Best Location to open an Italian Restaurant in San Francisco

Capstone Proyect - The Battle of Neighborhoods (Week 2)

Table of contents

Introduction: Business Problem

In this project we will try to find an optimal location for a restaurant. Specifically, this report will be targeted to stakeholders and investors interested in opening an **Italian restaurant** in **San Francisco**, California.

Since there are lots of restaurants in San Francisco we will try to detect **locations that are not already crowded with restaurants**. We are also particularly interested in **areas with no Italian restaurants in vicinity**. We would also prefer locations **as close to city center as possible**,and with **low crime rate**.

We will use our data science powers to generate a few most promissing neighborhoods based on this criteria. Advantages of each area will then be clearly expressed so that best possible final location can be chosen by stakeholders.

Data

Based on definition of our problem, factors that will influence our decission are:

- number of existing restaurants in the neighborhood (any type of restaurant)
- number of and distance to Italian restaurants in the neighborhood, if any
- distance of neighborhood from city center
- zone crime rate

We decided to use regularly spaced grid of locations, centered around city center, to define our neighborhoods.

Following data sources will be needed to extract/generate the required information:

- centers of candidate areas will be generated algorithmically and approximate addresses of centers of those areas will be obtained using **Google Maps API reverse geocoding**
- number of restaurants and their type and location in every neighborhood will be obtained using **Foursquare API**
- coordinate of Berlin center will be obtained using **Google Maps API geocoding** of well known San Francisco location (Union Square)

Neighborhood Candidates

Let's create latitude and longitude coordinates for centroids of our candidate neighborhoods. We will create a gro of cells covering our area of interest which is aprox 12 x 12 kilometers centered around San Francisco city center.

Let's first find a latitude and longitude of San Francisco city center, usign specific, well know address and Google Maps geocoding API

In [190]:
```
import requests
import googlemaps

google_api_key = 'AIzaSyCJx-7G2i18Lk583v9r-km8TsEgsfw2UR4'
api_key        = 'AIzaSyCJx-7G2i18Lk583v9r-km8TsEgsfw2UR4'

def get_coordinates(api_key, address, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&address=
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        geographical_data = results[0]['geometry']['location'] # get geographica
        lat = geographical_data['lat']
        lon = geographical_data['lng']
        return [lat, lon]
    except:
        return [None, None]

address = 'Union Square, San Francisco, California'
sf_center = get_coordinates(google_api_key, address) # [LAT, LON]
print('Coordinate of {}: {}'.format(address, sf_center))
```

```
Coordinate of Union Square, San Francisco, California: [37.7879938, -122.407437
4]
```

In [191]:
```
sf_center
```

Out[191]:
```
[37.7879938, -122.4074374]
```

Now let's create a grid of area candidates, equaly spaced, centered around city center and within 4 km from Union Square. Our neighborhoods will de defined as circular areas with a radius of 300 meters, so our neighborhood center will be 600 meters apart.

To accurate calculate distances we need to create our grid of locations in Cartesian 2D coordinate system wich allow us to calculate distances in meters (not in latitude/longitude degrees). Then we will project those coordinates back to latitude/longitude degrees to be shown on Folium map. So let's create funcitons to convert between WGS84 spherical coordinate sytems (latitude/longitude degrees) and UTM Cartesian coordinate system (X/Y coordinates in meters)

In [192]:
```
pip install shapely
```

```
Requirement already satisfied: shapely in c:\programdata\anaconda3\lib\site-pac
kages (1.7.0)
Note: you may need to restart the kernel to use updated packages.
```

In [193]:
```
import shapely.geometry

#!pip install pyproj
import pyproj

import math

def lonlat_to_xy(lon, lat):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    xy = pyproj.transform(proj_latlon, proj_xy, lon, lat)
    return xy[0], xy[1]

def xy_to_lonlat(x, y):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    lonlat = pyproj.transform(proj_xy, proj_latlon, x, y)
    return lonlat[0], lonlat[1]

def calc_xy_distance(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    return math.sqrt(dx*dx + dy*dy)

print('Coordinate transformation check')
print('-------------------------------')
print('San Francisco center longitude = {}, latitude = {}'.format(sf_center[1], s
x, y = lonlat_to_xy(sf_center[1], sf_center[0])
print('San Francisco center UTM X = {}, Y = {}'.format(x, y))
lo, la = xy_to_lonlat(x, y)
print('San Francisco center longitude = {}, latitude = {}'.format(lo, la))
```

```
Coordinate transformation check
-------------------------------
San Francisco center longitude = -122.4074374, latitude = 37.7879938
San Francisco center UTM X = -3310893.21088001, Y = 14843808.790182108
San Francisco center longitude = -122.40743740000003, latitude = 37.78799380000
002
```

Let's create a **hexagonal grid of cells**: we offset every other row, and adjust vertical row spacing so that **every cell center is equally distant from all it's neighbors**.

In [194]:
```
# City center in Cartesian coordinates
sf_center_x, sf_center_y = lonlat_to_xy(sf_center[1],sf_center[0])

k = math.sqrt(3) / 2 #Vertical offset for hexagonal grid cells
x_min = sf_center_x - 4000
x_step = 720
y_min = sf_center_y - 4000 - (int(21/k)*k*600-8000)/1
y_step = 720 * k

latitudes = []
longitudes = []
distances_from_center = []
xs = []
ys = []
for i in range(0, int(21/k)):
    y = y_min + i * y_step
    x_offset = 300 if i%2==0 else 0
    for j in range(0, 21):
        x = x_min + j * x_step + x_offset
        distance_from_center = calc_xy_distance(sf_center_x, sf_center_y, x, y)
        if (distance_from_center <= 4000):
            lon, lat = xy_to_lonlat(x, y)
            latitudes.append(lat)
            longitudes.append(lon)
            distances_from_center.append(distance_from_center)
            xs.append(x)
            ys.append(y)

print(len(latitudes), 'candidate neighborhood centers generated.')
#-----------------------------------------------------------------------------
```

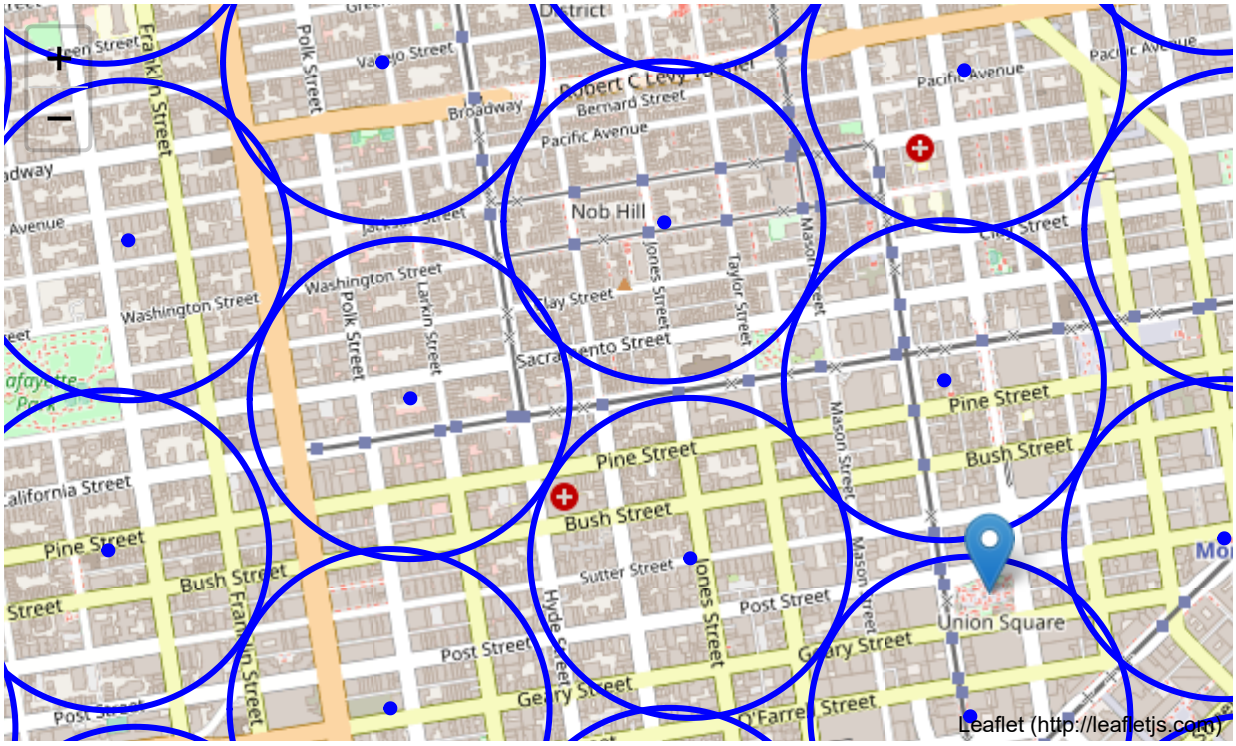
```
110 candidate neighborhood centers generated.
```

Let's visualize the data we have so far city center location and candidate neighborhood center

In [195]:
```
import folium
```

In [196]:
```
map_sf = folium.Map(location=sf_center, zoom_start=15)
folium.Marker(sf_center, popup='Union Square').add_to(map_sf)
for lat, lon in zip(latitudes, longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
    folium.Circle([lat, lon], radius=300, color='blue', fill=False).add_to(map_s
    #folium.Marker([lat, lon]).add_to(map_berlin)
map_sf
```

Out[196]:



We now have the coordinates of centers of neighborhoods areas to be evaluated, equally spaced (distace from every poin to it's neighborhood is exactly tue same) and within 4 km from Union Square.

Let's use Google Maps API to get approximate addresses of those locations.

