Assignment #5

Chicago Food Inspections - NoSQL

Author: Atef Bader Last Edit: 11/26/2018

Deliverables:

- Submit a single zip-compressed file that has the name: YourLastName_Assignment_5 that has the following files:

    1. Your **PDF document** that has your Source code and output
    2. Your **ipynb script** that has your Source code and output

Objectives:

In this assignment, you will:

- Interact with a **NoSQL** (document-oriented) database engine, ElasticSearch
- Experient with different NoSQL queries and evaluate the output to fine-tune results for better precision/accuracy /relevance
- Create and run NoSQL queries required for this assignment requirements

Submission Formats :

Create a folder or directory with all supplementary files with your last name at the beginning of the folder name, compress that folder with zip compression, and post the zip-archived folder under the assignment link in Canvas. The following files should be included in an archive folder/directory that is uploaded as a single zip-compressed file. (Use zip, not StuffIt or any 7z or any other compression method.)

1. Complete IPYNB script that has the source code in Python used to access and analyze the data. The code should be submitted as an IPYNB script that can be be loaded and run in Jupyter Notebook for Python
2. Output from the program, such as console listing/logs, text files, and graphics output for visualizations. If you use the Data Science Computing Cluster or School of Professional Studies database servers or systems, include Linux logs of your sessions as plain text files. Linux logs may be generated by using the script process at the beginning of your session, as demonstrated in tutorial handouts for the DSCC servers.
3. List file names and descriptions of files in the zip-compressed folder/directory.

Formatting Python Code When programming in Python, refer to Kenneth Reitz' PEP 8: The Style Guide for Python Code: http://pep8.org/ (http://pep8.org/) (Links to an external site.)Links to an external site. There is the Google style guide for Python at https://google.github.io/styleguide/pyguide.html (https://google.github.io/styleguide/pyguide.html) (Links to an external site.)Links to an external site. Comment often and in detail.

Assignment Description and Requirement Specifications

Chicago Food Inspections

Recent watchdog report published by **[Chicago Tribune (http://www.chicagotribune.com/news/watchdog/ct-daycare-food-inspections-met-20150516-story.html)](http://www.chicagotribune.com/news/watchdog/ct-daycare-food-inspections-met-20150516-story.html)** indcated that food safety inspectors overlook hundreds of day cares in the city of Chicago.

The key take away from the Chicago Tribune watchdog report is that the city had only 33 working field inspectors to cover the entire city of Chicago. Many of the facilties serve food for Children, and while few fail inspectionns, many escape routine inspections.

This is a classic resource allocation problem. In this assignment, our goal is to identify the **hot-spots** (areas that have facilities serving food to children and have failed inspections in the past) on the Chicago map to dispatch inspectors to.

To achive our goal, we need the following:

1. Dataset for Chicago Food Inspections
2. NoSQL database Egnine (ElasticSearch) for indexing and data retrieval
3. HeatMap to plot the children facilties that failed Chicago Food Inspections

The CSV file for dataset of the city of chicago is obtained from the data portal for the city of Chicago. Here th elink for the city of Chicago data portal **[City of Chicago Data Portal (https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5)](https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5)**

Loading the Dataset CSV file

Lets load the CSV file into a DataFrame object and see the nature of the data that we have.

Description of the dataset:

1. It has 164953 inspection records
2. It has inspection records from 2010 to 2018
3. It has 17 fields

In [181]:
```
# Lets load the CSV Chicago Food Inspections dataset into a dataframe
import pandas as pd

df = pd.read_csv("Chicago_Food_Inspections.csv")
```

In [182]:
```
df.head()
```

Out[182]:

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City	State	Zip	Ins
0	2144807	SAMMY'S RED HOT	SAMMY'S RED HOT	2578852.0	Restaurant	Risk 1 (High)	238 W DIVISION ST	CHICAGO	IL	60610.0	02/
1	2144802	CAFE NILLY	NILLY CAFE	2578631.0	Restaurant	Risk 1 (High)	60 E ADAMS ST	CHICAGO	IL	60603.0	02/
2	2144800	EVITA ARGENTINIAN STEAKHOUSE	EVITA	2464488.0	Restaurant	Risk 1 (High)	6112 N LINCOLN AVE	CHICAGO	IL	60659.0	02/
3	2144791	PHO SPICIER THAI	PHO SPICIER THAI	2578881.0	NaN	All	1320 W DEVON AVE	CHICAGO	IL	60660.0	02/
4	2144789	RED SNAPPER	JIMMY'S BEST	2232836.0	Restaurant	Risk 1 (High)	1347 E 87TH ST	CHICAGO	IL	60619.0	02/

There are few fields in the dataset of interest for us:

1. Risk
2. Results
3. Latitude
4. Longtitude
5. Inspection ID

We are also interested in any field that mentioned (or misspelled) the word **Children**

There are possibilities that the data entry clerk might've made some typos and misspellings and there are different words meant to indicate the same thing, some examples of this:

- Children
- Children's
- Childrens

To perform different queries to retrieve the relevant inspection records, we will store the dataset in a NoSQL database engine ElasticSearch.

For more information on elastic search visit [ElasticSearch (https://www.elastic.co/webinars/getting-started-elasticsearch?elektra=home&storm=sub1)](https://www.elastic.co/webinars/getting-started-elasticsearch?elektra=home&storm=sub1)

Please note that in this version of the assignment, the index for Chicago food inspections dataset already created on ElasticSearch on DSCC

- you do NOT need to create an index; its already created
- you are connecting to DSCC/ElasticSearch server thru the VPN to access the food_inspections index

ElasticSearch

- Download [elasticsearch (https://www.elastic.co/downloads/elasticsearch)](https://www.elastic.co/downloads/elasticsearch) to your laptop
- Getting Started with [elasticsearch (https://www.elastic.co/start)](https://www.elastic.co/start)

The three major platofrms are supported:

1. Windows
2. MacOS
3. Linux

Startup ElasticSearch Server

After you install ElasticSearch, go to the directory where you installed ElasticSearch under elasticsearch-6.2.3\bin directory and type from the terminal/command prompt the following command: **elasticsearch**

elasticsearch package

We need **elasticsearch (https://anaconda.org/anaconda/elasticsearch)** package to connect to ElasticSearch Servers

To install elastic search pakage, execute following command from the command/terminal windows:

- **conda install -c anaconda elasticsearch**

In [183]:
```
#Import Elascticsearch and helpers from  elasticsearch

from elasticsearch import Elasticsearch, helpers


es=Elasticsearch('http://student:spsdata@129.105.88.91:9200')
```

Load and Index the Inspection Records into ElasticSearch

Inspection records are insreted into ElasticSearch engine using the bulk Api of elastic search.

Here is the link **API DOCS (http://elasticsearch-py.readthedocs.io/en/master/helpers.html)** for the API documentation.

Query is used to retieve data from ElasticSearch server

The query is used to retrieve data from ElasticSearch servers that match certain filters.

For information about the syntax and semantics for query, you can read the docs at the following URL **QUERY DOCS (https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html)**

We will also use the scroll to retrive the data matching the our query. For more information about scroll, you can read the docs ta the following URL **Scroll DOCS (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-scroll.html)**

We create our query to rertieve the inspections records we are interested in three experiements and will compare the results for each:

1. Experiment #1: Using Regular Expressions using the term Children
2. Experiment #2: Using Fuziness using the term Children's
3. Experiment #3: Using Fuziness using the term Children

Experiment #1: Create the query using regex

In [184]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
{"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clause in SQL


                              {"query_string": {
                                            "query": "*Children*",  #using regex o
f children  to match all posssible combinations of "Children"
                                            "fields": ["Facility Type","Violation
s","DBA Name"] #Multi-field matching query
                                             }
                             }



                             ]
                    }
                }
        }
results = es.search(index='food_inspections', body=query, scroll='1h')
```

In [185]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']

print('sid = ', sid)
print('Scroll Size = ', scroll_size)
```

```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFw6FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcO
xZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXDwWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAg
FxAFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcPRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXD4WZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgFw_FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAIBcQRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXEIWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgFxDFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  601
```

In [186]:
```
type(results)
```

Out[186]:
```
dict
```

In [188]:
```
len(results['hits']['hits'])
```

Out[188]:
```
601
```

Process the retrieved documents and filter fields we need for the Heatmap

We need to create a list-of-lists of the two fields, (Latitude and Longitude) for the HeatMap

In [189]:
```
len(results['hits']['hits'])
```

Out[189]:
```
601
```

In [190]:
```
count = 0
list_of_lAT_LONG_pairs = []
while(scroll_size > 0):

    for inspection in results['hits']['hits']:                    #Iterating each
results of  the qurey
        current_location_lAT_LONG = []
        document = inspection['_source']
        count = count +1

        #defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document.keys():
            if 'Longitude' in document.keys():
                if 'Address' in document.keys():
                    if(document['Latitude'] != None and document['Longitude'] != N
one  and document['Address'] != None):
                        current_location_lAT_LONG.append(float(document['Latitude
']))    #Appending Latitude and Longitude into the list
                        current_location_lAT_LONG.append(float(document['Longitude
']))
                        list_of_lAT_LONG_pairs.append(current_location_lAT_LONG)

    results = es.scroll(scroll_id = sid, scroll = '2m')
    sid = results['_scroll_id']                                  #Changing the scro
ll-id
    scroll_size = len(results['hits']['hits'])

print("the total number of match with children using wild card:",count)
```

```
the total number of match with children using wild card: 601
```

