Comparing Helsinki Neighbourhoods

We will explore the neighoubour hoods of my home city - Helsinki, Finland. This is somewhat like the analysis we did as practise for New Your and Toronto, but with some twist! In addition to venues from Foursquare, we will using some other, socio-economical indicators, namely:

- Occupational structure of each region (relative number of children / students / unemployed / working / pensioners...)
- Average household income

This information, as well as postal codes of Helsinki, are available as open data.

I feel that adding socio-economical indicators - in addition to Foursquare venue data, that is, service offering of the area - will give a more complete picture of "nature" of each neighbourhood.

In [2]:
```
# Install and import required libraries, as in New Your/Toronto clustering notebook
import numpy as np # library to handle data in a vectorized manner

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

import json # library to handle JSON files

!conda install -c conda-forge geopy --yes # uncomment this line if you haven't com
pleted the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longit
ude values

import requests # library to handle requests
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

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

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

!conda install -c conda-forge geocoder --yes
import geocoder

!conda install -c conda-forge pyproj
import pyproj
```

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

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

    geographiclib: 1.49-py_0   conda-forge
    geopy:         1.17.0-py_0 conda-forge

geographiclib- 100% |################################| Time: 0:00:00  23.91 MB/s
geopy-1.17.0-p 100% |################################| Time: 0:00:00  24.34 MB/s
Fetching package metadata ............
Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

    altair:  2.2.2-py35_1 conda-forge
    branca:  0.3.1-py_0   conda-forge
    folium:  0.5.0-py_0   conda-forge
    vincent: 0.4.4-py_1   conda-forge

altair-2.2.2-p 100% |################################| Time: 0:00:00  49.74 MB/s
branca-0.3.1-p 100% |################################| Time: 0:00:00  33.67 MB/s
vincent-0.4.4- 100% |################################| Time: 0:00:00  39.97 MB/s
folium-0.5.0-p 100% |################################| Time: 0:00:00  47.21 MB/s
Fetching package metadata ............
Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following packages will be UPDATED:

    beautifulsoup4: 4.6.0-py35h442a8c9_1 --> 4.6.3-py35_0 conda-forge

beautifulsoup4 100% |################################| Time: 0:00:00  40.88 MB/s
Fetching package metadata ............
Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

    geocoder:   1.38.1-py_0  conda-forge
    orderedset: 2.0-py35_0   conda-forge
    ratelim:    0.1.6-py35_0 conda-forge

orderedset-2.0 100% |################################| Time: 0:00:00  57.12 MB/s
ratelim-0.1.6- 100% |################################| Time: 0:00:00   9.21 MB/s
geocoder-1.38. 100% |################################| Time: 0:00:00  39.15 MB/s
Fetching package metadata ............
Solving package specifications: .

Package plan for installation in environment /opt/conda/envs/DSX-Python35:

The following NEW packages will be INSTALLED:

    proj4:  4.9.3-h470a237_8       conda-forge
    pyproj: 1.9.5.1-py35h508ed2a_5 conda-forge

proj4-4.9.3-h4 100% |################################| Time: 0:00:00  67.69 MB/s
pyproj-1.9.5.1 100% |################################| Time: 0:00:00  55.24 MB/s
```

Postal codes and neighbourhood names of Helsinki

Helsinki Postal code data is available for feree from Helsinki Region Infoshare, see: https://hri.fi/data/en_GB/dataset/paakaupunkiseudun-postinumeroalueet (https://hri.fi/data/en_GB/dataset/paakaupunkiseudun-postinumeroalueet)

No API keys or such are needesd. Data is in JSON format, and needs to be parsed a little.

In [3]:
```
urlPostalCodes = 'https://hri.fi/data/api/action/datastore_search?resource_id=cbc11e4a-f695-4efa-93d7-9446066a07dd&limit=84'
results = requests.get(urlPostalCodes).json()['result']['records']
postal_code_list=[]
for p in results:
    postal_code_list.append((
            '00'+str(p['Postinumero']),
            p['Nimi']))
postal_codes = pd.DataFrame.from_records(postal_code_list, columns=['PostalCode', 'NeighbourhoodName'], index='PostalCode')
postal_codes.sort_index(inplace=True)
postal_codes.head()
```

Out[3]:

	NeighbourhoodName
PostalCode	
00100	Helsinki Keskusta - Etu-Töölö
00120	Punavuori
00130	Kaartinkaupunki
00140	Kaivopuisto - Ullanlinna
00150	Eira - Hernesaari

Occupation data for Helsinki region

Occupational data by postal code - and a wealth of other pieces of data - is available for free from Statistics Finland.

The data can also be browsed via a web interface in here:
http://pxnet2.stat.fi/PXWeb/pxweb/en/Postinumeroalueittainen_avoin_tieto/Postinumeroalueittainen_avoin_tieto__2018/paavo_8_pt_2018.px/?rxid=39840011-c10c-4e00-8cd1-7015d2e09479 (http://pxnet2.stat.fi/PXWeb/pxweb/en/Postinumeroalueittainen_avoin_tieto/Postinumeroalueittainen_avoin_tieto__2018/paavo_8_pt_2018.px/?rxid=39840011-c10c-4e00-8cd1-7015d2e09479).

The data received is in JSON "list format", that is, not grouped by postal code. We need to do some pivoting, as well as some further handling, to make the data usable. We need to have the relative share of various groups. The division used is the Finnish "standard division of labour", which is the following:

- Children (aged 0-14)
- Students
- Unemployed
- Workforce (which contains both employed and unemployed)
- Other (this means for example housewives)
- Pensioners

We want to separate between employed and unemployed, so we calculate employed=workforce-unemployed.

In [59]:
```
# Occupation data by postal code from Statistics Finland web service

# Define a function for this purpose
def fetchDataFromStatFinland(url, postalCodeList, dataItemList):
    postData={
      "query": [
        {
          "code": "Postinumeroalue",
          "selection": {
            "filter": "item",
            "values": postalCodeList

          }
        },
        {
          "code": "Tiedot",
          "selection": {
            "filter": "item",
            "values": dataItemList
          }
        }
      ],
      "response": {
        "format": "json"
      }
    }
    results=requests.post(url, json=postData).json()['data']
    data_list = []
    data_list.append([(
            row['key'][0],
            row['key'][1],
           row['values'][0]) for row in results])
    return data_list

# These are the data items we need
urlOccupationData = 'http://pxnet2.stat.fi/PXWeb/api/v1/en/Postinumeroalueittainen_
avoin_tieto/2018/paavo_8_pt_2018.px'
occupationDataItems = ["Pt_vakiy", "Pt_tyovy","Pt_tyott","Pt_0_14","Pt_opisk","Pt_e
lakel","Pt_muut"]
occupation_data_list = fetchDataFromStatFinland(urlOccupationData, postal_codes.ind
ex.values.tolist(), occupationDataItems)