In [197]:
```
def get_address(api_key, latitude, longitude, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&latlng={}
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        address = results[0]['formatted_address']
        return address
    except:
        return None

google_api_key = 'AIzaSyCJx-7G2i18Lk583v9r-km8TsEgsfw2UR4'
api_key = 'AIzaSyCJx-7G2i18Lk583v9r-km8TsEgsfw2UR4'
addr = get_address(google_api_key, sf_center[0], sf_center[1])
print('Reverse geocoding check')
print('-----------------------')
print('Address of [{}, {}] is: {}'.format(sf_center[0], sf_center[1], addr))
```

```
Reverse geocoding check
-----------------------
Address of [37.7879938, -122.4074374] is: 301 Post St, San Francisco, CA 94108,
USA
```

In [198]:
```
print('Obtaining location addresses: ', end='')
addresses = []
for lat, lon in zip(latitudes, longitudes):
    address = get_address(google_api_key, lat, lon)
    if address is None:
        address = 'NO ADDRESS'
    address = address.replace(', USA', '') # We don't need country part of addres
    addresses.append(address)
    print(' .', end='')
print(' done.')
```

```
Obtaining location addresses:  . . . . . .
. . . . . .
. . . . . .
. . . . . done.
```

Looking good, Let's now place all this into a Pandas dataframe

In [199]:
```
import pandas as pd

df_locations = pd.DataFrame({'Address': addresses,
                             'Latitude': latitudes,
                             'Longitude': longitudes,
                             'X': xs,
                             'Y': ys,
                             'Distance from center': distances_from_center})

df_locations.head(10)
```

Out[199]:

	Address	Latitude	Longitude	X	Y	Distance from center
0	San Francisco, CA	37.805313	-122.378149	-3.312433e+06	1.484033e+07	3807.771539
1	San Francisco, CA	37.808009	-122.384163	-3.311713e+06	1.484033e+07	3577.698155
2	San Francisco, CA	37.810704	-122.390177	-3.310993e+06	1.484033e+07	3483.894960
3	San Francisco, CA	37.813400	-122.396193	-3.310273e+06	1.484033e+07	3537.219825
4	San Francisco, CA	37.816095	-122.402209	-3.309553e+06	1.484033e+07	3731.370270
5	San Francisco, CA	37.797362	-122.372573	-3.313453e+06	1.484095e+07	3837.581324
6	San Francisco – Oakland Bay Bridge, San Franci...	37.800057	-122.378585	-3.312733e+06	1.484095e+07	3399.857411
7	San Francisco, CA	37.802753	-122.384599	-3.312013e+06	1.484095e+07	3070.477229
8	San Francisco, CA	37.805448	-122.390613	-3.311293e+06	1.484095e+07	2886.768161
9	San Francisco, CA	37.808144	-122.396628	-3.310573e+06	1.484095e+07	2876.774307

In [200]:
```
df_locations.shape
```

Out[200]:
```
(110, 6)
```

and let's now save/persist this data into local file.

In [201]:
```
df_locations.to_pickle('./locations.pkl')
```

Foursquare

Now that we have our location candidates, let's use Foursquare API to get info on restaurants in each neighborhood.

We're interested in venues in 'food' category, but only those that are proper restaurants - coffe shops, pizza places, bakeries etc. are not direct competitors so we don't care about those. So we will include in out list only venues that have 'restaurant' in category name, and we'll make sure to detect and include all the subcategories of specific 'Italian restaurant' category, as we need info on Italian restaurants in the neighborhood.

In [202]:
```
# The code was removed by Watson Studio for sharing.
CLIENT_ID = 'COZN2ICX1YF0PCQ5MS5E5OBRXV2BMTF0BBFX3AKU4M5BWLR3' # Foursquare ID
CLIENT_SECRET = 'NHQMSPEPYOJTD5M2X50EXSU4PRM5BN3N2OE3HFXPUNO15HVV' # Foursquare S
VERSION = '201800604' #in the lab was 20180605
print('Your credentials:')
print('CLIENT_ID: '+ CLIENT_ID)
print('CLIENT_SECRET: '+ CLIENT_SECRET)
```

```
Your credentials:
CLIENT_ID: COZN2ICX1YF0PCQ5MS5E5OBRXV2BMTF0BBFX3AKU4M5BWLR3
CLIENT_SECRET: NHQMSPEPYOJTD5M2X50EXSU4PRM5BN3N2OE3HFXPUNO15HVV
```

In [203]:
```
# Category IDs corresponding to Italian restaurants were taken from Foursquare we

food_category = '4d4b7105d754a06374d81259' # 'Root' category for all food-relate

italian_restaurant_categories = ['4bf58dd8d48988d110941735','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cba7','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cb95','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cb98','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cbb0','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cbaa','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cb92','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cbb9','55a5a1ebe4b01390908
                                 '55a5a1ebe4b013909087cb9e','55a5a1ebe4b01390908

def is_restaurant(categories, specific_filter=None):
    restaurant_words = ['restaurant', 'diner', 'taverna', 'steakhouse']
    restaurant = False
    specific = False
    for c in categories:
        category_name = c[0].lower()
        category_id = c[1]
        for r in restaurant_words:
            if r in category_name:
                restaurant = True
        if 'fast food' in category_name:
            restaurant = False
        if not(specific_filter is None) and (category_id in specific_filter):
            specific = True
            restaurant = True
    return restaurant, specific

def get_categories(categories):
    return [(cat['name'], cat['id']) for cat in categories]

def format_address(location):
    address = ', '.join(location['formattedAddress'])
    address = address.replace(', Deutschland', '')
    address = address.replace(', Germany', '')
    return address

def get_venues_near_location(lat, lon, category, client_id, client_secret, radius
    version = '20180724'
    url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secre
        client_id, client_secret, version, lat, lon, category, radius, limit)
    try:
        results = requests.get(url).json()['response']['groups'][0]['items']
        venues = [(item['venue']['id'],
                   item['venue']['name'],
                   get_categories(item['venue']['categories']),
                   (item['venue']['location']['lat'], item['venue']['location'][
                   format_address(item['venue']['location']),
                   item['venue']['location']['distance']) for item in results]
    except:
        venues = []
    return venues
```

In [204]:
```
# Let's now go over our neighborhood locations and get nearby restaurants; we'll
foursquare_client_id = 'COZN2ICX1YF0PCQ5MS5E5OBRXV2BMTF0BBFX3AKU4M5BWLR3' # Fours
foursquare_client_secret = 'NHQMSPEPYOJTD5M2X50EXSU4PRM5BN3N2OE3HFXPUNO15HVV' # 
VERSION = '201800604' #in the lab was 20180605
import pickle

def get_restaurants(lats, lons):
    restaurants = {}
    italian_restaurants = {}
    location_restaurants = []

    print('Obtaining venues around candidate locations:', end='')
    for lat, lon in zip(lats, lons):
        # Using radius=350 to meke sure we have overlaps/full coverage so we don
        venues = get_venues_near_location(lat, lon, food_category, foursquare_cl
        area_restaurants = []
        for venue in venues:
            venue_id = venue[0]
            venue_name = venue[1]
            venue_categories = venue[2]
            venue_latlon = venue[3]
            venue_address = venue[4]
            venue_distance = venue[5]
            is_res, is_italian = is_restaurant(venue_categories, specific_filter
            if is_res:
                x, y = lonlat_to_xy(venue_latlon[1], venue_latlon[0])
                restaurant = (venue_id, venue_name, venue_latlon[0], venue_latlo
                if venue_distance<=300:
                    area_restaurants.append(restaurant)
                restaurants[venue_id] = restaurant
                if is_italian:
                    italian_restaurants[venue_id] = restaurant
        location_restaurants.append(area_restaurants)
        print(' .', end='')
    print(' done.')
    return restaurants, italian_restaurants, location_restaurants

# Try to load from local file system in case we did this before
restaurants = {}
italian_restaurants = {}
location_restaurants = []
loaded = False
try:
    with open('restaurants_350.pkl', 'rb') as f:
        restaurants = pickle.load(f)
    with open('italian_restaurants_350.pkl', 'rb') as f:
        italian_restaurants = pickle.load(f)
    with open('location_restaurants_350.pkl', 'rb') as f:
        location_restaurants = pickle.load(f)
    print('Restaurant data loaded.')
    loaded = True
except:
    pass

# If load failed use the Foursquare API to get the data
if not loaded:
```

```
    restaurants, italian_restaurants, location_restaurants = get_restaurants(lat

    # Let's persists this in local file system
    with open('restaurants_350.pkl', 'wb') as f:
        pickle.dump(restaurants, f)
    with open('italian_restaurants_350.pkl', 'wb') as f:
        pickle.dump(italian_restaurants, f)
    with open('location_restaurants_350.pkl', 'wb') as f:
        pickle.dump(location_restaurants, f)
```

Restaurant data loaded.

In [205]:
```
import numpy as np

print('Total number of restaurants:', len(restaurants))
print('Total number of Italian restaurants:', len(italian_restaurants))
print('Percentage of Italian restaurants: {:.2f}%'.format(len(italian_restaurant
print('Average number of restaurants in neighborhood:', np.array([len(r) for r i
```

```
Total number of restaurants: 989
Total number of Italian restaurants: 79
Percentage of Italian restaurants: 7.99%
Average number of restaurants in neighborhood: 7.6454545454545455
```

In [206]:
```
print('List of all restaurants')
print('----------------------')
for r in list(restaurants.values())[:10]:
    print(r)
print('...')
print('Total:', len(restaurants))
```

```
List of all restaurants
----------------------
('4a975e5cf964a5207e2920e3', 'Bubba Gump Shrimp Co.', 37.8110504, -122.4104181,
'39 Piers Box M-211, San Francisco, CA 94133, United States', 329, False, -3309
134.1087442287, 14841323.935208522)
('56bfb134cd10f62cdec858b1', 'Chart House', 37.8109343, -122.4108872, 'Pier 39,
Space 206-8, Building K, San Francisco, CA 94133, United States', 287, False, -
3309099.0362220122, 14841361.2437554)
('4d9baddf2e6d8cfa87742c17', 'Hana Zen Sushi & Yakitori Bar', 37.8111402572289
7, -122.41065040036543, 'Pier 39, M209 (Beach & Embarcadero), San Francisco, CA
94133, United States', 311, False, -3309107.226987691, 14841325.556576401)
('4a10af0af964a520f4761fe3', 'Waterfront Restaurant', 37.799346, -122.395618,
'7 The Embarcadero, San Francisco, CA 94111, United States', 252, False, -33112
33.169987426, 14841904.446479393)
('51575e96e4b0fa0f8b310fe8', 'Seaglass Restaurant', 37.80214747492143, -122.396
87482007305, 'Exploratorium Pier 15/17 (Embarcadero), San Francisco, CA, United
States', 78, False, -3310938.1472279127, 14841648.392604528)
('41044980f964a520700b1fe3', 'Hillstone', 37.80612149051469, -122.404939131139,
'1800 Montgomery St (btwn Bay St & Francisco St), San Francisco, CA 94111, Unit
ed States', 336, False, -3309949.5859764493, 14841607.038326712)
('3fd66200f964a5205bec1ee3', 'Pier 23 Cafe', 37.80335180922726, -122.4008559469
9735, 'Pier 23 (btwn Battery St & Green St), San Francisco, CA 94111, United St
ates', 225, False, -3310499.0285219317, 14841714.642242081)
('523bc00411d20b49a72fc444', 'Fog City', 37.803538902224695, -122.4018813335202
7, '1300 Battery St (The Embarcadero), San Francisco, CA 94111, United States',
178, False, -3310393.8808595957, 14841745.782243771)
('45e9666df964a5207c431fe3', 'Il Fornaio', 37.80264398142964, -122.402048427041
62, '1265 Battery St (btwn Filbert & Greenwich St), San Francisco, CA 94111, Un
ited States', 310, True, -3310436.507748886, 14841856.713561349)
('4bb78e967421a59307a6c040', 'Fog Harbor Fish House', 37.808903545091106, -122.
41037933382874, 'Pier 39 (at The Embarcadero), San Francisco, CA 94133, United
States', 234, False, -3309276.2384019014, 14841567.510787517)
...
Total: 989
```