In [191]:
```
document.keys()
```

Out[191]:
```
dict_keys(['Inspection ID', 'DBA Name', 'AKA Name', 'License #', 'Facility Type
', 'Risk', 'Address', 'City', 'State', 'Zip', 'Inspection Date', 'Inspection Typ
e', 'Results', 'Violations', 'Latitude', 'Longitude', 'Location'])
```

In [192]:
```
list_of_lAT_LONG_pairs[:3]
```

Out[192]:
```
[[41.8814369069, -87.6659213595],
 [41.760441801, -87.6735652436],
 [41.9531127244, -87.7800185741]]
```

In [193]:
```
len(list_of_lAT_LONG_pairs)
```

Out[193]:
```
601
```

We need to install folium package to plot the Map and Heatmaps

The official documentation can be accessed at this URL: Folium (https://github.com/python-visualization/folium)

To install Folium package execute following command from the Command/Terminal window:

- **conda install folium**

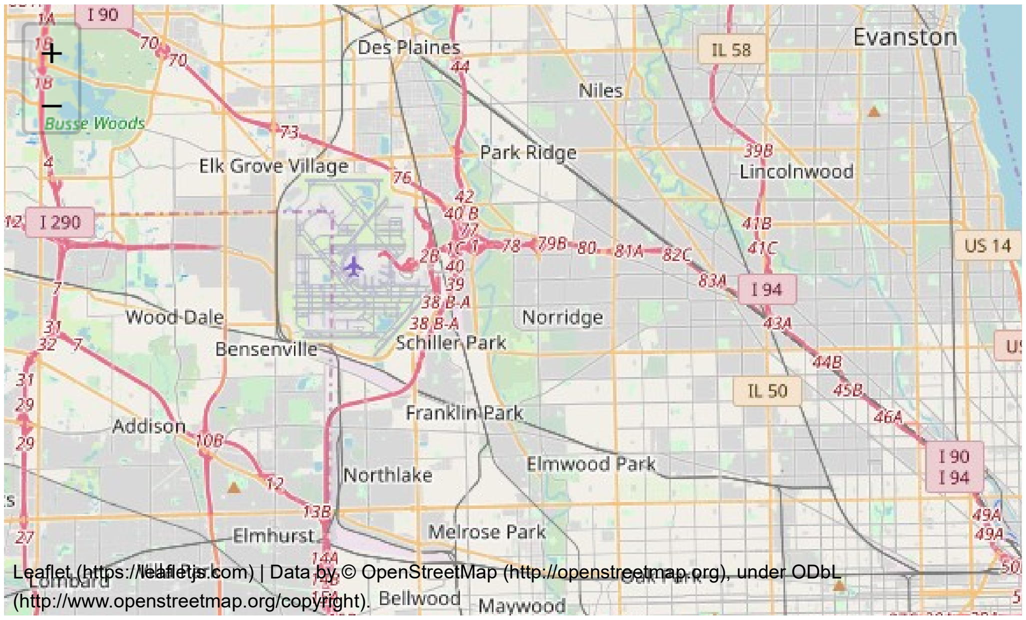
For the different configuration paramteres for HeatMap, you can access the docs at this URL: HeatMap (https://github.com/python-visualization/folium/blob/master/folium/plugins/heat_map.py)

In [194]:
```
import folium
from folium import plugins

print(folium.__version__)
```

```
0.10.0
```

In [195]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

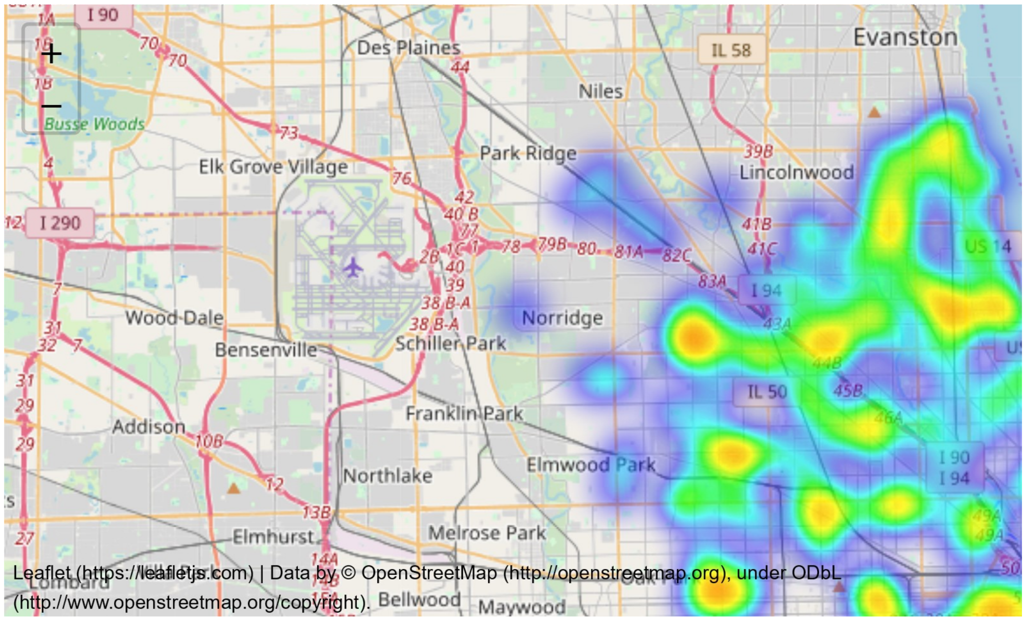
Out[195]:



Create the HeatMap

In [196]:
```
# Lets plot the query matches on Chicago HeatMap

chicago_map.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs, radius=15))
chicago_map
```

Out[196]:



Create the query using fuzziness

Now lets try to retrieve documents using ElasticSearch fuzziness

The fuzzy query generates all possible matching terms that are within the maximum edit distance specified in fuzziness.

For information about the syntax and semantics for fuziness, you can read the docs at the following URL [fuzziness (https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html)](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html)

Experiment #2: We will first build our query with the parameters:

1. "query": "Children",
2. "fuzziness": "1",

In [197]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {

                  'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
{"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clasue in SQL

                            {"query_string": {
                                            "query": "Children",
                                            "fuzziness": "1",
                                            "fields": ["Facility Type","Violation
s","DBA Name"]

                                             }
                            }

                            ]
                     }
                 }
        }
results = es.search(index='food_inspections', body=query,scroll='1h')
```

In [198]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']
```

In [199]:
```
count = 0
list_of_lAT_LONG_pairs = []

while(scroll_size > 0):

    for inspection in results['hits']['hits']:
        current_location_lAT_LONG = []
        document = inspection['_source']
        count = count +1

        #defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document.keys():
            if 'Longitude' in document.keys():
                if 'Address' in document.keys():
                    if(document['Latitude'] != None and document['Longitude'] != N
one  and document['Address'] != None):
                        current_location_lAT_LONG.append(float(document['Latitude
']))
                        current_location_lAT_LONG.append(float(document['Longitude
']))
                        list_of_lAT_LONG_pairs.append(current_location_lAT_LONG)

    results = es.scroll(scroll_id = sid, scroll = '2m')
    sid = results['_scroll_id']
    scroll_size = len(results['hits']['hits'])

print("Total number of query matches with children using fuziness:",count)
```

```
Total number of query matches with children using fuziness: 141
```

Experiment #3: Lets now build our query with the parameters:

1. "query": "Children's",
2. "fuzziness": "1",

In [200]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
{"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clasue in SQL

                            {"query_string": {
                                            "query": "Children's",
                                            "fuzziness": "1",
                                            "fields": ["Facility Type","Violation
s","DBA Name"]
                                             }
                            }

                            ]
                    }
                }
        }
results = es.search(index='food_inspections', body=query,scroll='1h')
```

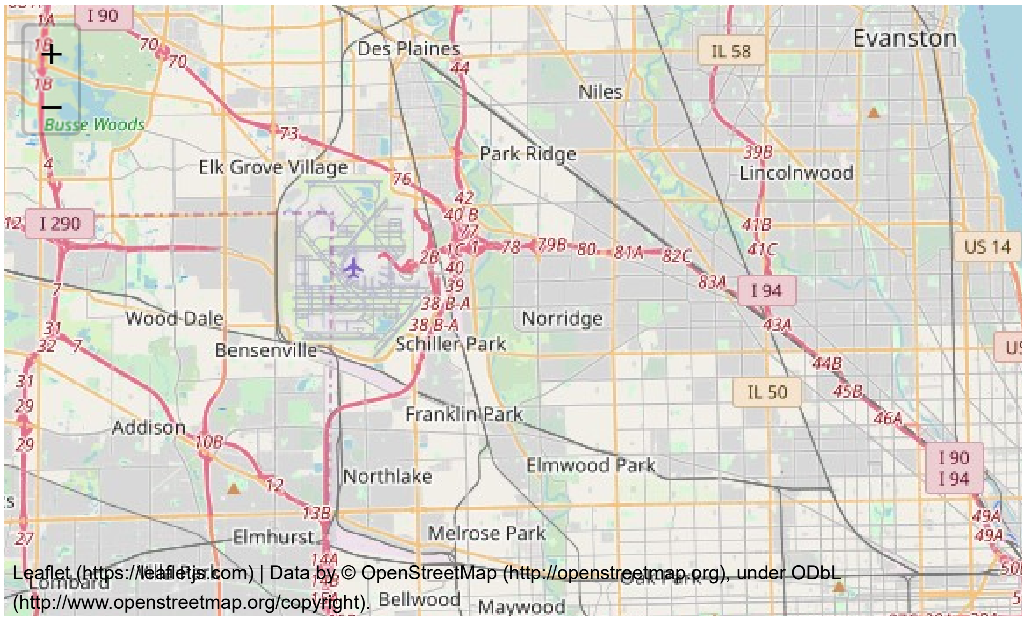
In [201]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']
```

