Importing Libraries

In [1]:

```
import pandas as pd # library to process data as dataframes
! pip install lxml
import lxml
```

```
Collecting lxml
  Downloading https://files.pythonhosted.org/packages/64/28/0b761b64ecbd63d272ed0e7a6ae6e4402fc37886b59181bfdf274424d
693/lxml-4.6.1-cp36-cp36m-manylinux1_x86_64.whl (5.5MB)
     |████████████████████████████████| 5.5MB 6.7MB/s eta 0:00:01
Installing collected packages: lxml
Successfully installed lxml-4.6.1
```

In [2]:

```
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
```

In [3]:

```
import matplotlib.pyplot as plt # plotting library
# backend for rendering plots within the browser
%matplotlib inline

from sklearn.cluster import KMeans # import k-means from clustering stage
from sklearn.datasets.samples_generator import make_blobs
```

In [4]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [5]:

```
!conda install -c conda-forge geopy --yes
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.9.1
  latest version: 4.9.2

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2020.11.8  |       ha878542_0         145 KB  conda-forge
    certifi-2020.11.8          |   py36h5fab9bb_0         150 KB  conda-forge
    geographiclib-1.50         |             py_0          34 KB  conda-forge
    geopy-2.0.0                |     pyh9f0ad1d_0          63 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         392 KB

The following NEW packages will be INSTALLED:

  geographiclib      conda-forge/noarch::geographiclib-1.50-py_0
  geopy              conda-forge/noarch::geopy-2.0.0-pyh9f0ad1d_0

The following packages will be UPDATED:

  ca-certificates                      2020.6.20-hecda079_0 --> 2020.11.8-ha878542_0
  certifi                          2020.6.20-py36h9880bd3_2 --> 2020.11.8-py36h5fab9bb_0



Downloading and Extracting Packages
certifi-2020.11.8    | 150 KB    | ##################################### | 100%
ca-certificates-2020 | 145 KB    | ##################################### | 100%
geopy-2.0.0          | 63 KB     | ##################################### | 100%
geographiclib-1.50   | 34 KB     | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

In [6]:

```
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
```

In [7]:

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

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

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

from folium import plugins
from folium.plugins import HeatMap
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): failed

# >>>>>>>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<<<<<<<

    Traceback (most recent call last):
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 436, in _error_catcher
        yield
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 763, in read_chunked
        self._update_chunk_length()
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 693, in _update_chunk_lengt
h
        line = self._fp.fp.readline()
      File "/home/jupyterlab/conda/lib/python3.8/socket.py", line 669, in readinto
        return self._sock.recv_into(b)
      File "/home/jupyterlab/conda/lib/python3.8/ssl.py", line 1241, in recv_into
        return self.read(nbytes, buffer)
      File "/home/jupyterlab/conda/lib/python3.8/ssl.py", line 1099, in read
        return self._sslobj.read(len, buffer)
    ConnectionResetError: [Errno 104] Connection reset by peer

    During handling of the above exception, another exception occurred:

    Traceback (most recent call last):
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/models.py", line 751, in generate
        for chunk in self.raw.stream(chunk_size, decode_content=True):
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 571, in stream
        for line in self.read_chunked(amt, decode_content=decode_content):
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 792, in read_chunked
        self._original_response.close()
      File "/home/jupyterlab/conda/lib/python3.8/contextlib.py", line 131, in __exit__
        self.gen.throw(type, value, traceback)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/urllib3/response.py", line 454, in _error_catcher
        raise ProtocolError("Connection broken: %r" % e, e)
    urllib3.exceptions.ProtocolError: ("Connection broken: ConnectionResetError(104, 'Connection reset by peer')", Co
nnectionResetError(104, 'Connection reset by peer'))

    During handling of the above exception, another exception occurred:

    Traceback (most recent call last):
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/exceptions.py", line 1079, in __call__
        return func(*args, **kwargs)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/cli/main.py", line 84, in _main
        exit_code = do_call(args, p)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/cli/conda_argparse.py", line 83, in do_call
        return getattr(module, func_name)(args, parser)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/cli/main_install.py", line 20, in execute
        install(args, parser, 'install')
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/cli/install.py", line 261, in install
        unlink_link_transaction = solver.solve_for_transaction(
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/solve.py", line 114, in solve_for_transacti
on
        unlink_precs, link_precs = self.solve_for_diff(update_modifier, deps_modifier,
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/solve.py", line 157, in solve_for_diff
        final_precs = self.solve_final_state(update_modifier, deps_modifier, prune, ignore_pinned,
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/solve.py", line 262, in solve_final_state
        ssc = self._collect_all_metadata(ssc)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/common/io.py", line 88, in decorated
        return f(*args, **kwds)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/solve.py", line 425, in _collect_all_metada
ta
        index, r = self._prepare(prepared_specs)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/solve.py", line 1020, in _prepare
        reduced_index = get_reduced_index(self.prefix, self.channels,
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/index.py", line 276, in get_reduced_index
        new_records = SubdirData.query_all(spec, channels=channels, subdirs=subdirs,
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 120, in query_all
        result = tuple(concat(executor.map(subdir_query, channel_urls)))
      File "/home/jupyterlab/conda/lib/python3.8/concurrent/futures/_base.py", line 611, in result_iterator
        yield fs.pop().result()
      File "/home/jupyterlab/conda/lib/python3.8/concurrent/futures/_base.py", line 439, in result
        return self.__get_result()
      File "/home/jupyterlab/conda/lib/python3.8/concurrent/futures/_base.py", line 388, in __get_result
        raise self._exception
      File "/home/jupyterlab/conda/lib/python3.8/concurrent/futures/thread.py", line 57, in run
        result = self.fn(*self.args, **self.kwargs)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 112, in <lambda>
        subdir_query = lambda url: tuple(SubdirData(Channel(url), repodata_fn=repodata_fn).query(
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 125, in query
        self.load()
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 189, in load
        _internal_state = self._load()
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 259, in _load
        raw_repodata_str = fetch_repodata_remote_request(
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/conda/core/subdir_data.py", line 499, in fetch_repodat
a_remote_request
        resp = session.get(join_url(url, filename), headers=headers, proxies=session.proxies,
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/sessions.py", line 543, in get
        return self.request('GET', url, **kwargs)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/sessions.py", line 530, in request
        resp = self.send(prep, **send_kwargs)
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/sessions.py", line 685, in send
        r.content
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/models.py", line 829, in content
        self._content = b''.join(self.iter_content(CONTENT_CHUNK_SIZE)) or b''
      File "/home/jupyterlab/conda/lib/python3.8/site-packages/requests/models.py", line 754, in generate
        raise ChunkedEncodingError(e)
    requests.exceptions.ChunkedEncodingError: ("Connection broken: ConnectionResetError(104, 'Connection reset by pee
r')", ConnectionResetError(104, 'Connection reset by peer'))

`$ /home/jupyterlab/conda/condabin/conda install -c conda-forge folium=0.5.0 --yes`

  environment variables:
                 CIO_TEST=<not set>
        CONDA_BACKUP_HOST=x86_64-conda_cos6-linux-gnu
   CONDA_BACKUP_JAVA_HOME=/usr/jre1.8.0_211
CONDA_BACKUP_JAVA_LD_LIBRARY_PATH=
        CONDA_DEFAULT_ENV=python
                CONDA_DIR=/home/jupyterlab/conda
                CONDA_EXE=/home/jupyterlab/conda/bin/conda
             CONDA_PREFIX=/home/jupyterlab/conda/envs/python
           CONDA_PREFIX_1=/home/jupyterlab/conda/envs/jupyterlab
    CONDA_PROMPT_MODIFIER=(python)
         CONDA_PYTHON_EXE=/home/jupyterlab/conda/bin/python
               CONDA_ROOT=/home/jupyterlab/conda
              CONDA_SHLVL=2
           CURL_CA_BUNDLE=<not set>
     JAVA_LD_LIBRARY_PATH=/home/jupyterlab/conda/envs/python/jre/lib/amd64/server
                     PATH=/home/jupyterlab/conda/envs/python/bin:/home/jupyterlab/conda/condabin
                          :/home/jupyterlab/conda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/
                          usr/bin:/sbin:/bin:/usr/jre1.8.0_211/bin:/home/jupyterlab/hadoop-2.9.2
                          /bin:/home/jupyterlab/spark-2.4.3/bin
           PYTHONHASHSEED=0
         PYTHONIOENCODING=UTF-8
```

```
       REQUESTS_CA_BUNDLE=<not set>
     SPARK_DIST_CLASSPATH=/home/jupyterlab/hadoop-2.9.2/etc/hadoop/*:/home/jupyterlab/hadoop-2.9
                          .2/share/hadoop/common/lib/*:/home/jupyterlab/hadoop-2.9.2/share/hadoo
                          p/common/*:/home/jupyterlab/hadoop-2.9.2/share/hadoop/hdfs/*:/home/jup
                          yterlab/hadoop-2.9.2/share/hadoop/hdfs/lib/*:/home/jupyterlab/hadoop-2
                          .9.2/share/hadoop/hdfs/*:/home/jupyterlab/hadoop-2.9.2/share/hadoop/ya
                          rn/lib/*:/home/jupyterlab/hadoop-2.9.2/share/hadoop/yarn/*:/home/jupyt
                          erlab/hadoop-2.9.2/share/hadoop/mapreduce/lib/*:/home/jupyterlab/hadoo
                          p-2.9.2/share/hadoop/mapreduce/*:/home/jupyterlab/hadoop-2.9.2/share/h
                          adoop/tools/lib/*
            SSL_CERT_FILE=<not set>

     active environment : python
    active env location : /home/jupyterlab/conda/envs/python
            shell level : 2
       user config file : /home/jupyterlab/.condarc
 populated config files : 
          conda version : 4.9.1
    conda-build version : not installed
         python version : 3.8.5.final.0
       virtual packages : __glibc=2.28=0
                          __unix=0=0
                          __archspec=1=x86_64
       base environment : /home/jupyterlab/conda  (writable)
           channel URLs : https://conda.anaconda.org/conda-forge/linux-64
                          https://conda.anaconda.org/conda-forge/noarch
                          https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /home/jupyterlab/conda/pkgs
                          /home/jupyterlab/.conda/pkgs
       envs directories : /home/jupyterlab/conda/envs
                          /home/jupyterlab/.conda/envs
               platform : linux-64
             user-agent : conda/4.9.1 requests/2.24.0 CPython/3.8.5 Linux/4.15.0-112-generic debian/10 glibc/2.28
                UID:GID : 1000:2000
             netrc file : None
           offline mode : False


An unexpected error has occurred. Conda has prepared the above report.
```

```
Upload successful.
```