In [207]:
```
print('List of Italian restaurants')
print('---------------------------')
for r in list(italian_restaurants.values())[:10]:
    print(r)
print('...')
print('Total:', len(italian_restaurants))
```

```
List of Italian restaurants
---------------------------
('45e9666df964a5207c431fe3', 'Il Fornaio', 37.80264398142964, -122.40204842704162, '1265 Battery St (btwn Filbert & Greenwich St), San Francisco, CA 94111, United States', 310, True, -3310436.507748886, 14841856.713561349)
('4ba53d01f964a5201df038e3', 'Swiss Louis', 37.810184493448745, -122.41076485542048, 'Pier 39, San Francisco, CA 94133, United States', 296, True, -3309158.5494127274, 14841440.740662636)
('4c1d815feac020a12f6b48c2', 'Louis Italian', 37.81031583, -122.41054236888885, 'San Francisco, CA 94133, United States', 314, True, -3309170.2596488995, 14841414.321621116)
('531f8334498e87dbcc7bb241', "alioto's San Francisco", 37.808685302734375, -122.41633605957031, 'San Francisco, CA, United States', 283, True, -3308749.7534648296, 14841897.571628133)
('51e06fd6498ee38914f05ec0', 'Bouli Bar', 37.79510408225698, -122.39305800788188, '1 Ferry Building #35, San Francisco, CA 94111, United States', 236, True, -3311739.540721074, 14842258.495830642)
('4cd0a38e5b5ca35de6317df0', 'Cotogna', 37.797345664268256, -122.40362430843288, '490 Pacific Ave (at Montgomery St), San Francisco, CA 94133, United States', 298, True, -3310635.6014788197, 14842543.551379733)
('49e05c38f964a52052611fe3', "Tommaso's", 37.797806, -122.405316, '1042 Kearny St (at Broadway), San Francisco, CA 94133, United States', 254, True, -3310452.316044035, 14842577.565610107)
('53f914d8498e1ef0fe04474c', 'The Italian Homemade Company', 37.80149747181165, -122.41179456578453, '716 Columbus Ave, San Francisco, CA 94133, United States', 288, True, -3309625.930131172, 14842487.168225084)
('4b24598cf964a5204a6624e3', 'Baonecci Ristorante', 37.79969698045158, -122.4077831292636, '516 Green St (at Grant Ave), San Francisco, CA 94133, United States', 315, True, -3310106.278841428, 14842487.642252445)
('4a10d76bf964a52003771fe3', 'Caffe Sport', 37.79974610518378, -122.40863641330579, '574 Green St (Columbus Avenue), San Francisco, CA 94133, United States', 326, True, -3310025.654773329, 14842525.735498734)
...
Total: 79
```

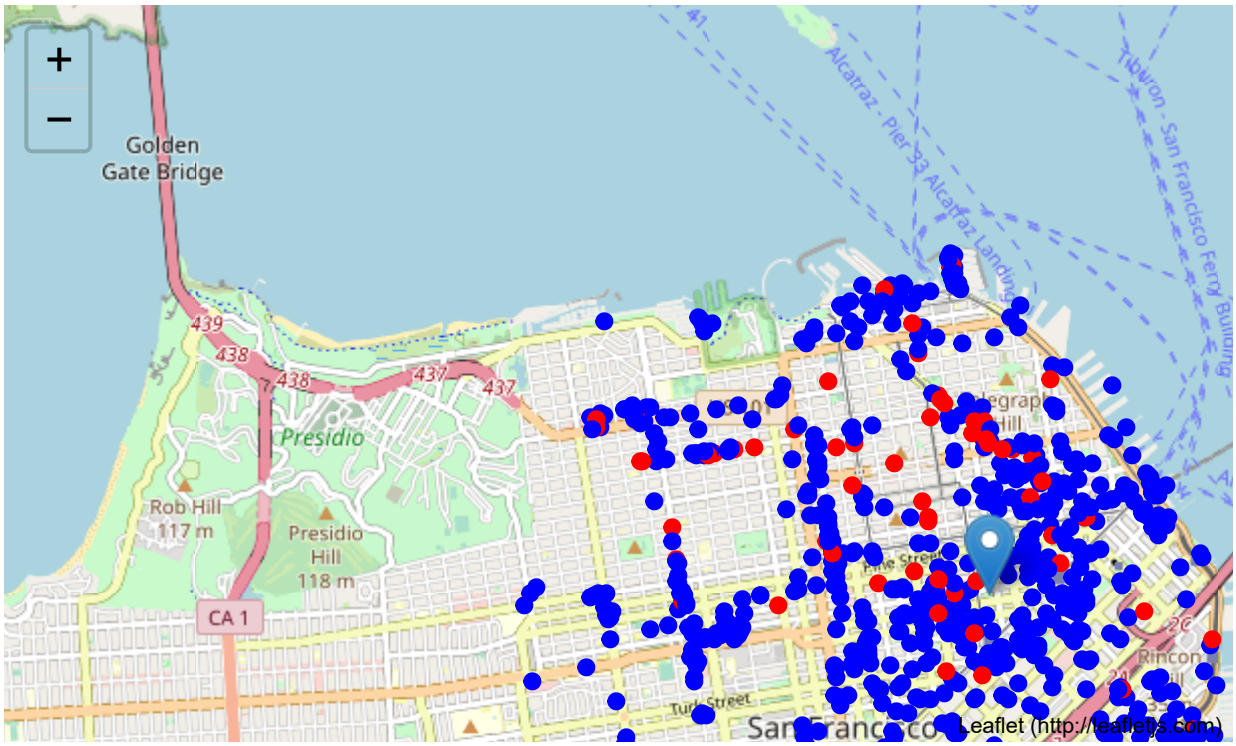
In [208]:
```
print('Restaurants around location')
print('--------------------------')
for i in range(100, 110):
    rs = location_restaurants[i][:8]
    names = ', '.join([r[1] for r in rs])
    print('Restaurants around location {}: {}'.format(i+1, names))
```

```
Restaurants around location
--------------------------
Restaurants around location 101: Kantine, Sushi Zone, Izakaya Roku, Destino Lat
in Bistro/Pisco Bar, Kibatsu, Two Jacks Seafood, Sushi Delight
Restaurants around location 102: Alamo Square Seafood Grill, Nara, Yh Beijing
Restaurants around location 103: jū-ni, Brenda's Meat & Three, Bar Crudo, Tsuna
mi, Kung Food, Che Fico, barvale, Nopa
Restaurants around location 104:
Restaurants around location 105: Central Kitchen, Pho On Bryant, Flour+Water Pa
sta Shop
Restaurants around location 106: Farmhouse Kitchen, flour + water, Gallardo's,
Evergreen Garden Restaurant, El Tepa Taqueria, El Faro, Chuy's Fiestas Restaura
nt, Las Cocineras Restaurant
Restaurants around location 107: Pancho Villa Taqueria, Panchita's Pupuseria Re
staurant #2, Limón Rotisserie, Truly Mediterranean, La Oaxaqueña Bakery & Resta
urant, Kitava, Puerto Alegre, RAW Sugar Factory
Restaurants around location 108: Woodhouse Fish Co., Pakwan, Jasmine Garden, Ra
men Izakaya Goku, Lers Ros Thai, Warakubune Sushi Restaurant, KRUA THAI, Red Ja
de
Restaurants around location 109: El Castillito, Beit Rima, Palmyra, Uva Enotec
a, Iza Ramen, Purple Rice, Thep Phanom, Hot Zushi
Restaurants around location 110: RT Rotisserie, Nopalito, Indian Paradox, Namu
Stonepot, Club Waziema, Mangrove Thai Kitchen, Phuket Thai, wazima
```

Lets' now see all the colelcted restaurants in our area if interest on map, and let0s also show Italian restaurants in differente color.

In [209]:
```
map_sf = folium.Map(location=sf_center, zoom_start=13)
folium.Marker(sf_center, popup='Union Square').add_to(map_sf)
for res in restaurants.values():
    lat = res[2]; lon = res[3]
    is_italian = res[6]
    color = 'red' if is_italian else 'blue'
    folium.CircleMarker([lat, lon], radius=3, color=color, fill=True, fill_color=
map_sf
```

Out[209]:



Looking good. So now we have all the restaurats in area within few kilometers from Union Square, and we know whoch ones area Italian restaurants, we also know which restaurants exactly are in vicinity if every neighborhood candidate center.

This conclude the data gathering phase, we ara now ready to use this data for analysis to produce the report on optimal locations for a new italian restaurant.

Methodology

In this project we will direct our efforts on detecting areas of San Francisco that have low restaurant density, particularly those with low number of Italian restaurants. We will limit our analysis to area ~4km around city center.

In first step we have collected the required **data: location and type (category) of every restaurant within 4km from San Francisco center** (Union Square). We have also **identified Italian restaurants** (according to Foursquare categorization).

Second step in our analysis will be calculation and exploration of '**restaurant density**' across different areas of San Francisco - we will use **heatmaps** to identify a few promising areas close to center with low number of restaurants in general (*and* no Italian restaurants in vicinity) and focus our attention on those areas.

In third and final step we will focus on most promising areas and within those create **clusters of locations that meet some basic requirements** established in discussion with stakeholders: we will take into consideration locations with **no more than two restaurants in radius of 250 meters**, and we want locations **without Italian restaurants in radius of 400 meters**. We will present map of all such locations but also create clusters (using **k-means clustering**) of those locations to identify general zones / neighborhoods / addresses which should be a starting point for final 'street level' exploration and search for optimal venue location by stakeholders.