In [202]:
```
count = 0
list_of_lAT_LONG_pairs = []

while(scroll_size > 0):

    for inspection in results['hits']['hits']:
        current_location_lAT_LONG = []
        document = inspection['_source']
        count = count +1

        #defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document.keys():
            if 'Longitude' in document.keys():
                if 'Address' in document.keys():
                    if(document['Latitude'] != None and document['Longitude'] != N
one  and document['Address'] != None):
                        current_location_lAT_LONG.append(float(document['Latitude
']))
                        current_location_lAT_LONG.append(float(document['Longitude
']))
                        list_of_lAT_LONG_pairs.append(current_location_lAT_LONG)

    results = es.scroll(scroll_id = sid, scroll = '2m')
    sid = results['_scroll_id']
    scroll_size = len(results['hits']['hits'])

print("Total number of match with Children's using fuziness:",count)
```

```
Total number of match with Children's using fuziness: 451
```

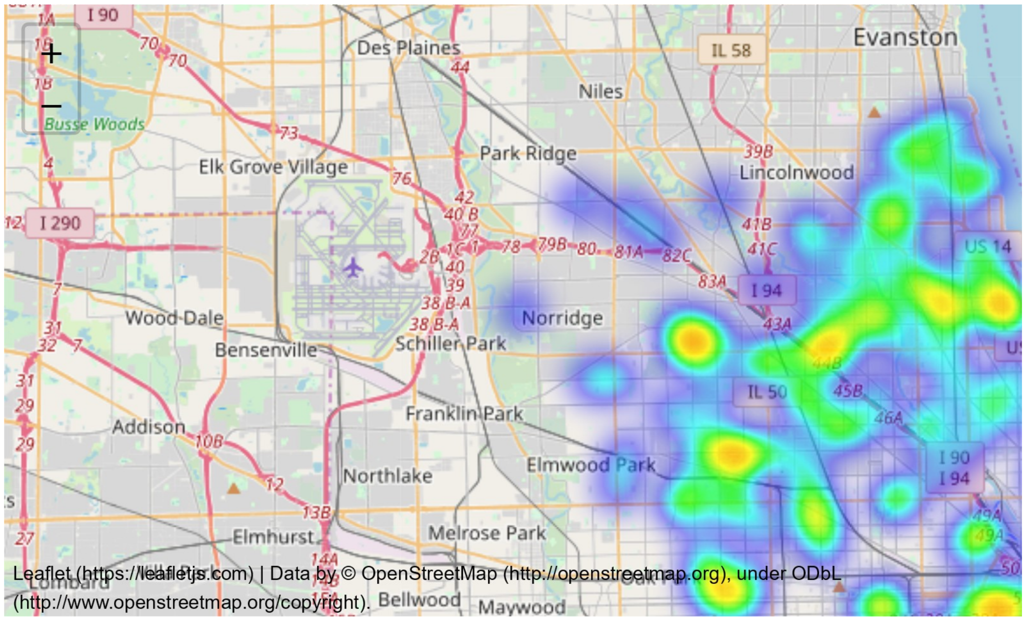
In [203]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

Out[203]:



In [204]:
```
# Lets plot the  query matches for  "Children's" on Chicago HeatMap

chicago_map.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs, radius=15))
chicago_map
```

Out[204]:



Frequent Violators:

Despite the fact that the city of Chicago has the department of **Business Affairs and Consumer Protection (https://www.cityofchicago.org/city/en/depts/bacp/provdrs/pros_adj.html)** to revoke business licensses to protect consumers, it appears many businesses with frequent violations have obtained new licenses under the same DBA name

Experiment #4: Lets get the top list of frequent violators:

Facilities that serve children can be classified under different Facility Types:

1. Daycare Above and Under 2 Years
2. Children's Services Facility
3. Daycare (2 - 6 Years)

We will use ELasticSearch and Folium to plot on the map those facilities that **failed inspection at least 5 times with risk high**.

In [205]:
```
query ={
      'size' : 10000,
      'query': {
          "bool" : {
              "should":[    {'match' : {'Facility Type': {"query" : 'Daycare (2 - 6 Years)',"operator":"and"}}},
                            {'match' : {'Facility Type':{"query" : 'Daycare Above and Under 2 Years',"operator": "and"}}},
                            {'match' : {'Facility Type':{"query" : 'CHILDRENS SERVICES FACILITY',"operator" : "and"}}},
                       ],
                       "minimum_should_match" : 1,
                       "filter" : [{"match" : {'Results': {"query": 'Fail', "operator": "and"}}},
                                    {"match" : {'Risk': {"query": 'Risk 1 (High)', "operator": "and"}}}
                                  ]

              }
      },
      "aggs" : {
          "selected_dbas" :{
                          "terms" : {
                                        "field" : "DBA Name.keyword",
                                        "min_doc_count": 5,
                                        "size" :10000

                                   },
                          "aggs": {
                              "top_dba_hits": {
                                  "top_hits": {
                                  "size": 10
                                  }
                              }
                  }



          }
     }
}

results = es.search(index='food_inspections', body=query,scroll='1h')
```

In [206]:
```
list_of_lAT_LONG_pairs = []

for dba_bucket in results["aggregations"]["selected_dbas"]["buckets"]:
    if "top_dba_hits" in dba_bucket and "hits" in dba_bucket["top_dba_hits"] and "hits" in dba_bucket["top_dba_hits"]["hits"]:

        for hit in dba_bucket["top_dba_hits"]["hits"]["hits"]:

            if "_source" in hit:

                if "Latitude" in hit["_source"] and "Longitude" in hit["_source"]:
                    list_of_lAT_LONG_pairs.append([hit["_source"]["Latitude"], hit["_source"]["Longitude"]])


# Lets dumps the LAt and LONG
# list_of_lAT_LONG_pairs
```

In [207]:
```
# Lets dump the hits per bucket into a datframe object for all buckets

row_index =0
df_top_frequent_violators = pd.DataFrame()
for dba_bucket in results["aggregations"]["selected_dbas"]["buckets"]:
    if "top_dba_hits" in dba_bucket and "hits" in dba_bucket["top_dba_hits"] and "
hits" in dba_bucket["top_dba_hits"]["hits"]:
        doc_count = dba_bucket['doc_count']
        for hit in dba_bucket["top_dba_hits"]["hits"]["hits"]:
            score = hit['_score']
            if "_source" in hit:
                row_index += 1
                df_frequent_violator = pd.DataFrame(hit['_source'],index =[row_ind
ex])
                df_frequent_violator['doc_count'] = doc_count
                df_frequent_violator['score'] = score
                df_top_frequent_violators = df_top_frequent_violators.append(df_fr
equent_violator)
```

In [208]:
```
df_top_frequent_violators.head()
```

Out[208]:

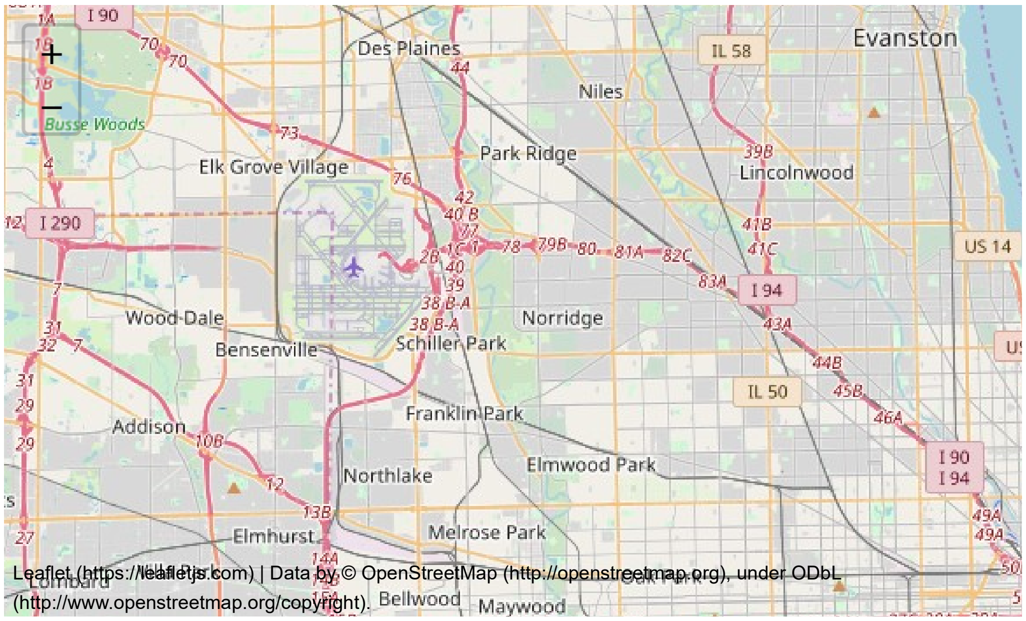
	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City	State	Zip	Inspecti D
1	1319663	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	2215472.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO	IL	60637.0	07/17/2(
2	1229713	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	3793.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO	IL	60637.0	06/20/2(
3	1515476	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	2215472.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO	IL	60637.0	12/29/2(
4	1229852	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	1194190.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO	IL	60637.0	06/28/2(
5	1386187	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	2215472.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO	IL	60637.0	06/08/2(

