Capstone - Battle of the Neighborhoods - Part 2

Full Data Analysis Report

Asian Tappas Bar

Source:[https://www.needpix.com/photo/1136722/appetizers-food-instagram-food-tasty (https://www.needpix.com/photo/1136722/appetizers-food-instagram-food-tasty)](https://www.needpix.com/photo/1136722/appetizers-food-instagram-food-tasty)

Introduction

Background

A group of Swiss investors are interested to open a **Asian Tapas Bar** in Zürich (Switzerland). The venie shall leverage *modern food offering* and a *relaxed atmosphere* and is open from mid-morning to late evening. It is supposed to attract people for various occasions such spend time for a short break, taking lunch, go for after work drinks, take dinner or just meet and hang out with friends.

An experienced person have been appointed to manage all aspects of this project from planning to execution and finally shall run the bar. This person, currently living in London, will move to Zürich and needs some advise where to look for an appartement.

Assignments

1 Select Location for Venue

Zürich is in internatioal terms a small city and has a high density of restaurants and bars. Choosing the right location is paramount to make the success of this project. It is desired to have the venue centrally located with a good connectivity to public transport.

2 Select as good Living Area for the Venue Manager

The appointed person needs an appartement in Zürich. In an interview the following requirement have been captured: Residential area, well connected to the city, short commute time to the venuw and moderate comfort with respect to nearby groceries, gym amd recreation area.

Data section

Source:Wikipedia -- [https://pixabay.com/users/johnsongoh-3978075/ (https://pixabay.com/users/johnsongoh-3978075/)](https://pixabay.com/users/johnsongoh-3978075/)

Description of data required

To perform both the listed assignments different kind of datasets are requried:

Structural Data

Describing how is the city organized e.g. Districts, Neighborhoods, etc. and how well areas are connected with the public transport system of Zürich.

Demographics Data

Population and growth per area including some data on average income per area, nationalitities.

Density and Type of Venues

To explore venues across the city and its areas data is fetched from Foursquare

Usage of collected data

Once the data has been collected, cleansed and understood it is being used to build a picture about the city. With the data and insights gained, the following questions shall be answered:

- How is Zürich structured?
- Where people living?
- What kind of people is living where?
- How are areas connected to public transport?
- Where would be a good place to open a new bar?
- Where are assumed competitiors located?

The anserws for these questions build the foundation for a educated decision making.

Data Sources

Based on a first evaluation the following data source will likely provide all the required data sets (sorry some are in german only):

Open Data Portal of Zürich https://data.stadt-zuerich.ch/dataset (https://data.stadt-zuerich.ch/dataset)

Geo-Data Portal of Zürich https://www.ogd.stadt-zuerich.ch/geodaten/ (https://www.ogd.stadt-zuerich.ch/geodaten/)

Wikipedia https://de.wikipedia.org/wiki/Stadtteile_der_Stadt_Z%C3%BCrich (https://de.wikipedia.org/wiki/Stadtteile_der_Stadt_Z%C3%BCrich)

Data Inventory

Districts (German: Stadtkreise)

List and coordinates of the districts of Zürich. This information is constantly updated while quite static anyway:

- https://data.stadt-zuerich.ch/dataset/geo_stadtkreise (https://data.stadt-zuerich.ch/dataset/geo_stadtkreise)
- https://www.ogd.stadt-zuerich.ch/geodaten/Stadtkreise?format=10009 (https://www.ogd.stadt-zuerich.ch/geodaten/Stadtkreise?format=10009)

The data must be downloaded manually. From the package the file *stzh.adm_stadtkreise_beschr_p.json* is being used as it contains standardized coordinate information for the districts and the shapes.

Neighborhood (German: Statistische Quartiere)

List and coordinates of the neighberhoods of Zürich. This information is constantly updated while quite static anyway:

- https://data.stadt-zuerich.ch/dataset/geo_statistische_quartiere (https://data.stadt-zuerich.ch/dataset/geo_statistische_quartiere)
- https://www.ogd.stadt-zuerich.ch/geodaten/Statistische_Quartiere?format=10009 (https://www.ogd.stadt-zuerich.ch/geodaten/Statistische_Quartiere?format=10009)

The data must be downloaded manually. From the package two files are being used

1. *stzh.adm_statistische_quartiere_b_p.json* contains standardized information
2. *stzh.adm_statistische_quartiere_a.json* contains neighborhood shapes.

Methodology

Source: [http://www.thebluediamondgallery.com/typewriter/m/methodology.html (http://www.thebluediamondgallery.com/typewriter/m/methodology.html)](http://www.thebluediamondgallery.com/typewriter/m/methodology.html)

Overview

The methods chosen have to address the two aspects of the task:

1. Selecting a living area
2. Select Venue location

Technically the methods are accumulative, meaning that the selection of the living area will feed into the selection of the venue location, as it delivers general informatiuon about the city of Zürich.

Selection of Living Area

General approach is to map collected data on structures, population, etc. onto the city and pick the right fight for the area which fulfills the requirements.

- Collect and inspect structural and demographical information
- Explore and understand Data
- Prepare, process data
- Visualize data for decision making

Selection of Venue Location

Complement the collected and visualized information with information about the venues across the city and analyze areas where similar venues are located, denisty of venues, etc. to identfy a good location for out "**Asian Tappas Bar**".

- Collect, inspect and combine venue data with structural data
- Explore and understand Data
- Prepare data for processing
- Build a model to find the right location (using Elbow Method to optimize clustering)
- Consoildate and visualize data for decision making

On the modeling side the selection of the venue location goes further than the selection of the living area as clustering algorithm K-means will be being used review the city from a "venue-perspective". To find the optimal number of clusters the "Elbow Method" has been applied This insight should help to find the best place for the Asian Tappas Bar.

Result

Result on "Where to life in Zürich?"

From all the analysis and balancing out the needs **Alt-Wiedikon** seems to be the area that fits the best and is the the environment to live in. It excels with the following characteristics:

- It is close to the center of Zürich and the lake
- Has a average population (18k), which is moderately growing (7%)
- It is a multi cultural environment hosting 114 nationalities
- Living costs seem to be moderate because also people with normal incomes seem to live there (76k)

Results on "Where in Zürich to open the Asian Tapas Bar?"

Based on the analysis **Zürich City** is the area to open our **Asian Tapas Bar**. The rational for this selection comes from comparing the four clusters build along the venues categories in Zürich. Following how these clusters compare:

- **Cluster 0** Residential area (also including the above mentioned **Alt-Wiedikon**) - good place to live but not suitable to open an Asian Tappas Bar.
- **Cluster 1** Residential area like Cluster 0 and also a good place to live but not suitable to open an Asian Tappas Bar.
- **Cluster 2** Selected area - It is a quite busy area, where people hang out
- **Cluster 3** Similar to Cluster 2 - It is a quite busy area, where people hang out

What makes **Cluster 2** standing out is the fact, that Asian Food offerings seem not to be a common offering as it is in Cluter 3. Given this the area gives the popularity and high people frequency and the time offers out Asian Tappas Bar the opportunity to be special/unique.

Conclusions

Source: http://thebluediamondgallery.com/hand-held-card/c/conclusions.html (http://thebluediamondgallery.com/hand-held-card/c/conclusions.html)

Learning and Conclusion on Data availability

During the evaluation of the project content it became obvious how many cities and countries open themself and offer open data platforms. Moreover multiple city across countries are using common data platforms. This was rather surprising than expected. To a certain extend the information fetched from Foursquare is redundant to what the cities are offering themselves

To address some data access issues I contacted the listed points and was again surprised, that I received immediate support by the staff from the government. **Big thanks!**

Given this I conclude that there can be much more done in this area and the tool set from the training is extremely helpful and actually quite sufficient to start this data science journey.

Conclusion on Results

The results are arbitrary at first and judgment if they are valid by any means is difficult if you have no relation to the topic or the city.

I am not living in Zürich, but I have a friend living there and so I used the opportunity to present this outcome and here the feedback:

Place of Living

Rational and reasoning why I selected the living was confirmed. To the surprise of all it actually was the area the friend is living in. Also confirmed were the more detailed information, which has not been presented in this report, e.g. changes in the population.

Place for the "Asian Tappas Bar"

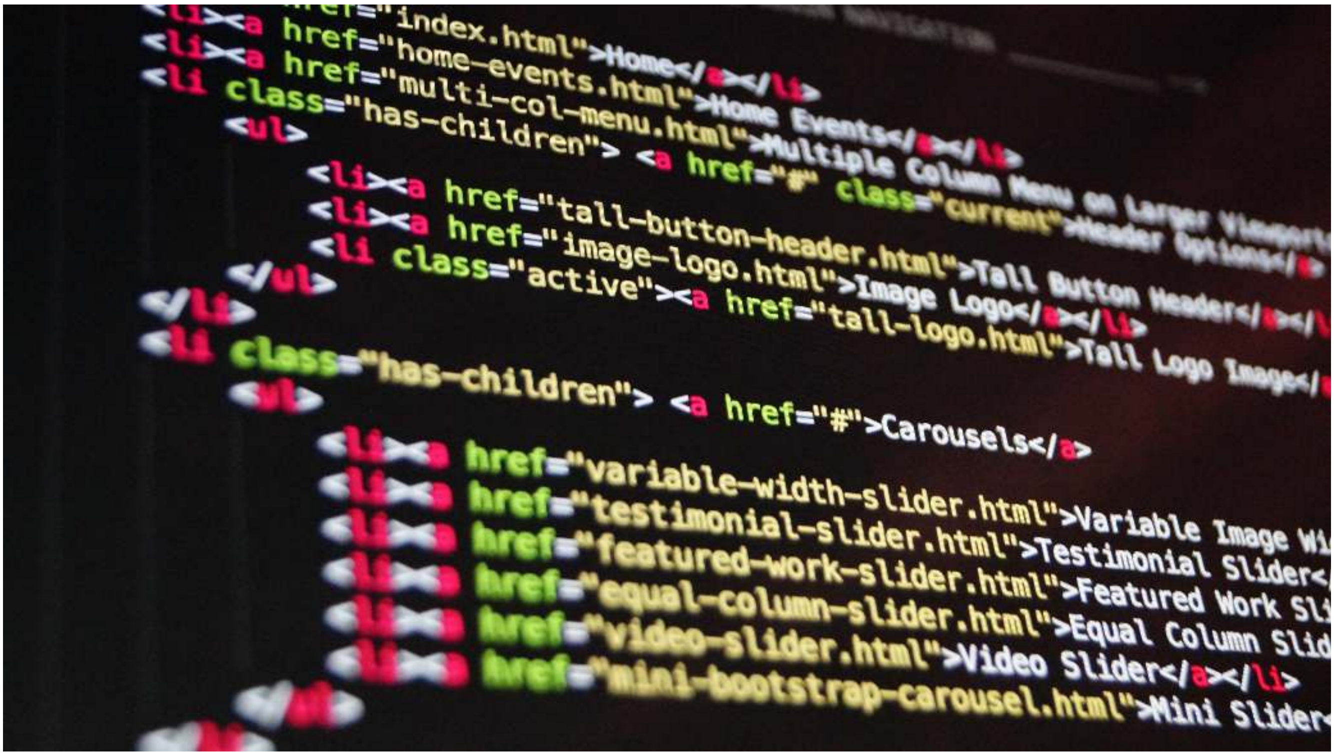
Feedback here is, Zürich City fits! it is actually a "no brainer".

However...

a significant question is about cost and availability of facilities in this area. Availability of data about costs of business facility is an issue and would require far more investigation - issue spotted. Another comment was related to the detailed analysis and the identified top tier areas. Here it seems that the analysis misses an aspect: Some of the areas are very close to the university and other schools of Zürich. These areas one offer well priced food places and are frequented during daytime, but in the evening these areas are not very active - I didn't spot this.

Data Analysis & Code

From here onwards the data collection, preparation and processing can be found.