Analysis

Let's perform some basic explanatory data analysis and derive some additional into from our raw data. First let's count the **number of restaurants in every area candidate**:

In [210]:
```
location_restaurants_count = [len(res) for res in location_restaurants]

df_locations['Restaurants in area'] = location_restaurants_count

print('Average number of restaurants in every area with radius=300m:', np.array(

df_locations.head(10)
```

```
Average number of restaurants in every area with radius=300m: 7.645454545454545
5
```

Out[210]:

	Address	Latitude	Longitude	X	Y	Distance from center	Restaurants in area
0	San Francisco, CA	37.805313	-122.378149	-3.312433e+06	1.484033e+07	3807.771539	0
1	San Francisco, CA	37.808009	-122.384163	-3.311713e+06	1.484033e+07	3577.698155	0
2	San Francisco, CA	37.810704	-122.390177	-3.310993e+06	1.484033e+07	3483.894960	0
3	San Francisco, CA	37.813400	-122.396193	-3.310273e+06	1.484033e+07	3537.219825	0
4	San Francisco, CA	37.816095	-122.402209	-3.309553e+06	1.484033e+07	3731.370270	0
5	San Francisco, CA	37.797362	-122.372573	-3.313453e+06	1.484095e+07	3837.581324	0
6	San Francisco – Oakland Bay Bridge, San Franci...	37.800057	-122.378585	-3.312733e+06	1.484095e+07	3399.857411	0
7	San Francisco, CA	37.802753	-122.384599	-3.312013e+06	1.484095e+07	3070.477229	0
8	San Francisco, CA	37.805448	-122.390613	-3.311293e+06	1.484095e+07	2886.768161	0
9	San Francisco, CA	37.808144	-122.396628	-3.310573e+06	1.484095e+07	2876.774307	0

In [211]:
```
df_locations.shape
```

Out[211]:
```
(110, 7)
```

Now let's calculate the distance ti nearest Italian restaurant from every area candidate center (not only thise within 300 m, we want distance to closest one, regardless of how distant it is)

In [212]:
```
distances_to_italian_restaurant = []

for area_x, area_y in zip(xs, ys):
    min_distance = 10000
    for res in italian_restaurants.values():
        res_x = res[7]
        res_y = res[8]
        d = calc_xy_distance(area_x, area_y, res_x, res_y)
        if d<min_distance:
            min_distance = d
    distances_to_italian_restaurant.append(min_distance)

df_locations['Distance to Italian restaurant'] = distances_to_italian_restaurant
```

In [213]:
```
df_locations.head(10)
```

Out[213]:

	Address	Latitude	Longitude	X	Y	Distance from center	Restaurants in area	Dist res
0	San Francisco, CA	37.805313	-122.378149	-3.312433e+06	1.484033e+07	3807.771539	0	2052
1	San Francisco, CA	37.808009	-122.384163	-3.311713e+06	1.484033e+07	3577.698155	0	1932
2	San Francisco, CA	37.810704	-122.390177	-3.310993e+06	1.484033e+07	3483.894960	0	1628
3	San Francisco, CA	37.813400	-122.396193	-3.310273e+06	1.484033e+07	3537.219825	0	1539
4	San Francisco, CA	37.816095	-122.402209	-3.309553e+06	1.484033e+07	3731.370270	0	1153
5	San Francisco, CA	37.797362	-122.372573	-3.313453e+06	1.484095e+07	3837.581324	0	2156
6	San Francisco – Oakland Bay Bridge, San Franci...	37.800057	-122.378585	-3.312733e+06	1.484095e+07	3399.857411	0	1643
7	San Francisco, CA	37.802753	-122.384599	-3.312013e+06	1.484095e+07	3070.477229	0	1336
8	San Francisco, CA	37.805448	-122.390613	-3.311293e+06	1.484095e+07	2886.768161	0	1247
9	San Francisco, CA	37.808144	-122.396628	-3.310573e+06	1.484095e+07	2876.774307	0	917

In [214]:
```
print('Average distance to closest Italian restaurat from each area center:', df_
```

```
Average distance to closest Italian restaurat from each area center: 700.892680
0609331
```

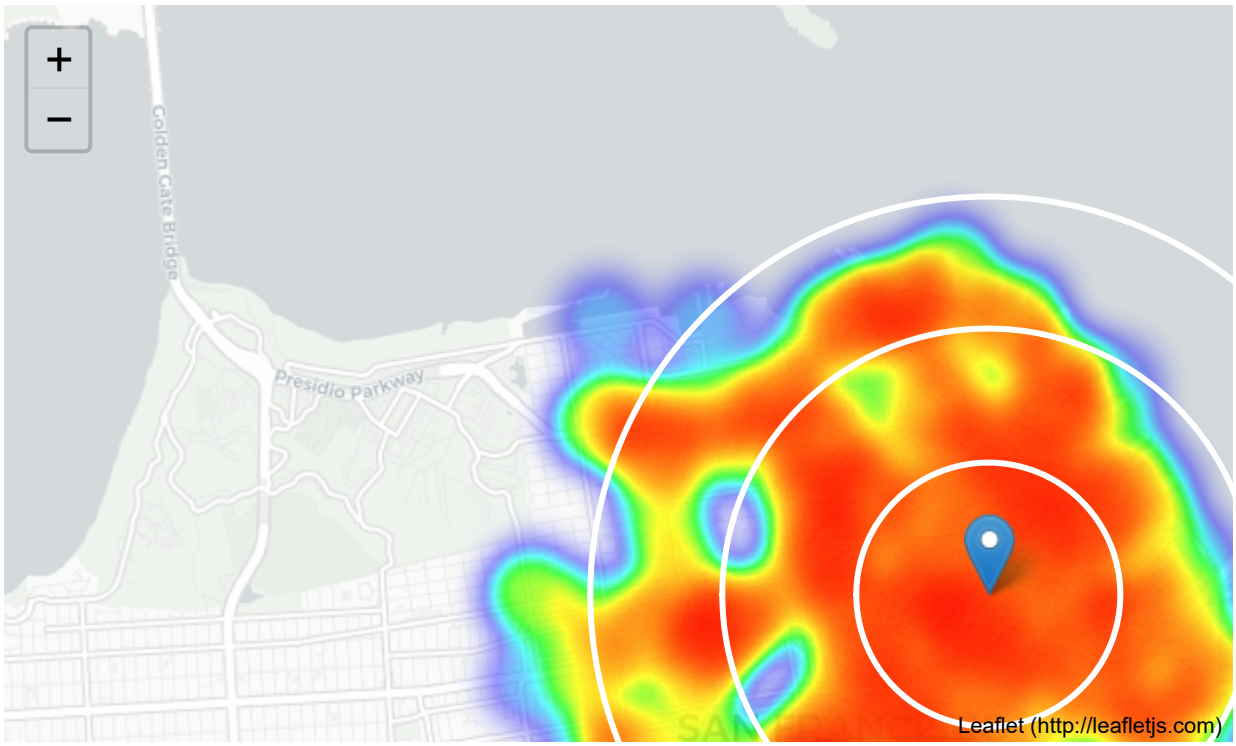
So on average Italian restaurat can be found within **700m** from every area center candidate. That's a little close, so we need to filter our area carefully.

Let's create a map showing heatmap density of restaurants and try to extract some meaningfull info from that. Also, let's show borders of San francisco boroughs on our map and few circles indicating distance from 1km, 2km and 3km from Union Square.

In [215]:
```
restaurant_latlons = [[res[2], res[3]] for res in restaurants.values()]

italian_latlons = [[res[2], res[3]] for res in italian_restaurants.values()]
```

In [216]:
```
from folium import plugins
from folium.plugins import HeatMap

map_sf = folium.Map(location=sf_center, zoom_start=13)
folium.TileLayer('cartodbpositron').add_to(map_sf) #cartodbpositron cartodbdark_
HeatMap(restaurant_latlons).add_to(map_sf)
folium.Marker(sf_center).add_to(map_sf)
folium.Circle(sf_center, radius=1000, fill=False, color='white').add_to(map_sf)
folium.Circle(sf_center, radius=2000, fill=False, color='white').add_to(map_sf)
folium.Circle(sf_center, radius=3000, fill=False, color='white').add_to(map_sf)
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[216]:



In [217]:
```
# add
restaurant_latlons
```

Out[217]:
```
[[37.8110504, -122.4104181],
 [37.8109343, -122.4108872],
 [37.81114025722897, -122.41065040036543],
 [37.799346, -122.395618],
 [37.80214747492143, -122.39687482007305],
 [37.80612149051469, -122.404939131139],
 [37.80335180922726, -122.40085594699735],
 [37.803538902224695, -122.40188133352027],
 [37.80264398142964, -122.40204842704162],
 [37.808903545091106, -122.41037933382874],
 [37.808765, -122.4098099],
 [37.80987127905576, -122.41047313758463],
 [37.809361697644135, -122.41044185347539],
 [37.809092261940044, -122.41005903126526],
 [37.806817471598656, -122.40562026751266],
 [37.80761115223452, -122.40468005039423],
 [37.810127, -122.410414],
 [37.80908833730518, -122.414845988984],
 [37.808618698378055, -122.41500961963726],
```

In [218]:
```
restaurant_latlons[0]
```

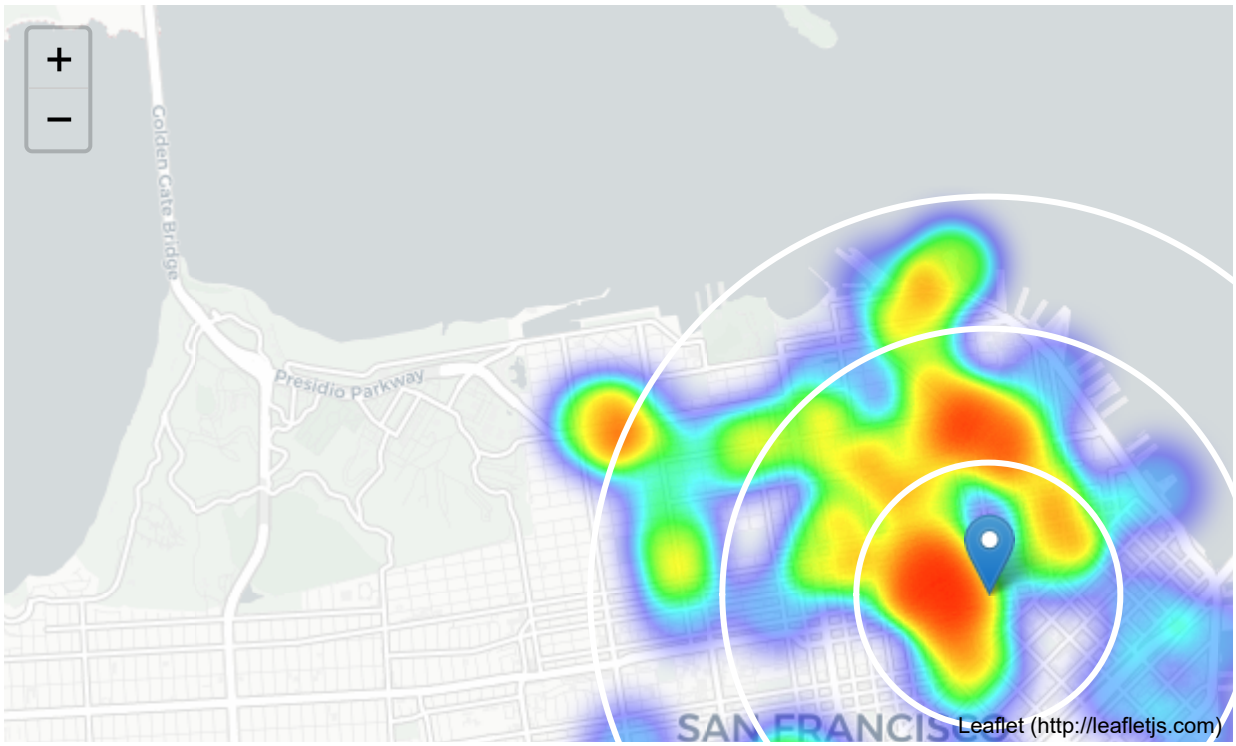
Out[218]:
```
[37.8110504, -122.4104181]
```

Looks like a few pockets of low restaurat density closest to city center can be found **north, north-east** fron Union Square.

Let's create another heatmap showing heatmap on **only Italian** restaurants.

In [219]:
```
map_sf = folium.Map(location=sf_center, zoom_start=13)
folium.TileLayer('cartodbpositron').add_to(map_sf) #cartodbpositron cartodbdark_
HeatMap(italian_latlons).add_to(map_sf)
folium.Marker(sf_center).add_to(map_sf)
folium.Circle(sf_center, radius=1000, fill=False, color='white').add_to(map_sf)
folium.Circle(sf_center, radius=2000, fill=False, color='white').add_to(map_sf)
folium.Circle(sf_center, radius=3000, fill=False, color='white').add_to(map_sf)
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[219]:



This map is not so 'hot', italian restaurants represent a subset of 8% of all restaruants in San Francisco, but it also indicate higher density of existing Italian restaurants directly from center and north from Union Square. However if we zoom in, we can observe that there are areas with out Italian restaurants nearby.

Based on this, we will focus our analysis on areas *center* and *south-east* from San Francisco center.

We will move the center of our area of interest and reduce it's size.

This places our location candidates mostly in boroughs **South Beach**, **West Soma**, and **Hayes Valley**.

South Beach, West Soma and Hayes Valley

South Beach is a neighborhood in San Francisco, California with a population of 5,041. South Beach is in San Francisco County and is one of the best places to live in California. Living in South Beach offers residents a dense urban feel and most residents rent their homes. In South Beach there are a lot of bars, restaurants, coffee shops, and parks. Many young professionals live in South Beach and residents tend to be liberal. The public schools in South Beach are highly rated.

Soma was once a light industrial and warehouse district. Today, it's one of the coolest neighborhoods in San Francisco. SOMA is packed with happening bars, boutique shopping, alternative culture, museums, art galleries, and tasty restaurants. It also has great parks and is conveniently located next to lots of public transportation.

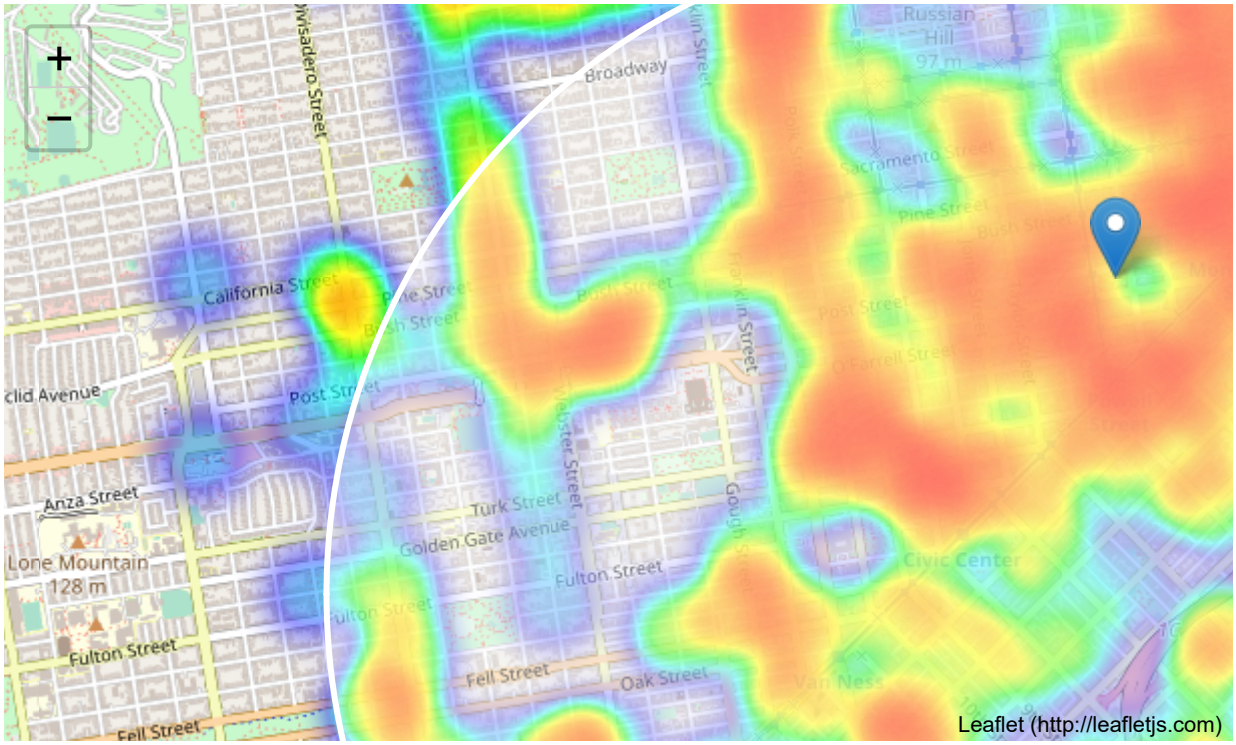
Hayes Valley’s transformation started back in 2003, when a committed group of citizens succeeded in taking down the Central Freeway that had divided their neighborhood. The efforts paid off massively: Since then, the area’s seen major investment from local brands and restaurants. The mix of creative boutiques and shops makes for a very stroll-able shopping neighborhood, not to mention what’s become one of the best food scenes in the city.

Analysing deeper this information and mora, we decide that one potenctional place coulb be South Beach, because it has a lot os hotels, which bring the most tourist.

Let's define, more narrow region if interest, which will include low-restaurants parts of **South Beach**.

In [220]:
```
roi_x_min = sf_center_x - 2700
roi_y_max = sf_center_y + 4000
roi_width = 5000
roi_height = 5000
roi_center_x = roi_x_min + 2500
roi_center_y = roi_y_max - 2500
roi_center_lon, roi_center_lat = xy_to_lonlat(roi_center_x, roi_center_y)
roi_center = [roi_center_lat, roi_center_lon]

map_sf = folium.Map(location=roi_center, zoom_start=14)
HeatMap(restaurant_latlons).add_to(map_sf)
folium.Marker(sf_center).add_to(map_sf)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[220]:



Not bat, this nicely covers all the pockets of low restaurant density in South West, at Union Square south.

Let's also crete nwe, more dense grid of location candidates restricted to our new region of interest.

In [221]:
```
k = math.sqrt(3) / 2 # Vertical offset for hexagonal grid cells
x_step = 100
y_step = 100 * k
roi_y_min = roi_center_y - 2500

roi_latitudes = []
roi_longitudes = []
roi_xs = []
roi_ys = []
for i in range(0, int(51/k)):
    y = roi_y_min + i * y_step
    x_offset = 50 if i%2==0 else 0
    for j in range(0, 51):
        x = roi_x_min + j * x_step + x_offset
        d = calc_xy_distance(roi_center_x, roi_center_y, x, y)
        if (d <= 2501):
            lon, lat = xy_to_lonlat(x, y)
            roi_latitudes.append(lat)
            roi_longitudes.append(lon)
            roi_xs.append(x)
            roi_ys.append(y)

print(len(roi_latitudes), 'candidate neighborhood centers generated.')
```

```
2261 candidate neighborhood centers generated.
```

OK. Now let's calculate two most important things for each location candidate: **number of restaurants in vicinity** (we'll use radius of **250 meters**) and **distance to closest Italian restaurant**.

In [222]:
```
def count_restaurants_nearby(x, y, restaurants, radius=250):
    count = 0
    for res in restaurants.values():
        res_x = res[7]; res_y = res[8]
        d = calc_xy_distance(x, y, res_x, res_y)
        if d<=radius:
            count += 1
    return count

def find_nearest_restaurant(x, y, restaurants):
    d_min = 100000
    for res in restaurants.values():
        res_x = res[7]; res_y = res[8]
        d = calc_xy_distance(x, y, res_x, res_y)
        if d<=d_min:
            d_min = d
    return d_min

roi_restaurant_counts = []
roi_italian_distances = []

print('Generating data on location candidates... ', end='')
for x, y in zip(roi_xs, roi_ys):
    count = count_restaurants_nearby(x, y, restaurants, radius=250)
    roi_restaurant_counts.append(count)
    distance = find_nearest_restaurant(x, y, italian_restaurants)
    roi_italian_distances.append(distance)
print('done.')
```