In [209]:
```
# Lets print the number of violations for every DBA NAME

df_top_frequent_violators['DBA Name'].value_counts()
```

Out[209]:
```
BUSY BUMBLE BEE ACADEMY DAYCARE                            9
BOTTLES TO BOOKS LEARNING CENTER                           8
AMAZING GRACE DAYCARE CENTER                               7
A CHILD'S WORLD EARLY LEARNING CENTER                      7
LINCOLN KING DAY CARE                                      6
EARLY CHILDHOOD EDUCARE CENTER                             6
Little People's Day Care & Kindergarten, Inc.              6
COMMONWEALTH DAYCARE CENTER                                6
LITTLE KIDS VILLAGE LEARNING                               6
KIDS R FIRST LEARNING ACADEMY                              6
THE WORLD IS YOUR'S CHILD CARE & LEARNING CENTER INC.     6
DISCOVERY LEARNING ACADEMY, INC.                           6
FIRMAN COMMUNITY SERVICES                                  6
JELLYBEAN LEARNING CENTER                                  6
ANGELS                                                     5
ADA S MCKINLEY MAGGIE DRUMMON                              5
LAKE & PULASKI CHILD DEVELOPMENT CENTER                    5
GREATER INSTITUTE AME CHURCH                               5
CENTRO INFANTIL                                            5
THE CRYSTAL PALACE EARLY LITERACY ZONE                     5
MONTESSORI ACDY. INFT/TOD. CNT                             5
GRANT DAY CARE INC                                         5
EZZARD CHARLES DAYCARE CENTER                              5
THE EDSEL ALBERT AMMONS NURSER                             5
KENYATTA'S DAYCARE                                         5
MOLADE' CHILD DEVELOPMENT CENTER                           5
Name: DBA Name, dtype: int64
```

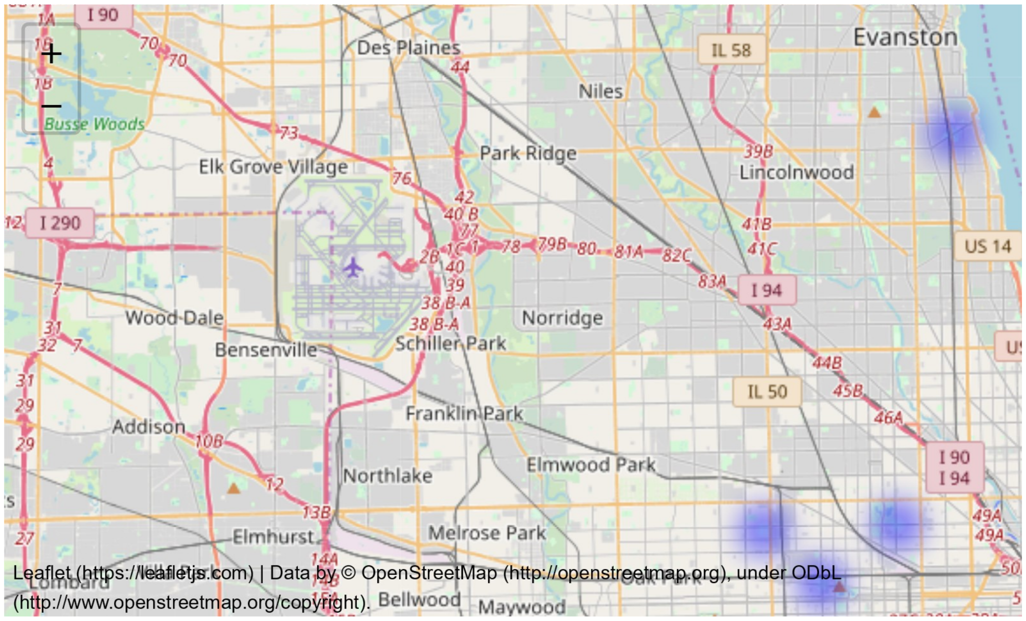
In [210]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

Out[210]:



In [211]:
```
# Lets plot the top frequent violators on Chicago HeatMap

chicago_map.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs, radius=15))
chicago_map
```

Out[211]:



Loopholes

- **How much the fee to apply for business license for Children services type facility?**

As you might have guessed by now, it must be really cheap to do so, those frequent violators reobtain business license multiple times under the same business name for only **$165** application fee based on the official numbers published on the [City of Chicago - Business Licensing (https://www.cityofchicago.org/city/en/depts/bacp/sbc/business_licensing.html#Children)](https://www.cityofchicago.org/city/en/depts/bacp/sbc/business_licensing.html#Children)

And it appears the city of Chicago is willinig to rubber-stamp the approval of the application for only **$165**, rather than imposing the very simple rule: **(3 strikes and you are out)**

Requirements

The PDF document your are submitting must have the source code and the output for the following four requirements

Requirement #1:

Provide your comparative analysis for the results obtained from 3 experiments you executed above

The first experiment using regex and the wildcard operator produced all possible combinations of "Children", such as CHILDREN, Children's, CHILDRN'S, and so on. The search seems to have been the widest out of the three. Although, this produced many results, the wildcard search pattern can produced results that are not as relevant. Additionally, a search such as this increases the iterations needed to find matching terms and will thus slow search performance. However, in this case, it seems as though the search may have brought in the right results, yet the caveat is that the * will always match zero or more characters. That is, if the query was adjusted to '**Chil**', it would have brought in irrelevant terms such as 'Chili's' along with 'Children'. In the matter of the concept and context of precision and recall, the regex search using the wildcard operator seems to "focus" more on completeness, which affects both precision and recall.

The second experiment using fuzziness and the word, "Children", did not produce as many matches as regex, though fuzzy matching is very similar to wildcard matching. Fuzziness is used to approximate string matching in the attempt to match one string to another one. It is very useful for queries that also need to match based on spelling errors in words. The degree of fuzziness is specified based on the Levenshtein distance from the original word, or the the number of one-character changes that need to be made to one string to make it the same as another string. Using elasticsearch, a fuzzy query creates a set of all possible variations of the search term within a specified edit (Levenshtein) distance, which then returns exact matches for each variation or expansion. However, in this case it seems that the fuzzy query did not "seem" to be as effective as the results seem to be too low. One explanation may be that it is a matter of how the fuzzy query works. Although it goes through all the variations to approximate a string that will likely match a search term(s), a fuzzy search will still return a list of results based on **likely** relevance. Therefore, it seems that the fuzziness query still somewhat takes into account relevancy in terms of precision and recall. In the second experiment, the max relevancy score was 11.80244. On the other hand, the max relevancy score for the first experiment was 3.5609713 (constant scoring usually applied to wildcard search by default), and all documents were assigned a similar low score until the end. However, it is not recommended to rely on fuzzy matching based on scoring purposes.

In the third experiment using fuzziness and the word, "Children's", did not produce as many results as the first experiment, yet it produced many more results than the second experiment. There are a couple of reasons why this may have happened. The second experiment searched based off of "Children", which in this case, was a search term that may not have had room for much variation in the documents. In other words, using the word "Children" with fuzziness may have been too narrow where an approximate match had the same result as an exact match. This is evident when the same query is run without fuzziness using "\"Children\"" (exact matching), which produces the same results. Therefore, it seems that the search criteria may have been too precise. However, when the criteria was changed to "Children's", this allowed for a wider search, and it was also a term that had many more possibilities for variation in the documents. Like the first experiment, the third experiment seemed to have produced matches based on CHILDREN, Childrens, Children's, and even CHILDRN'S as well. In the context of precision and recall, the search query with "Children's" seemed to have been the most balanced out of the three with regard to relevancy.

