so on.

The results of the London clusters showed that:

1. Cafes, coffee shops, restaurants, supermarkets dominant Cluster 1. It suggests that the cluster could be either residential or commercial areas as the cluster spanned across almost the entire London area.
2. Clusters 2, 3 and 4 could be small towns or villages as they have common local amenities such as pub, discount store, restaurants and zoos.
3. There are pubs, parks, hotels, train stations, bus stops and gas stations in Cluster 5. It suggests that it could be a cluster for local commuters.

## Conclusion

Since my client was looking for vibrant city lifestyle, both cities provide the lifestyle that they are after, particularly the Cluster 3 in Toronto and Cluster 1 in London. When I drilled down into these two clusters, the Cluster 1 in London was more appealing as it covered significant area of London which means there would be more places to explore and choose from to settle down.

However, London is known as an expensive place to live, albeit it's one of the three financial hubs in the world meaning there will be more job opportunities and getting higher pay than Toronto. There were also other constraints and factors that didn't take into consideration in this analysis, e.g., it only explored downtown Toronto and London post town, the commuting time and cost, catchment area, house price and so on didn't take into account in the analysis. But it should be a good starting point and built a foundation to discuss further with my client.