```
Generating data on location candidates... done.
```

In [223]:
```
# Let's put this into dataframe
df_roi_locations = pd.DataFrame({'Latitude':roi_latitudes,
                                 'Longitude':roi_longitudes,
                                 'X':roi_xs,
                                 'Y':roi_ys,
                                 'Restaurants nearby':roi_restaurant_counts,
                                 'Distance to Italian restaurant':roi_italian_di

df_roi_locations.head(10)
```

Out[223]:

	Latitude	Longitude	X	Y	Restaurants nearby	Distance to Italian restaurant
0	37.793688	-122.400634	-3.311143e+06	1.484281e+07	18	165.020058
1	37.794062	-122.401470	-3.311043e+06	1.484281e+07	17	245.166091
2	37.791056	-122.396448	-3.311693e+06	1.484290e+07	2	394.984386
3	37.791430	-122.397284	-3.311593e+06	1.484290e+07	4	300.350146
4	37.791804	-122.398119	-3.311493e+06	1.484290e+07	8	210.731476
5	37.792178	-122.398954	-3.311393e+06	1.484290e+07	13	136.401245
6	37.792553	-122.399790	-3.311293e+06	1.484290e+07	13	113.149656
7	37.792927	-122.400625	-3.311193e+06	1.484290e+07	15	162.537039
8	37.793301	-122.401460	-3.311093e+06	1.484290e+07	18	181.083895
9	37.793675	-122.402296	-3.310993e+06	1.484290e+07	19	201.832546

In [224]:
```
df_roi_locations.shape
```

Out[224]:
```
(2261, 6)
```

OK. Let us now filter those locations: we're interested only in locations with no more than two restaurants in radius of 250 meters, and no Italian restaurants in radius of 400 meters.

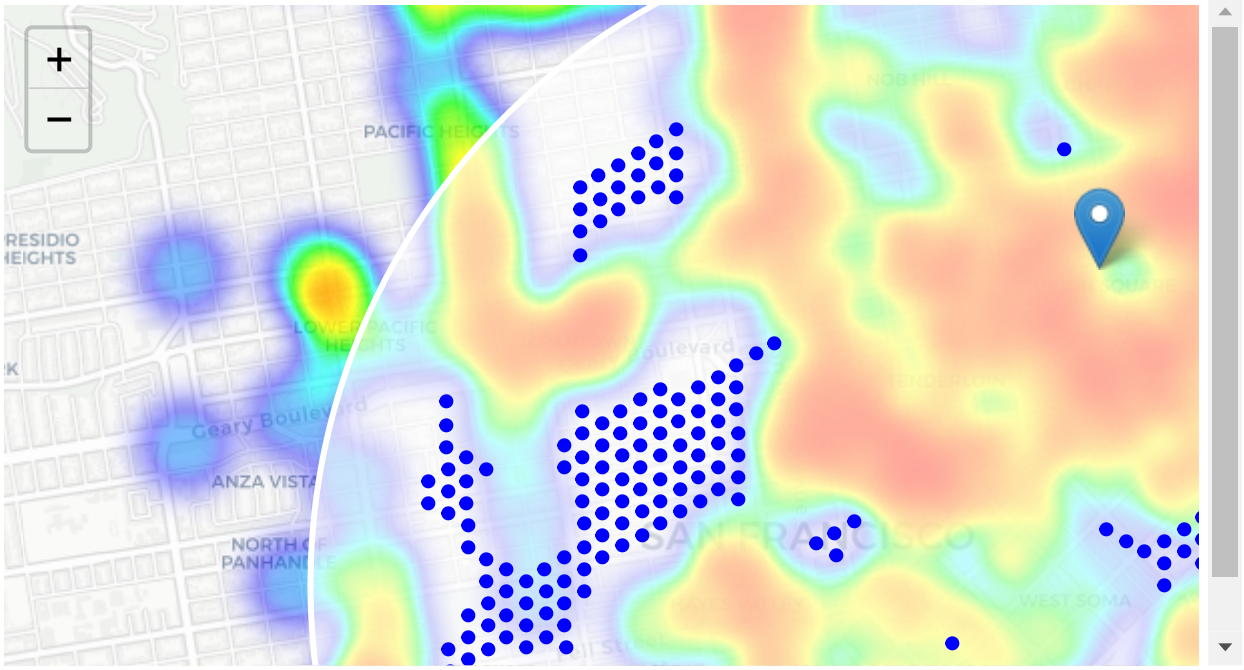
In [225]:
```
good_res_count = np.array((df_roi_locations['Restaurants nearby']<=2))
print('Locations with no more than two restaurants nearby:', good_res_count.sum(

good_ita_distance = np.array(df_roi_locations['Distance to Italian restaurant']>
print('Locations with no Italian restaurants within 400m:', good_ita_distance.su

good_locations = np.logical_and(good_res_count, good_ita_distance)
print('Locations with both conditions met:', good_locations.sum())

df_good_locations = df_roi_locations[good_locations]
```

```
Locations with no more than two restaurants nearby: 718
Locations with no Italian restaurants within 400m: 1182
Locations with both conditions met: 520
```

Let's se how this looks in a map.

In [226]:
```
good_latitudes = df_good_locations['Latitude'].values
good_longitudes = df_good_locations['Longitude'].values

good_locations = [[lat, lon] for lat, lon in zip(good_latitudes, good_longitudes

map_sf = folium.Map(location=roi_center, zoom_start=14)
folium.TileLayer('cartodbpositron').add_to(map_sf)
HeatMap(restaurant_latlons).add_to(map_sf)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.
folium.Marker(sf_center).add_to(map_sf)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[226]:



Looks goos, we now have a bunch of locations at the south of Union Square, mostly in South Beach, and we nkow that each of those locations has no mora than two restauratns in radious if 250m, and no Italian restaurant closer than 400m. So any of those location is a potential candidate for a new Italian restauratn, at leas based on neraby competition.

Let's now show those goog locations in a form of heatmap

In [227]:
```
map_sf = folium.Map(location=roi_center, zoom_start=14)
HeatMap(good_locations, radius=25).add_to(map_sf)
folium.Marker(sf_center).add_to(map_sf)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

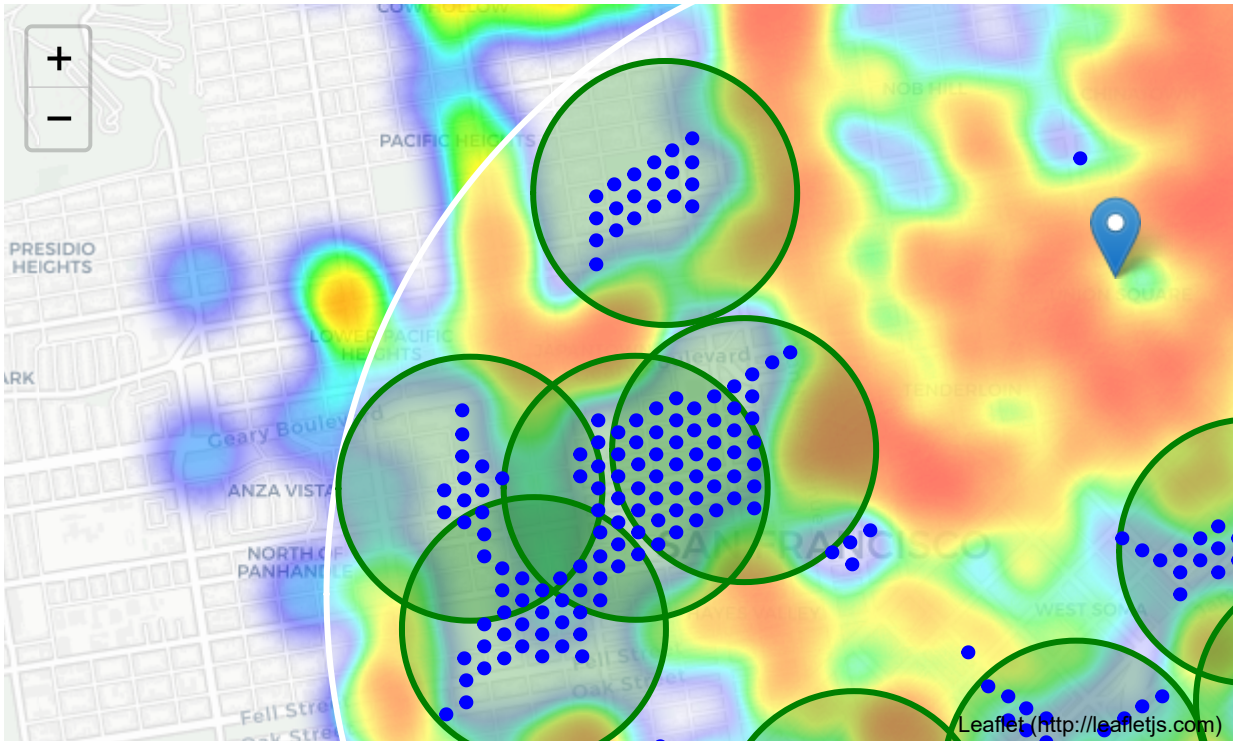
Out[227]:

Looking good, what we have now is a clear aindication of zones with low number of restaurants in vicinity ans NO italian restaurants nearby.

Let us cluster those locations to create **center of zones containing good locations**, those zones, their centerts and addresses will be the final result of our analysis.

In [228]:
```
from sklearn.cluster import KMeans

number_of_clusters = 15

good_xys = df_good_locations[['X', 'Y']].values
kmeans = KMeans(n_clusters=number_of_clusters, random_state=0).fit(good_xys)

cluster_centers = [xy_to_lonlat(cc[0], cc[1]) for cc in kmeans.cluster_centers_]

map_sf = folium.Map(location=roi_center, zoom_start=14)
folium.TileLayer('cartodbpositron').add_to(map_sf)
HeatMap(restaurant_latlons).add_to(map_sf)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.4
folium.Marker(sf_center).add_to(map_sf)
for lon, lat in cluster_centers:
    folium.Circle([lat, lon], radius=500, color='green', fill=True, fill_opacity=
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[228]:



Not bad - our clusters represent groupings of most of the candidate locations and cluster centers are placed nicely in the middle of the zones 'rich' with location candidates.

Addresses of those cluster centers will be a good starting point for exploring the neighborhoods to find the best possible location based on neighborhood specifics.