Requirement #2:

Rerun Experiments #1, #2, #3 but searching for "Child" matches

Experiment #1: Query using regex:

In [212]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
{"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clasue in SQL


                              {"query_string": {
                                           "query": "*Child*",  #using regex of c
hildren  to match all posssible combinations of "Children"
                                           "fields": ["Facility Type","Violation
s","DBA Name"] #Multi-field matching query
                                            }
                             }



                             ]
                    }
                }
        }
results1 = es.search(index='food_inspections', body=query, scroll='1h')
```

In [213]:
```
sid1 = results1['_scroll_id']
scroll_size1 = results1['hits']['total']

print('sid = ', sid1)
print('Scroll Size = ', scroll_size1)
```

```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFxlFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcY
xZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXGQWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAg
FxoFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcaxZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXGYWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgFxnFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAIBcaRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXGwWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgFxqFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  774
```

In [214]:
```
len(results1['hits']['hits'])
```

Out[214]:
```
774
```

In [215]:
```
count1 = 0
list_of_lAT_LONG_pairs1 = []
while(scroll_size1 > 0):

    for inspection1 in results1['hits']['hits']:                    #Iterating each
results of  the query
        current_location_lAT_LONG1 = []
        document1 = inspection1['_source']
        count1 = count1 + 1

        #Defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document1.keys():
            if 'Longitude' in document1.keys():
                if 'Address' in document1.keys():
                    if(document1['Latitude'] != None and document1['Longitude'] !=
None  and document1['Address'] != None):
                        current_location_lAT_LONG1.append(float(document1['Latitud
e']))     #Appending Latitude and Longitude into the list
                        current_location_lAT_LONG1.append(float(document1['Longitu
de']))
                        list_of_lAT_LONG_pairs1.append(current_location_lAT_LONG1)

    results1 = es.scroll(scroll_id = sid1, scroll = '2m')
    sid1 = results1['_scroll_id']                                        #Changing the sc
roll-id
    scroll_size1 = len(results1['hits']['hits'])

print("the total number of match with child using wild card:",count1)
```

```
the total number of match with child using wild card: 774
```

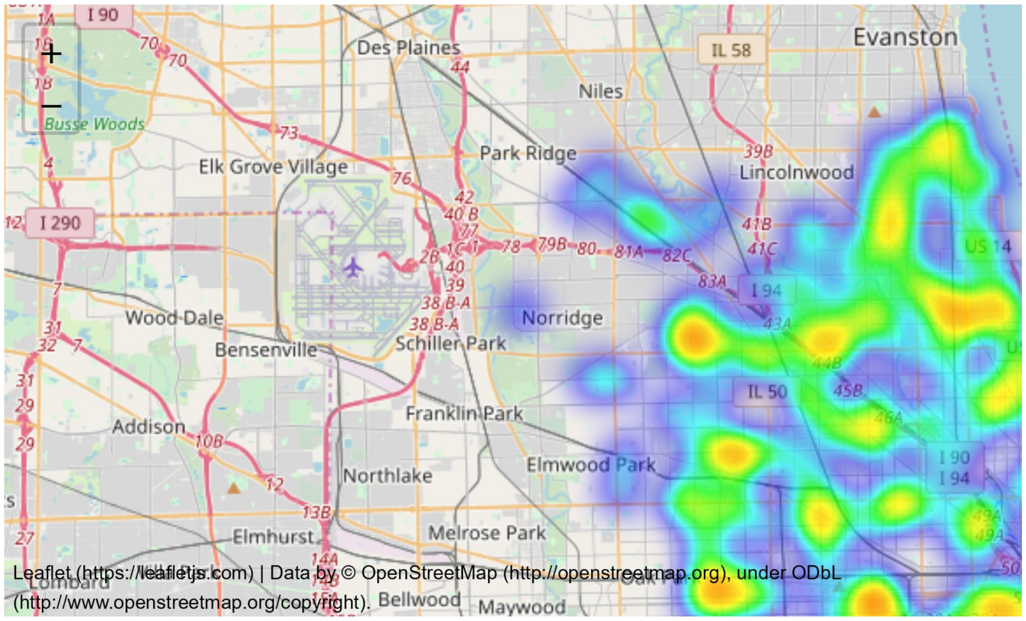
In [216]:
```
len(list_of_lAT_LONG_pairs1)
```

Out[216]:
```
774
```

In [217]:
```
import folium
from folium import plugins
```

```
In [218]: #Plot of query matches on Chicago HeatMap using Wild Card
          chicago_map1 = folium.Map([41.90293279, -87.70769386], zoom_start=11)
          chicago_map1.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs1, radius=15))
          chicago_map1
```

Out[218]:



Experiment #2: Query using "Child" and "fuzziness" = "1":

```
In [219]: query = {
                  'size' : 10000,
                  'query': {
                      'bool': {
                              'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
          {"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clasue in SQL

                                    {"query_string": {
                                                    "query": "Child",
                                                    "fuzziness": "1",
                                                    "fields": ["Facility Type","Violation
          s","DBA Name"]
                                                     }
                                    }

                                    ]
                               }
                          }
                  }
          results2 = es.search(index='food_inspections', body=query,scroll='1h')
```

In [220]:
```
sid2 = results2['_scroll_id']
scroll_size2 = results2['hits']['total']

print('sid = ', sid2)
print('Scroll Size = ', scroll_size2)
```

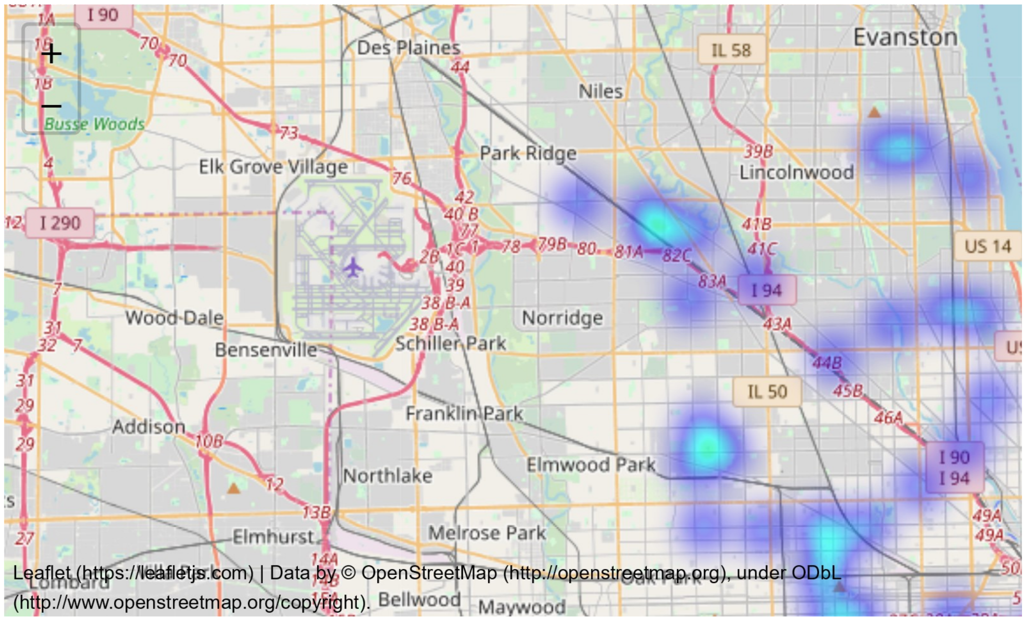
```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFxvFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcb
hZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXHEWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAg
FxzFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBccBZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXHIWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgFx1FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAIBcdhZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXHQWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgFx3FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  158
```

In [221]:
```
count2 = 0
list_of_lAT_LONG_pairs2 = []

while(scroll_size2 > 0):

    for inspection2 in results2['hits']['hits']:                              #Iterating
each  results of  the query
        current_location_lAT_LONG2 = []
        document2 = inspection2['_source']
        count2 = count2 + 1

        #defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document2.keys():
            if 'Longitude' in document2.keys():
                if 'Address' in document2.keys():
                    if(document2['Latitude'] != None and document2['Longitude'] !=
None  and document2['Address'] != None):
                        current_location_lAT_LONG2.append(float(document2['Latitud
e']))  #Appending Latitude and Longitude into the list
                        current_location_lAT_LONG2.append(float(document2['Longitu
de']))
                        list_of_lAT_LONG_pairs2.append(current_location_lAT_LONG2)

    results2 = es.scroll(scroll_id = sid2, scroll = '2m')
    sid2 = results2['_scroll_id']                                   #Changing the scroll-i
d
    scroll_size2 = len(results2['hits']['hits'])

print("Total number of match with Child using Fuzziness:",count2)
```

```
Total number of match with Child using Fuzziness: 158
```

In [222]:
```
#Plot of query matches for "Child" on Chicago HeatMap using Fuzziness
chicago_map2 = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map2.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs2, radius=15))
chicago_map2
```

Out[222]:



Experiment #3: Query using "Child's" and "fuzziness" = "1":

In [223]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}}, {"match" : {'Risk':
{"query": 'Risk 1 (High)', "operator": "and"}} }, # same as where clasue in SQL

                            {"query_string": {
                                            "query": "Child's",
                                            "fuzziness": "1",
                                            "fields": ["Facility Type","Violation
s","DBA Name"]
                                             }
                            }

                            ]
                     }
                 }
        }
results3 = es.search(index='food_inspections', body=query,scroll='1h')
```

In [224]:
```
sid3 = results3['_scroll_id']
scroll_size3 = results3['hits']['total']

print('sid = ', sid3)
print('Scroll Size = ', scroll_size3)
```

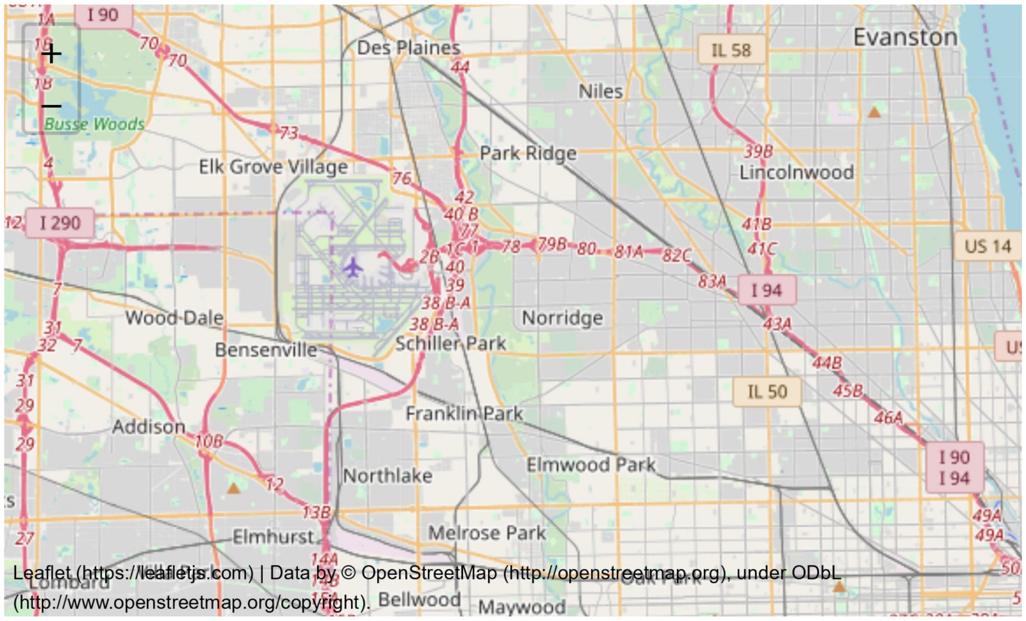
```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFx4FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcf
hZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXHoWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAg
Fx5FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcfRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXHwWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgFx7FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAIBcgRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXH8WZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgFyAFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  8
```

In [225]:
```
count3 = 0
list_of_lAT_LONG_pairs3 = []

while(scroll_size3 > 0):

    for inspection3 in results3['hits']['hits']:                         #Iterating
each  results of  the query
        current_location_lAT_LONG3 = []
        document3 = inspection3['_source']
        count3 = count3 + 1

        #defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document3.keys():
            if 'Longitude' in document3.keys():
                if 'Address' in document3.keys():
                    if(document3['Latitude'] != None and document3['Longitude'] !=
None  and document3['Address'] != None):
                        current_location_lAT_LONG3.append(float(document3['Latitud
e']))  #Appending Latitude and Longitude into the list
                        current_location_lAT_LONG3.append(float(document3['Longitu
de']))
                        list_of_lAT_LONG_pairs3.append(current_location_lAT_LONG3)

    results3 = es.scroll(scroll_id = sid3, scroll = '2m')
    sid3 = results3['_scroll_id']                             #Changing the scroll-i
d
    scroll_size3 = len(results3['hits']['hits'])

print("Total number of match with Child's using Fuzziness:",count3)
```

```
Total number of match with Child's using Fuzziness: 8
```

```
In [226]: #Plot of query matches for "Child" on Chicago HeatMap using Fuzziness
          chicago_map3 = folium.Map([41.90293279, -87.70769386], zoom_start=11)
          chicago_map3.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs3, radius=15))
          chicago_map3
```

Out[226]:



Requirement #3:

In Experiment #4 we have obtained the list of frequent vilators, produce a table that shows DBA Name, number of violations and number of licenses issued for every DBA Name

In [227]:
```
dba_violators = df_top_frequent_violators.groupby('DBA Name', as_index=False).coun
t()
dba_violators = dba_violators[['DBA Name', 'Inspection ID']]
dba_violators.rename(columns={'Inspection ID':'# of Violations'}, inplace=True)
dba_licenses = df_top_frequent_violators[['DBA Name', 'License #']]
dba_licenses_unique = dba_licenses.drop_duplicates()
dba_licenses_issued = dba_licenses_unique.groupby('DBA Name', as_index=False).coun
t()[['DBA Name', 'License #']]
dba_licenses_issued.rename(columns={'License #':'# of Licenses Issued'}, inplace=T
rue)
dba_violators_licenses_count = dba_violators.merge(dba_licenses_issued, left_on='D
BA Name', right_on='DBA Name')
dba_violators_licenses_count.sort_values(by='# of Violations', ascending=False)
```

Out[227]:

	DBA Name	# of Violations	# of Licenses Issued
5	BUSY BUMBLE BEE ACADEMY DAYCARE	9	3
4	BOTTLES TO BOOKS LEARNING CENTER	8	2
0	A CHILD'S WORLD EARLY LEARNING CENTER	7	2
2	AMAZING GRACE DAYCARE CENTER	7	2
11	FIRMAN COMMUNITY SERVICES	6	4
20	Little People's Day Care & Kindergarten, Inc.	6	2
19	LITTLE KIDS VILLAGE LEARNING	6	2
18	LINCOLN KING DAY CARE	6	1
16	KIDS R FIRST LEARNING ACADEMY	6	2
14	JELLYBEAN LEARNING CENTER	6	4
25	THE WORLD IS YOUR'S CHILD CARE & LEARNING CENT...	6	2
9	EARLY CHILDHOOD EDUCARE CENTER	6	3
8	DISCOVERY LEARNING ACADEMY, INC.	6	2
7	COMMONWEALTH DAYCARE CENTER	6	1
10	EZZARD CHARLES DAYCARE CENTER	5	2
12	GRANT DAY CARE INC	5	1
1	ADA S MCKINLEY MAGGIE DRUMMON	5	2
15	KENYATTA'S DAYCARE	5	3
17	LAKE & PULASKI CHILD DEVELOPMENT CENTER	5	2
6	CENTRO INFANTIL	5	3
3	ANGELS	5	2
21	MOLADE' CHILD DEVELOPMENT CENTER	5	1
22	MONTESSORI ACDY. INFT/TOD. CNT	5	3
23	THE CRYSTAL PALACE EARLY LITERACY ZONE	5	2
24	THE EDSEL ALBERT AMMONS NURSER	5	1
13	GREATER INSTITUTE AME CHURCH	5	2

Requirement #4:

Use the results of Experiment #4 to plot on the Heatmap those frequent violators who have obtained 3 business licenses or more under the same DBA Name through out the liftime of their business

In [228]:
```
#All records for facilities that serve children
query ={
      'size' : 10000,
      'query': {
          "bool" : {
              "should":[    {'match' : {'Facility Type': {"query" : 'Daycare (2 -
6 Years)',"operator":"and"}}},
                            {'match' : {'Facility Type':{"query" : 'Daycare Above
and Under 2 Years',"operator": "and"}}},
                            {'match' : {'Facility Type':{"query" : 'CHILDRENS SERV
ICES FACILITY',"operator" : "and"}}},
                       ],
                       "minimum_should_match" : 1,

             }
      }
}

results4 = es.search(index='food_inspections', body=query,scroll='1h')
```

In [229]:
```
sid4 = results4['_scroll_id']
scroll_size4 = results4['hits']['total']

print('sid = ', sid4)
print('Scroll Size = ', scroll_size4)
```

```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFyJFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBch
RZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXIQWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAg
FyDFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBcghZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXIcWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgFyGFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAIBcixZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXIgWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgFyKFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  4402
```

In [230]:
```
#All records for every facility dataframe
index_row = 0
all_child_facilities = pd.DataFrame()
for dba_all in results4['hits']['hits']:
    if '_source' in dba_all:
        index_row += 1
        all_child_facility = pd.DataFrame(dba_all['_source'], index=[index_row])
        all_child_facilities = all_child_facilities.append(all_child_facility)
```

In [231]:
```
all_child_facilities.head()
```

Out[231]:

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City	State	Zip	Ins
1	2129448	PATHWAYS TO LEARNING CHILD CARE CENTER	PATHWAYS TO LEARNING CHILD CARE CENTER	2215780.0	CHILDRENS SERVICES FACILITY	Risk 1 (High)	3450 - 3454 W 79TH ST	CHICAGO	IL	60652.0	12/
2	2135260	PATHWAYS TO LEARNING CHILD CARE CENTER	PATHWAYS TO LEARNING CHILD CARE CENTER	2215780.0	CHILDRENS SERVICES FACILITY	Risk 1 (High)	3450 - 3454 W 79TH ST	CHICAGO	IL	60652.0	01/
3	2116981	PATHWAYS TO LEARNING CHILD CARE CENTER	PATHWAYS TO LEARNING CHILD CARE CENTER	2215780.0	CHILDRENS SERVICES FACILITY	Risk 1 (High)	3450 - 3454 W 79TH ST	CHICAGO	IL	60652.0	12/
4	1931940	PATHWAYS TO LEARNING CHILD CARE CENTER	PATHWAYS TO LEARNING CHILD CARE CENTER	2215780.0	CHILDRENS SERVICES FACILITY	Risk 1 (High)	3450 - 3454 W 79TH ST	CHICAGO	IL	60652.0	06/
5	1516621	PATHWAYS TO LEARNING CHILD CARE CENTER	PATHWAYS TO LEARNING CHILD CARE CENTER	2215780.0	CHILDRENS SERVICES FACILITY	Risk 1 (High)	3450 - 3454 W 79TH ST	CHICAGO	IL	60652.0	10/

In [232]:
```
#Top violators lifetime
violators_list = dba_violators_licenses_count['DBA Name'].tolist()

top_violators_life = []

for violator in violators_list:
    for i in range(0, len(all_child_facilities)):
        if all_child_facilities.iloc[i]['DBA Name']==violator:
            top_violators_life.append(all_child_facilities.iloc[i])

top_violators_lifetime = pd.DataFrame(top_violators_life)
```

In [233]:
```
top_violators_lifetime.head()
```

Out[233]:

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City	State	Zip	Inspe
2314	1285266	A CHILD'S WORLD EARLY LEARNING CENTER	A CHILD'S WORLD EARLY LEARNING CENTER	1357825.0	Daycare (2 - 6 Years)	Risk 1 (High)	2145 E 83RD ST	CHICAGO	IL	60617.0	09/14
2328	1214996	A CHILD'S WORLD EARLY LEARNING CENTER	A CHILD'S WORLD EARLY LEARNING CENTER	1768092.0	Daycare (2 - 6 Years)	Risk 1 (High)	2145 E 83RD ST	CHICAGO	IL	60617.0	05/18
2569	1235192	A CHILD'S WORLD EARLY LEARNING CENTER	A CHILD'S WORLD EARLY LEARNING CENTER	1357825.0	Daycare (2 - 6 Years)	Risk 1 (High)	2145 E 83RD ST	CHICAGO	IL	60617.0	09/12
2658	68455	A CHILD'S WORLD EARLY LEARNING CENTER	A CHILD'S WORLD EARLY LEARNING CENTER	1357825.0	Daycare (2 - 6 Years)	Risk 1 (High)	2145 E 83RD ST	CHICAGO	IL	60617.0	06/11
2887	537211	A CHILD'S WORLD EARLY LEARNING CENTER	A CHILD'S WORLD EARLY LEARNING CENTER	1768092.0	Daycare (2 - 6 Years)	Risk 1 (High)	2145 E 83RD ST	CHICAGO	IL	60617.0	02/07

In [234]:
```
#Latitude and longitude for those violators that have 3 or more issued licenses. M
ultiple DBAs under same name
#are not duplicates because they have different latitude, longitude, and addresses
meaning they have different
#location recorded, which is included in the heatmap

top_dba_lic = top_violators_lifetime[['DBA Name', 'License #', 'Latitude', 'Longit
ude']].drop_duplicates()
top_dba_number = top_dba_lic.groupby('DBA Name', as_index=False).count()[['DBA Nam
e', 'License #']]
top_dba_number_3 = top_dba_number.loc[top_dba_number['License #'] >= 3]
top_dba_loc = top_dba_lic[['DBA Name', 'Latitude', 'Longitude']].drop_duplicates()

top_dba_list = top_dba_number_3['DBA Name'].tolist()

top_dba_number_3_loc = []

for top_dba in top_dba_list:
    for r in range(0, len(top_dba_loc)):
        if top_dba_loc.iloc[r]['DBA Name']==top_dba:
            top_dba_number_3_loc.append(top_dba_loc.iloc[r])

top_dba_number_3_loc_df = pd.DataFrame(top_dba_number_3_loc)
top_dba_number_3_loc_df
```

Out[234]:

	DBA Name	Latitude	Longitude
26	AMAZING GRACE DAYCARE CENTER	41.691646	-87.642214
2009	BUSY BUMBLE BEE ACADEMY DAYCARE	41.777092	-87.606004
46	CENTRO INFANTIL	41.902822	-87.695990
8	EARLY CHILDHOOD EDUCARE CENTER	41.802458	-87.624433
1828	EARLY CHILDHOOD EDUCARE CENTER	41.802143	-87.625747
1881	FIRMAN COMMUNITY SERVICES	41.809097	-87.627599
1956	FIRMAN COMMUNITY SERVICES	41.805367	-87.616633
2051	FIRMAN COMMUNITY SERVICES	41.796235	-87.630405
223	JELLYBEAN LEARNING CENTER	41.765802	-87.616183
1923	JELLYBEAN LEARNING CENTER	41.739343	-87.663008
113	KENYATTA'S DAYCARE	41.759085	-87.567448
643	LITTLE KIDS VILLAGE LEARNING	41.778861	-87.716745
1913	LITTLE KIDS VILLAGE LEARNING	41.764689	-87.690441
141	MONTESSORI ACDY. INFT/TOD. CNT	41.707740	-87.643003

In [235]:
```
top_dba_lat_lng = top_dba_number_3_loc_df[['Latitude', 'Longitude']]
top_dba_lat_lng.head()
```

Out[235]:

	Latitude	Longitude
26	41.691646	-87.642214
2009	41.777092	-87.606004
46	41.902822	-87.695990
8	41.802458	-87.624433
1828	41.802143	-87.625747

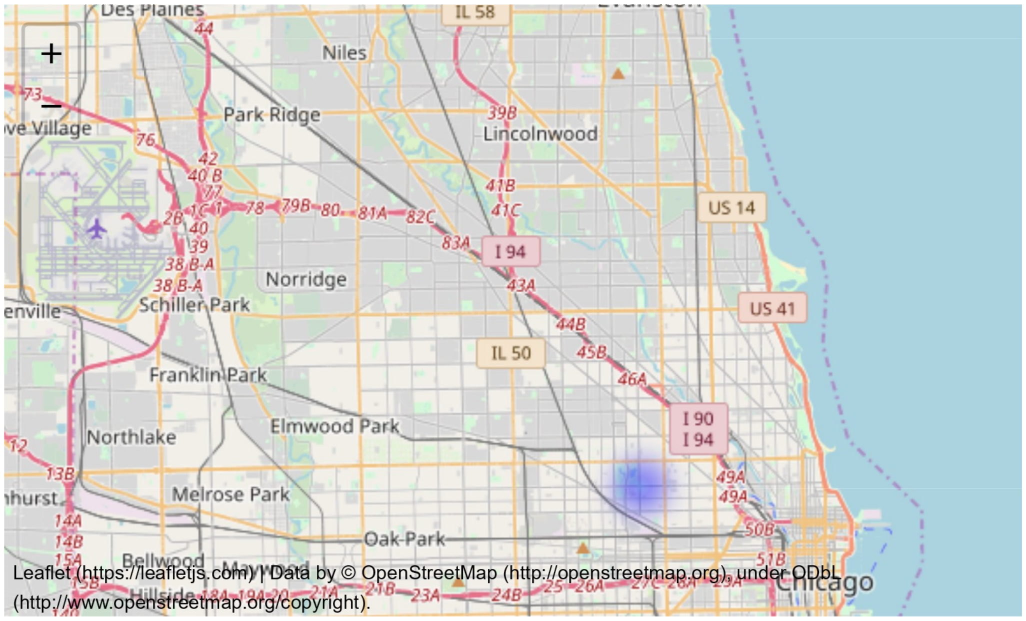
In [236]:
```
#Check for null values:
top_dba_lat_lng.isnull().sum()
```

Out[236]:
```
Latitude     0
Longitude    0
dtype: int64
```

In [237]:
```
top_dba_lic_3_coord = []

for e in range(0, len(top_dba_lat_lng)):
    location_ll = []
    location_ll.append(float(top_dba_lat_lng.iloc[e]['Latitude']))
    location_ll.append(float(top_dba_lat_lng.iloc[e]['Longitude']))
    top_dba_lic_3_coord.append(location_ll)
```

In [238]:
```
#Heatmap of frequent violators who have obtained 3 business licenses or more under the same DBA Name
#throughout the lifetime of their business
top_dba_lic_3_heat_map = folium.Map([41.891551, -87.607375],zoom_start = 11)
top_dba_lic_3_heat_map.add_child(plugins.HeatMap(top_dba_lic_3_coord ,radius=15))
```

Out[238]:



Requirement #5:

Plot on the Heatmap those facilites that serve children but failed inspections with high risk, and **MICE DROPPINGS were OBSERVED** in the Violations; you have to **execlude** violations that stated **NO MICE DROPPINGS were OBSERVED**

In [239]:
```
# Facilites that serve children but failed inspections with high risk, and "MICE D
ROPPINGS were OBSERVED", excluding
# violations that stated "NO MICE DROPPINGS were OBSERVED"
query ={
      'size' : 10000,
      'query': {
          "bool" : {
              "should":[    {'match' : {'Facility Type': {"query" : 'Daycare (2 -
6 Years)',"operator":"and"}}},
                            {'match' : {'Facility Type':{"query" : 'Daycare Above
and Under 2 Years',"operator": "and"}}},
                            {'match' : {'Facility Type':{"query" : 'CHILDRENS SERV
ICES FACILITY',"operator" : "and"}}},
                       ],
                       "minimum_should_match" : 1,

              "must":[{'match' : {'Results': 'Fail'}}, {"match" : {'Risk': {"quer
y": 'Risk 1 (High)', "operator": "and"}}},

                           {"match_phrase": {
                               "Violations": {
                                "query": "MICE DROPPINGS were OBSERVED"
                                   }
                                }
                            }
                    ],

              "must_not":[

                            {"match_phrase": {
                                "Violations": {
                                "query": "NO MICE DROPPINGS were OBSERVED"
                                    }
                                }
                            }
                        ]

                    }
                }
        }
results5 = es.search(index='food_inspections', body=query,scroll='1h')
```

In [240]:
```
sid5 = results5['_scroll_id']
scroll_size5 = results5['hits']['total']

print('sid = ', sid5)
print('Scroll Size = ', scroll_size5)
```

```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFz_FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBc_
RZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXQYWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAg
Fz-FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBdABZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
ACAXQIWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgF0BFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAIBdAxZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXQQWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAgF0FFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  3
```

In [241]:
```
results5
```

Out[241]:

```
{'_scroll_id': 'DnF1ZXJ5VGhlbkZldGNoCgAAAAAAgFz_FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEA
AAAAIBc_RZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXQYWZ3FCbVVtNHFSNnFKN1BocUlZLW1m
UQAAAAAAgFz-FmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAIBdABZncUJtVW00cVI2cUo3UGhxSVkt
bWZRAAAAAACAXQIWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAgF0BFmdxQm1VbTRxUjZxSjdQaHFJ
WS1tZlEAAAAAAIBdAxZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAACAXQQWZ3FCbVVtNHFSNnFKN1Bo
cUlZLW1mUQAAAAAAgF0FFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=',
 'took': 24,
 'timed_out': False,
 '_shards': {'total': 10, 'successful': 10, 'skipped': 0, 'failed': 0},
 'hits': {'total': 3,
  'max_score': 23.559385,
  'hits': [{'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '251208',
    '_score': 23.559385,
    '_source': {'Inspection ID': 251208,
     'DBA Name': 'CHICAGO MONTESSORI, NFP',
     'AKA Name': 'CHICAGO MONTESSORI, NFP',
     'License #': 1845360.0,
     'Facility Type': 'Daycare (2 - 6 Years)',
     'Risk': 'Risk 1 (High)',
     'Address': '1713 W CULLOM AVE ',
     'City': 'CHICAGO',
     'State': 'IL',
     'Zip': 60613.0,
     'Inspection Date': '06/23/2010',
     'Inspection Type': 'Canvass',
     'Results': 'Fail',
     'Violations': "18. NO EVIDENCE OF RODENT OR INSECT OUTER OPENINGS PROTECTED
/RODENT PROOFED, A WRITTEN LOG SHALL BE MAINTAINED AVAILABLE TO THE INSPECTORS -
Comments: All necessary control measures shall be used to effectively minimize o
r eliminate the presence of rodents, roaches, and other vermin and insects on th
e premises of all food establishments, in food-transporting vehicles, and in ven
ding machines. \nWE (MISS BELL)OBSERVED EVIDENCE OF MICE DROPPINGS IN BOTH CLASS
ROOMS.CLASS ROOM(MS.HILGEN)MICE DROPPINGS WERE OBSERVED UNDER 1 COMPARTMENT SIN
K,(25 OR MORE). CLASS ROOM(ELLIS)MICE DROPPINGS WERE OBSERVED UNDER 1 COMPARTMEN
T SINK AND STORAGE AREA(40 OR MORE).INSTRUCTED TO REMOVE DROPPINGS AND SANITIZE
AREA.ELEVATE ITEMS FROM UNDER THE SINKS AND STORAGE AREA,6'OFF AND ABOVE.SERIOUS
VIOLATION:7-38-020 HOOO062879 | 34. FLOORS: CONSTRUCTED PER CODE, CLEANED, GOOD
REPAIR, COVING INSTALLED, DUST-LESS CLEANING METHODS USED - Comments: The floors
shall be constructed per code, be smooth and easily cleaned, and be kept clean a
nd in good repair.SEAL PLASTIC COVING IN PREP AREA, SOUTH WALL.",
     'Latitude': 41.9596298877,
     'Longitude': -87.6713590304,
     'Location': '(41.959629887701624, -87.6713590303878)'}},
   {'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '1106884',
    '_score': 20.213863,
    '_source': {'Inspection ID': 1106884,
     'DBA Name': 'LITTLE ANGELS DAY CARE CENTER',
     'AKA Name': 'LITTLE ANGELS DAY CARE CENTER',
     'License #': 2215473.0,
     'Facility Type': 'Daycare Above and Under 2 Years',
     'Risk': 'Risk 1 (High)',
     'Address': '6407 N MAPLEWOOD AVE ',
     'City': 'CHICAGO',
     'State': 'IL',
     'Zip': 60645.0,
     'Inspection Date': '09/27/2013',
     'Inspection Type': 'License',
     'Results': 'Fail',
     'Violations': "12. HAND WASHING FACILITIES: WITH SOAP AND SANITARY HAND DRY
ING DEVICES, CONVENIENT AND ACCESSIBLE TO FOOD PREP AREA - Comments: NO PAPER TO
```

In [242]:
```
index_row = 0
all_mice_facilities = pd.DataFrame()
for dba_all in results5['hits']['hits']:
    if '_source' in dba_all:
        index_row += 1
        all_mice_facility = pd.DataFrame(dba_all['_source'], index=[index_row])
        all_mice_facilities = all_mice_facilities.append(all_mice_facility)
```

In [243]:
```
#Sanity test of facilities
all_mice_facilities['Facility Type'].value_counts()
```

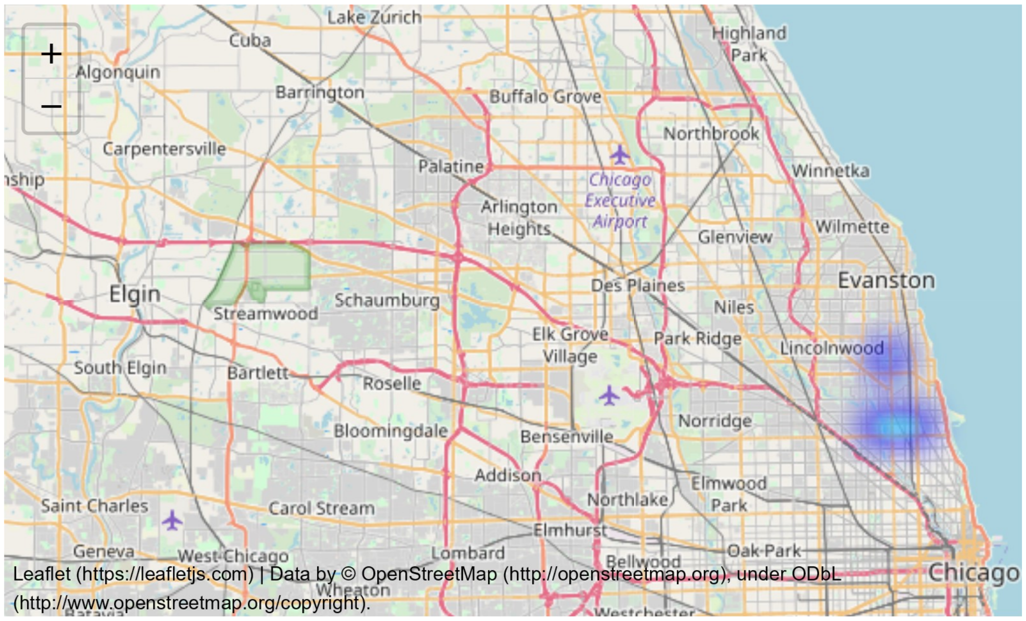
Out[243]:
```
Daycare (2 - 6 Years)              2
Daycare Above and Under 2 Years    1
Name: Facility Type, dtype: int64
```

In [244]:
```
count5 = 0
list_of_lAT_LONG_pairs5 = []
while(scroll_size5 > 0):

    for inspection5 in results5['hits']['hits']:                      #Iterating each
results of  the query
        current_location_lAT_LONG5 = []
        document5 = inspection5['_source']
        count5 = count5 + 1

        #Defensive coding to ensure we have the fields in the inspection documents
        if 'Latitude' in document5.keys():
            if 'Longitude' in document5.keys():
                if 'Address' in document5.keys():
                    if(document5['Latitude'] != None and document5['Longitude'] !=
None  and document5['Address'] != None):
                        current_location_lAT_LONG5.append(float(document5['Latitud
e']))    #Appending Latitude and Longitude into the list
                        current_location_lAT_LONG5.append(float(document5['Longitu
de']))
                        list_of_lAT_LONG_pairs5.append(current_location_lAT_LONG5)

    results5 = es.scroll(scroll_id = sid5, scroll = '2m')
    sid5 = results5['_scroll_id']                                    #Changing the sc
roll-id
    scroll_size5 = len(results5['hits']['hits'])

print("The total number of failed inspections where exactly 'MICE DROPPINGS were O
BSERVED' stated:",count5)
```

```
The total number of failed inspections where exactly 'MICE DROPPINGS were OBSERV
ED' stated: 3
```

```
In [245]: #Plot of query matches for failed inspections where "MICE DROPPINGS were OBSERVED"
          on Chicago HeatMap
          chicago_map5 = folium.Map([41.90293279, -87.70769386], zoom_start=10)
          chicago_map5.add_child(plugins.HeatMap(list_of_lAT_LONG_pairs5, radius=15))
          chicago_map5
```

Out[245]:


Lake Zurich
Cuba
Highland Park
Algonquin
Barrington
Buffalo Grove
Northbrook
Carpentersville
Palatine
Chicago Executive Airport
Winnetka
Arlington Heights
Wilmette
Glenview
Evanston
Elgin
Des Plaines
Schaumburg
Streamwood
Niles
Elk Grove Village
Park Ridge
Lincolnwood
South Elgin
Bartlett
Roselle
Norridge
Bloomingdale
Bensenville
Addison
Elmwood Park
Saint Charles
Carol Stream
Northlake
Elmhurst
Geneva
West Chicago
Lombard
Bellwood
Oak Park
Chicago
Wheaton
Westchester
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).