# We need to wrangle with the data a bit, since is is "list" format, that is, not g
rouped by postal code => pivoting the data frame does the trick
occupation_data = pd.DataFrame.from_records(occupation_data_list[0], columns=['Post
alCode', 'OccupationCategory', 'NumberInThisOccupation'])
occupation_data_pivoted = occupation_data.pivot(index='PostalCode', columns='Occupa
tionCategory', values='NumberInThisOccupation')
# Translate the column names
occupation_data_pivoted.rename(columns={'Pt_vakiy':'Total','Pt_0_14':'Child','Pt_el
akel':'Pensioner','Pt_muut':'Others','Pt_opisk':'Student','Pt_tyott':'Unemployed','
Pt_tyovy':'Workforce'}, inplace=True)
occupation_data_pivoted = occupation_data_pivoted.replace('.','0')
columnNames = ['Total','Child','Pensioner','Others','Student','Unemployed','Workfor
ce']
# Now calculate the relative share of different occupations - noting what we are to
ld workforce and unemployed - we have to calculate the relative share of working pe
ople
occupation_data_pivoted[columnNames] = occupation_data_pivoted[columnNames].apply(p
d.to_numeric)
occupation_data_pivoted['EmployedR'] = (occupation_data_pivoted['Workforce']-occupa
tion_data_pivoted['Unemployed'])  / occupation_data_pivoted['Total']
occupation_data_pivoted['UnemployedR'] = occupation_data_pivoted['Unemployed']  / o
ccupation_data_pivoted['Total']
```

Out[59]:

OccupationCategory	EmployedR	UnemployedR	ChildR	StudentR	OthersR	PensionerR
PostalCode						
00100	0.552440	0.048355	0.099228	0.070405	0.045165	0.184408
00120	0.546114	0.049815	0.113009	0.061771	0.051950	0.177341
00130	0.561198	0.045573	0.110677	0.072266	0.046224	0.164062
00140	0.525905	0.048996	0.114110	0.064987	0.054880	0.191122
00150	0.569631	0.057641	0.098075	0.060652	0.054414	0.159587

In [60]:

```
# Now, merge this with post number data
helsinki_data = pd.concat([postal_codes, occupation_data_pivoted], axis=1)
helsinki_data.head()
```

Out[60]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958

Median household income for Helsinki region

This we can also fetch from Statistics Finland, from a different data set: http://pxnet2.stat.fi/PXWeb/pxweb/en/Postinumeroalueittainen_avoin_tieto/Postinumeroalueittainen_avoin_tieto__2018/paavo_5_tr_2018.px/?rxid=39840011-c10c-4e00-8cd1-7015d2e09479 (http://pxnet2.stat.fi/PXWeb/pxweb/en/Postinumeroalueittainen_avoin_tieto/Postinumeroalueittainen_avoin_tieto__2018/paavo_5_tr_2018.px/?rxid=39840011-c10c-4e00-8cd1-7015d2e09479)

Data handling is analogous to data handling of the occupational data.

In [61]:

```
# Next, get median income per household per postal code area, from Statistics Finla
nd
urlIncomeData = 'http://pxnet2.stat.fi/PXWeb/api/v1/en/Postinumeroalueittainen_avoi
n_tieto/2018/paavo_5_tr_2018.px'
incomeDataList = ['Tr_mtu']
income_data_list = fetchDataFromStatFinland(urlIncomeData, postal_codes.index.value
s.tolist(), incomeDataList)
income_data = pd.DataFrame.from_records(income_data_list[0], columns=['PostalCode',
'IncomeCategory', 'MedianHouseholdIncome'])

income_data_pivoted = income_data.pivot(index='PostalCode', columns='IncomeCategory
', values='MedianHouseholdIncome')
# Translate the column names
income_data_pivoted.rename(columns={'Tr_mtu':'MedianHouseholdIncome'}, inplace=True
)
income_data_pivoted = income_data_pivoted.replace('.','0')
columnNames = ['MedianHouseholdIncome']
# Now normalize the income
income_data_pivoted[columnNames] = income_data_pivoted[columnNames].apply(pd.to_num
eric)
#Standard scaled
income_data_pivoted['MedianHouseholdIncomeNorm'] = \
    (income_data_pivoted['MedianHouseholdIncome']-income_data_pivoted['MedianHouseh
oldIncome'].mean())/ \
    income_data_pivoted['MedianHouseholdIncome'].std()
#Min-max scaled
#income_data_pivoted['MedianHouseholdIncomeNorm'] = \
#    (income_data_pivoted['MedianHouseholdIncome']-income_data_pivoted['MedianHouse
holdIncome'].min()) / \
#    (income_data_pivoted['MedianHouseholdIncome'].max() - income_data_pivoted['Med
ianHouseholdIncome'].min())
income_data_pivoted.drop(columnNames, inplace=True, axis=1)
income_data_pivoted.head()
```

Out[61]:

IncomeCategory	MedianHouseholdIncomeNorm
PostalCode	
00100	0.135282
00120	0.311709
00130	0.701486
00140	0.397746
00150	-0.284276

In [62]:
```
# Merge this, too with the existing dataset
helsinki_data = pd.concat([helsinki_data, income_data_pivoted], axis=1)
helsinki_data.head()
```

Out[62]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958

Housing type and average size per neighborhood

We use the same style as above, but this time we'll fetch some housing type (apartment/house) and average apartment/house size. I believe this differentiates neighborhoods pretty much.

In [63]:

```
# Next, get median income per household per postal code area, from Statistics Finla
nd
urlHousingData = 'http://pxnet2.stat.fi/PXWeb/api/v1/en/Postinumeroalueittainen_avo
in_tieto/2018/paavo_6_ra_2018.px'
housingDataList = ['Ra_as_kpa','Ra_pt_as','Ra_kt_as']
housing_data_list = fetchDataFromStatFinland(urlHousingData, postal_codes.index.val
ues.tolist(), housingDataList)
housing_data = pd.DataFrame.from_records(housing_data_list[0], columns=['PostalCode
', 'DataKey', 'DataValue'])

housing_data_pivoted = housing_data.pivot(index='PostalCode', columns='DataKey', va
lues='DataValue')
# Translate the column names
housing_data_pivoted.rename(columns={'Ra_as_kpa':'AverageFloorSize','Ra_pt_as':'Dwe
llingsHouse','Ra_kt_as':'DwellingsApartment'}, inplace=True)
housing_data_pivoted = housing_data_pivoted.replace('.','0')
columnNames = ['AverageFloorSize','DwellingsHouse','DwellingsApartment']
housing_data_pivoted[columnNames] = housing_data_pivoted[columnNames].apply(pd.to_n
umeric)
# Normalize average housing size & relative shares of house / apartment dwellings
housing_data_pivoted['AverageFloorSizeR'] = \
    (housing_data_pivoted['AverageFloorSize']-housing_data_pivoted['AverageFloorSiz
e'].min()) / \
    (housing_data_pivoted['AverageFloorSize'].max() - housing_data_pivoted['Average
FloorSize'].min())
housing_data_pivoted['DwellingsHouseR'] = \
    housing_data_pivoted['DwellingsHouse']/(housing_data_pivoted['DwellingsHouse']+
housing_data_pivoted['DwellingsApartment'])
housing_data_pivoted['DwellingsApartmentR'] = \
    housing_data_pivoted['DwellingsApartment']/(housing_data_pivoted['DwellingsHous
e']+housing_data_pivoted['DwellingsApartment'])
housing_data_pivoted.drop(columnNames, inplace=True, axis=1)
housing_data_pivoted.head()
```

Out[63]:

DataKey	AverageFloorSizeR	DwellingsHouseR	DwellingsApartmentR
PostalCode			
00100	0.408390	0.000170	0.999830
00120	0.426897	0.001583	0.998417
00130	0.462060	0.000000	1.000000
00140	0.455891	0.002580	0.997420
00150	0.342998	0.004730	0.995270

In [64]:
```
# Merge this, too with the existing dataset
helsinki_data = pd.concat([helsinki_data, housing_data_pivoted], axis=1)
helsinki_data.head()
```

Out[64]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958

Coordinates of neighbourhoods

Now we proceed in the same way as we did with Toronto - we fetch coordinates of each postal code by Bing geocoding. That seems to work quite reliably.

In [10]:
```
# @hidden_cell
BING_KEY = 'AimCCc_XOtX3tc1vDbyskkGUEj3C8Uq-GydnRGUixxFqdvy8yRE-zaj7NQz-LGDt'
FOURSQUARE_CLIENT_ID = 'QZHB1I2ZGTCB4HUYONOFZBW5ZSS4X10UFM3OBUJEFOSMVOCC'
FOURSQUARE_CLIENT_SECRET = 'PRHHWELKYMDYR045BR3C2P4NYINH2YUPKJHBDXESPLDFTV44'
```

In [65]:
```
# A function to return lat and long given postal code
def fetchCoordinatesByAddress(address):
    coords = None
    n_times = 0
    while((coords is None) & (n_times < 5)):
        print('Trying to find {} from Bing'.format(address))
        g = geocoder.bing(address, key=BING_KEY)
        coords = g.latlng
        n_times = n_times + 1
    if coords != None:
        return coords[0], coords[1]
    else:
        # Open Street Map address not found, let us try Bing instead
        print('Trying to find {} from Open Street Map'.format(address))
        g = geocoder.osm(address)
        coords = g.osm
        if coords != None:
            return coords['y'], coords['x']
        else:
            return 0.0, 0.0

# Loop through all areas and fetch coordinates by postal code
lats=[]
longs=[]
for index, area in postal_codes.iterrows():
    lat, lon = fetchCoordinatesByAddress('{}, Helsinki, Finland'.format(index))
    lats.append(lat)
    longs.append(lon)
helsinki_data = helsinki_data.assign(Latitude=lats, Longitude=longs)
helsinki_data.head()
```

```
Trying to find 00100, Helsinki, Finland from Bing
Trying to find 00120, Helsinki, Finland from Bing
Trying to find 00130, Helsinki, Finland from Bing
Trying to find 00140, Helsinki, Finland from Bing
Trying to find 00150, Helsinki, Finland from Bing
Trying to find 00160, Helsinki, Finland from Bing
Trying to find 00170, Helsinki, Finland from Bing
Trying to find 00180, Helsinki, Finland from Bing
Trying to find 00190, Helsinki, Finland from Bing
Trying to find 00200, Helsinki, Finland from Bing
Trying to find 00210, Helsinki, Finland from Bing
Trying to find 00220, Helsinki, Finland from Bing
Trying to find 00230, Helsinki, Finland from Bing
Trying to find 00240, Helsinki, Finland from Bing
Trying to find 00250, Helsinki, Finland from Bing
Trying to find 00260, Helsinki, Finland from Bing
Trying to find 00270, Helsinki, Finland from Bing
Trying to find 00280, Helsinki, Finland from Bing
Trying to find 00290, Helsinki, Finland from Bing
Trying to find 00300, Helsinki, Finland from Bing
Trying to find 00310, Helsinki, Finland from Bing
Trying to find 00320, Helsinki, Finland from Bing
Trying to find 00330, Helsinki, Finland from Bing
Trying to find 00340, Helsinki, Finland from Bing
Trying to find 00350, Helsinki, Finland from Bing
Trying to find 00360, Helsinki, Finland from Bing
Trying to find 00370, Helsinki, Finland from Bing
Trying to find 00380, Helsinki, Finland from Bing
Trying to find 00390, Helsinki, Finland from Bing
Trying to find 00400, Helsinki, Finland from Bing
Trying to find 00410, Helsinki, Finland from Bing
Trying to find 00420, Helsinki, Finland from Bing
Trying to find 00430, Helsinki, Finland from Bing
Trying to find 00440, Helsinki, Finland from Bing
Trying to find 00500, Helsinki, Finland from Bing
Trying to find 00510, Helsinki, Finland from Bing
Trying to find 00520, Helsinki, Finland from Bing
Trying to find 00530, Helsinki, Finland from Bing
Trying to find 00540, Helsinki, Finland from Bing
Trying to find 00550, Helsinki, Finland from Bing
Trying to find 00560, Helsinki, Finland from Bing
Trying to find 00570, Helsinki, Finland from Bing
Trying to find 00580, Helsinki, Finland from Bing
Trying to find 00590, Helsinki, Finland from Bing
Trying to find 00600, Helsinki, Finland from Bing
Trying to find 00610, Helsinki, Finland from Bing
Trying to find 00620, Helsinki, Finland from Bing
Trying to find 00630, Helsinki, Finland from Bing
Trying to find 00640, Helsinki, Finland from Bing
Trying to find 00650, Helsinki, Finland from Bing
Trying to find 00660, Helsinki, Finland from Bing
Trying to find 00670, Helsinki, Finland from Bing
Trying to find 00680, Helsinki, Finland from Bing
Trying to find 00690, Helsinki, Finland from Bing
Trying to find 00700, Helsinki, Finland from Bing
Trying to find 00710, Helsinki, Finland from Bing
Trying to find 00720, Helsinki, Finland from Bing
Trying to find 00730, Helsinki, Finland from Bing
Trying to find 00740, Helsinki, Finland from Bing
Trying to find 00750, Helsinki, Finland from Bing
Trying to find 00760, Helsinki, Finland from Bing
Trying to find 00770, Helsinki, Finland from Bing
Trying to find 00780, Helsinki, Finland from Bing
Trying to find 00790, Helsinki, Finland from Bing
```

Out[65]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pension
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958

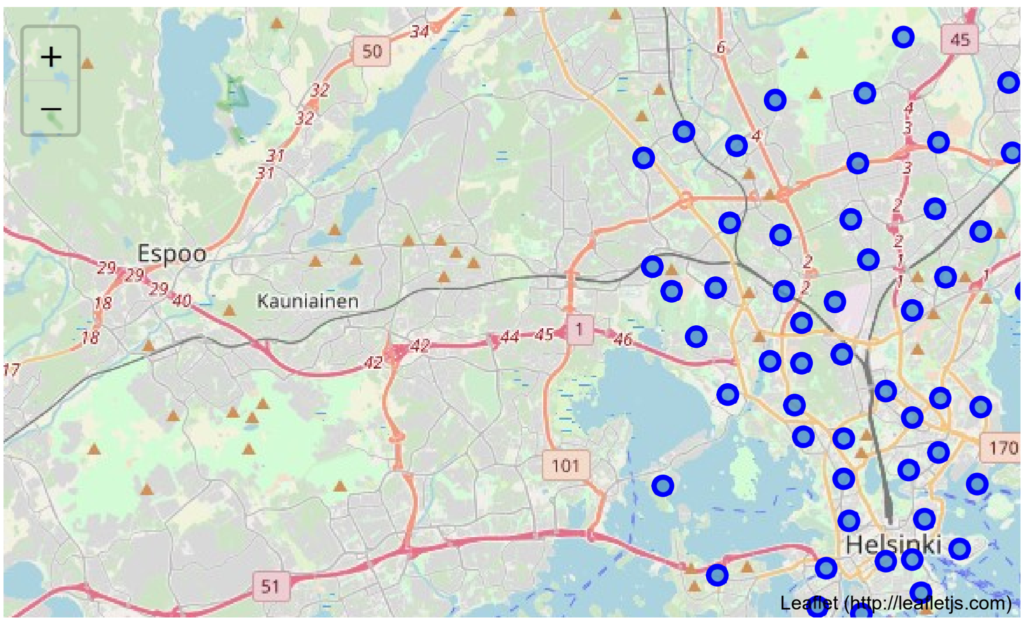
Put the Helsinki Neighbourhoods on the map

We'll make a Folium map showing all the neighbourhoods on the map.

In [66]:
```
# We first need the address of Helsinki, Finland

helsinkiLat, helsinkiLon = fetchCoordinatesByAddress('Helsinki, Finland')
map_helsinki = folium.Map(location=[helsinkiLat, helsinkiLon], zoom_start=11)

# add markers to map
for lat, lng, postalCode, label in zip(helsinki_data['Latitude'], helsinki_data['Lo
ngitude'], helsinki_data.index, helsinki_data['NeighbourhoodName']):
    label = folium.Popup(postalCode + " " + 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_helsinki)

map_helsinki
```

Trying to find Helsinki, Finland from Bing

Out[66]:



Neighbourhood venues from Foursquare

This we do just as we did for New York and Toronto.

In [69]:
```
VERSION = '20180605' # Foursquare API version
LIMIT=100

# This is copied from the New York lab
def getNearbyVenues(postalCodes, names, latitudes, longitudes, radius=500):
    venues_list=[]
    for postalCode, name, lat, lng in zip(postalCodes, 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(
            FOURSQUARE_CLIENT_ID,
            FOURSQUARE_CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)
        #print(url)
        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            postalCode,
            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 = ['PostalCode',
                  'NeighbourhoodName',
                  'Neighbourhood Latitude',
                  'Neighbourhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)

helsinki_venues = getNearbyVenues(helsinki_data.index, helsinki_data['Neighbourhood
Name'], helsinki_data['Latitude'], helsinki_data['Longitude'])
helsinki_venues.head()
```

```
Helsinki Keskusta - Etu-Töölö
Punavuori
Kaartinkaupunki
Kaivopuisto - Ullanlinna
Eira - Hernesaari
Katajanokka
Kruununhaka
Kamppi - Ruoholahti
Suomenlinna
Lauttasaari
Vattuniemi
Jätkäsaari
Ilmala
Länsi-Pasila
Taka-Töölö
Keski-Töölö
Pohjois-Meilahti
Ruskeasuo
Meilahden sairaala-alue
Pikku Huopalahti
Kivihaka
Etelä-Haaga
Munkkiniemi
Kuusisaari-Lehtisaari
Munkkivuori-Niemenmäki
Pajamäki
Reimarla
Pitäjänmäen teollisuusalue
Konala
Pohjois-Haaga
Malminkartano
Kannelmäki
Maununneva
Lassila
Sörnäinen
Etu-Vallila - Alppila
Itä-Pasila
Kallio
Kalasatama
Vallila
Toukola-Vanhakaupunki
Kulosaari
Verkkosaari
Kaitalahti
Koskela-Helsinki
Käpylä
Metsälä-Etelä-Oulunkylä
Maunula-Suursuo
Oulunkylä-Patola
Veräjämäki
Länsi-Pakila
Paloheinä
Itä-Pakila
Tuomarinkylä-Torpparinmäki
Malmi
Pihlajamäki
Pukinmäki-Savela
Tapanila
Siltamäki
Puistola
Suurmetsä
Jakomäki - Alppikylä
Tapaninvainio
Viikki
```

Out[69]:

	PostalCode	NeighbourhoodName	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	V Long
0	00100	Helsinki Keskusta - Etu-Töölö	60.17202	24.925289	Cafetoria	60.173203	24.92
1	00100	Helsinki Keskusta - Etu-Töölö	60.17202	24.925289	Ateljé Finne	60.171198	24.92
2	00100	Helsinki Keskusta - Etu-Töölö	60.17202	24.925289	Twisted Street Kitchen	60.170641	24.92
3	00100	Helsinki Keskusta - Etu-Töölö	60.17202	24.925289	Hoshito	60.171347	24.92
4	00100	Helsinki Keskusta - Etu-Töölö	60.17202	24.925289	Temppeliaukio	60.172552	24.92

In [70]:

```
helsinki_venues.shape
```

Out[70]: (1518, 8)

In [71]:
```
# See number of venues by neighbourhood
helsinki_venues.groupby('PostalCode').count()
```

Out[71]:

	NeighbourhoodName	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	V Cate
PostalCode							
00100	40	40	40	40	40	40	40
00120	100	100	100	100	100	100	100
00130	91	91	91	91	91	91	91
00140	50	50	50	50	50	50	50
00150	11	11	11	11	11	11	11
00160	31	31	31	31	31	31	31
00170	90	90	90	90	90	90	90
00180	30	30	30	30	30	30	30
00190	21	21	21	21	21	21	21
00200	27	27	27	27	27	27	27
00210	24	24	24	24	24	24	24
00220	16	16	16	16	16	16	16
00230	3	3	3	3	3	3	3
00240	20	20	20	20	20	20	20
00250	39	39	39	39	39	39	39
00260	73	73	73	73	73	73	73
00270	20	20	20	20	20	20	20
00280	8	8	8	8	8	8	8
00290	20	20	20	20	20	20	20
00300	8	8	8	8	8	8	8
00310	6	6	6	6	6	6	6
00320	10	10	10	10	10	10	10
00330	19	19	19	19	19	19	19
00340	7	7	7	7	7	7	7
00350	5	5	5	5	5	5	5
00360	6	6	6	6	6	6	6
00370	15	15	15	15	15	15	15
00380	12	12	12	12	12	12	12
00390	23	23	23	23	23	23	23
00400	10	10	10	10	10	10	10
00410	5	5	5	5	5	5	5
00420	5	5	5	5	5	5	5
00430	3	3	3	3	3	3	3
00440	11	11	11	11	11	11	11
00500	58	58	58	58	58	58	58

In [72]:
```
# See how many neighbourhoods got venues
helsinki_venues.groupby('PostalCode').count().shape
```

Out[72]: (83, 7)

In [73]:
```
helsinki_data.shape
```

Out[73]: (84, 13)

In [74]:
```
# Oh well, we have a problem, since not all of the postal have venues - luckily onl
y one. Let us remove it from helsinki_data so we can combine info later
helsinki_venues.groupby('PostalCode').count().index.values
```

Out[74]:
```
array(['00100', '00120', '00130', '00140', '00150', '00160', '00170',
       '00180', '00190', '00200', '00210', '00220', '00230', '00240',
       '00250', '00260', '00270', '00280', '00290', '00300', '00310',
       '00320', '00330', '00340', '00350', '00360', '00370', '00380',
       '00390', '00400', '00410', '00420', '00430', '00440', '00500',
       '00510', '00520', '00530', '00540', '00550', '00560', '00570',
       '00580', '00590', '00600', '00610', '00620', '00630', '00640',
       '00650', '00660', '00670', '00680', '00690', '00700', '00710',
       '00720', '00730', '00740', '00750', '00760', '00770', '00780',
       '00790', '00800', '00810', '00820', '00830', '00840', '00850',
       '00860', '00870', '00880', '00900', '00910', '00920', '00930',
       '00940', '00950', '00960', '00970', '00980', '00990'], dtype=object)
```

In [75]:
```
helsinki_data.index.values
```

Out[75]:
```
array(['00100', '00120', '00130', '00140', '00150', '00160', '00170',
       '00180', '00190', '00200', '00210', '00220', '00230', '00240',
       '00250', '00260', '00270', '00280', '00290', '00300', '00310',
       '00320', '00330', '00340', '00350', '00360', '00370', '00380',
       '00390', '00400', '00410', '00420', '00430', '00440', '00500',
       '00510', '00520', '00530', '00540', '00550', '00560', '00570',
       '00580', '00590', '00600', '00610', '00620', '00630', '00640',
       '00650', '00660', '00670', '00680', '00690', '00700', '00710',
       '00720', '00730', '00740', '00750', '00760', '00770', '00780',
       '00790', '00800', '00810', '00820', '00830', '00840', '00850',
       '00860', '00870', '00880', '00890', '00900', '00910', '00920',
       '00930', '00940', '00950', '00960', '00970', '00980', '00990'], dtype=obj
ect)
```

In [76]:
```
# We can see that postal code 00890 in helsinki_data has no values. We'll delete it
to stay away from harm. Sorry, 00890.
helsinki_data.drop('00890', inplace=True)
helsinki_data.index.values
```

Out[76]:
```
array(['00100', '00120', '00130', '00140', '00150', '00160', '00170',
       '00180', '00190', '00200', '00210', '00220', '00230', '00240',
       '00250', '00260', '00270', '00280', '00290', '00300', '00310',
       '00320', '00330', '00340', '00350', '00360', '00370', '00380',
       '00390', '00400', '00410', '00420', '00430', '00440', '00500',
       '00510', '00520', '00530', '00540', '00550', '00560', '00570',
       '00580', '00590', '00600', '00610', '00620', '00630', '00640',
       '00650', '00660', '00670', '00680', '00690', '00700', '00710',
       '00720', '00730', '00740', '00750', '00760', '00770', '00780',
       '00790', '00800', '00810', '00820', '00830', '00840', '00850',
       '00860', '00870', '00880', '00900', '00910', '00920', '00930',
       '00940', '00950', '00960', '00970', '00980', '00990'], dtype=object)
```

Reshape venue data according to venue type & analyze

Next we'll see what types of venues are most typical in each area. To do this, we one-hot encode the venues.

In [77]:
```
# One-hot encoding of the venues
helsinki_onehot = pd.get_dummies(helsinki_venues[['Venue Category']], prefix="", pr
efix_sep="")

helsinki_onehot['PostalCode'] = helsinki_venues['PostalCode']

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

helsinki_onehot.head()
```

Out[77]:

	PostalCode	ATM	American Restaurant	Antique Shop	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Auditorium	Auto Worksho
0	00100	0	0	0	0	0	0	0	0	0
1	00100	0	0	0	0	0	0	0	0	0
2	00100	0	0	0	0	0	0	1	0	0
3	00100	0	0	0	0	0	0	0	0	0
4	00100	0	0	0	0	0	0	0	0	0

In [78]:
```
# Venue types per Neighbourhood
helsinki_grouped = helsinki_onehot.groupby('PostalCode').mean().reset_index()
helsinki_grouped.set_index('PostalCode', inplace=True)
helsinki_grouped
```

Out[78]:

	ATM	American Restaurant	Antique Shop	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Auditorium	Aut Worksho
PostalCode									
00100	0.0	0.000000	0.000000	0.025000	0.025000	0.00	0.050000	0.000000	0.000000
00120	0.0	0.010000	0.000000	0.020000	0.000000	0.01	0.010000	0.000000	0.000000
00130	0.0	0.010989	0.010989	0.010989	0.010989	0.00	0.010989	0.000000	0.000000
00140	0.0	0.000000	0.020000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00150	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00160	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.032258	0.000000
00170	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00180	0.0	0.000000	0.000000	0.033333	0.000000	0.00	0.000000	0.000000	0.000000
00190	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00200	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00210	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.041667	0.000000	0.000000
00220	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00230	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00240	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00250	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00260	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.013699	0.000000	0.000000
00270	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.050000	0.000000	0.000000
00280	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00290	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00300	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00310	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00320	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00330	0.0	0.000000	0.000000	0.000000	0.052632	0.00	0.000000	0.000000	0.000000
00340	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00350	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00360	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00370	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00380	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00390	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.043478
00400	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00410	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00420	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000
00430	0.0	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.000000

In [79]:
```
# Get the top 5 venue types per neighbourhood
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]

num_top_venues = 5

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

# create columns according to number of top venues
columns = ['PostalCode']
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['PostalCode'] = helsinki_grouped.index

for ind in np.arange(helsinki_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(helsinki_
grouped.iloc[ind, :], num_top_venues)
neighborhoods_venues_sorted.set_index('PostalCode', inplace=True)
neighborhoods_venues_sorted
```

Out[79]:

	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue
PostalCode					
00100	Pub	Coffee Shop	Café	Scandinavian Restaurant	Multiplex
00120	Café	Scandinavian Restaurant	Coffee Shop	Bar	Vietnamese Restaurant
00130	Scandinavian Restaurant	Hotel	Café	Restaurant	Coffee Shop
00140	Park	Coffee Shop	Ice Cream Shop	Grocery Store	Playground
00150	Scandinavian Restaurant	Modern European Restaurant	Gym / Fitness Center	Park	Turkish Restaurant
00160	Park	Scandinavian Restaurant	Hotel	Bar	Tram Station
00170	Pizza Place	Café	Boat or Ferry	Coffee Shop	Scandinavian Restaurant
00180	Restaurant	Gym	Hotel	Grocery Store	Bar
00190	History Museum	Café	Restaurant	Scenic Lookout	Castle
00200	Bus Stop	Pizza Place	Skate Park	Flea Market	Supermarket
00210	Gym / Fitness Center	Restaurant	Supermarket	Italian Restaurant	Bar
00220	Electronics Store	Tram Station	Park	Cruise	Café
00230	Gym	Café	Forest	Zoo	Food Court
00240	Bus Stop	Gym / Fitness Center	Hockey Arena	Restaurant	Bar
00250	Thai Restaurant	Indian Restaurant	Soccer Stadium	Music Venue	Himalayan Restaurant
00260	Sushi Restaurant	Café	Coffee Shop	Hotel	Italian Restaurant
00270	Park	Bar	Playground	Scandinavian Restaurant	Gym
00280	Park	Pharmacy	Bus Stop	Garden	Himalayan Restaurant
00290	Bus Stop	Park	Café	Scandinavian Restaurant	Spa
00300	Plaza	Bus Line	Flea Market	Himalayan Restaurant	Convenience Store
00310	Bus Stop	Café	Garden	Tunnel	Zoo
00320	Bus Stop	Restaurant	Supermarket	Gym / Fitness Center	Hotel
00330	Café	Pizza Place	Gastropub	Himalayan Restaurant	Art Museum
00340	Coffee Shop	Bus Stop	Grocery Store	Café	Falafel Restaurant

In [80]:
```
# Finally, add the grouped venues data to our master dataframe
# Merge this, too with the existing dataset
helsinki_data = pd.concat([helsinki_data, helsinki_grouped], axis=1)
helsinki_data.head(20)
```

Out[80]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958
00160	Katajanokka	0.461521	0.044519	0.138926	0.062416	0.044966	0.24765
00170	Kruununhaka	0.556894	0.041943	0.117440	0.068597	0.041402	0.17372
00180	Kamppi - Ruoholahti	0.554610	0.058448	0.115832	0.065212	0.050467	0.15543
00190	Suomenlinna	0.477273	0.056818	0.236111	0.068182	0.046717	0.11489
00200	Lauttasaari	0.531131	0.046177	0.137323	0.061256	0.035587	0.18852
00210	Vattuniemi	0.499156	0.037937	0.152007	0.045862	0.030401	0.23463
00220	Jätkäsaari	0.568197	0.048321	0.152502	0.120631	0.033585	0.07676
00230	Ilmala	NaN	NaN	NaN	NaN	NaN	NaN
00240	Länsi-Pasila	0.482094	0.061892	0.111717	0.076294	0.036785	0.23121
00250	Taka-Töölö	0.583807	0.051449	0.090441	0.066522	0.036285	0.17149
00260	Keski-Töölö	0.504795	0.046846	0.088528	0.057174	0.050350	0.25230
00270	Pohjois-Meilahti	0.536107	0.062925	0.125850	0.072344	0.034668	0.16810
00280	Ruskeasuo	0.530667	0.049000	0.110333	0.104333	0.027333	0.17833
00290	Meilahden sairaala-alue	0.546584	0.049689	0.105590	0.024845	0.080745	0.19254
00300	Pikku Huopalahti	0.445450	0.069319	0.168516	0.080417	0.036537	0.19976

In [81]:
```
# We see that postal code area 00230 does not contain any relevant data - drop it
helsinki_data.drop('00230', inplace=True)
```

In [116]:
```
helsinki_data.shape
```

Out[116]: (82, 260)

Cluster analysis of our Helsinki data

Now we will perform a cluster analysis of our data.

K-means clustering

In [82]:
```
# set number of clusters
kclusters = 6

# Drop columns that are not in numeric 0..1 range from the cluster analysis
helsinki_grouped_clustering = helsinki_data.drop({'NeighbourhoodName', 'Latitude',
'Longitude'}, 1)
helsinki_grouped_clustering.fillna(0, inplace=True)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, init='random', n_init=20, random_state=2).fit
(helsinki_grouped_clustering)

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

Out[82]:
```
array([5, 5, 5, 5, 4, 5, 5, 4, 5, 4], dtype=int32)
```

Try DBSCAN

In [100]:
```
from sklearn.cluster import DBSCAN
dbscanResult = DBSCAN(eps=4, min_samples=3).fit(helsinki_grouped_clustering)
dbscanResult.labels_
```

Out[100]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Try hierarchical clustering

In [107]:
```
from scipy import ndimage
from scipy.cluster import hierarchy
from scipy.spatial import distance_matrix
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
import pylab
import scipy.cluster.hierarchy
Z = hierarchy.linkage(helsinki_grouped_clustering, 'complete')
```

In [110]:
```
from scipy.cluster.hierarchy import fcluster
max_d = 2
clusters = fcluster(Z, max_d, criterion='distance')
clusters
```

Out[110]:
```
array([1, 3, 3, 3, 1, 3, 3, 1, 3, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, 1, 1,
       1, 1, 1, 2, 2, 2, 3, 1, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1, 2,
       3, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1], dtype=int32)
```

It is going to be K-means clustering...

Since I was not able to get proper results from other algorithms...

In [83]:
```
# combine datasets
helsinki_merged = helsinki_data

# add clustering labels
helsinki_merged['Cluster Labels'] = kmeans.labels_

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighb
orhood
helsinki_merged = helsinki_merged.join(neighborhoods_venues_sorted)

helsinki_merged.head(20) # check the last columns!
```

Out[83]:

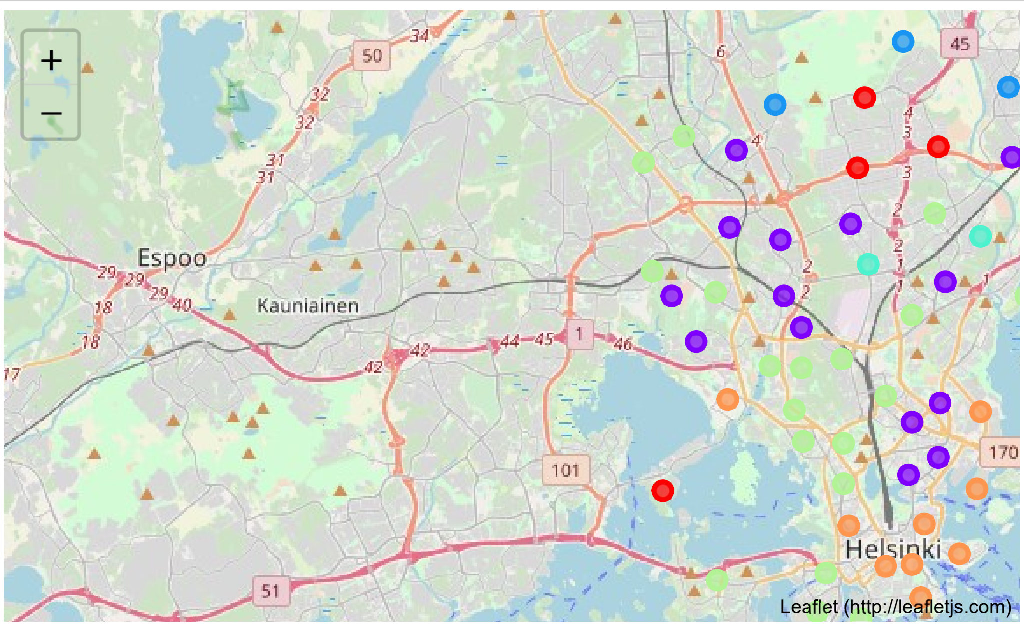
	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958
00160	Katajanokka	0.461521	0.044519	0.138926	0.062416	0.044966	0.24765
00170	Kruununhaka	0.556894	0.041943	0.117440	0.068597	0.041402	0.17372
00180	Kamppi - Ruoholahti	0.554610	0.058448	0.115832	0.065212	0.050467	0.15543
00190	Suomenlinna	0.477273	0.056818	0.236111	0.068182	0.046717	0.11489
00200	Lauttasaari	0.531131	0.046177	0.137323	0.061256	0.035587	0.18852
00210	Vattuniemi	0.499156	0.037937	0.152007	0.045862	0.030401	0.23463
00220	Jätkäsaari	0.568197	0.048321	0.152502	0.120631	0.033585	0.07676
00240	Länsi-Pasila	0.482094	0.061892	0.111717	0.076294	0.036785	0.23121
00250	Taka-Töölö	0.583807	0.051449	0.090441	0.066522	0.036285	0.17149
00260	Keski-Töölö	0.504795	0.046846	0.088528	0.057174	0.050350	0.25230
00270	Pohjois-Meilahti	0.536107	0.062925	0.125850	0.072344	0.034668	0.16810
00280	Ruskeasuo	0.530667	0.049000	0.110333	0.104333	0.027333	0.17833
00290	Meilahden sairaala-alue	0.546584	0.049689	0.105590	0.024845	0.080745	0.19254
00300	Pikku Huopalahti	0.445450	0.069319	0.168516	0.080417	0.036537	0.19976

Visualizing and analyzing the clustering results

Now we are ready to show our results on map, as well as try to make sense out of the clusters

In [84]:
```
# Visualize the clusters - straight from New York lab

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

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

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(helsinki_merged['Latitude'], helsinki_merged['Lon
gitude'], helsinki_merged['NeighbourhoodName'], helsinki_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
```

Out[84]:



In [85]:
```
# Examine cluster 0
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 0]
```

Out[85]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00340	Kuusisaari-Lehtisaari	0.437727	0.037485	0.162636	0.083434	0.056227	0.22249
00590	Kaitalahti	0.452632	0.050000	0.205263	0.086842	0.042105	0.16315
00660	Länsi-Pakila	0.433841	0.034349	0.182497	0.076912	0.025687	0.24671
00670	Paloheinä	0.475312	0.031920	0.211804	0.070657	0.022610	0.18769
00680	Itä-Pakila	0.448626	0.041209	0.198077	0.087912	0.029945	0.19423
00830	Tammisalo	0.412417	0.030155	0.188470	0.068736	0.050998	0.24922
00850	Jollas	0.459442	0.041680	0.222187	0.080795	0.035268	0.16062

In [86]:
```
# Examine cluster 1
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 1]
```

Out[86]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00310	Kivihaka	0.529018	0.061384	0.108259	0.068080	0.042411	0.19084
00320	Etelä-Haaga	0.547399	0.052437	0.104051	0.070327	0.032079	0.19370
00350	Munkkivuori-Niemenmäki	0.501020	0.052281	0.136983	0.080301	0.036071	0.19334
00360	Pajamäki	0.501867	0.066667	0.112000	0.078933	0.041600	0.19893
00400	Pohjois-Haaga	0.453545	0.064762	0.113545	0.087090	0.049206	0.23185
00420	Kannelmäki	0.434278	0.074194	0.127046	0.085803	0.054943	0.22373
00440	Lassila	0.433415	0.054066	0.113228	0.074673	0.031908	0.29271
00500	Sörnäinen	0.619195	0.098431	0.044619	0.073625	0.044619	0.11951
00510	Etu-Vallila - Alppila	0.589441	0.095456	0.061372	0.064380	0.040655	0.14869
00530	Kallio	0.568558	0.078176	0.050849	0.067245	0.042169	0.19300
00550	Vallila	0.562660	0.074829	0.077498	0.094257	0.037468	0.15328
00600	Koskela-Helsinki	0.361893	0.055583	0.129854	0.095388	0.048058	0.30922
00630	Maunula-Suursuo	0.418243	0.075069	0.142427	0.065189	0.043017	0.25605
00700	Malmi	0.431618	0.083485	0.151032	0.081740	0.046372	0.20575
00710	Pihlajamäki	0.427244	0.084215	0.151442	0.064183	0.048397	0.22451
00720	Pukinmäki-Savela	0.441548	0.068038	0.139818	0.081015	0.042319	0.22726
00770	Jakomäki - Alppikylä	0.404489	0.085869	0.170684	0.084513	0.051973	0.20247
00800	Länsi-Herttoniemi	0.460483	0.086661	0.133943	0.073627	0.041875	0.20341
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In [87]:
```
# Examine cluster 2
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 2]
```

Out[87]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00430	Maununneva	0.477887	0.037469	0.186732	0.093161	0.026618	0.17813
00690	Tuomarinkylä-Torpparinmäki	0.459683	0.043197	0.218862	0.083153	0.028078	0.16702
00760	Suurmetsä	0.476937	0.040814	0.196942	0.081886	0.028893	0.17452
00780	Tapaninvainio	0.452929	0.054608	0.155432	0.064562	0.030148	0.24232
00950	Vartioharju	0.472908	0.044946	0.174276	0.073548	0.030734	0.20358

In [88]:
```
# Examine cluster 3
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 3]
```

Out[88]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensior
PostalCode							
00570	Kulosaari	0.456851	0.040272	0.152458	0.081067	0.049948	0.21940
00620	Metsälä-Etelä-Oulunkylä	0.469312	0.042467	0.132052	0.065716	0.022629	0.26782
00650	Veräjämäki	0.453395	0.066283	0.188493	0.082394	0.039586	0.16985
00730	Tapanila	0.489783	0.056701	0.184962	0.075630	0.034056	0.15886
00740	Siltamäki	0.442310	0.054406	0.178316	0.089305	0.031468	0.20419
00750	Puistola	0.455799	0.075261	0.175608	0.085686	0.042246	0.16540
00840	Laajasalo	0.425452	0.048854	0.153559	0.067551	0.028468	0.27611

In [89]:
```
# Examine cluster 4
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 4]
```

Out[89]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensio
PostalCode							
00150	Eira - Hernesaari	0.569631	0.057641	0.098075	0.060652	0.054414	0.15958
00180	Kamppi - Ruoholahti	0.554610	0.058448	0.115832	0.065212	0.050467	0.15543
00200	Lauttasaari	0.531131	0.046177	0.137323	0.061256	0.035587	0.18852
00220	Jätkäsaari	0.568197	0.048321	0.152502	0.120631	0.033585	0.07676
00240	Länsi-Pasila	0.482094	0.061892	0.111717	0.076294	0.036785	0.23121
00250	Taka-Töölö	0.583807	0.051449	0.090441	0.066522	0.036285	0.17149
00260	Keski-Töölö	0.504795	0.046846	0.088528	0.057174	0.050350	0.25230
00270	Pohjois-Meilahti	0.536107	0.062925	0.125850	0.072344	0.034668	0.16810
00280	Ruskeasuo	0.530667	0.049000	0.110333	0.104333	0.027333	0.17833
00290	Meilahden sairaala-alue	0.546584	0.049689	0.105590	0.024845	0.080745	0.19254
00300	Pikku Huopalahti	0.445450	0.069319	0.168516	0.080417	0.036537	0.19976
00370	Reimarla	0.449640	0.059952	0.170114	0.081385	0.045713	0.19319
00380	Pitäjänmäen teollisuusalue	0.479339	0.073691	0.150138	0.080119	0.039486	0.17722
00390	Konala	0.502897	0.067267	0.148214	0.067589	0.036852	0.17718
00410	Malminkartano	0.476130	0.075463	0.155758	0.109513	0.045669	0.13746
00520	Itä-Pasila	0.494909	0.075434	0.108848	0.074573	0.056647	0.18958
00560	Toukola-Vanhakaupunki	0.532949	0.049670	0.165997	0.092440	0.028589	0.13035
00610	Käpylä	0.475796	0.060237	0.156079	0.063041	0.033411	0.21143

In [90]:
```
# Examine cluster 5
helsinki_merged.loc[helsinki_merged['Cluster Labels'] == 5]
```

Out[90]:

	NeighbourhoodName	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pension
PostalCode							
00100	Helsinki Keskusta - Etu-Töölö	0.552440	0.048355	0.099228	0.070405	0.045165	0.18440
00120	Punavuori	0.546114	0.049815	0.113009	0.061771	0.051950	0.17734
00130	Kaartinkaupunki	0.561198	0.045573	0.110677	0.072266	0.046224	0.16406
00140	Kaivopuisto - Ullanlinna	0.525905	0.048996	0.114110	0.064987	0.054880	0.19112
00160	Katajanokka	0.461521	0.044519	0.138926	0.062416	0.044966	0.24765
00170	Kruununhaka	0.556894	0.041943	0.117440	0.068597	0.041402	0.17372
00190	Suomenlinna	0.477273	0.056818	0.236111	0.068182	0.046717	0.11489
00210	Vattuniemi	0.499156	0.037937	0.152007	0.045862	0.030401	0.23463
00330	Munkkiniemi	0.454535	0.039486	0.140569	0.069495	0.040050	0.25586
00540	Kalasatama	0.568480	0.041276	0.121951	0.107411	0.021576	0.13930
00580	Verkkosaari	0.534730	0.036016	0.158030	0.079383	0.042264	0.14957
00860	Santahamina	0.590588	0.018824	0.301176	0.054118	0.021176	0.01411
00990	Aurinkolahti	0.478641	0.056583	0.161004	0.053710	0.041844	0.20821

In [115]:
```
helsinki_data.corr()
```

Out[115]:

	EmployedR	UnemployedR	ChildR	StudentR	OthersR	Pensione
EmployedR	1.000000	-0.246359	-0.405733	-0.208461	-0.205343	-0.652993
UnemployedR	-0.246359	1.000000	-0.360091	0.114670	0.450477	0.113408
ChildR	-0.405733	-0.360091	1.000000	0.182867	-0.308608	-0.252621
StudentR	-0.208461	0.114670	0.182867	1.000000	-0.201543	-0.215605
OthersR	-0.205343	0.450477	-0.308608	-0.201543	1.000000	0.198215
PensionerR	-0.652993	0.113408	-0.252621	-0.215605	0.198215	1.000000
MedianHouseholdIncomeNorm	-0.021108	-0.732189	0.560558	-0.041624	-0.313530	-0.119505
AverageFloorSizeR	-0.389472	-0.565060	0.586388	0.046404	-0.145860	0.179074
DwellingsHouseR	-0.336947	-0.409933	0.594327	0.117908	-0.378389	0.079686
DwellingsApartmentR	0.336947	0.409933	-0.594327	-0.117908	0.378389	-0.079686
Latitude	-0.468928	0.233346	0.203626	0.312951	-0.197763	0.249139
Longitude	-0.395068	0.252056	0.328543	0.029919	0.114068	0.059042
ATM	0.017665	-0.017288	0.101494	-0.005605	-0.076859	-0.086889
American Restaurant	0.202948	-0.106874	-0.131940	-0.099488	0.117673	-0.084700
Antique Shop	0.153448	-0.101009	-0.122709	-0.092369	0.154835	-0.045319
Art Gallery	0.228835	-0.133253	-0.107195	0.201684	-0.170709	-0.157389
Art Museum	0.043027	-0.159701	-0.086162	-0.072610	0.016639	0.102733
Arts & Crafts Store	0.127396	-0.061068	-0.089582	-0.112288	0.114915	-0.043115
Asian Restaurant	0.211668	-0.130112	-0.151847	-0.206736	-0.038215	-0.001405
Auditorium	-0.037388	-0.094740	-0.020757	-0.107319	0.040072	0.123485
Auto Workshop	0.110303	-0.044929	0.050225	0.099108	-0.143728	-0.160976
Automotive Shop	0.043209	0.049898	0.003906	-0.067499	-0.046888	-0.043496
BBQ Joint	0.120956	-0.155679	0.018204	0.008622	0.025871	-0.113732
Badminton Court	-0.046485	-0.121744	0.015177	0.036244	0.093457	0.056553
Bagel Shop	0.127396	-0.061068	-0.089582	-0.112288	0.114915	-0.043115
Bakery	0.220141	0.000035	-0.178978	-0.180011	0.012536	-0.055451
Bar	0.173027	0.088419	-0.299359	-0.132167	-0.112475	0.089247
Basketball Court	0.046908	-0.079944	-0.154591	-0.147666	0.097773	0.134514
Bay	0.021177	0.110607	0.047699	-0.017285	-0.027880	-0.098056
Beach	-0.094736	-0.034608	0.233769	-0.031585	0.015672	-0.074197
Beer Bar	0.257955	0.243673	-0.356238	-0.236253	0.246512	-0.068497
Beer Garden	0.201908	0.080416	-0.046195	-0.157324	0.038749	-0.194490
Bike Shop	-0.190964	0.193284	0.102747	0.080793	0.139516	0.012860
Bistro	0.326888	-0.019328	-0.239689	0.194384	-0.106737	-0.212793