Let's see those zones on a city map without heatmap, using shaded areas to indicate our clusters:

In [229]:
```
map_sf = folium.Map(location=roi_center, zoom_start=14)
folium.Marker(sf_center).add_to(map_sf)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.Circle([lat, lon], radius=250, color='#00000000', fill=True, fill_colo
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
for lon, lat in cluster_centers:
    folium.Circle([lat, lon], radius=500, color='green', fill=False).add_to(map_
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[229]:

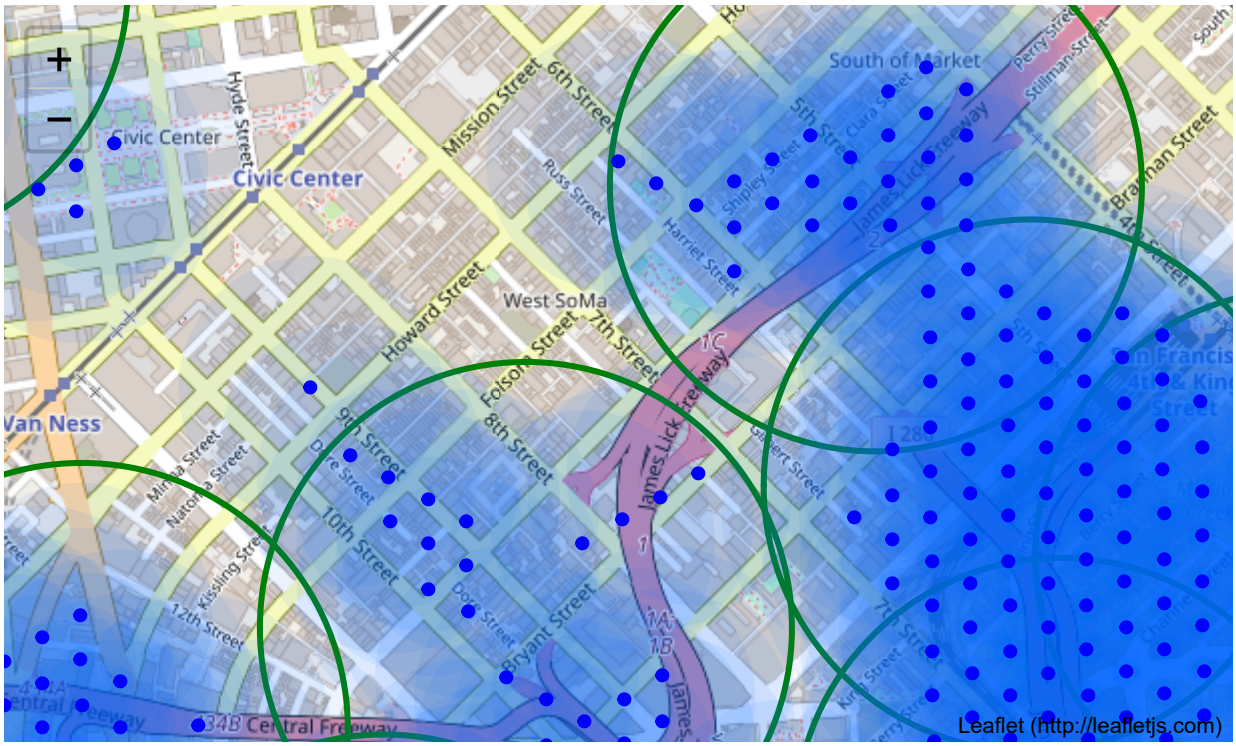
Let's zoom in on candidata areas near South Beach center

*Here we can convert between coordinates (grados decmimales y grados, minutos, segundos)

https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps (https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps)

In [230]:
```
map_sf = folium.Map(location=[37.77181370069208, -122.39915103200892], zoom_star
folium.Marker(sf_center).add_to(map_sf)
for lon, lat in cluster_centers:
    folium.Circle([lat, lon], radius=500, color='green', fill=False).add_to(map_
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.Circle([lat, lon], radius=250, color='#0000ff00', fill=True, fill_col
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_colo
#folium.GeoJson(berlin_boroughs, style_function=boroughs_style, name='geojson').
map_sf
```

Out[230]:



Finally, let's **reverse geocode those candidate area centers to get the addresses** which can be presented to stakeholders.

In [231]:
```
candidate_area_addresses = []
print('============================================================')
print('Addresses of centers of areas recommended for further analysis')
print('============================================================\n')
for lon, lat in cluster_centers:
    addr = get_address(google_api_key, lat, lon).replace(', USA', '')
    candidate_area_addresses.append(addr)
    x, y = lonlat_to_xy(lon, lat)
    d = calc_xy_distance(x, y, sf_center_x, sf_center_y)
    print('{}{} => {:.1f}km from Union Square'.format(addr, ' '*(50-len(addr)), d
```

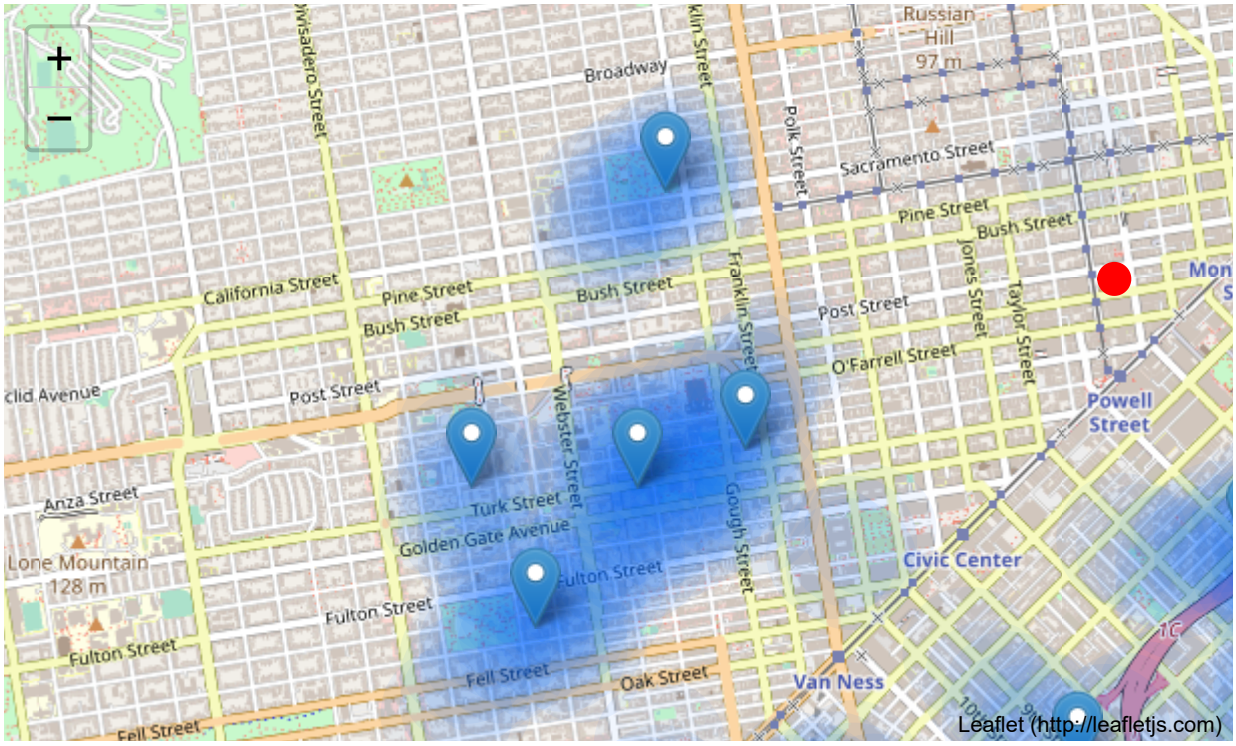
```
============================================================
Addresses of centers of areas recommended for further analysis
============================================================

1675 Eddy St, San Francisco, CA 94115              => 3.0km from Union Square
1200 7th St, San Francisco, CA 94107               => 2.8km from Union Square
333 Alabama St, San Francisco, CA 94110            => 3.1km from Union Square
610 Long Bridge Street, San Francisco, CA 94158    => 2.6km from Union Square
1141 Turk St, San Francisco, CA 94115              => 2.3km from Union Square
1104 Bryant St, San Francisco, CA 94103            => 2.2km from Union Square
2049 Sacramento St, San Francisco, CA 94109        => 2.0km from Union Square
2145 18th St, San Francisco, CA 94107              => 3.5km from Union Square
420 5th St, San Francisco, CA 94107                => 1.4km from Union Square
3484 19th St, San Francisco, CA 94110              => 3.9km from Union Square
435 Townsend St, San Francisco, CA 94107           => 2.1km from Union Square
295 Church St, San Francisco, CA 94114             => 3.6km from Union Square
958 Hayes St, San Francisco, CA 94117              => 3.0km from Union Square
951 Eddy St, San Francisco, CA 94109               => 1.8km from Union Square
224 US-101, San Francisco, CA 94103                => 2.7km from Union Square
```

Here concludes our analysis, we have created 15 addresses representing centers of zones containgi locations with low number of restaurants ans NO italian restaurants nearby, all zones being faily close to city center (at least 4 km form Union Square).

Most of the zones are located in the center of South Beach, which is a popular zone for tourist.

In [232]:
```
map_sf = folium.Map(location=roi_center, zoom_start=14)
folium.Circle(sf_center, radius=50, color='red', fill=True, fill_color='red', fi
for lonlat, addr in zip(cluster_centers, candidate_area_addresses):
    folium.Marker([lonlat[1], lonlat[0]], popup=addr).add_to(map_sf)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.Circle([lat, lon], radius=250, color='#0000ff00', fill=True, fill_colo
map_sf
```

Out[232]:



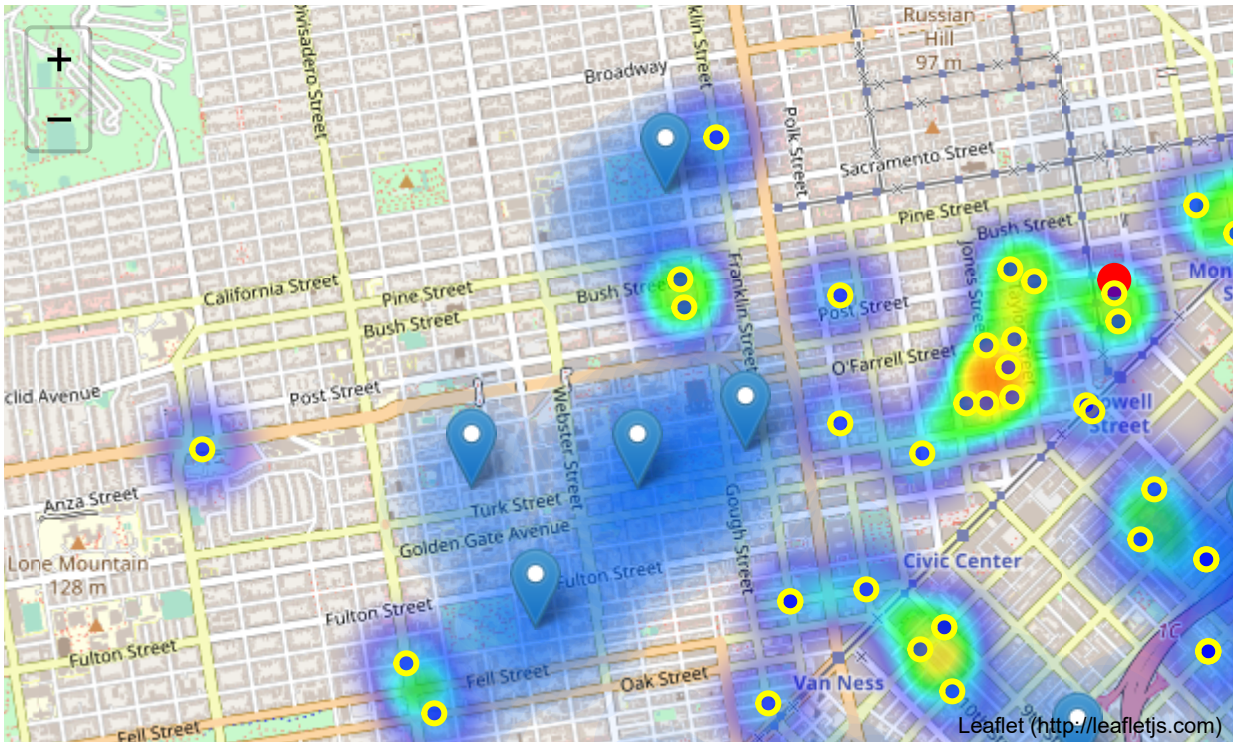
Analyzing the Crime Zone

We will import crime rate data for San Francisco, provided in a previous lab, in such a way to look if one of our potential candidate location is located near or relative far from the most areas where crimes have been reported.

In [233]:
```
# Import the dataset (150500, 13)
df_incidents = pd.read_csv('csv_files/Police_Department_Incidents_-_Previous_Yea
# Limit the dataset to only 100 rows
df_incidents = df_incidents.iloc[0:100,:]
```

In [234]:
```
# instantiate a feature group for the incidents in the dataframe
incidents = folium.map.FeatureGroup()