Data

Getting the rental price per square meter per Borough from de.statista.com

In [8]:

```
url='https://de.statista.com/statistik/daten/studie/262505/umfrage/mietpreise-in-frankfurt-am-main-nach-bezirken/#professional'
dfs1 = pd.read_html(url)

# Get first table
df1 = dfs1[0]

# correcting the prices according to the number of digits
df1['Mietpreis in Euro pro m²']=df1['Mietpreis in Euro pro m²'].astype(float)
for i in range(len(df1)):
    if df1['Mietpreis in Euro pro m²'][i] > 999:
        df1['Mietpreis in Euro pro m²'][i]=df1['Mietpreis in Euro pro m²'][i]/100
    elif df1['Mietpreis in Euro pro m²'][i] > 99:
        df1['Mietpreis in Euro pro m²'][i]=df1['Mietpreis in Euro pro m²'][i]/10

# Renaming the columns
df1.rename(columns={'Unnamed: 0': 'Borough', 'Mietpreis in Euro pro m²': 'Rental Price'}, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/ipykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  del sys.path[0]
```

In [9]:

```
# Splitting the rows with multiple boroughs
df1=df1.set_index(['Rental Price']).apply(lambda x: x.str.split(',').explode())

# Remove leading spaces and trailing *
df1['Borough'] = df1['Borough'].str.strip()
df1['Borough'] = df1['Borough'].str.strip('*')
```

In [10]:

```
df1=df1.reset_index()
```

In [11]:

```
# Split Sachsenhausen-Nord/-Süd into 2 boroughs
df1=df1.append({'Rental Price': 13.50, 'Borough':'Sachsenhausen-Nord'}, ignore_index=True)
df1=df1.append({'Rental Price': 13.50, 'Borough':'Sachsenhausen-Süd'}, ignore_index=True)
```

In [12]:

```
# drop row with joint boroughs
# drop borough 'airport'
df1 = df1[df1.Borough != 'Sachsenhausen-Nord/-Süd']
frankfurt_rental_data = df1[df1.Borough != 'Flughafen']
```

In [13]:

```
# Sort by boroughs
frankfurt_rental_data=frankfurt_rental_data.sort_values(by=['Borough'])
frankfurt_rental_data=frankfurt_rental_data.reset_index(drop=True)

#frankfurt_rental_data.head()
```

In [14]:

```
print('We have rental prices for {} boroughs'.format(len(frankfurt_rental_data)))
```

```
We have rental prices for 45 boroughs
```

From the portal of the city of Frankfurt, we get data of the density of the boroughs (population per ha)

In [15]:

```
!wget --quiet https://offenedaten.frankfurt.de/dataset/a0feb40c-b5f5-4ba2-a1fc-217229f65a96/resource/8153b993-ee1b-462a-abd8-ed19bc94dcb0/download/bauenwohnen.json -O fra_density.json
#nbh_density = r'fra_density.json'
with open('fra_density.json') as json_data:
    fra_density_data = json.load(json_data)

# define the dataframe columns
column_names = ['Borough', 'Population Density']

# instantiate the dataframe
frankfurt_density_data = pd.DataFrame(columns=column_names)

for i in range(0,45):
    frankfurt_density_data = frankfurt_density_data.append({'Borough': fra_density_data[i]['Stadtteil'],
                                                            'Population Density': fra_density_data[i]['Bauen und Wohnen Einwohnerdichte je ha  2012']}, ignore_index=True)
```

In [16]:

```
# Replacing decimal commas
frankfurt_density_data['Population Density'] = [x.replace(',', '.') for x in frankfurt_density_data['Population Density']]
frankfurt_density_data['Population Density'] = pd.to_numeric(frankfurt_density_data['Population Density'],errors='coerce')

# adding the value of Sachsenhausen-Süd to Sachsenhausen-Nord
frankfurt_density_data.loc[(frankfurt_density_data.Borough == 'Sachsenhausen-Nord'),'Population Density']= 9.4
```

In [17]:

```
# Replacing the missing value of population density by the mean value
frankfurt_density_data['Population Density'].fillna(value=frankfurt_density_data['Population Density'].mean(), inplace=True)
```

From the portal of the city of Frankfurt, we get other population data of the boroughs:

average age

percentage of the population between 18 and 64

percentage of single-person households

In [19]:

```
!wget --quiet https://offenedaten.frankfurt.de/dataset/3be1af84-12d5-4d91-979a-3a468c77ed4e/resource/d4fc2f98-43cd-4a6c-8511-02ee1
d1165a2/download/bevoelkerung.json -O fra_population.json

with open('fra_population.json') as json_data:
    fra_population_data = json.load(json_data)

# define the dataframe columns
column_names = ['Borough', 'Average Age','Percentage 18-64','Percentage Single Households']

# instantiate the dataframe
frankfurt_population_data = pd.DataFrame(columns=column_names)

for i in range(0,45):
    frankfurt_population_data = frankfurt_population_data.append({'Borough': fra_population_data[i]['Stadtteil'],
                                                                  'Average Age': fra_population_data[i]['Bevölkerung Durchschnitts
alter  2012'],
                                                                  'Percentage 18-64': fra_population_data[i]['Bevölkerung Einwohne
rinnen und Einwohner von 18 bis 64 Jahren in %  2012'],
                                                                  'Percentage Single Households': fra_population_data[i]['Bevölker
ung Einpersonenhaushalte in %  2012']}, ignore_index=True)
# Replacing decimal commas
frankfurt_population_data['Average Age'] = [x.replace(',', '.') for x in frankfurt_population_data['Average Age']]
frankfurt_population_data['Percentage 18-64'] = [x.replace(',', '.') for x in frankfurt_population_data['Percentage 18-64']]
frankfurt_population_data['Percentage Single Households'] = [x.replace(',', '.') for x in frankfurt_population_data['Percentage Si
ngle Households']]

frankfurt_population_data['Average Age'] = pd.to_numeric(frankfurt_population_data['Average Age'],errors='coerce')
frankfurt_population_data['Percentage 18-64'] = pd.to_numeric(frankfurt_population_data['Percentage 18-64'],errors='coerce')
frankfurt_population_data['Percentage Single Households'] = pd.to_numeric(frankfurt_population_data['Percentage Single Households'
],errors='coerce')
```

Merging the dataframes

In [20]:

```
frankfurt_district_data = frankfurt_density_data.merge(frankfurt_population_data)
```

In [21]:

```
frankfurt_district_data=frankfurt_district_data.merge(frankfurt_rental_data)
```

In [22]:

```
frankfurt_district_data=frankfurt_district_data.sort_values(by=['Borough'])
frankfurt_district_data=frankfurt_district_data.reset_index(drop=True)
```

In [23]:

```
frankfurt_district_data.head()
```

Out[23]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price
0	Altstadt	48.7	43.4	73.0	66.7	14.75
1	Bahnhofsviertel	59.3	37.5	85.6	71.6	14.75
2	Bergen-Enkheim	14.0	44.3	63.6	43.8	11.45
3	Berkersheim	11.5	38.9	60.8	36.4	12.20
4	Bockenheim	66.1	38.9	75.0	60.4	15.95

In [24]:

```
print('We have population data for {} boroughs'.format(len(frankfurt_district_data)))
```

```
We have population data for 45 boroughs
```

Getting the geospatial data for Frankfurt

In [26]:

```
!wget --quiet https://offenedaten.frankfurt.de/dataset/85b38876-729c-4a78-910c-a52d5c6df8d2/resource/84dff094-ab75-431f-8c64-39606
672f1da/download/ffmstadtteilewahlen.geojson -O frankfurt.json
frankfurt_geo = r'frankfurt.json' # geojson file
with open('frankfurt.json') as json_data:
    frankfurt_data = json.load(json_data)
Borough_data = frankfurt_data['features']
```

In [27]:

```
# define the dataframe columns
column_names = ['STTLNR', 'STTLNAME', 'STTLLAT','STTLLON']

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)
geolocator = Nominatim(user_agent="fra_explorer")
for data in Borough_data:
    address = 'Frankfurt, ' + data['properties']['STTLNAME'] + ', Germany'
    location = geolocator.geocode(address)
    lat = location.latitude
    long = location.longitude
    neighborhoods = neighborhoods.append({'STTLNR': data['properties']['STTLNR'],
                                          'STTLNAME': data['properties']['STTLNAME'],
                                          'STTLLAT': lat,'STTLLON':long }, ignore_index=True)
```

In [28]:

```
neighborhoods.rename(columns={'STTLNAME': 'Borough'}, inplace=True)
```

Separating 'Gutleut-/Bahnhofsviertel'

In [29]:

```
neighborhoods=neighborhoods.append({'STTLNR': 10, 'Borough':'Gutleutviertel', 'STTLLAT': 50.107193, 'STTLLON': 8.670254}, ignore_index=True)
neighborhoods=neighborhoods.append({'STTLNR': 10, 'Borough':'Bahnhofsviertel', 'STTLLAT': 50.107193, 'STTLLON': 8.670254}, ignore_index=True)
neighborhoods = neighborhoods[neighborhoods.Borough != 'Gutleut-/Bahnhofsviertel']
neighborhoods=neighborhoods.sort_values(by=['Borough'])
neighborhoods=neighborhoods.reset_index(drop=True)
```

Merging the population data, clusters and the geospatial data

In [32]:

```
ffm_data=frankfurt_district_data.merge(neighborhoods)
```

In [33]:

```
ffm_data.head()
```

Out[33]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON
0	Altstadt	48.7	43.4	73.0	66.7	14.75	1	50.110644	8.682092
1	Bahnhofsviertel	59.3	37.5	85.6	71.6	14.75	10	50.107193	8.670254
2	Bergen-Enkheim	14.0	44.3	63.6	43.8	11.45	46	50.139567	8.747393
3	Berkersheim	11.5	38.9	60.8	36.4	12.20	32	50.176219	8.697437
4	Bockenheim	66.1	38.9	75.0	60.4	15.95	12	50.120524	8.653046

Using the Foursquare API to explore the neighborhoods and segment them.

In [34]:

```
# @hidden_cell
CLIENT_ID = 'XAGIT2LAU3HQ0FOGYQJARFLZYMWVA0C1NTCJ0DZVS55ZX50H'
CLIENT_SECRET = 'SDCZOXVHJOU0RWWPH1NWF0C0IWVSKE2PXA0QPFGR1FBOBOKS'
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value
```

In [35]:

```
# We define a function to get the top 100 venues for each neighborhood within a radius of 500 m

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

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

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.forma
t(
            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_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                   'Venue Category']

    return(nearby_venues)
```

In [36]:

```
# We run the function getNearbyVenues

frankfurt_venues = getNearbyVenues(names=ffm_data['Borough'],
                                   latitudes=ffm_data['STTLLAT'],
                                   longitudes=ffm_data['STTLLON'])
print('{} venues were returned.'.format(frankfurt_venues.shape[0]))
```

```
1125 venues were returned.
```

In [37]:

```
frankfurt_venues.head()
```

Out[37]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Altstadt	50.110644	8.682092	SCHIRN Kunsthalle	50.110291	8.683542	Art Museum
1	Altstadt	50.110644	8.682092	Römerberg	50.110489	8.682131	Plaza
2	Altstadt	50.110644	8.682092	Weinterasse Rollanderhof	50.112473	8.682164	Wine Bar
3	Altstadt	50.110644	8.682092	Hoppenworth & Ploch	50.110891	8.683701	Café
4	Altstadt	50.110644	8.682092	Kleinmarkthalle	50.112778	8.682958	Market

In [ ]:

```
# We check how many venues were returned for each neighborhood

frankfurt_venues.groupby('Neighborhood').count()
```

In [38]:

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

```
There are 181 unique categories.
```

We count the venues of entertainment & nightlife venues "fun index"

In [39]:

```
entertainment = ['Comedy Club', 'Concert Hall', 'Indie Movie Theater', 'Laser Tag', 'Movie Theater', 'Opera House', 'Performing Ar
ts Venue', 'Rock Club', 'Theater']
nightlife = ['Bar', 'Beer Bar', 'Beer Garden', 'Cocktail Bar', 'Dive Bar', 'Hotel Bar', 'Lounge', 'Nightclub', 'Pub', 'Sports Bar'
, 'Whisky Bar', 'Wine Bar']
food = ['Apple Wine Pub', 'BBQ Joint', 'Bistro', 'Breakfast Spot', 'Burger Joint', 'Cafeteria', 'Café', 'Coffee Shop', 'Currywurst
Joint', 'Deli/Bodega',
        'Diner', 'Gastropub', 'Irish Pub', 'Juice Bar', 'Pizza Place', 'Salad Place', 'Soup Place', 'Steakhouse', 'Trattoria/Oster
ia']
options = entertainment + nightlife + food
# selecting rows based on condition
events_df1 = frankfurt_venues[frankfurt_venues['Venue Category'].isin(options)]
ev=events_df1.groupby('Neighborhood').count()['Venue']
data = ev.index
df_fun = pd.DataFrame(data)
fun=[]
for i in range(len(df_fun)):
    fun.append(ev[i])
df_fun['Fun Index'] = fun
df_fun.rename(columns={'Neighborhood': 'Borough'}, inplace=True)
```

We select the restaurants

In [64]:

```
frankfurt_restaurants = frankfurt_venues[frankfurt_venues['Venue Category'].str.contains('Restaurant')]
frankfurt_restaurants.reset_index(drop=True, inplace=True)
frankfurt_restaurants.head()
```

Out[64]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Altstadt	50.110644	8.682092	Superkato	50.111664	8.679153	Sushi Restaurant
1	Altstadt	50.110644	8.682092	Heimat – Essen und Weine	50.111125	8.678286	German Restaurant
2	Altstadt	50.110644	8.682092	Góc Phố	50.113509	8.681686	Vietnamese Restaurant
3	Altstadt	50.110644	8.682092	Questione Di Gusto	50.112424	8.682045	Italian Restaurant
4	Altstadt	50.110644	8.682092	Picknickbank	50.111534	8.678509	Moroccan Restaurant

In [78]:

```
frankfurt_restaurants_latlon=frankfurt_restaurants[['Venue Latitude', 'Venue Longitude']].to_numpy() # array of latitude and longi
tude of restaurants
frankfurt_restaurants_latlon_list=frankfurt_restaurants_latlon.tolist() # list coordinates of restaurants
```

We establish a list of french restaurants

In [203]:

```
French_restaurants = frankfurt_venues[frankfurt_venues['Venue Category'] == 'French Restaurant']
French_restaurants = French_restaurants.drop_duplicates(subset=['Venue'])
French_restaurants=French_restaurants.reset_index(drop=True)
French_restaurants.head()
```

Out[203]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Altstadt	50.110644	8.682092	Restaurant Français	50.110217	8.675702	French Restaurant
1	Bahnhofsviertel	50.107193	8.670254	The Legacy	50.105702	8.666243	French Restaurant
2	Bockenheim	50.120524	8.653046	Lafleur	50.121445	8.655832	French Restaurant
3	Sachsenhausen-Nord	50.107332	8.687672	Lobster	50.105224	8.687848	French Restaurant
4	Westend-Süd	50.117517	8.652180	Brasserie ici	50.114454	8.651004	French Restaurant

We add the "Fun Index" to the data_frame ffm_data

In [201]:

```
ffm_data=ffm_data.merge(df_fun, how='left')
ffm_data['Fun Index'].fillna(value=0, inplace=True) # NA values are replaced by 0
```

In [211]:

```
ffm_data.head()
```

Out[211]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index
0	Altstadt	48.7	43.4	73.0	66.7	14.75	1	50.110644	8.682092	35.0
1	Bahnhofsviertel	59.3	37.5	85.6	71.6	14.75	10	50.107193	8.670254	27.0
2	Bergen-Enkheim	14.0	44.3	63.6	43.8	11.45	46	50.139567	8.747393	2.0
3	Berkersheim	11.5	38.9	60.8	36.4	12.20	32	50.176219	8.697437	0.0
4	Bockenheim	66.1	38.9	75.0	60.4	15.95	12	50.120524	8.653046	17.0

Clustering the data

We use StandardScaler() to normalize our dataset.

In [212]:

```
ffm_cluster_data = ffm_data.loc[:,['Borough', 'Population Density', 'Average Age', 'Percentage 18-64', 'Percentage Single Househol
ds', 'Rental Price', 'Fun Index']]
X = ffm_cluster_data.values[:,1:]
X = np.nan_to_num(X)
cluster_dataset = StandardScaler().fit_transform(X)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/sklearn/utils/validation.py:595: DataConversionWarnin
g: Data with input dtype object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/sklearn/utils/validation.py:595: DataConversionWarnin
g: Data with input dtype object was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [214]:

```
num_clusters = 3

k_means = KMeans(init="k-means++", n_clusters=num_clusters, n_init=12)
k_means.fit(cluster_dataset)
labels = k_means.labels_
```

In [216]:

```
ffm_data["Labels"] = labels
```

Out[216]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index	Labels
0	Altstadt	48.7	43.4	73.0	66.7	14.75	1	50.110644	8.682092	35.0	1
1	Bahnhofsviertel	59.3	37.5	85.6	71.6	14.75	10	50.107193	8.670254	27.0	1
2	Bergen-Enkheim	14.0	44.3	63.6	43.8	11.45	46	50.139567	8.747393	2.0	0
3	Berkersheim	11.5	38.9	60.8	36.4	12.20	32	50.176219	8.697437	0.0	0
4	Bockenheim	66.1	38.9	75.0	60.4	15.95	12	50.120524	8.653046	17.0	1

Bar Chart of the clusters

In [218]:

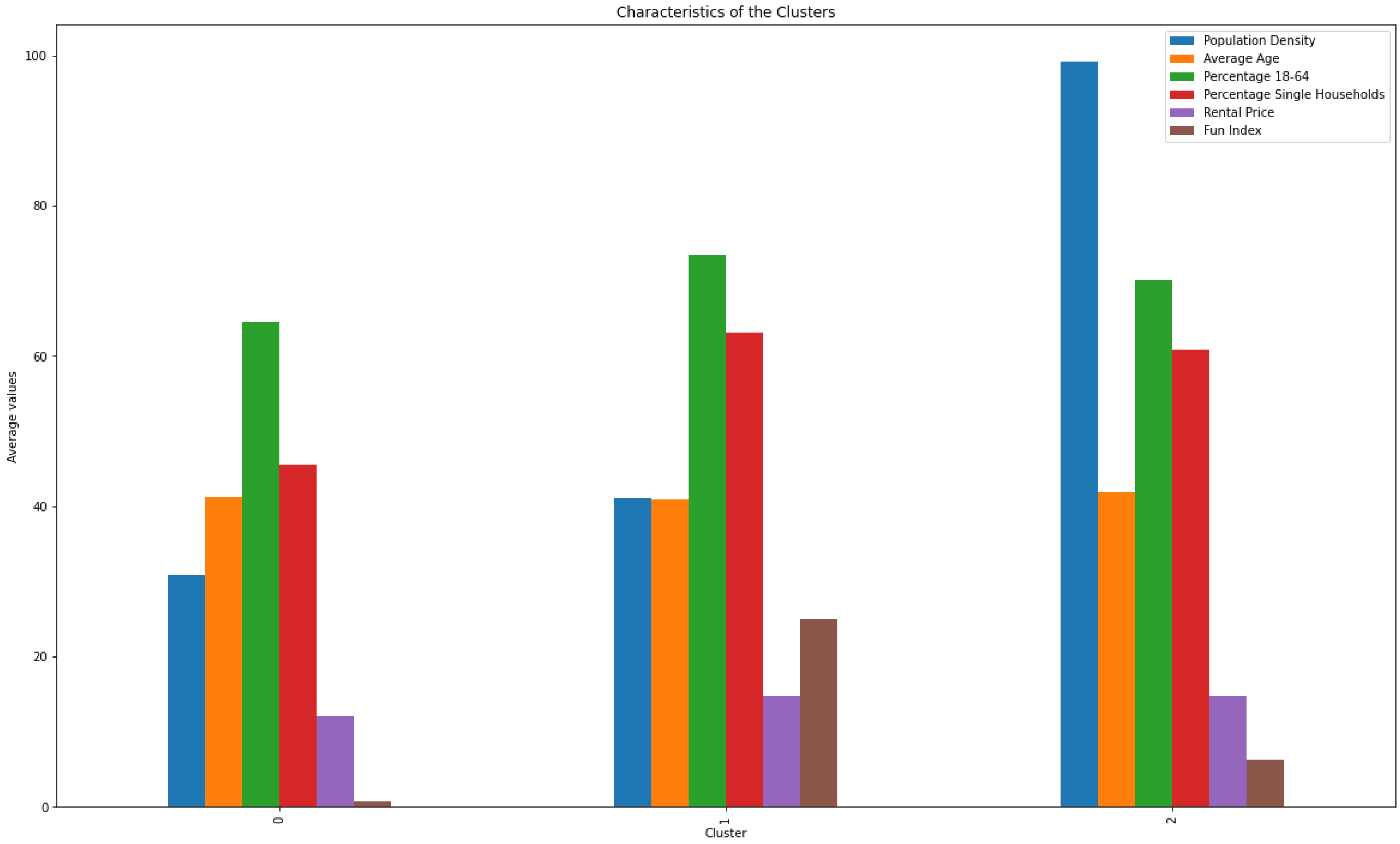
```
ff1=ffm_data.groupby('Labels').mean()
ff1=ff1.drop('STTLLAT', axis=1)
ff1=ff1.drop('STTLLON', axis=1)
ff1
```

Out[218]:

	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	Fun Index
Labels						
0	30.819052	41.293103	64.503448	45.606897	12.048276	0.758621
1	41.113500	40.890000	73.500000	63.070000	14.770000	25.000000
2	99.150000	41.916667	70.066667	60.900000	14.766667	6.333333

In [219]:

```
ff1.plot(kind='bar', figsize=(20,12))
plt.xlabel('Cluster')
plt.ylabel('Average values')
plt.title('Characteristics of the Clusters')
plt.show()
```


Characteristics of the Clusters
Population Density
Average Age
Percentage 18-64
Percentage Single Households
Rental Price
Fun Index
Average values
100
80
60
40
20
0
0
1
2
Cluster

Interpretation

Visualizing the results

In [84]:

```
address = 'Frankfurt, Germany'

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

```
The geograpical coordinates of Frankfurt are 50.1106444, 8.6820917.
```

In [221]:

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

# set color scheme for the clusters
x = np.arange(num_clusters)
ys = [i + x + (i*x)**2 for i in range(num_clusters)]
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(ffm_data['STTLLAT'], ffm_data['STTLLON'], ffm_data['Borough'], ffm_data['Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

# add the French restaurants as markers
for lat, lng, label in zip(French_restaurants['Venue Latitude'], French_restaurants['Venue Longitude'], French_restaurants['Venue'
]):
    folium.Marker(
        [lat, lng],
        popup=label).add_to(map_clusters)

map_clusters
```

```
Out[221]:
```

Make this Notebook Trusted to load map: File -> Trust Notebook

In [222]:

```
ffm_data.loc[ffm_data['Labels'] == 0]
```

Out[222]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index	Labels
2	Bergen-Enkheim	14.0000	44.3	63.6	43.8	11.45	46	50.139567	8.747393	2.0	0
3	Berkersheim	11.5000	38.9	60.8	36.4	12.20	32	50.176219	8.697437	0.0	0
5	Bonames	40.5000	43.1	63.4	44.7	12.20	31	50.181347	8.663331	2.0	0
8	Eckenheim	42.2175	41.9	65.9	51.2	12.20	29	50.145077	8.689725	0.0	0
9	Eschersheim	43.1000	42.4	65.5	53.8	13.60	28	50.158438	8.655319	0.0	0
10	Fechenheim	22.2000	39.7	64.8	47.8	12.00	35	50.125715	8.750796	1.0	0
11	Frankfurter Berg	42.2175	39.0	65.9	34.2	12.20	47	50.170230	8.676782	1.0	0
13	Ginnheim	45.9000	40.4	63.9	46.5	13.60	26	50.141706	8.643167	0.0	0
14	Griesheim	47.2000	40.0	70.1	53.0	11.50	19	50.101258	8.606512	1.0	0
16	Harheim	8.6000	42.4	62.7	38.4	11.45	44	50.187171	8.700933	0.0	0
17	Hausen	57.1000	40.8	66.0	45.4	12.25	21	50.134767	8.619715	0.0	0
18	Heddernheim	67.3000	42.1	63.5	45.5	12.85	24	50.161636	8.636703	2.0	0
19	Höchst	29.9000	37.0	70.7	50.7	11.50	36	50.098506	8.528433	1.0	0
21	Kalbach-Riedberg	17.4000	35.9	66.7	29.5	11.45	43	50.186279	8.639055	0.0	0
22	Nied	47.9000	41.1	65.9	46.7	11.50	37	50.102322	8.571419	2.0	0
23	Nieder-Erlenbach	5.5000	42.5	62.7	38.1	11.45	42	50.200454	8.710470	2.0	0
24	Nieder-Eschbach	17.9000	42.5	63.4	43.2	11.45	45	50.207230	8.657373	0.0	0
26	Niederursel	20.9000	42.9	60.6	44.5	12.85	25	50.158750	8.622585	0.0	0
29	Oberrad	45.9000	42.8	66.7	54.0	13.50	16	50.058148	8.727985	1.0	0
31	Praunheim	34.5000	43.2	62.1	46.8	12.25	22	50.158750	8.622585	0.0	0
32	Preungesheim	37.3000	38.7	64.8	44.8	12.20	30	50.159900	8.686182	0.0	0

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index	Labels
33	Riederwald	42.2175	42.6	65.7	55.9	12.00	33	50.135842	8.735708	2.0	0
34	Rödelheim	33.1000	41.8	67.3	55.5	12.25	20	50.124411	8.607307	1.0	0
37	Schwanheim	11.5000	42.4	61.0	42.8	11.50	18	50.068214	8.633047	2.0	0
38	Seckbach	12.4000	44.4	62.5	49.3	12.00	34	50.144764	8.709590	0.0	0
39	Sindlingen	17.2000	41.4	64.1	46.7	11.50	38	50.087538	8.512303	1.0	0
40	Sossenheim	25.6000	40.7	64.2	44.6	11.50	41	50.115937	8.555881	0.0	0
41	Unterliederbach	25.0000	40.6	64.7	47.0	11.50	40	50.101248	8.527213	1.0	0
44	Zeilsheim	27.7000	42.0	61.4	41.8	11.50	39	50.090283	8.506643	0.0	0

In [223]:

```
ffm_data.loc[ffm_data['Labels'] == 1]
```

Out[223]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index	Labels
0	Altstadt	48.7000	43.4	73.0	66.7	14.75	1	50.110644	8.682092	35.0	1
1	Bahnhofsviertel	59.3000	37.5	85.6	71.6	14.75	10	50.107193	8.670254	27.0	1
4	Bockenheim	66.1000	38.9	75.0	60.4	15.95	12	50.120524	8.653046	17.0	1
12	Gallus	31.0000	38.6	73.7	60.0	14.75	11	50.106654	8.662581	26.0	1
15	Gutleutviertel	42.2175	41.1	74.0	66.1	14.75	10	50.107193	8.670254	27.0	1
20	Innenstadt	42.2175	41.6	76.5	71.3	14.75	2	50.106654	8.662581	26.0	1
30	Ostend	48.5000	42.5	71.9	62.6	15.05	8	50.115651	8.701897	14.0	1
35	Sachsenhausen-Nord	9.4000	40.6	71.5	60.4	13.50	13	50.107332	8.687672	31.0	1
36	Sachsenhausen-Süd	9.4000	44.7	64.8	56.3	13.50	14	50.107332	8.687672	31.0	1
42	Westend-Nord	54.3000	40.0	69.0	55.3	15.95	5	50.120988	8.673486	16.0	1

In [224]:

```
ffm_data.loc[ffm_data['Labels'] == 2]
```

Out[224]:

	Borough	Population Density	Average Age	Percentage 18-64	Percentage Single Households	Rental Price	STTLNR	STTLLAT	STTLLON	Fun Index	Labels
6	Bornheim	120.7	43.2	69.0	62.0	15.05	9	50.115651	8.701897	14.0	2
7	Dornbusch	84.5	44.0	63.5	55.1	13.60	27	50.135764	8.672073	6.0	2
25	Niederrad	76.5	41.7	69.7	57.5	13.50	17	50.080774	8.637245	1.0	2
27	Nordend-Ost	150.6	40.7	75.2	65.4	15.25	7	50.133655	8.699082	1.0	2
28	Nordend-West	92.0	41.2	72.5	63.0	15.25	6	50.132620	8.680232	3.0	2
43	Westend-Süd	70.6	40.7	70.5	62.4	15.95	4	50.117517	8.652180	13.0	2

Adding a heat map with the density of restaurants

In [226]:

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

# set color scheme for the clusters
x = np.arange(num_clusters)
ys = [i + x + (i*x)**2 for i in range(num_clusters)]
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(ffm_data['STTLLAT'], ffm_data['STTLLON'], ffm_data['Borough'], ffm_data['Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

HeatMap(frankfurt_restaurants_latlon_list).add_to(folium.FeatureGroup(name='Heat Map').add_to(map_clusters))
folium.LayerControl().add_to(map_clusters)
map_clusters
```

```
Out[226]:
```

Make this Notebook Trusted to load map: File -> Trust Notebook

```
In [ ]:
```

In [ ]:

```
```