Source: Wikipedia - [https://pixabay.com/en/users/JOSBORNE_-1640589/ (https://pixabay.com/en/users/JOSBORNE_-1640589/)](https://pixabay.com/en/users/JOSBORNE_-1640589/)

Import used libraries

In [30]:
```
import numpy as np

import pandas as pd
from pandas.io.json import json_normalize

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

# library to handle requests
import requests

# library to process xml
import xml

# library to handle JSON files
import json

# tranform JSON file into a pandas dataframe
from pandas.io.json import json_normalize

# for webscraping import Beautiful Soup
from bs4 import BeautifulSoup

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

# Matplotlib and associated plotting modules
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
```

Setup global variables and application switches

In [31]:
```
# Switch to bypass Foursquare API and load data from file
FOURSQUARE_ONLINE = False

# your Foursquare ID
CLIENT_ID = '...'
# your Foursquare Secret
CLIENT_SECRET = '...'
# Foursquare API version
VERSION = '20180605'
```

Collect data files

In [32]:
```
DATA_PATH ='./data/'
#!wget -q -O './data/stops.txt' https://data.stadt-zuerich.ch/dataset/ec7bb57c-f0aa-4e8e-9266-f0b7112f6355/resource/c2054388-eba0-416f-9601-b2824226c24c/download/stops.txt
#print('Data downloaded!')

URL_NEIGHBORHOODS_BUILDINGS  = 'https://data.stadt-zuerich.ch/dataset/bf46df6f-4a63-4bb2-b434-df7e3e1fdf09/resource/3850add1-264c-4993-98cd-d8a9ba87ee25/download/bau_best_geb_whg_bev_gebaeudeart_quartier_seit2008.csv'
URL_NEIGHBORHOODS_POPULATION = 'https://data.stadt-zuerich.ch/dataset/24c0fa4e-4e27-4bec-9e49-cf2e9ace0707/resource/d1f89135-caee-43cd-b94f-34ad841968a2/download/bev336od3363.csv'
URL_NEIGHBORHOODS_INCOME = 'https://data.stadt-zuerich.ch/dataset/15deecbb-b7f7-4ee8-b312-eabb649c55d7/resource/8c68847c-d433-4802-817f-9abea76ace7a/download/wir100od1003.csv'
URL_ZH_PUBLIC_TRANSPORT = 'https://data.stadt-zuerich.ch/dataset/ec7bb57c-f0aa-4e8e-9266-f0b7112f6355/resource/c2054388-eba0-416f-9601-b2824226c24c/download/stops.txt'
```

Data Analysis on "Where to live in Zürich?"

Source: Wikipedia https://commons.wikimedia.org/wiki/File:Altstadt_Z%C3%BCrich_2015.jpg (https://commons.wikimedia.org/wiki/File:Altstadt_Z%C3%BCrich_2015.jpg)

Collect and prepare geographical data about Zürich

Administrativ and Geographical Structure of Zürich

Districts (Stadtkreise)

Load raw JSON file and extract **districts** and their coordinates

In [33]:
```
# reading the JSON data using json.load()
district_file = DATA_PATH + 'stzh.adm_stadtkreise_beschr_p.json'

with open(district_file) as datafile:
    d = json.load(datafile)

# process JSON data
df_district_raw = json_normalize(data = d['features'])

print('Shape: {}'.format(df_district_raw.shape))
df_district_raw.head()
```

```
Shape: (12, 9)
```

Out[33]:

	type	geometry.type	geometry.coordinates	properties.objid	properties.bezeichnung	properties.name	properties.or
0	Feature	Point	[8.5319800266, 47.3458196873]	1	Kreis 2	2	(
1	Feature	Point	[8.5578558879, 47.3521329109]	2	Kreis 8	8	(
2	Feature	Point	[8.5254433999, 47.4212437242]	3	Kreis 11	11	(
3	Feature	Point	[8.5743532965, 47.4028865247]	4	Kreis 12	12	(
4	Feature	Point	[8.4995069785, 47.4064859987]	5	Kreis 10	10	(

Extract district coordiantes

In [34]:
```
# Define what information shall be extracted
filtered_columns = ['properties.bezeichnung', 'properties.name', 'geometry.coordinates']
df_district = df_district_raw.loc[:,filtered_columns]

# Adjust column naming
df_district.rename(columns={'properties.bezeichnung': 'District'}, inplace=True)
df_district.rename(columns={'properties.name': 'DistrictID'}, inplace=True)
df_district.rename(columns={'geometry.coordinates': 'Coordinates'}, inplace=True)

# Extract the data array with the coordinates
df_district[['Longitude','Latitude']] = pd.DataFrame(df_district.Coordinates.values.toli
st(), index = df_district.index)

# The coordinates array is not longer used
df_district.drop('Coordinates', axis=1, inplace=True)

# Convert District into string
df_district['District']   = df_district['District'].astype(str)
df_district['DistrictID'] = df_district['DistrictID'].astype(int)
df_district.reset_index(drop=True)

# Reorder List
df_district.sort_values(by = ['DistrictID'], ascending=True, inplace=True)
df_district.reset_index(drop=True, inplace=True)

print('Shape: {}'.format(df_district.shape))
df_district
```

```
Shape: (12, 4)
```

Out[34]:

	District	DistrictID	Longitude	Latitude
0	Kreis 1	1	8.541120	47.372853
1	Kreis 2	2	8.531980	47.345820
2	Kreis 3	3	8.506778	47.362094
3	Kreis 4	4	8.518935	47.379624
4	Kreis 5	5	8.519968	47.388217
5	Kreis 6	6	8.546674	47.392192
6	Kreis 7	7	8.577787	47.371184
7	Kreis 8	8	8.557856	47.352133
8	Kreis 9	9	8.482312	47.383897
9	Kreis 10	10	8.499507	47.406486
10	Kreis 11	11	8.525443	47.421244
11	Kreis 12	12	8.574353	47.402887

Neighborhood (Stadtquartiere)

Load JSON files with **neighborhood** information

In [35]:
```
# reading the JSON data using json.load()
neighborhood_file = DATA_PATH + 'stzh.adm_statistische_quartiere_b_p.json'

with open(neighborhood_file) as datafile:
    d = json.load(datafile)

df_neighborhood_raw = json_normalize(data = d['features'])

print('Shape: {}'.format(df_neighborhood_raw.shape))
df_neighborhood_raw.head()
```

```
Shape: (34, 9)
```

Out[35]:

	type	geometry.type	geometry.coordinates	properties.objid	properties.name	properties.kuerzel	properties.ori	pro
0	Feature	Point	[8.5064997018, 47.4231890593]	1	Affoltern	111	0	
1	Feature	Point	[8.539599435, 47.4238990203]	2	Seebach	119	0	
2	Feature	Point	[8.5644810557, 47.4118491554]	3	Saatlen	121	0	
3	Feature	Point	[8.4956736729, 47.4081249683]	4	Höngg	101	0	
4	Feature	Point	[8.5233724127, 47.3972245722]	5	Wipkingen	102	0	

Extract neighborhood coordinates

In [36]:
```
filtered_columns = ['properties.name','geometry.coordinates']

df_neighborhood = df_neighborhood_raw.loc[:,filtered_columns]

df_neighborhood.rename(columns={'properties.name': 'Neighborhood'}, inplace=True)
df_neighborhood.rename(columns={'geometry.coordinates': 'Coordinates'}, inplace=True)

df_neighborhood[['Longitude','Latitude']] = pd.DataFrame(df_neighborhood.Coordinates.val
ues.tolist(), index = df_neighborhood.index)

df_neighborhood.drop('Coordinates', axis=1, inplace=True)
df_neighborhood['Neighborhood'] = df_neighborhood['Neighborhood'].astype(str)

print('Shape: {}'.format(df_neighborhood.shape))
df_neighborhood.head()
```

```
Shape: (34, 3)
```

Out[36]:

	Neighborhood	Longitude	Latitude
0	Affoltern	8.506500	47.423189
1	Seebach	8.539599	47.423899
2	Saatlen	8.564481	47.411849
3	Höngg	8.495674	47.408125
4	Wipkingen	8.523372	47.397225

Districts vs Neighborhood

Load matching table to link district and neighborhoods

In [37]:
```
df_district_neighborhood_raw = pd.read_csv(DATA_PATH + 'zrh_district_neighborhood.csv')
df_district_neighborhood = df_district_neighborhood_raw[['District', 'Neighborhood']]

print('Shape: {}'.format(df_district_neighborhood.shape))
df_district_neighborhood.head()
```

```
Shape: (34, 2)
```

Out[37]:

	District	Neighborhood
0	Kreis 1	Rathaus
1	Kreis 1	Hochschulen
2	Kreis 1	Lindenhof
3	Kreis 1	City
4	Kreis 2	Wollishofen

Link Districts and Neighborhood and retain Neighborhood coordinates

In [38]:
```
df_district_neighborhood = df_district_neighborhood.join(df_neighborhood.set_index('Neig
hborhood'), on='Neighborhood')

print('Shape: {}'.format(df_district_neighborhood.shape))
df_district_neighborhood.head()
```

```
Shape: (34, 4)
```

Out[38]:

	District	Neighborhood	Longitude	Latitude
0	Kreis 1	Rathaus	8.544455	47.371933
1	Kreis 1	Hochschulen	8.544603	47.365484
2	Kreis 1	Lindenhof	8.539873	47.373063
3	Kreis 1	City	8.534951	47.371386
4	Kreis 2	Wollishofen	8.532078	47.339917

Add a numerical representation of Districts

In [39]:
```
df_district_neighborhood['DistrictNb'] = pd.to_numeric(df_district_neighborhood['Distric
t'].str[-2:])
df_district_neighborhood.dropna(inplace=True)
df_district_neighborhood.head()
```

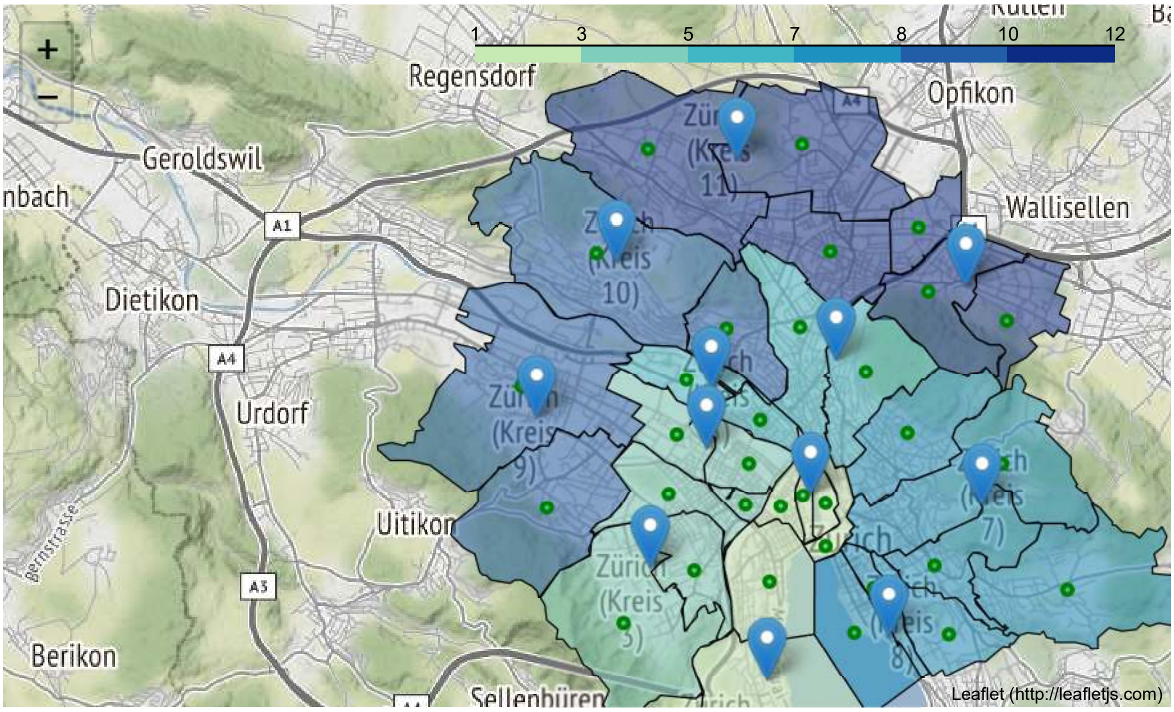
Out[39]:

	District	Neighborhood	Longitude	Latitude	DistrictNb
0	Kreis 1	Rathaus	8.544455	47.371933	1
1	Kreis 1	Hochschulen	8.544603	47.365484	1
2	Kreis 1	Lindenhof	8.539873	47.373063	1
3	Kreis 1	City	8.534951	47.371386	1
4	Kreis 2	Wollishofen	8.532078	47.339917	2

Visualization Districts and Neighborhoods

In [40]:
```
# Color Palettes
# 'BuGn', 'BuPu', 'GnBu', 'OrRd', 'PuBu', 'PuBuGn', 'PuRd', 'RdPu', 'YlGn', 'YlGnBu', 'Y
lOrBr', and 'YlOrRd'.

# evaluate map center
latitude = df_district_neighborhood['Latitude'].median()
longitude = df_district_neighborhood['Longitude'].median()

# build map
map = folium.Map(location=[latitude, longitude],
                 tiles='Stamen Terrain',
                 zoom_start=12)

# draw Neighborhood markers on map
zh_geo = DATA_PATH + 'stzh.adm_statistische_quartiere_a.json'

## add chloropleth layer

map.choropleth(
    geo_data=zh_geo,
    data=df_district_neighborhood,
    columns=['Neighborhood', 'DistrictNb'],
    key_on='feature.properties.name',
    fill_color='YlGnBu',
    fill_opacity=0.5,
    line_opacity=1
    )

# New way to setup Choropleth
'''
folium.Choropleth(
    geo_data=zh_geo,
    name='choropleth',
    data=df_district_neighborhood,
    columns=['Neighborhood', 'DistrictNb'],
    key_on='feature.properties.name',
    fill_color='YlGnBu',
    fill_opacity=0.5,
    line_opacity=1
).add_to(map)
'''
fg = folium.FeatureGroup(name='Neighborhood')
for lat, lon, name in zip(df_district['Latitude'].tolist(),
                          df_district['Longitude'].tolist(),
                          df_district['District'].tolist()):

    marker_text = '{}'.format(name)
    fg.add_child(folium.Marker(location=[lat, lon], popup=marker_text))

map.add_child(fg)

# draw neighborhood markers on map
for lat, lng, neighborhood, district in zip(df_district_neighborhood['Latitude'],
                                            df_district_neighborhood['Longitude'],
                                            df_district_neighborhood['Neighborhood'],
                                            df_district_neighborhood['District']):

    label = '{} {}'.format(neighborhood, district)

    label = folium.Popup(label, parse_html=True)

    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='green',
        fill=True,
        fill_color='#31cc77',
        fill_opacity=0.7,
        parse_html=False).add_to(map)

map
```

Out[40]:



Prepare facts and figures about Neighborhoods of Zürich

Neighborhood Population incl. Nationalities 2018 and 2013

Extract information about population and nationalities per neighborhood and compare numbers 2018 and 2013

In [41]:
```
# Load data file and rename columns
df_neighborhood_population_raw = pd.read_csv(URL_NEIGHBORHOODS_POPULATION)

df_neighborhood_population_raw.columns = ['Year','Sort','Code','Neighborhood','Populatio
n','Nationalities']

# Select data for 2018 and drop unused columns
df_neighborhood_population = df_neighborhood_population_raw[df_neighborhood_population_r
aw.Year == 2018]
df_neighborhood_population.loc['Population'] = pd.to_numeric(df_neighborhood_population
['Population'])
df_neighborhood_population.drop('Year', axis=1, inplace=True)
df_neighborhood_population.drop('Code', axis=1, inplace=True)
df_neighborhood_population.drop('Sort', axis=1, inplace=True)

# Select data for 2013 and drop unused columns
df_neighborhood_population_hist = df_neighborhood_population_raw[df_neighborhood_populat
ion_raw.Year == 2013]
df_neighborhood_population_hist.rename(columns = {'Population':'Population_Past'}, inpla
ce=True)
df_neighborhood_population_hist.drop('Year', axis=1, inplace=True)
df_neighborhood_population_hist.drop('Code', axis=1, inplace=True)
df_neighborhood_population_hist.drop('Sort', axis=1, inplace=True)
df_neighborhood_population_hist.drop('Nationalities', axis=1, inplace=True)

# Rename columns to avoid overlap with 2018 data
df_neighborhood_population_hist.loc['Population_Past'] = pd.to_numeric(df_neighborhood_p
opulation_hist['Population_Past'])

# Compute changes in population and number of nationalities
df_neighborhood_population = df_neighborhood_population.join(df_neighborhood_population_
hist.set_index('Neighborhood'), on='Neighborhood')
df_neighborhood_population['Population_Growth'] = 100*(df_neighborhood_population['Popul
ation'] - df_neighborhood_population['Population_Past'])/df_neighborhood_population['Pop
ulation_Past']

# Drop NaN, sort and reindex table
df_neighborhood_population.dropna(inplace=True)
df_neighborhood_population.set_index('Neighborhood', inplace=True)
df_neighborhood_population.sort_values('Population_Growth', ascending=False, inplace=Tru
e)
df_neighborhood_population.reset_index(inplace=True)

print('Shape: {}'.format(df_neighborhood_population.shape))
df_neighborhood_population.head()
```

```
Shape: (34, 5)
```

Out[41]:

	Neighborhood	Population	Nationalities	Population_Past	Population_Growth
0	Escher Wyss	6066.0	96.0	4010.0	51.271820
1	Wollishofen	18923.0	113.0	15937.0	18.736274
2	Saatlen	8582.0	102.0	7280.0	17.884615
3	Albisrieden	22304.0	117.0	19146.0	16.494307
4	Hirzenbach	12801.0	116.0	11153.0	14.776293

Evaluate Housing situation of the Neighborhoods in 2017

Extract information about housing from building information per neighborhoods in 2017

In [42]:
```
# Load data file and rename columns
df_neighborhood_buildings_raw = pd.read_csv(URL_NEIGHBORHOODS_BUILDINGS)
df_neighborhood_buildings_raw.columns = ['Year',
                                          'Sort',
                                          'Neighborhood',
                                          'Building_Type_Code',
                                          'Building_Type',
                                          'Building_Count',
                                          'Volume',
                                          'Persons',
                                          'Appartements',
                                          'Appartement_Area',
                                          'Appartement_Persons',
                                          'District_Code',
                                          'District',
                                          'Building_Type_2',
                                          'Building_Type_3',
                                          'Building_Type_Pub',
                                          'Building_Class']

# Select data for 2017, drop unusded colums and remove unusable data entries (rows)
df_neighborhood_buildings = df_neighborhood_buildings_raw[['Year','Neighborhood','Buildi
ng_Class', 'Appartements']]
df_neighborhood_buildings = df_neighborhood_buildings[df_neighborhood_buildings_raw.Year
== 2017]
df_neighborhood_buildings = df_neighborhood_buildings[df_neighborhood_buildings.Neighbor
hood != 'Unbekannt']
df_neighborhood_buildings.drop('Year', axis=1, inplace=True)

print('Shape: {}'.format(df_neighborhood_buildings.shape))
df_neighborhood_buildings.head()
```

```
Shape: (1146, 3)
```

Out[42]:

	Neighborhood	Building_Class	Appartements
9	Rathaus	Einfamilienhäuser	1
19	Rathaus	Einfamilienhäuser	24
29	Rathaus	Mehrfamilienhäuser	10
39	Rathaus	Mehrfamilienhäuser	164
49	Rathaus	Mehrfamilienhäuser	1920

Determine relevant building classes based on the variations

In [43]:
```
df_neighborhood_buildings.Building_Class.unique()
```

Out[43]:
```
array(['Einfamilienhäuser', 'Mehrfamilienhäuser', 'Spezielle Wohngebäude',
       'Kommerzielle Gebäude', 'Kleingebäude', 'Infrastrukturgebäude',
       'Produktions- und Lagergebäude'], dtype=object)
```

Select and consolidate appartements in Einfamilienhäuser and Mehrfamilienhäuser per Neighborhood

In [44]:
```
# Select Building Classes
df_neighborhood_housing = df_neighborhood_buildings[(df_neighborhood_buildings.Building_
Class == "Mehrfamilienhäuser") |
                                                    (df_neighborhood_buildings.Building_
Class == "Einfamilienhäuser")]

# After selection drop Building Classes columns
df_neighborhood_housing.drop('Building_Class', axis=1, inplace=True)

# Consolidate number of appartements
df_neighborhood_housing_grouped = df_neighborhood_housing.groupby('Neighborhood').sum().
reset_index()

print('Shape: {}'.format(df_neighborhood_housing_grouped.shape))
df_neighborhood_housing_grouped.head()
```

```
Shape: (34, 2)
```

Out[44]:

	Neighborhood	Appartements
0	Affoltern	11578
1	Albisrieden	11184
2	Alt-Wiedikon	9454
3	Altstetten	16381
4	City	314

Evaluate Income situation across Neighborhoods in 2015

Extract information about the tax income 2015 within neighborhoods (75 percentile)

In [45]:
```
df_neighborhood_income_raw = pd.read_csv(URL_NEIGHBORHOODS_INCOME)

df_neighborhood_income_raw.columns = ['Year',
                                      'Sort',
                                      'Neighborhood',
                                      'Tax_Tarif_Code',
                                      'Tax_Tarif',
                                      'Tax_p50_kCHF',
                                      'Tax_p25_kCHF',
                                      'Tax_p75_kCHF']

df_neighborhood_income_raw = df_neighborhood_income_raw[df_neighborhood_income_raw.Year
== 2015]
df_neighborhood_income_raw.drop('Year', axis=1, inplace=True)
df_neighborhood_income_raw.drop('Sort', axis=1, inplace=True)
df_neighborhood_income_raw.drop('Tax_Tarif_Code', axis=1, inplace=True)
df_neighborhood_income_raw.drop('Tax_p50_kCHF', axis=1, inplace=True)
df_neighborhood_income_raw.drop('Tax_p25_kCHF', axis=1, inplace=True)

print('Shape: {}'.format(df_neighborhood_income_raw.shape))
df_neighborhood_income_raw.head()
```

```
Shape: (102, 3)
```

Out[45]:

	Neighborhood	Tax_Tarif	Tax_p75_kCHF
1632	Rathaus	Grundtarif	84.60
1633	Rathaus	Verheiratetentarif	190.30
1634	Rathaus	Einelternfamilientarif	91.75
1635	Hochschulen	Grundtarif	70.40
1636	Hochschulen	Verheiratetentarif	225.40

Segregate the different tax tarifs

In [46]:
```
tax_tarif = df_neighborhood_income_raw.Tax_Tarif.unique()

df_neighborhood_income = []
for n in range(len(tax_tarif)):
    df_neighborhood_income.append(df_neighborhood_income_raw[df_neighborhood_income_raw.
Tax_Tarif == tax_tarif[n]])
    df_neighborhood_income[n].drop('Tax_Tarif', axis=1, inplace=True)
    df_neighborhood_income[n].rename(columns = {'Tax_p75_kCHF':tax_tarif[n]}, inplace=Tr
ue)
    df_neighborhood_income[n].fillna(df_neighborhood_income[n].mean(), inplace=True)

df_neighborhood_income[0].head()
```

Out[46]:

	Neighborhood	Grundtarif
1632	Rathaus	84.6
1635	Hochschulen	70.4
1638	Lindenhof	99.1
1641	City	70.1
1644	Wollishofen	67.3

Prepare information on Public Transport

In [47]:
```
df_public_transport = pd.read_csv(URL_ZH_PUBLIC_TRANSPORT)
df_public_transport.drop('stop_id', axis=1, inplace=True)
df_public_transport.drop('stop_url', axis=1, inplace=True)
df_public_transport.drop('location_type', axis=1, inplace=True)
df_public_transport.drop('parent_station', axis=1, inplace=True)
df_public_transport.drop_duplicates(['stop_name'], keep='last', inplace=True)
df_public_transport.rename(columns={'stop_lat':'Latitude','stop_lon':'Longitude'}, inpla
ce=True)

df_public_transport.head()
```

Out[47]:

	stop_name	Latitude	Longitude
13	Oberrieden, Tannenbach	47.269117	8.582621
42	Islisberg, Dorf	47.323023	8.439016
43	Kindhausen AG	47.394117	8.376035
45	Oberlunkhofen, Oberdorf	47.312585	8.392555
49	Jonen, Taverne	47.296333	8.395583

Consolidate all collected fact and figures of Zürich

Consolidate all information collected into a single dataframe for further processing

In [48]:
```
df_neighborhood = df_district_neighborhood.copy()

# Complement population information
df_neighborhood = df_neighborhood.join(df_neighborhood_population.set_index('Neighborhoo
d'), on='Neighborhood')

# Complement housing information
df_neighborhood = df_neighborhood.join(df_neighborhood_housing_grouped.set_index('Neighb
orhood'), on='Neighborhood')

for n in range(len(df_neighborhood_income)):
    df_neighborhood = df_neighborhood.join(df_neighborhood_income[n].set_index('Neighbor
hood'), on='Neighborhood')

print('Shape: {}'.format(df_neighborhood.shape))
df_neighborhood[['Neighborhood', 'Population','Nationalities','Population_Growth','Appar
tements','Grundtarif']]
```

```
Shape: (34, 13)
```

Out[48]:

	Neighborhood	Population	Nationalities	Population_Growth	Appartements	Grundtarif
0	Rathaus	3267.0	79.0	2.285535	2119	84.60
1	Hochschulen	664.0	44.0	-0.150376	258	70.40
2	Lindenhof	990.0	47.0	7.258938	692	99.10
3	City	829.0	53.0	5.874840	314	70.10
4	Wollishofen	18923.0	113.0	18.736274	9017	67.30
5	Leimbach	6320.0	101.0	10.296684	2628	59.70
6	Enge	9634.0	97.0	9.031236	4962	97.10
7	Alt-Wiedikon	17956.0	114.0	7.482342	9454	76.50
8	Friesenberg	10933.0	97.0	2.215782	4316	52.50
9	Sihlfeld	21680.0	119.0	3.578424	11899	65.00
10	Werd	4455.0	88.0	6.375358	2260	72.50
11	Langstrasse	11111.0	114.0	3.977166	6515	68.90
12	Hard	13163.0	116.0	-0.589079	6771	55.00
13	Gewerbeschule	9513.0	107.0	-1.173904	4939	67.70
14	Escher Wyss	6066.0	96.0	51.271820	3067	94.60
15	Unterstrass	23394.0	125.0	6.939111	11482	71.50
16	Oberstrass	10927.0	102.0	4.544585	5402	82.00
17	Fluntern	8485.0	93.0	8.006619	3978	96.70
18	Hottingen	11265.0	103.0	5.201718	5693	87.70
19	Hirslanden	7488.0	90.0	2.786548	3980	80.40
20	Witikon	10953.0	100.0	6.681601	5384	70.10
21	Seefeld	5253.0	83.0	5.524307	3437	100.25
22	Mühlebach	6315.0	90.0	6.816644	3603	88.75
23	Weinegg	5220.0	89.0	3.942652	2453	73.90
24	Albisrieden	22304.0	117.0	16.494307	11184	62.70
25	Altstetten	33461.0	124.0	7.539772	16381	60.50
26	Höngg	24020.0	119.0	11.301608	11520	71.80
27	Wipkingen	16321.0	116.0	3.069151	8912	67.80
28	Affoltern	26562.0	122.0	5.900646	11578	57.80
29	Oerlikon	23214.0	121.0	7.184412	11495	69.10
30	Seebach	25568.0	128.0	6.497834	12026	59.50
31	Saatlen	8582.0	102.0	17.884615	3382	51.10
32	Schwamendingen-Mitte	11100.0	113.0	-0.972433	5807	52.00
33	Hirzenbach	12801.0	116.0	14.776293	5732	48.20

Visualize facts and figures of Zürich across neighborhoods

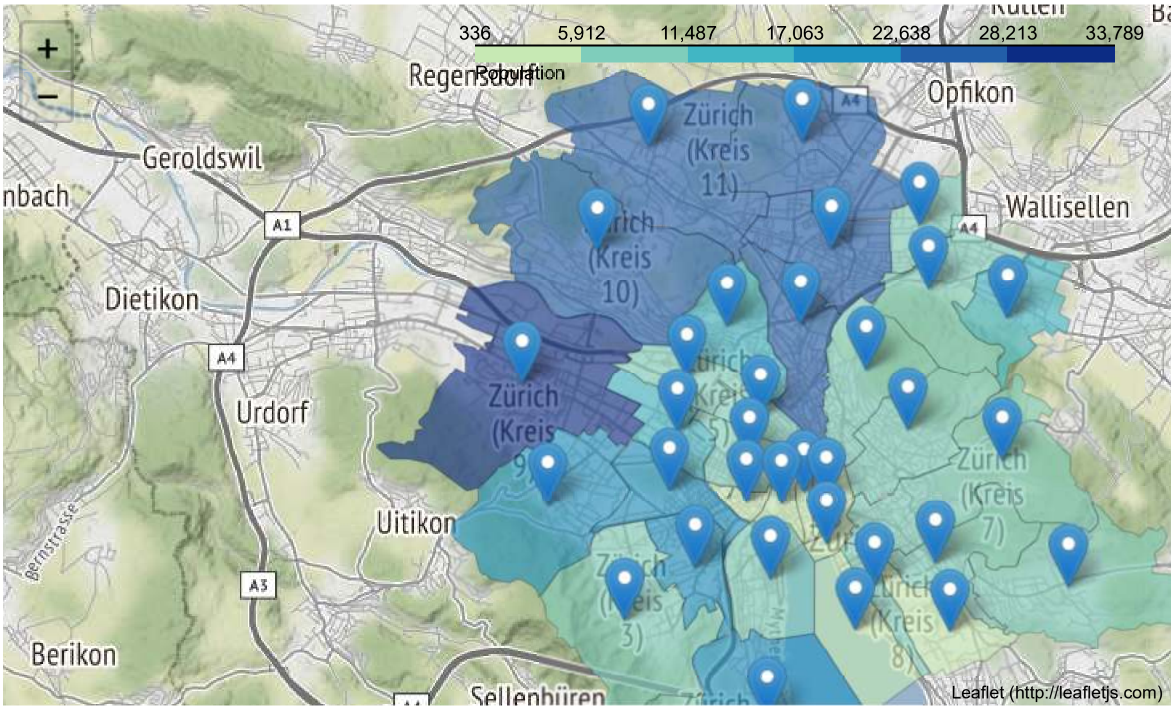
Build function to visualize facts and figures in a common and efficient way

In [49]:
```
def show_neighborhood_data(data_column, data_unit, legend_title, color_palette='YlGnBu
'):

        # Color Palettes
        # 'BuGn', 'BuPu', 'GnBu', 'OrRd', 'PuBu', 'PuBuGn', 'PuRd', 'RdPu', 'YlGn', 'YlG
nBu', 'YlOrBr', and 'YlOrRd'.

        # evaluate map center
        latitude = df_neighborhood['Latitude'].median()
        longitude = df_neighborhood['Longitude'].median()

        # build map
        map = folium.Map(location=[latitude, longitude],
                         tiles='Stamen Terrain',
                         zoom_start=12)

        # draw Neighborhood markers on map
        zh_geo = DATA_PATH+'stzh.adm_statistische_quartiere_a.json'

        # add chloropleth layer
        map.choropleth(
            geo_data=zh_geo,
            name='choropleth',
            data = df_neighborhood,
            columns = ['Neighborhood', data_column],
            key_on = 'feature.properties.name',
            fill_color = color_palette,
            fill_opacity = 0.6,
            line_opacity = 0.2,
            legend_name = legend_title
        )

        # New way to setup Choropleth
        '''
        folium.Choropleth(
            geo_data=zh_geo,
            name='choropleth',
            data = df_neighborhood,
            columns = ['Neighborhood', data_column],
            key_on = 'feature.properties.name',
            fill_color = color_palette,
            fill_opacity = 0.6,
            line_opacity = 0.2,
            legend_name = legend_title
        ).add_to(map)
        '''

        # add markers with basic information
        fg = folium.FeatureGroup(name='Neighborhood Info')
        for lat, lon, val, name in zip(df_neighborhood['Latitude'].tolist(),
                                       df_neighborhood['Longitude'].tolist(),
                                       df_neighborhood[data_column].tolist(),
                                       df_neighborhood['Neighborhood'].tolist()):

            marker_text = '{}<br/>{}: {:,.0f} {}'.format(name, legend_title, round(val,
0), data_unit)
            fg.add_child(folium.Marker(location=[lat, lon], popup = marker_text))

        map.add_child(fg)

        # Not yet required
        # folium.LayerControl().add_to(map)

        return map
```

Population across Neighborhoods

In [50]:
```
show_neighborhood_data('Population', '', 'Population', 'YlGnBu')
```

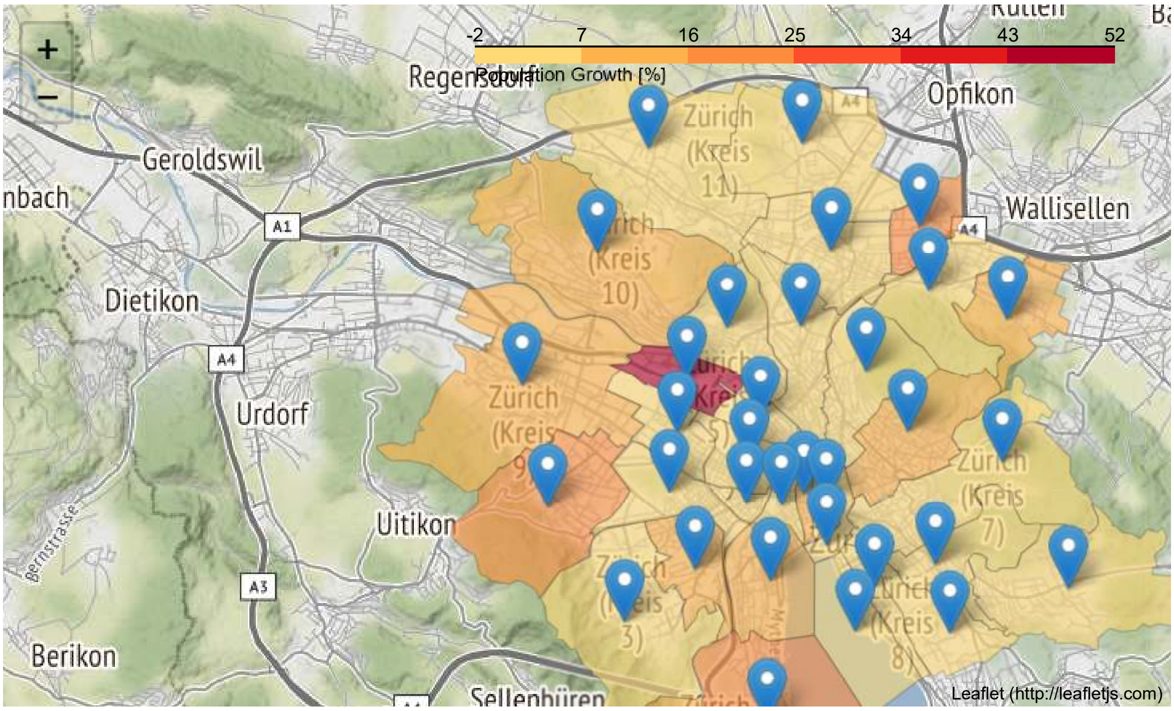
Out[50]:



Population Growth 2013 - 2018 across Neighborhoods

In [51]:
```
show_neighborhood_data('Population_Growth', '', 'Population Growth [%]', 'YlOrRd')
```

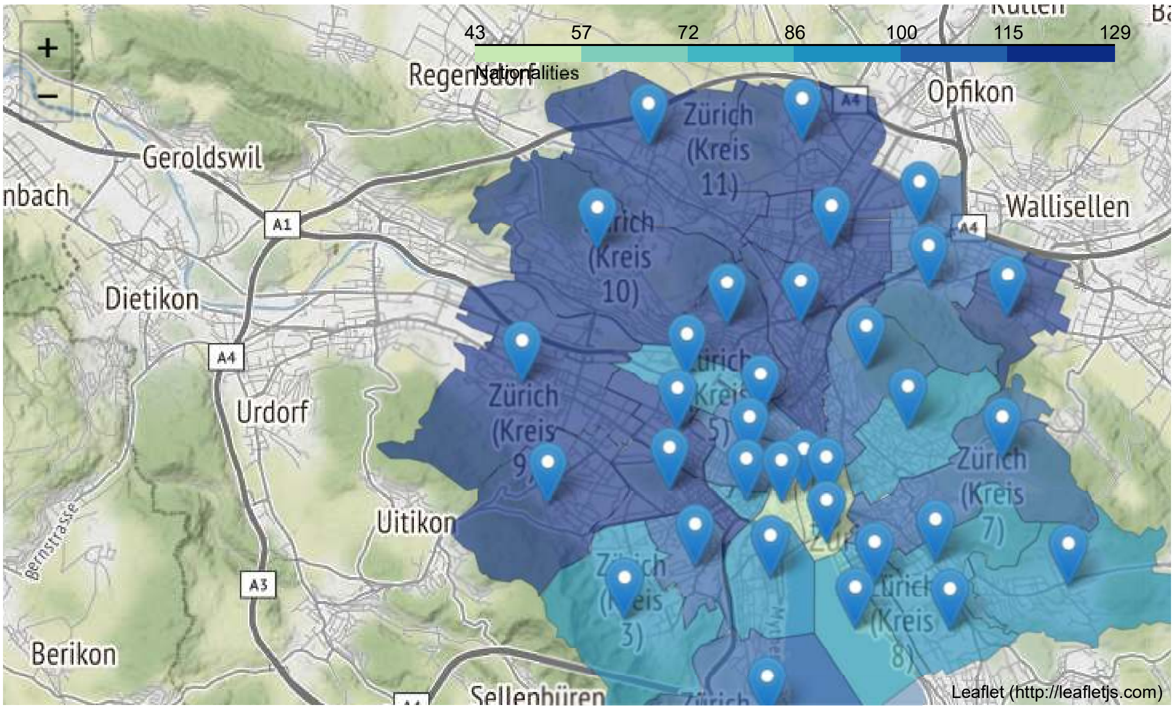
Out[51]:



Nationalities across Neighborhoods

In [52]:
```
show_neighborhood_data('Nationalities', '', 'Nationalities', 'YlGnBu')
```

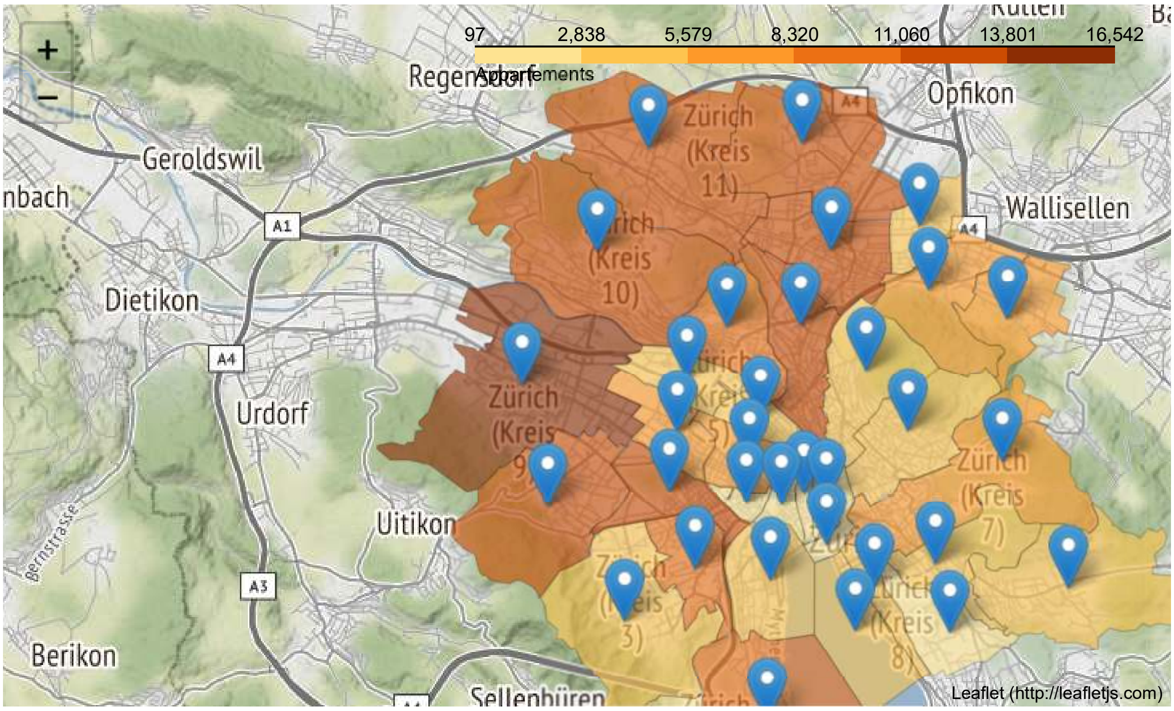
Out[52]:



Appartments across Neighborhoods

In [53]:
```
show_neighborhood_data('Appartements', '', 'Appartements', 'YlOrBr')
```

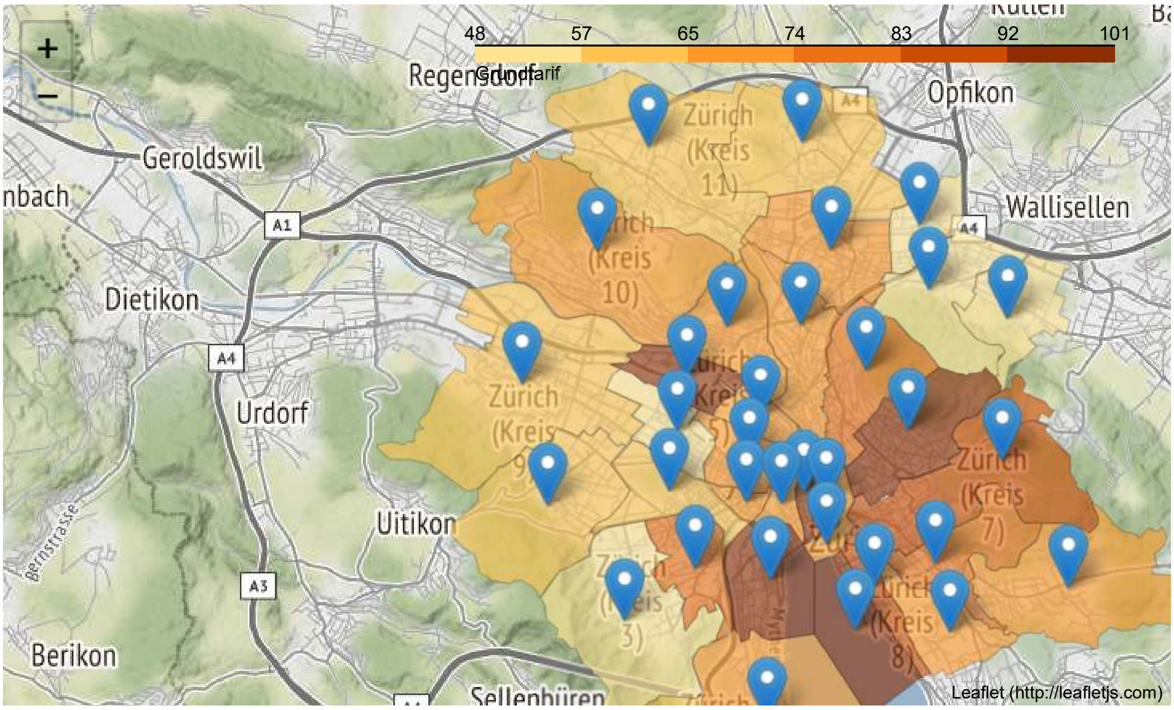
Out[53]:



Tax-Income 'Grundtarif' across Neighborhoods

In [54]:
```
show_neighborhood_data('Grundtarif', 'kCHF', 'Grundtarif', 'YlOrBr')
#show_neighborhood_data('Verheiratetentarif', 'kCHF', 'Verheiratetentarif', 'YlOrBr')
#show_neighborhood_data('Einelternfamilientarif', 'kCHF', 'Einelternfamilientarif', 'YlO
rBr')
```

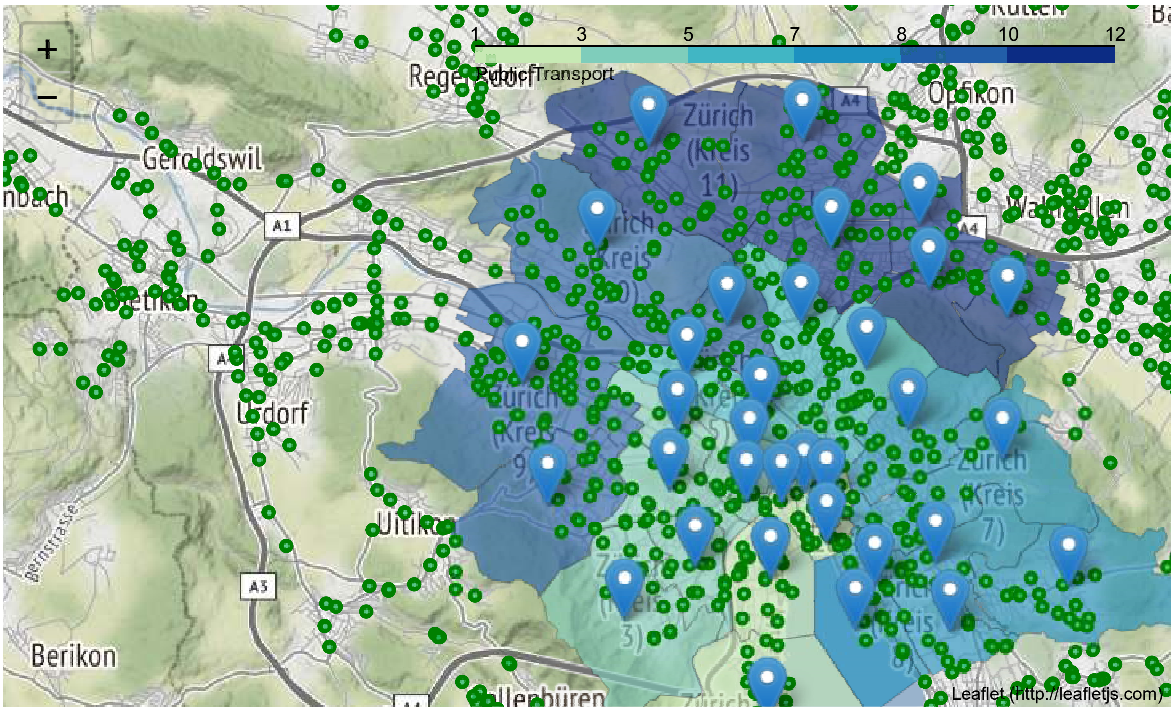
Out[54]:



Distribution Public Transport in Zürich

In [55]:
```
# Color Palettes
# 'BuGn', 'BuPu', 'GnBu', 'OrRd', 'PuBu', 'PuBuGn', 'PuRd', 'RdPu', 'YlGn', 'YlGnBu', 'Y
lOrBr', and 'YlOrRd'.

# evaluate map center
latitude = df_neighborhood['Latitude'].median()
longitude = df_neighborhood['Longitude'].median()

# build map
map = folium.Map(location=[latitude, longitude],
                 tiles='Stamen Terrain',
                 zoom_start=12)

# draw Neighborhood markers on map
zh_geo = DATA_PATH+'stzh.adm_statistische_quartiere_a.json'

# add chloropleth layer
map.choropleth(
    geo_data=zh_geo,
    name='choropleth',
    data = df_neighborhood,
    columns = ['Neighborhood', 'DistrictNb'],
    key_on = 'feature.properties.name',
    fill_color = 'YlGnBu',
    fill_opacity = 0.6,
    line_opacity = 0.2,
    legend_name = 'Public Transport'
)

# draw Venue markers on map
for lat, lng, stop_name in zip(df_public_transport['Latitude'],
                               df_public_transport['Longitude'],
                               df_public_transport['stop_name']):

    label = '{}'.format(stop_name)

    label = folium.Popup(label, parse_html=True)

    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='green',
        fill=True,
        fill_color='#31cc77',
        fill_opacity=0.7,
        parse_html=False).add_to(map)

# add markers with neighborhood information
fg = folium.FeatureGroup(name='Neighborhood Info')
for lat, lon, name in zip(df_neighborhood['Latitude'].tolist(),
                          df_neighborhood['Longitude'].tolist(),
                          df_neighborhood['Neighborhood'].tolist()):

    marker_text = '{}'.format(name)
    fg.add_child(folium.Marker(location=[lat, lon], popup = marker_text))

map.add_child(fg)

map
```

Out[55]:



Data Analysis on Where to open Asian Tappas Bar?

Source: [https://en.wikipedia.org/wiki/Pincho#/media/File:Bar_de_pinchos_Donosti_01.JPG (https://en.wikipedia.org/wiki/Pincho#/media/File:Bar_de_pinchos_Donosti_01.JPG)](https://en.wikipedia.org/wiki/Pincho#/media/File:Bar_de_pinchos_Donosti_01.JPG) [https://commons.wikimedia.org/wiki/User:Basotxerri (https://commons.wikimedia.org/wiki/User:Basotxerri)](https://commons.wikimedia.org/wiki/User:Basotxerri)

Collect and prepare data about venues across Zürich's neighborhoods

Fetch venues from Foursquare

Define function to load venues from Foursquare

In [56]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    LIMIT = 200

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secre
t={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

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

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

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

    return(nearby_venues)
```

Based on a application switch either fetch data from Foursquare or load data from a previous fetch and stored in a csv file

In [57]:
```
# Pull data from foursquare
if FOURSQUARE_ONLINE == True:
    nearby_venues = getNearbyVenues(names=df_neighborhood['Neighborhood'],
                                    latitudes=df_neighborhood['Latitude'],
                                    longitudes=df_neighborhood['Longitude'],
                                    radius = 999)
    # Store result in file
    nearby_venues.to_csv(DATA_PATH+'zrh_nearby_venues.csv', index = False)
    print("Result pulled, processed and saved.")
else:
    nearby_venues = pd.read_csv(DATA_PATH+'zrh_nearby_venues.csv')
    print('Data loaded.')
```

```
Data loaded.
```

Venue information fetched/loaded for analysis

In [58]:
```
print('Shape:',nearby_venues.shape)
nearby_venues.head()
```

```
Shape: (1984, 7)
```

Out[58]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Latitude	Longitude	Category
0	Rathaus	47.371933	8.544455	Fitnesspark Münstergasse	47.370888	8.544999	Gym / Fitness Center
1	Rathaus	47.371933	8.544455	Café Schober	47.371400	8.544149	Café
2	Rathaus	47.371933	8.544455	Schwarzenbach Kolonialwaren	47.371444	8.544091	Gourmet Shop
3	Rathaus	47.371933	8.544455	Old Crow	47.372092	8.541024	Cocktail Bar
4	Rathaus	47.371933	8.544455	Äss-Bar	47.372561	8.543693	Bakery

Filter out specific categories

Zürich has many Tram and Bus Stations. During would dominate clustering and distort the result

In [59]:
```
# Filter out venues categories
nearby_venues_filtered = nearby_venues
nearby_venues_filtered.dropna(inplace=True)
nearby_venues_filtered = nearby_venues_filtered[nearby_venues_filtered.Category != 'Tram Station']
nearby_venues_filtered = nearby_venues_filtered[nearby_venues_filtered.Category != 'Bus Station']

print('Shape:',nearby_venues_filtered.shape)
nearby_venues_filtered.head()
```

```
Shape: (1882, 7)
```

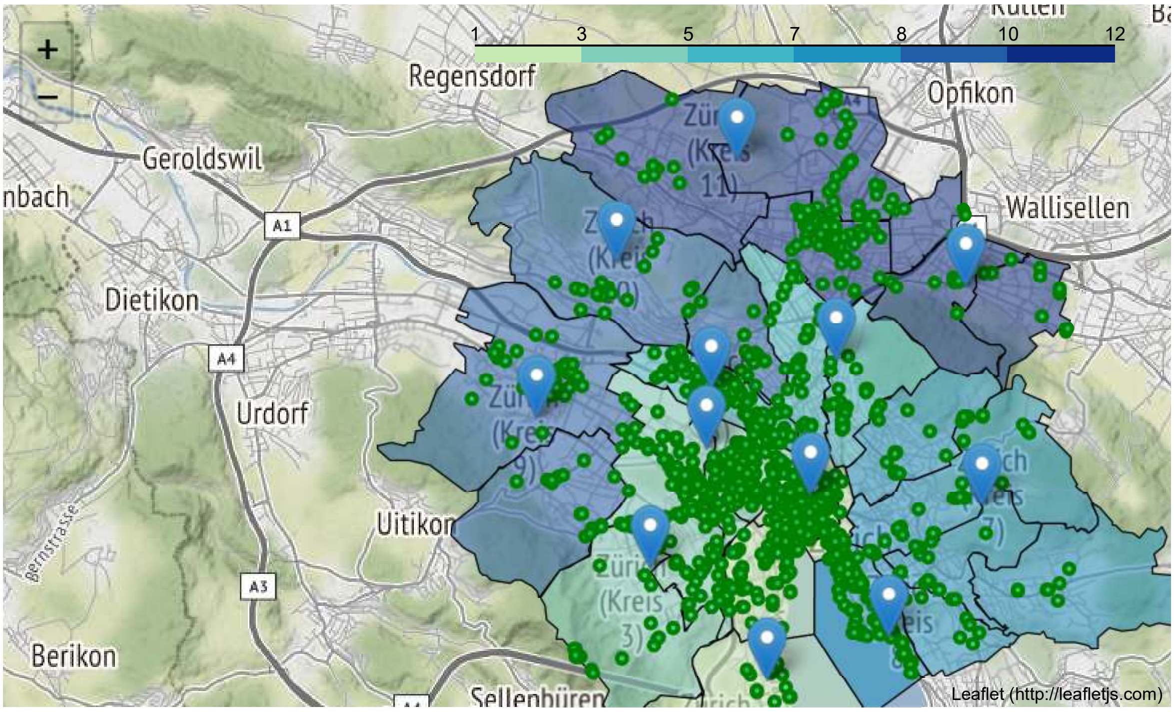
Out[59]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Latitude	Longitude	Category
0	Rathaus	47.371933	8.544455	Fitnesspark Münstergasse	47.370888	8.544999	Gym / Fitness Center
1	Rathaus	47.371933	8.544455	Café Schober	47.371400	8.544149	Café
2	Rathaus	47.371933	8.544455	Schwarzenbach Kolonialwaren	47.371444	8.544091	Gourmet Shop
3	Rathaus	47.371933	8.544455	Old Crow	47.372092	8.541024	Cocktail Bar
4	Rathaus	47.371933	8.544455	Äss-Bar	47.372561	8.543693	Bakery

Visualize the venues across Zürich's neighborhood

In [60]:
```
# evaluate map center
latitude = df_neighborhood['Latitude'].median()
longitude = df_neighborhood['Longitude'].median()

# build map
map = folium.Map(location=[latitude, longitude],
                 tiles='Stamen Terrain',
                 zoom_start=12)

# draw Neighborhood markers on map
zh_geo = DATA_PATH+'stzh.adm_statistische_quartiere_a.json'

## add chloropleth for Neighborhood
map.choropleth(
    geo_data=zh_geo,
    data=df_district_neighborhood,
    columns=['Neighborhood', 'DistrictNb'],
    key_on='feature.properties.name',
    fill_color='YlGnBu',
    fill_opacity=0.5,
    line_opacity=1
    )

# draw Neighborhood markers on map
fg = folium.FeatureGroup(name='Neighborhood')
for lat, lon, name in zip(df_district['Latitude'].tolist(),
                          df_district['Longitude'].tolist(),
                          df_district['District'].tolist()):

    marker_text = '{}'.format(name)
    fg.add_child(folium.Marker(location=[lat, lon], popup=marker_text))

map.add_child(fg)

# draw Venue markers on map
for lat, lng, categories, venue in zip(nearby_venues_filtered['Latitude'],
                                       nearby_venues_filtered['Longitude'],
                                       nearby_venues_filtered['Category'],
                                       nearby_venues_filtered['Venue']):

    label = '{}, {}'.format(venue, categories)

    label = folium.Popup(label, parse_html=True)

    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='green',
        fill=True,
        fill_color='#31cc77',
        fill_opacity=0.7,
        parse_html=False).add_to(map)
map
```

Out[60]:



Analyze Venues along Neighborhoods

Due to limitations with Foursquare the maximum number of venues returned is 100

In [64]:
```
nearby_venues_filtered.groupby('Neighborhood').count().sort_values('Category', ascendin
g=False)
```

Out[64]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Latitude	Longitude	Category
Hard	100	100	100	100	100	100
Rathaus	100	100	100	100	100	100
Werd	100	100	100	100	100	100
Hochschulen	100	100	100	100	100	100
Lindenhof	100	100	100	100	100	100
Gewerbeschule	100	100	100	100	100	100
Langstrasse	100	100	100	100	100	100
City	100	100	100	100	100	100
Mühlebach	99	99	99	99	99	99
Escher Wyss	99	99	99	99	99	99
Enge	98	98	98	98	98	98
Sihlfeld	98	98	98	98	98	98
Alt-Wiedikon	85	85	85	85	85	85
Oerlikon	80	80	80	80	80	80
Seefeld	66	66	66	66	66	66
Wipkingen	63	63	63	63	63	63
Altstetten	42	42	42	42	42	42
Weinegg	33	33	33	33	33	33
Hirslanden	33	33	33	33	33	33
Unterstrass	33	33	33	33	33	33
Oberstrass	31	31	31	31	31	31
Wollishofen	26	26	26	26	26	26
Hottingen	25	25	25	25	25	25
Fluntern	22	22	22	22	22	22
Saatlen	21	21	21	21	21	21
Seebach	21	21	21	21	21	21
Friesenberg	21	21	21	21	21	21
Hirzenbach	17	17	17	17	17	17
Höngg	17	17	17	17	17	17
Albisrieden	14	14	14	14	14	14
Affoltern	12	12	12	12	12	12
Schwamendingen-Mitte	12	12	12	12	12	12
Witikon	7	7	7	7	7	7
Leimbach	7	7	7	7	7	7

In [130]:
```
print('There are {} uniques venue categories.'.format(len(nearby_venues_filtered['Catego
ry'].unique())))
```

```
There are 220 uniques venue categories.
```

Build clusters of neighborhoods based on venues categories

Build a 'one-hot' representation

Rerquired to determine occurences of venue categories

In [131]:
```
# one hot encoding
venues_onehot = pd.get_dummies(nearby_venues_filtered[['Category']], prefix="", prefix_s
ep="")

# add neighborhood column back to dataframe
venues_onehot['Neighborhood'] = nearby_venues_filtered['Neighborhood']

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

print("Shape: ",venues_onehot.shape)
venues_onehot.head()
```

```
Shape:  (1882, 221)
```

Out[131]:

	Neighborhood	Accessories Store	Acupuncturist	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports
0	Rathaus	0	0	0	0	0	0	0	0	0
1	Rathaus	0	0	0	0	0	0	0	0	0
2	Rathaus	0	0	0	0	0	0	0	0	0
3	Rathaus	0	0	0	0	0	0	0	0	0
4	Rathaus	0	0	0	0	0	0	0	0	0

5 rows × 221 columns

Compute occurences of venue category per Neighborhood

Consolidate the 'one-hot' representation per neighborhood

In [132]:
```
# venues_grouped = venues_onehot.groupby('Neighborhood').mean().reset_index()
venues_grouped = venues_onehot.groupby('Neighborhood').sum().reset_index()
print("Shape: ", venues_grouped.shape)
venues_grouped.head()
```

```
Shape:  (34, 221)
```

Out[132]:

	Neighborhood	Accessories Store	Acupuncturist	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports
0	Affoltern	0	0	0	0	0	0	0	0	1
1	Albisrieden	0	0	0	0	0	0	0	0	0
2	Alt-Wiedikon	0	0	0	0	0	0	0	2	3
3	Altstetten	0	0	0	0	0	0	0	1	0
4	City	0	0	0	1	0	1	1	0	0

5 rows × 221 columns

Determine the most busy Neighborhoods

Accumulate the venue occurences per venue category and neighborhood and sort according to the total occurences

In [133]:
```
if 'Venues' in venues_grouped.columns:
    venues_grouped.drop('Venues', axis=1, inplace=True)

# Remove Index before totalling
venues_grouped.set_index('Neighborhood', inplace=True)

# sum entries for each Neighborhood
venues_grouped.loc[:,'Venues'] = venues_grouped.sum(axis=1,skipna=True,numeric_only=Tru
e)

# order Neighborhood by total
venues_grouped = venues_grouped.sort_values('Venues', ascending=False).reset_index()

# save the Neighborhood by total venues
venues_neighborhood = venues_grouped[['Neighborhood','Venues']]

# remove Total after sorting - not required any further
venues_grouped.drop('Venues', axis=1, inplace=True)

venues_grouped.head(8)
```

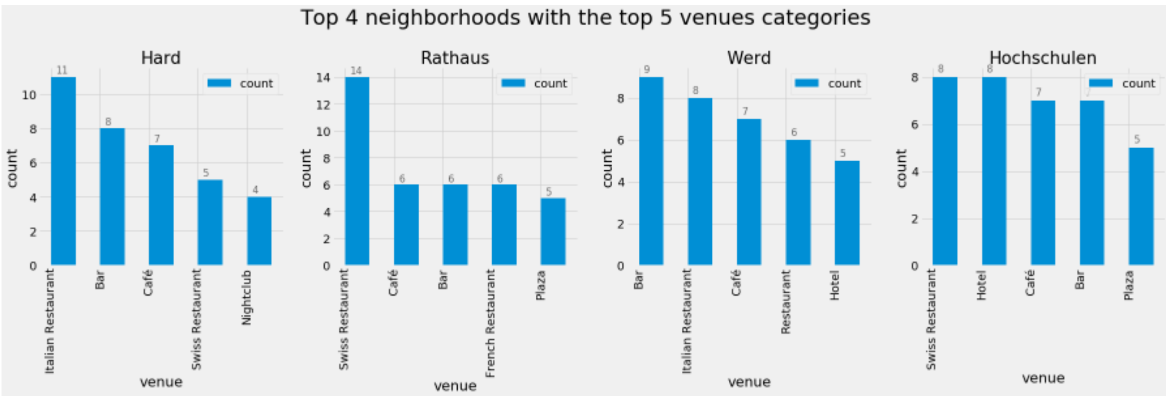
Out[133]:

	Neighborhood	Accessories Store	Acupuncturist	American Restaurant	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athletics & Sports
0	Hard	1	0	0	0	0	1	0	1	0
1	Rathaus	0	0	0	1	0	1	2	0	0
2	Werd	0	0	0	1	0	1	0	1	0
3	Hochschulen	0	0	0	1	0	1	0	1	0
4	Lindenhof	0	0	0	1	0	0	2	0	0
5	Gewerbeschule	2	0	0	0	1	0	1	3	0
6	Langstrasse	0	0	0	0	0	1	0	4	0
7	City	0	0	0	1	0	1	1	0	0

8 rows × 221 columns

Extract neighborhoods with the most often observed venue categories

In [134]:
```
# get the values to use in the graphic of the top 5 common venues
num_top_areas = 4
num_top_venues = 5

# optional: improve look and feel
mpl.style.use('fivethirtyeight')
plt.rcParams.update({'font.size': 18})

fig, axes = plt.subplots(nrows=1, ncols=num_top_areas)


# loop used to sum the quantity of venues and get the top 5 common venues
count = 0
for hood in venues_grouped['Neighborhood'].head(num_top_areas):

    # Pull and transpose neighborhoodrow
    df_tmp = venues_grouped[venues_grouped['Neighborhood'] == hood].T.reset_index()

    # rename column names
    df_tmp.columns = ['venue','count']

    # remove first row as it contains the neighborhood
    df_tmp = df_tmp.iloc[1:-1]

    # sort entries
    df_tmp = df_tmp.sort_values('count', ascending=False).reset_index(drop=True).head(nu
m_top_venues)

    # set index
    df_tmp.set_index('venue', inplace=True)

    # plot df_tmp data
    df_tmp.plot(kind='bar', figsize=(30, 6), position=0, ax=axes[count])

    axes[count].set_xlabel('venue') # add to x-label to the plot
    axes[count].set_ylabel('count') # add y-label to the plot
    axes[count].set_title(hood) # add title to the plot
    axes[count].set_alpha(0.8)

    for i in axes[count].patches:
        # get_x pulls left or right; get_height pushes up or down
        axes[count].text(i.get_x()+.1,
                         i.get_height()+.2,
                         str(i.get_height()),
                         fontsize=15, color='dimgrey')

    count = count + 1

fig.get_axes()[0].annotate('Top {} neighborhoods with the top {} venues categories'.form
at(num_top_areas, num_top_venues),
                            (0.5, 0.95),
                            xycoords='figure fraction',
                            ha='center',
                            fontsize=32)

plt.subplots_adjust(left=0.1, right=0.9, top=0.8, bottom=0.1)

print("")
```



Cluster Neighborhoods based on 'similar' distribution of venue categories

Function to pick a number of venues sorted by first row

In [135]:
```
def return_most_common_venues(row, num_top_venues):

    #remove first row
    row_categories = row.iloc[1:]

    #sort rows
    row_categories_sorted = row_categories.sort_values(ascending=False)

    #sort return only the defined number of entries
    return row_categories_sorted.index.values[0:num_top_venues]
```

Build a grid which illustrates the most common venues categories per neighborhood

In [137]:
```
num_top_venues = 10

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

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

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

# process all neighborhoods
for ind in np.arange(venues_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(venues_groupe
d.iloc[ind, :], num_top_venues)

print('Shape: {}'.format(neighborhoods_venues_sorted.shape))
neighborhoods_venues_sorted
```

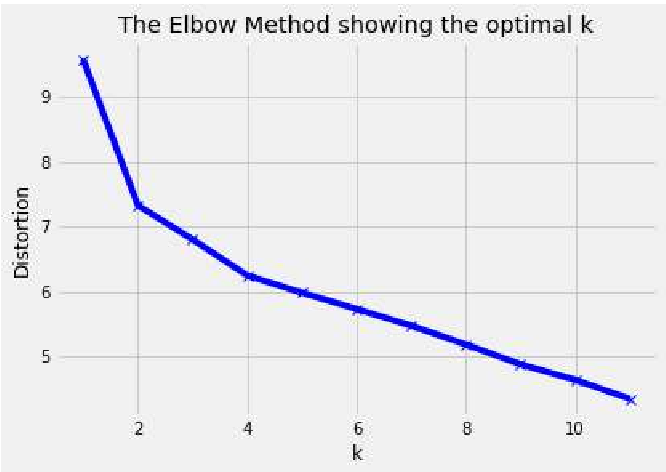
```
Shape: (34, 11)
```

Out[137]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	
0	Hard	Italian Restaurant	Bar	Café	Swiss Restaurant	Park	Mediterranean Restaurant	Nightclub	
1	Rathaus	Swiss Restaurant	Café	French Restaurant	Bar	Plaza	Cocktail Bar	Lounge	
2	Werd	Bar	Italian Restaurant	Café	Restaurant	Hotel	Cocktail Bar	Swiss Restaurant	F
3	Hochschulen	Swiss Restaurant	Hotel	Café	Bar	Plaza	Lounge	Dessert Shop	Cc
4	Lindenhof	Swiss Restaurant	Bar	Cocktail Bar	Café	Restaurant	French Restaurant	Lounge	Ve F
5	Gewerbeschule	Bar	Café	Thai Restaurant	Hotel	Middle Eastern Restaurant	Asian Restaurant	Swiss Restaurant	
6	Langstrasse	Bar	Swiss Restaurant	Café	Italian Restaurant	Thai Restaurant	Vegetarian / Vegan Restaurant	Asian Restaurant	
7	City	Bar	Hotel	Swiss Restaurant	Café	Cocktail Bar	Vegetarian / Vegan Restaurant	Restaurant	
8	Mühlebach	Swiss Restaurant	Hotel	Italian Restaurant	Restaurant	Coffee Shop	Bar	Movie Theater	
9	Escher Wyss	Café	Bar	Hotel	Restaurant	Nightclub	Swiss Restaurant	Art Museum	F
10	Enge	Hotel	Restaurant	Bar	Park	Swiss Restaurant	Italian Restaurant	Coffee Shop	Su
11	Sihlfeld	Bar	Café	Italian Restaurant	Swiss Restaurant	Hotel	Supermarket	Pizza Place	F
12	Alt-Wiedikon	Italian Restaurant	Supermarket	Hotel	Swiss Restaurant	Athletics & Sports	Pizza Place	Bakery	
13	Oerlikon	Supermarket	Hotel	Restaurant	Coffee Shop	Italian Restaurant	Chinese Restaurant	Indian Restaurant	
14	Seefeld	Italian Restaurant	Restaurant	Café	Swiss Restaurant	Bakery	Park	Supermarket	
15	Wipkingen	Restaurant	Italian Restaurant	Swiss Restaurant	Bar	Bakery	Supermarket	Plaza	
16	Altstetten	Supermarket	Platform	Bakery	Train Station	Middle Eastern Restaurant	Mexican Restaurant	French Restaurant	
17	Weinegg	Swiss Restaurant	Museum	Italian Restaurant	Restaurant	Café	French Restaurant	Supermarket	F
18	Hirslanden	Swiss Restaurant	Plaza	Italian Restaurant	Hotel	Park	Light Rail Station	Supermarket	
19	Unterstrass	Pizza Place	Italian Restaurant	Bakery	Park	Café	Grocery Store	Middle Eastern Restaurant	
20	Oberstrass	Swiss Restaurant	Italian Restaurant	Hotel	Supermarket	Bakery	Cable Car	Beer Store	Sr
21	Wollishofen	Supermarket	Swiss Restaurant	Gas Station	Mediterranean Restaurant	Restaurant	Harbor / Marina	Music Venue	
22	Hottingen	Restaurant	Swiss Restaurant	Zoo Exhibit	Pool	Sports Club	Spa	South American Restaurant	Sk
23	Fluntern	Hotel	Plaza	Grocery Store	Bakery	Zoo	Theater	Gastropub	
24	Saatlen	Swiss Restaurant	Pool Hall	Dance Studio	Restaurant	Stadium	Diner	Supermarket	F
25	Seebach	Hookah Bar	Supermarket	Bowling Alley	Auto Workshop	Mini Golf	Café	Massage Studio	Sc
26	Friesenberg	Swiss Restaurant	Trail	Scenic Lookout	Mountain	Outdoor Sculpture	Café	Tennis Court	Su

Apply multiple cluster attempts to find the optimal number of cluster using the elbow method

In [138]:
```
from scipy.spatial.distance import cdist

# optional: improve look and feel
mpl.style.use('fivethirtyeight')
plt.rcParams.update({'font.size': 10})

# drop neighborhood before clustering
venues_grouped_clustering = venues_grouped.drop('Neighborhood', 1)

# k means determine k
distortions = []
K = range(1,12)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0).fit(venues_grouped_clustering)
    distortions.append(sum(np.min(cdist(venues_grouped_clustering, kmeans.cluster_center
s_, 'euclidean'), axis=1)) / venues_grouped_clustering.shape[0])

# Plot the elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



Cluster with the chosen 'optimal' number of clusters

the elbow method didn't clearly outline an optmial value therfore I have chosen 4

In [139]:
```
# set number of clusters
kclusters = 4

# drop neighborhood before clustering
venues_grouped_clustering = venues_grouped.drop('Neighborhood', 1)

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

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

Out[139]:
```
array([3, 2, 3, 2, 2, 3, 3, 2, 0, 0], dtype=int32)
```

Merge the cluster with the city information

In [141]:

```
# remove clustering labels in case the column is already there
if 'Cluster' in neighborhoods_venues_sorted.columns:
    neighborhoods_venues_sorted.drop('Cluster', axis=1, inplace=True)

# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster', kmeans.labels_)

# Prepare dataframe to merge with cordinates
neighborhoods_merged = df_neighborhood

# merge neighborhoods_venues_sorted with df_explore to add latitude/longitude for each n
eighborhood
neighborhoods_merged = neighborhoods_merged.join(neighborhoods_venues_sorted.set_index('
Neighborhood'), on='Neighborhood')

# Cleanse some NaN from processing
neighborhoods_merged = neighborhoods_merged.dropna()

# Ensure cluster labels are int (can get changed due to NaN entries)
neighborhoods_merged['Cluster'] = neighborhoods_merged['Cluster'].astype(int)

# merge neighborhoods_venues_sorted with df_explore to add latitude/longitude for each n
eighborhood
neighborhoods_merged = neighborhoods_merged.join(venues_neighborhood.set_index('Neighbor
hood'), on='Neighborhood')

print('Shape: {}'.format(neighborhoods_merged.shape))
neighborhoods_merged.head()
```

```
Shape: (34, 25)
```

Out[141]:

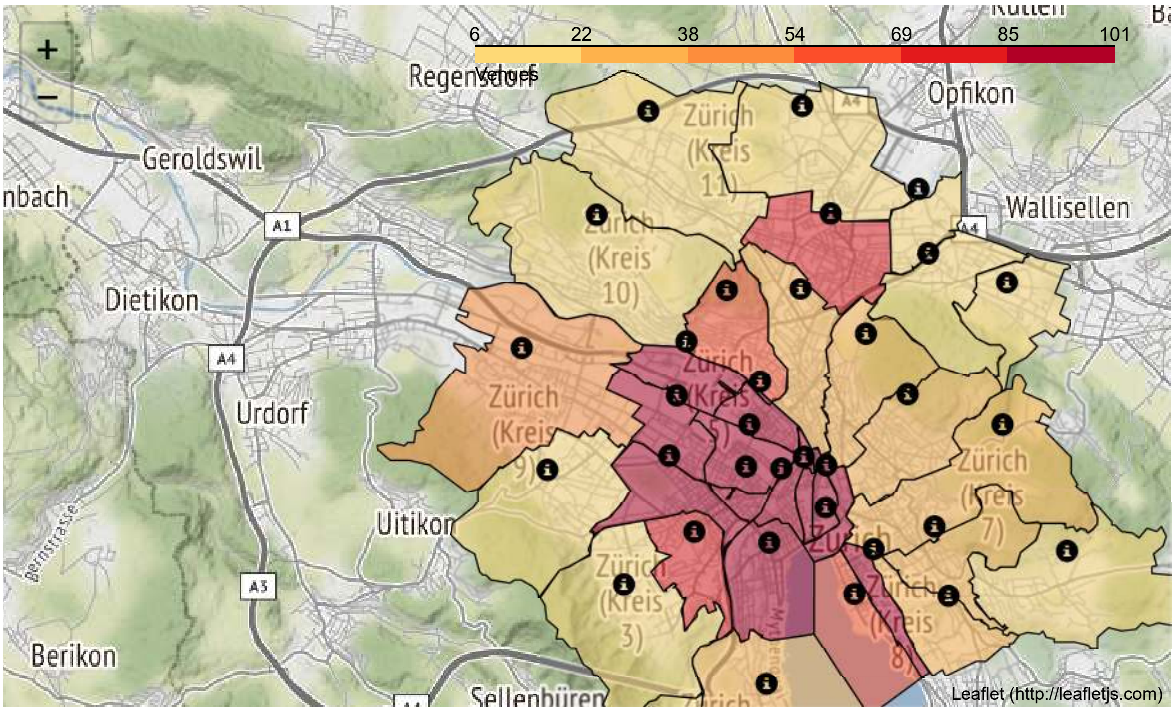
	District	Neighborhood	Longitude	Latitude	DistrictNb	Population	Nationalities	Population_Past	Population_Grov
0	Kreis 1	Rathaus	8.544455	47.371933	1	3267.0	79.0	3194.0	2.285
1	Kreis 1	Hochschulen	8.544603	47.365484	1	664.0	44.0	665.0	-0.150
2	Kreis 1	Lindenhof	8.539873	47.373063	1	990.0	47.0	923.0	7.258
3	Kreis 1	City	8.534951	47.371386	1	829.0	53.0	783.0	5.874
4	Kreis 2	Wollishofen	8.532078	47.339917	2	18923.0	113.0	15937.0	18.736

5 rows × 25 columns

Visualize Neighborhoods Clusters

In [143]:
```
# Color Palettes
# 'BuGn', 'BuPu', 'GnBu', 'OrRd', 'PuBu', 'PuBuGn', 'PuRd', 'RdPu', 'YlGn', 'YlGnBu', 'Y
lOrBr', and 'YlOrRd'.

# merge neighborhoods_venues_sorted with df_explore to add latitude/longitude for each n
eighborhood
df_district_neighborhood_ext = df_district_neighborhood.join(venues_neighborhood.set_ind
ex('Neighborhood'), on='Neighborhood')


# evaluate map center
latitude = df_neighborhood['Latitude'].median()
longitude = df_neighborhood['Longitude'].median()

# build map
map = folium.Map(location=[latitude, longitude],
                 tiles='Stamen Terrain',
                 zoom_start=12)

# draw Neighborhood markers on map
zh_geo = DATA_PATH+'stzh.adm_statistische_quartiere_a.json'

## add chloropleth for Neighborhood
map.choropleth(
    geo_data=zh_geo,
    data=df_district_neighborhood_ext,
    columns=['Neighborhood', 'Venues'],
    key_on='feature.properties.name',
    fill_color='YlOrRd',
    fill_opacity=0.5,
    line_opacity=1,
    legend_name="Venues"
    )

rainbow = ['blue','green', 'red', 'pink', 'black']

# draw Neighborhood markers on map
fg = folium.FeatureGroup(name='Neighborhood')
for lat, lon, neighborhood, venues, cluster in zip(neighborhoods_merged['Latitude'].toli
st(),
                                                   neighborhoods_merged['Longitude'].tol
ist(),
                                                   neighborhoods_merged['Neighborhood'].
tolist(),
                                                   neighborhoods_merged['Venues'].tolist
(),
                                                   neighborhoods_merged['Cluster'].tolis
t()):

    marker_text = 'Cluster {}<br/>{}<br/>Venues: {}'.format(cluster, neighborhood,venue
s)
    fg.add_child(folium.Marker(location = [lat, lon],
                               popup = marker_text,
                               icon=folium.Icon(color=rainbow[cluster]))
                )

map.add_child(fg)

map
```

Out[143]:



Review the Distribution of Venues in the various clusters

In [149]:
```
def showCluster(cluster):
    return neighborhoods_merged.loc[neighborhoods_merged['Cluster'] == cluster, neighbor
hoods_merged.columns[[0]+[1]+list(range(13, neighborhoods_merged.shape[1]-1))]]
```

Cluster 0 - Leisure and Shopping

In [150]:
```
showCluster(0)
```

Out[150]:

	District	Neighborhood	Cluster	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 M Comn Ve
6	Kreis 2	Enge	0	Hotel	Restaurant	Bar	Park	Swiss Restaurant	Italian Restaurant	Coffee S
7	Kreis 3	Alt-Wiedikon	0	Italian Restaurant	Supermarket	Hotel	Swiss Restaurant	Athletics & Sports	Pizza Place	Bak
14	Kreis 5	Escher Wyss	0	Café	Bar	Hotel	Restaurant	Nightclub	Swiss Restaurant	Art Muse
21	Kreis 8	Seefeld	0	Italian Restaurant	Restaurant	Café	Swiss Restaurant	Bakery	Park	Superma
22	Kreis 8	Mühlebach	0	Swiss Restaurant	Hotel	Italian Restaurant	Restaurant	Coffee Shop	Bar	Mc The
27	Kreis 10	Wipkingen	0	Restaurant	Italian Restaurant	Swiss Restaurant	Bar	Bakery	Supermarket	Pl
29	Kreis 11	Oerlikon	0	Supermarket	Hotel	Restaurant	Coffee Shop	Italian Restaurant	Chinese Restaurant	Inc Restau

Cluster 1 - Leisure, Sports and Shopping

In [151]:
```
showCluster(1)
```

Out[151]:

	District	Neighborhood	Cluster	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
4	Kreis 2	Wollishofen	1	Supermarket	Swiss Restaurant	Gas Station	Mediterranean Restaurant	Restaurant	Harbor / Marina
5	Kreis 2	Leimbach	1	Light Rail Station	Bakery	Gas Station	Supermarket	Trail	Organic Grocery
8	Kreis 3	Friesenberg	1	Swiss Restaurant	Trail	Scenic Lookout	Mountain	Outdoor Sculpture	Café
15	Kreis 6	Unterstrass	1	Pizza Place	Italian Restaurant	Bakery	Park	Café	Grocery Store
16	Kreis 6	Oberstrass	1	Swiss Restaurant	Italian Restaurant	Hotel	Supermarket	Bakery	Cable Car
17	Kreis 7	Fluntern	1	Hotel	Plaza	Grocery Store	Bakery	Zoo	Theater
18	Kreis 7	Hottingen	1	Restaurant	Swiss Restaurant	Zoo Exhibit	Pool	Sports Club	Spa
19	Kreis 7	Hirslanden	1	Swiss Restaurant	Plaza	Italian Restaurant	Hotel	Park	Light Rail Station
20	Kreis 7	Witikon	1	Church	Optical Shop	Soccer Field	Supermarket	Department Store	Bakery
23	Kreis 8	Weinegg	1	Swiss Restaurant	Museum	Italian Restaurant	Restaurant	Café	French Restaurant
24	Kreis 9	Albisrieden	1	Supermarket	Swiss Restaurant	Convenience Store	Thai Restaurant	Breakfast Spot	Café
25	Kreis 9	Altstetten	1	Supermarket	Platform	Bakery	Train Station	Middle Eastern Restaurant	Mexican Restaurant
26	Kreis 10	Höngg	1	Grocery Store	Indian Restaurant	Other Great Outdoors	Fast Food Restaurant	Spa	Mexican Restaurant
28	Kreis 11	Affoltern	1	Supermarket	Light Rail Station	Electronics Store	Lake	Department Store	Athletics & Sports
30	Kreis 11	Seebach	1	Hookah Bar	Supermarket	Bowling Alley	Auto Workshop	Mini Golf	Café
31	Kreis 12	Saatlen	1	Swiss Restaurant	Pool Hall	Dance Studio	Restaurant	Stadium	Diner
32	Kreis 12	Schwamendingen-Mitte	1	Swiss Restaurant	Trail	Asian Restaurant	Automotive Shop	Thai Restaurant	Supermarket
33	Kreis 12	Hirzenbach	1	Convenience Store	Soccer Field	Business Service	Swiss Restaurant	Baseball Field	Steakhouse

Cluster 2 - Busy Area for Food and Drinks

In [152]:

```
showCluster(2)
```

Out[152]:

	District	Neighborhood	Cluster	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 Con V
0	Kreis 1	Rathaus	2	Swiss Restaurant	Café	French Restaurant	Bar	Plaza	Cocktail Bar	Lounge	
1	Kreis 1	Hochschulen	2	Swiss Restaurant	Hotel	Café	Bar	Plaza	Lounge	Dessert Shop	C
2	Kreis 1	Lindenhof	2	Swiss Restaurant	Bar	Cocktail Bar	Café	Restaurant	French Restaurant	Lounge	Vege / \ Resta
3	Kreis 1	City	2	Bar	Hotel	Swiss Restaurant	Café	Cocktail Bar	Vegetarian / Vegan Restaurant	Restaurant	Lc

Cluster 3 - Busy Area for Drinks and Food

In [153]:

```
showCluster(3)
```

Out[153]:

	District	Neighborhood	Cluster	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
9	Kreis 3	Sihlfeld	3	Bar	Café	Italian Restaurant	Swiss Restaurant	Hotel	Supermarket	Pizza Place
10	Kreis 4	Werd	3	Bar	Italian Restaurant	Café	Restaurant	Hotel	Cocktail Bar	Swiss Restaurant
11	Kreis 4	Langstrasse	3	Bar	Swiss Restaurant	Café	Italian Restaurant	Thai Restaurant	Vegetarian / Vegan Restaurant	Asian Restaurant
12	Kreis 4	Hard	3	Italian Restaurant	Bar	Café	Swiss Restaurant	Park	Mediterranean Restaurant	Nightclub
13	Kreis 5	Gewerbeschule	3	Bar	Café	Thai Restaurant	Hotel	Middle Eastern Restaurant	Asian Restaurant	Swiss Restaurant

In [ ]:

```

```