# loop through the 100 crimes and add each to the incidents feature group
for lat, lng, in zip(df_incidents.Y, df_incidents.X):
    incidents.add_child(
        folium.features.CircleMarker(
            [lat, lng],
            radius=5, # define how big you want the circle markers to be
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6
        )
    )

# Select the locations for create the Heat Map
crime_latlon = df_incidents[['Y','X']].values.tolist()
# add incidents to map and the heatmap also
HeatMap(crime_latlon).add_to(map_sf)
map_sf.add_child(incidents)
```

Out[234]:



Looking thsi distribution we see that most crime are located at the center and near from Union Square, so we decided to eliminate the following candidates because are near to crime zones density.

420 5th St, San Francisco, CA 94107

333 Alabama St, San Francisco, CA 94110

3484 19th St, San Francisco, CA 94110

224 US-101, San Francisco, CA 94103

So our final candidates are shown below

In [235]:
```
candidate_area_addresses
```

Out[235]:
```
['1675 Eddy St, San Francisco, CA 94115',
 '1200 7th St, San Francisco, CA 94107',
 '333 Alabama St, San Francisco, CA 94110',
 '610 Long Bridge Street, San Francisco, CA 94158',
 '1141 Turk St, San Francisco, CA 94115',
 '1104 Bryant St, San Francisco, CA 94103',
 '2049 Sacramento St, San Francisco, CA 94109',
 '2145 18th St, San Francisco, CA 94107',
 '420 5th St, San Francisco, CA 94107',
 '3484 19th St, San Francisco, CA 94110',
 '435 Townsend St, San Francisco, CA 94107',
 '295 Church St, San Francisco, CA 94114',
 '958 Hayes St, San Francisco, CA 94117',
 '951 Eddy St, San Francisco, CA 94109',
 '224 US-101, San Francisco, CA 94103']
```

In [236]:
```
f = [candidate_area_addresses[0],
     candidate_area_addresses[1],
     candidate_area_addresses[3],
     candidate_area_addresses[4],
     candidate_area_addresses[5],
     candidate_area_addresses[6],
     candidate_area_addresses[7],
     candidate_area_addresses[10],
     candidate_area_addresses[11],
     candidate_area_addresses[13],
     candidate_area_addresses[13]
    ]
f
```

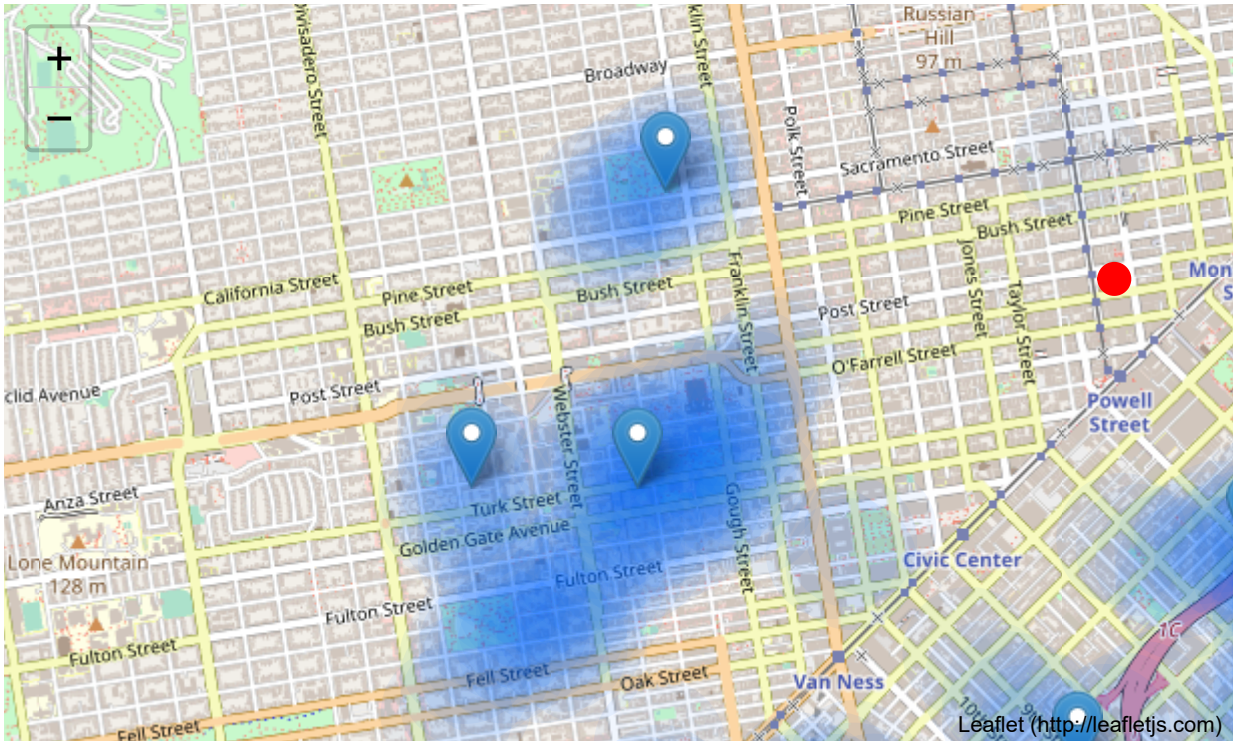
Out[236]:
```
['1675 Eddy St, San Francisco, CA 94115',
 '1200 7th St, San Francisco, CA 94107',
 '610 Long Bridge Street, San Francisco, CA 94158',
 '1141 Turk St, San Francisco, CA 94115',
 '1104 Bryant St, San Francisco, CA 94103',
 '2049 Sacramento St, San Francisco, CA 94109',
 '2145 18th St, San Francisco, CA 94107',
 '435 Townsend St, San Francisco, CA 94107',
 '295 Church St, San Francisco, CA 94114',
 '951 Eddy St, San Francisco, CA 94109',
 '951 Eddy St, San Francisco, CA 94109']
```

In [237]:
```
print(len(candidate_area_addresses))
print(len(f))
```

```
15
11
```

In [238]:
```
map_sf = folium.Map(location=roi_center, zoom_start=14)
folium.Circle(sf_center, radius=50, color='red', fill=True, fill_color='red', fi
for lonlat, addr in zip(cluster_centers, f):
    folium.Marker([lonlat[1], lonlat[0]], popup=addr).add_to(map_sf)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.Circle([lat, lon], radius=250, color='#0000ff00', fill=True, fill_colo
map_sf
```

Out[238]:



Results and Discussion

The previous analysis we did, sow that although there is a great number of restaurant in San Francisco, there are pockets of low restaurant density at the south less than 3 km, which is in our focus area that was of 4km around. Highest concentration of restaurant was detected center, north, north-east from Union Square, that is why we focus to south and south-west areas, most of them corresponging to South Beach borough, some other posilble candidates were located st south-east, but South Beach is more popular and atracts more tourists.

After directing out attention to this morr narrow area of interest, we first created a dense grid of location candidates spaced 100m appart, thosse locations were then filteres so that those with mora than two restaurats in radious of 250m and those with an Italian restaurant closet than 400 were removed.

Those location candidates were then clustered to create zones of interest which contain gratest number of location candidates. Addresses of centert of those zones were also generated usging reverso geocoding to be used as starting point for more detailed local analysis.

Additional, we made a heat map of crime rate zones in San Francisco, 4 of the candidates for the restaurant were close to the concurrent centers of crime, so we decided to eliminate that posible location candidates.

Resulting in 11 zones containing largest number of potential new restaurants location based on the number, distance and crime rate to existing venues, distance to restaurants in general and Italian restaurant particulary. This does not imply that those zones are actually optimal locations for a new restaurant.

Propouse of this analysis was to provide information on areas close to San Fracisco but not crowed with existing restaurants nearby. It is entire possible that there is a vero good reason for small number of restaurants in any of those areas, reason wich would make them usuitable for a new restaurant regardless od lack of competition in the area.

The recommended zones should therefore be condidered only as a starting point for more detailed analysis which could eventually result in location which has not only no nearby competition but also other factors taken into accoun and all other relevant condidions met.

Conclusion

The purpose of this proyect was to find the best locaton to open a new Italian Restaurant in San Francisco in order to aid stakholders in narrowing down the search for optimal location. This was made by calculating density distribution form Foursquare data we have identified general boroughs that justif further analysis (South Beach), then generate a collection of locations which satisfy some basic requirements regarding existing nearby restaurats.

Clustering this locations in order to create major zones of interest, and adresses of those zone centers were createdto be used as starting point.

The final decission on optimal restaurant location will be made by stakeholders based on specific charcteristics of neighborhoods and location in every one zone we recommended, taking in consideration additional factors like attractivess of each location, could by the proximity to park, leverl of noise, proximity to majer roads, abailability, prices, social and econoic behavors of every neighborhood.

In [ ]: