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)** 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)**

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 [1]:
```
# Lets load the CSV Chicago Food Inspections dataset into a dataframe
import pandas as pd

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

In [2]:
```
df.head()
```

Out[2]:

	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City	State	
0	2144807	SAMMY'S RED HOT	SAMMY'S RED HOT	2578852.0	Restaurant	Risk 1 (High)	238 W DIVISION ST	CHICAGO	IL	606
1	2144802	CAFE NILLY	NILLY CAFE	2578631.0	Restaurant	Risk 1 (High)	60 E ADAMS ST	CHICAGO	IL	606
2	2144800	EVITA ARGENTINIAN STEAKHOUSE	EVITA	2464488.0	Restaurant	Risk 1 (High)	6112 N LINCOLN AVE	CHICAGO	IL	606
3	2144791	PHO SPICIER THAI	PHO SPICIER THAI	2578881.0	NaN	All	1320 W DEVON AVE	CHICAGO	IL	606
4	2144789	RED SNAPPER	JIMMY'S BEST	2232836.0	Restaurant	Risk 1 (High)	1347 E 87TH ST	CHICAGO	IL	606

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)**

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)** to your laptop
- Getting Started with **elasticsearch (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 [3]:
```
#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 [5]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}},{'match' : {'Risk': '
Risk 1 (High)'}}, # same as where clasue in SQL

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



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

In [6]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']

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

```
sid =  DnF1ZXJ5VGhlbkZldGNoCgAAAAAAAqQYFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAAKkH
RZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpB4WZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAA
qQaFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAAKkGRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAA
AACpBsWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAAqQcFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAA
AAAAAKkHxZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpCAWZ3FCbVVtNHFSNnFKN1BocUlZLW1mU
QAAAAAAAqQhFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=
Scroll Size =  617
```

In [7]:
```
type(results)
```

Out[7]:
```
dict
```

In [8]:
```
results
```

Out[8]:

```
{'_scroll_id': 'DnF1ZXJ5VGhlbkZldGNoCgAAAAAAAqQYFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEA
AAAAAAKkHRZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpB4WZ3FCbVVtNHFSNnFKN1BocUlZLW1m
UQAAAAAAAqQaFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAAKkGRZncUJtVW00cVI2cUo3UGhxSVkt
bWZRAAAAAAACpBsWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAAqQcFmdxQm1VbTRxUjZxSjdQaHFJ
WS1tZlEAAAAAAAKkHxZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpCAWZ3FCbVVtNHFSNnFKN1Bo
cUlZLW1mUQAAAAAAAqQhFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=',
 'took': 27,
 'timed_out': False,
 '_shards': {'total': 10, 'successful': 10, 'skipped': 0, 'failed': 0},
 'hits': {'total': 617,
  'max_score': 3.5609713,
  'hits': [{'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '565537',
    '_score': 3.5609713,
    '_source': {'Inspection ID': 565537,
     'DBA Name': 'SALVATION ARMY TEMPLE HEAD START',
     'AKA Name': 'SALVATION ARMY TEMPLE HEAD START',
     'License #': 21716.0,
     'Facility Type': 'Daycare (2 Years)',
     'Risk': 'Risk 1 (High)',
     'Address': '1 N OGDEN AVE ',
     'City': 'CHICAGO',
     'State': 'IL',
     'Zip': 60607.0,
     'Inspection Date': '10/04/2011',
     'Inspection Type': 'License',
     'Results': 'Fail',
     'Violations': "27. TOILET ROOMS ENCLOSED CLEAN, PROVIDED WITH HAND CLEANSER
, SANITARY HAND DRYING DEVICES AND PROPER WASTE RECEPTACLES - Comments: In all f
ood establishments, toilet facilities shall be kept clean and in good repair and
shall include an adequate supply of hot and cold or tempered water, soap, and ap
proved sanitary towels or other approved hand-drying devices. \n \nTHE CHILDRENS
WASHROOM TOILET DOES NOT WORK, REPAIR THE TOILET SO IT IS FULLY FUNCTIONAL. | 32
. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND MAINTAIN
ED - Comments: All food and non-food contact equipment and utensils shall be smo
oth, easily cleanable, and durable, and shall be in good repair. \nTHE KITCHEN S
TOVE,REFRIGERATORS AND CAN OPENER ARE DIRTY; CLEAN, RINSE AND SANITIZE ALL EQUIP
MENT. \nTHE TURNED OFF MOLDY REFRIGERATOR, CABINETS AND SHEAVING IN THE CHILDREN
'S ROOM ARE DIRTY; WASH, RINSE AND SANITIZE THE REFRIGERATOR,CABINETS AND SHELVI
NG IN THE CHILDREN'S ROOM. | 34. FLOORS: CONSTRUCTED PER CODE, CLEANED, GOOD REP
AIR, COVING INSTALLED, DUST-LESS CLEANING METHODS USED - Comments: The floors sh
all be constructed per code, be smooth and easily cleaned, and be kept clean and
in good repair. \n \nTHE FLOORS ARE DIRTY AROUND THE CHILDRENS ROOM PERIMETER, C
LEAN THE FLOORS IN DETAIL AND REPLACE THE MISSING NORTH BASEBOARDS. | 35. WALLS,
CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAIR, SURFACES CLEAN A
ND DUST-LESS CLEANING METHODS - Comments: The walls and ceilings shall be in goo
d repair and easily cleaned. \n \nTHE CHILDREN'S ROOM WALLS AND CEILINGS ARE DIR
TY, CLEAN AND REPAINT THE WALLS AS NEEDED AND REPLACE THE DIRTY, WATER DAMGED CE
ILING.",
     'Latitude': 41.8814369069,
     'Longitude': -87.6659213595,
     'Location': '(41.88143690693931, -87.66592135954576)'}},
   {'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '543569',
    '_score': 3.5609713,
    '_source': {'Inspection ID': 543569,
     'DBA Name': 'CHILDREN OF TOMORROW LEARNING',
     'AKA Name': 'CHILDREN OF TOMORROW LEARNING',
     'License #': 2108390.0,
     'Facility Type': 'Daycare (2 - 6 Years)',
     'Risk': 'Risk 1 (High)',
     'Address': '7322 S DAMEN AVE ',
```

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 [9]:
```
results['hits']['hits']
```

```
Out[9]: [{'_index': 'food_inspections',
          '_type': 'food_inspection',
          '_id': '565537',
          '_score': 3.5609713,
          '_source': {'Inspection ID': 565537,
           'DBA Name': 'SALVATION ARMY TEMPLE HEAD START',
           'AKA Name': 'SALVATION ARMY TEMPLE HEAD START',
           'License #': 21716.0,
           'Facility Type': 'Daycare (2 Years)',
           'Risk': 'Risk 1 (High)',
           'Address': '1 N OGDEN AVE ',
           'City': 'CHICAGO',
           'State': 'IL',
           'Zip': 60607.0,
           'Inspection Date': '10/04/2011',
           'Inspection Type': 'License',
           'Results': 'Fail',
           'Violations': "27. TOILET ROOMS ENCLOSED CLEAN, PROVIDED WITH HAND CLEANSER,
        SANITARY HAND DRYING DEVICES AND PROPER WASTE RECEPTACLES - Comments: In all foo
        d establishments, toilet facilities shall be kept clean and in good repair and s
        hall include an adequate supply of hot and cold or tempered water, soap, and app
        roved sanitary towels or other approved hand-drying devices. \n \nTHE CHILDRENS
        WASHROOM TOILET DOES NOT WORK, REPAIR THE TOILET SO IT IS FULLY FUNCTIONAL. | 32
        . FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND MAINTAIN
        ED - Comments: All food and non-food contact equipment and utensils shall be smo
        oth, easily cleanable, and durable, and shall be in good repair. \nTHE KITCHEN S
        TOVE,REFRIGERATORS AND CAN OPENER ARE DIRTY; CLEAN, RINSE AND SANITIZE ALL EQUIP
        MENT. \nTHE TURNED OFF MOLDY REFRIGERATOR, CABINETS AND SHEAVING IN THE CHILDREN
        'S ROOM ARE DIRTY; WASH, RINSE AND SANITIZE THE REFRIGERATOR,CABINETS AND SHELVI
        NG IN THE CHILDREN'S ROOM. | 34. FLOORS: CONSTRUCTED PER CODE, CLEANED, GOOD REP
        AIR, COVING INSTALLED, DUST-LESS CLEANING METHODS USED - Comments: The floors sh
        all be constructed per code, be smooth and easily cleaned, and be kept clean and
        in good repair. \n \nTHE FLOORS ARE DIRTY AROUND THE CHILDRENS ROOM PERIMETER, C
        LEAN THE FLOORS IN DETAIL AND REPLACE THE MISSING NORTH BASEBOARDS. | 35. WALLS,
        CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAIR, SURFACES CLEAN A
        ND DUST-LESS CLEANING METHODS - Comments: The walls and ceilings shall be in goo
        d repair and easily cleaned. \n \nTHE CHILDREN'S ROOM WALLS AND CEILINGS ARE DIR
        TY, CLEAN AND REPAINT THE WALLS AS NEEDED AND REPLACE THE DIRTY, WATER DAMGED CE
        ILING.",
           'Latitude': 41.8814369069,
           'Longitude': -87.6659213595,
           'Location': '(41.88143690693931, -87.66592135954576)'}},
         {'_index': 'food_inspections',
          '_type': 'food_inspection',
          '_id': '543569',
          '_score': 3.5609713,
          '_source': {'Inspection ID': 543569,
           'DBA Name': 'CHILDREN OF TOMORROW LEARNING',
           'AKA Name': 'CHILDREN OF TOMORROW LEARNING',
           'License #': 2108390.0,
           'Facility Type': 'Daycare (2 - 6 Years)',
           'Risk': 'Risk 1 (High)',
           'Address': '7322 S DAMEN AVE ',
           'City': 'CHICAGO',
           'State': 'IL',
           'Zip': 60636.0,
           'Inspection Date': '08/25/2011',
           'Inspection Type': 'License',
           'Results': 'Fail',
           'Violations': '9. WATER SOURCE: SAFE, HOT & COLD UNDER CITY PRESSURE - Commen
        ts: All food establishments shall be provided with an adequate supply of hot and
        cold water under pressure properly connected to the city water supply. \nNO HOT
        WATER INSTRUCTED MANAGER TO PROVIDE HOT WATER UNDER CITY PRESSURE. | 12. HAND WA
        SHING FACILITIES: WITH SOAP AND SANITARY HAND DRYING DEVICES, CONVENIENT AND ACC
```

In [10]:
```
len(results['hits']['hits'])
```

Out[10]: 617

In [11]:
```
count = 0
list_of_lAT_LONG_pairs = []
while(scroll_size > 0):

    for inspection in results['hits']['hits']:                    #Iterating each  r
esults 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'] != No
ne  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 scrol
l-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: 617
```

In [12]:
```
document.keys()
```

Out[12]:
```
dict_keys(['Inspection ID', 'DBA Name', 'AKA Name', 'License #', 'Facility Type'
, 'Risk', 'Address', 'City', 'State', 'Zip', 'Inspection Date', 'Inspection Type
', 'Results', 'Violations', 'Latitude', 'Longitude', 'Location'])
```

In [13]:
```
list_of_lAT_LONG_pairs[:3]
```

Out[13]:
```
[[41.8814369069, -87.6659213595],
 [41.760441801, -87.6735652436],
 [41.9531127244, -87.7800185741]]
```

In [14]:
```
len(list_of_lAT_LONG_pairs)
```

Out[14]: 617

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**

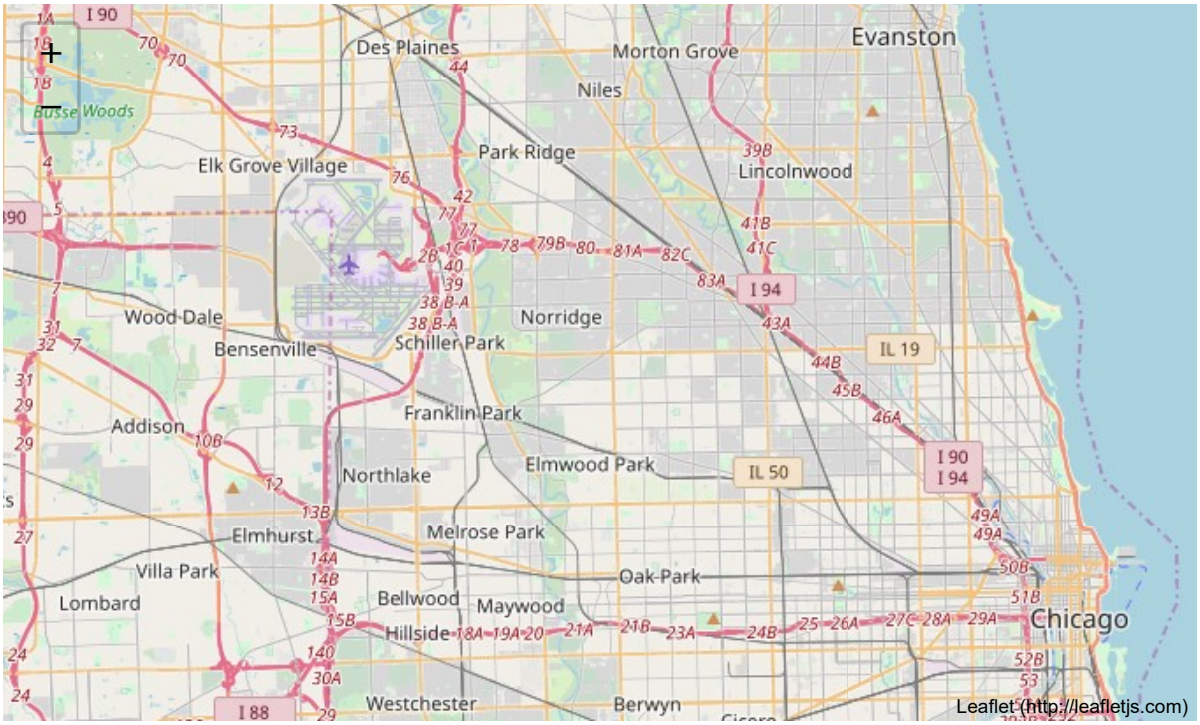
For the different configuration paramteres for HeatMap, you can access the docs at this URL: **HeatMap (https://github.com/python-visualization/folium/blob/master/folium/plugins/heat_map.py)**

In [15]:
```
import folium
from folium import plugins

print(folium.__version__)
```

```
0.5.0
```

In [16]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

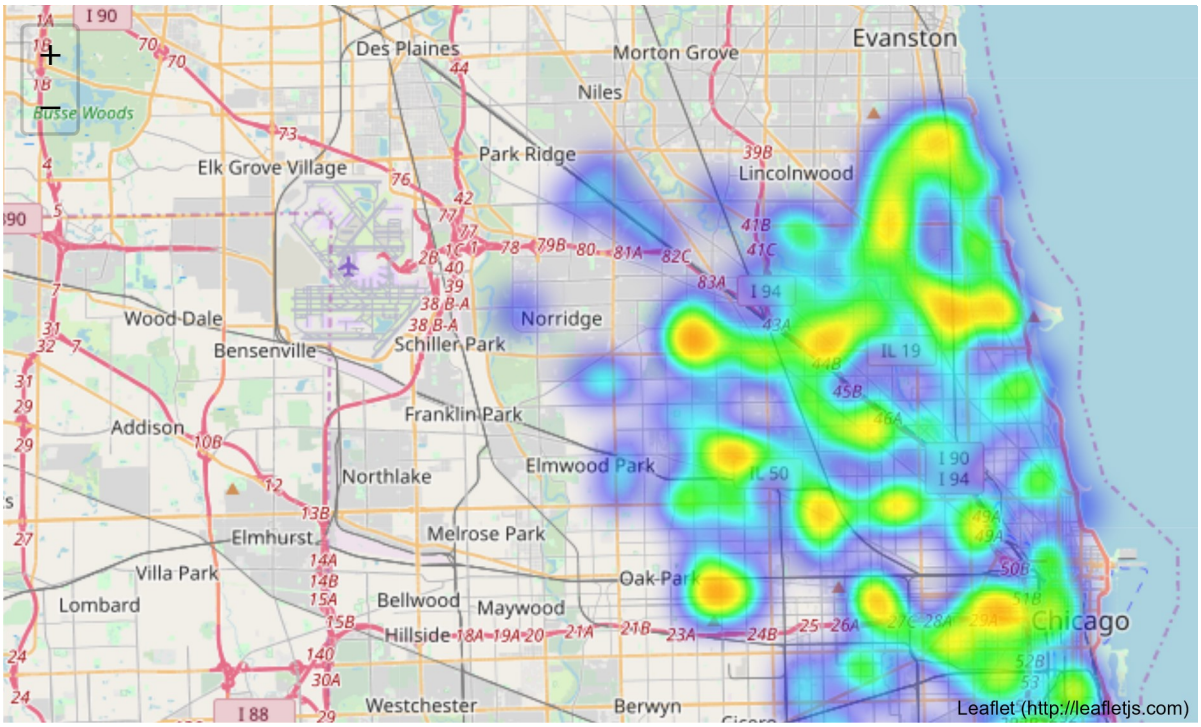
Out[16]:



Create the HeatMap

In [17]:
```
# Lets plot the query matches on Chicago HeatMap

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

Out[17]:



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)**

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

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

In [53]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}},{'match' : {'Risk': '
Risk 1 (High)'}},
                             {"query_string": {
                                             "query": "Children",
                                             "fuzziness": "1",
                                             "fields": ["Facility Type","inspection.
Violations","DBA Name"]
                                              }
                             }

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

In [54]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']
```

In [20]:
```
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'] != No
ne  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: 18
```

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

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

In [21]:
```
query = {
        'size' : 10000,
        'query': {
            'bool': {
                    'must' : [{'match' : {'Results': 'Fail'}},{'match' : {'Risk': '
Risk 1 (High)'}},
                             {"query_string": {
                                             "query": "Children's",
                                             "fuzziness": "1",
                                             "fields": ["Facility Type","inspection.
Violations","DBA Name"]
                                              }
                             }

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

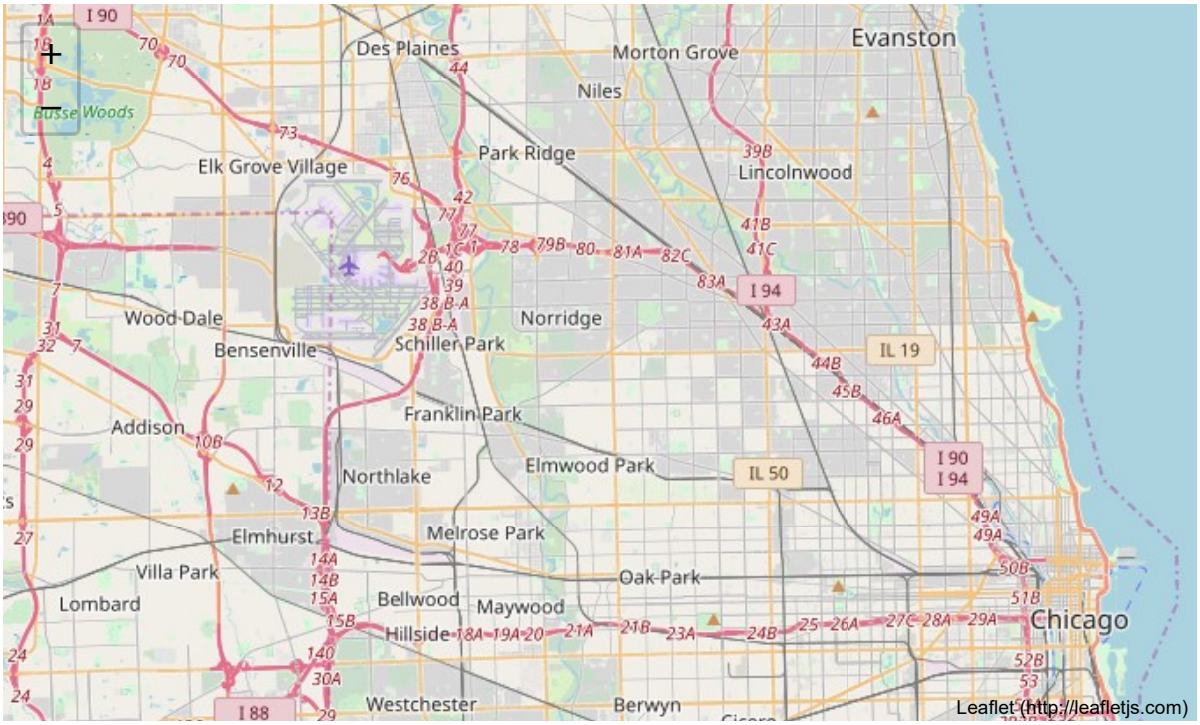
In [22]:
```
sid = results['_scroll_id']
scroll_size = results['hits']['total']
```

In [23]:
```
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'] != No
ne  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: 386
```

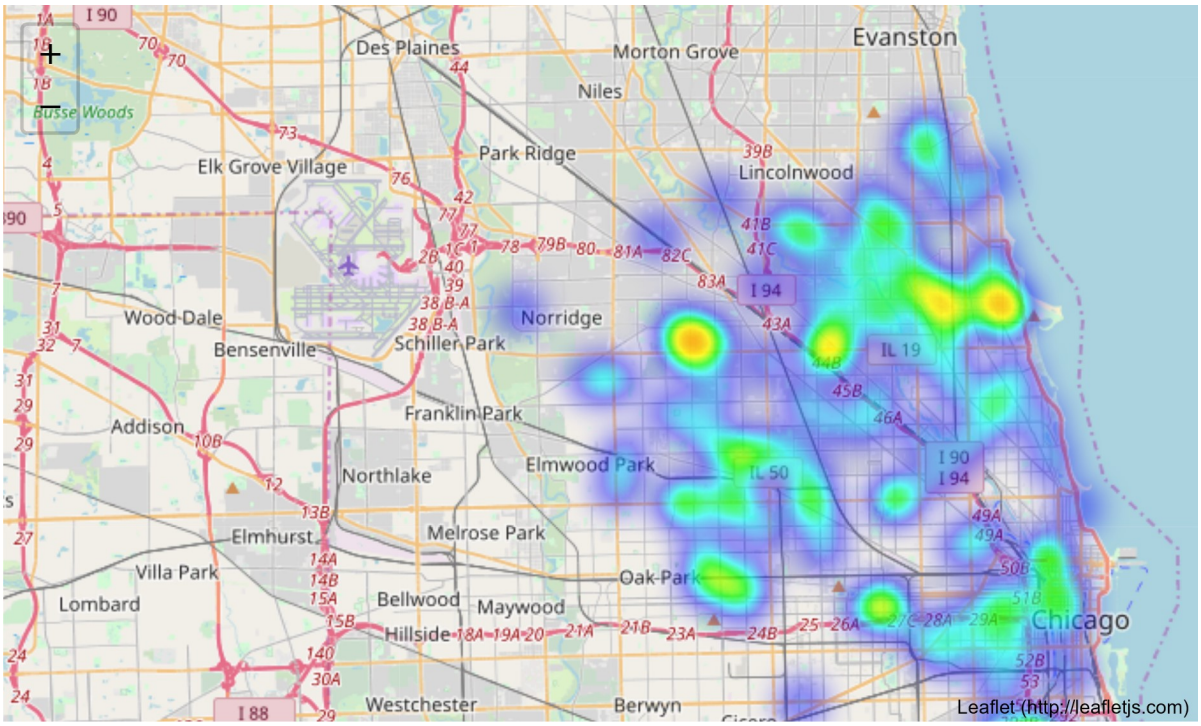
In [24]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

Out[24]:



In [25]:
```
# 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[25]:



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 [26]:
```
query ={
      'size' : 10000,
      'query': {
          "bool" : {
              "should":[    {'match' : {'Facility Type': {"query" : 'Daycare (2 - 6
Years)',"operator":"and"}}},
                            {'match' : {'Facility Type':{"query" : 'Daycare Above a
nd Under 2 Years',"operator": "and"}}},
                            {'match' : {'Facility Type':{"query" : 'CHILDRENS SERVI
CES FACILITY',"operator" : "and"}}},
                        ],
                        "minimum_should_match" : 1,
                        "filter" : [{"match" : {'Results': {"query": 'Fail', "operat
or": "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 [27]:
```
results
```

Out[27]:

```
{'_scroll_id': 'DnF1ZXJ5VGhlbkZldGNoCgAAAAAAAqTWFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEA
AAAAAAKk2BZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpNUWZ3FCbVVtNHFSNnFKN1BocUlZLW1m
UQAAAAAAAqTXFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlEAAAAAAAK1BZncUJtVW00cVI2cUo3UGhxSVkt
bWZRAAAAAAACpNkWZ3FCbVVtNHFSNnFKN1BocUlZLW1mUQAAAAAAAqTaFmdxQm1VbTRxUjZxSjdQaHFJ
WS1tZlEAAAAAAAK3RZncUJtVW00cVI2cUo3UGhxSVktbWZRAAAAAAACpNsWZ3FCbVVtNHFSNnFKN1Bo
cUlZLW1mUQAAAAAAAqTcFmdxQm1VbTRxUjZxSjdQaHFJWS1tZlE=',
 'took': 20,
 'timed_out': False,
 '_shards': {'total': 10, 'successful': 10, 'skipped': 0, 'failed': 0},
 'hits': {'total': 819,
  'max_score': 9.698278,
  'hits': [{'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '2116981',
    '_score': 9.698278,
    '_source': {'Inspection ID': 2116981,
     'DBA Name': 'PATHWAYS TO LEARNING CHILD CARE CENTER',
     'AKA Name': 'PATHWAYS TO LEARNING CHILD CARE CENTER',
     'License #': 2215780.0,
     'Facility Type': 'CHILDRENS SERVICES FACILITY',
     'Risk': 'Risk 1 (High)',
     'Address': '3450 - 3454 W 79TH ST ',
     'City': 'CHICAGO',
     'State': 'IL',
     'Zip': 60652.0,
     'Inspection Date': '12/11/2017',
     '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: 5 MICE DROPPING ALONG WALL BASE BEHIND BOOK CASE, 20 DROPPINGS SCATTER
ED ALONG WALLBASE UNDER STORAGE SHELVES IN PREP AREA. INSTD TO REMOVE DROPPINGS,
CLEAN AND SANITIZE ALL AFFECTED AREAS. CONTACT PEST CONTROL FOR SERVICE. VIOLATI
ON 7-38-020 SERIOUS. | 35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER C
ODE: GOOD REPAIR, SURFACES CLEAN AND DUST-LESS CLEANING METHODS - Comments: HALF
WALLS NOTED AT TODDLERS TOILET ROOM. INSTD TO HAVE TOILET ROOM FULLY ENCLOSED (W
ALLS FLOOR TO CEILING) AND INSTALL DOOR.  | 41. PREMISES MAINTAINED FREE OF LITT
ER, UNNECESSARY ARTICLES, CLEANING  EQUIPMENT PROPERLY STORED - Comments: EMPLOY
EE BELONGINGS IMPROPERLY STORED. INSTD TO STORE IN DESIGNATED AREA. | 31. CLEAN
MULTI-USE UTENSILS AND SINGLE SERVICE ARTICLES PROPERLY STORED: NO REUSE OF SING
LE SERVICE ARTICLES - Comments: CLEAN MULTI USE UTENSILS IMPROPERLY STORED. INST
D TO STORE HANDLE UP. | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED
, CONSTRUCTED AND MAINTAINED - Comments: CUTTING BOARD WORN, STAINED SURFACES. I
NSTD TO REPLACE AND MAINTAIN CUTTING BOARDS.',
     'Latitude': 41.7498738065,
     'Longitude': -87.709315712,
     'Location': '(41.749873806501896, -87.70931571203879)'}},
   {'_index': 'food_inspections',
    '_type': 'food_inspection',
    '_id': '419760',
    '_score': 8.72406,
    '_source': {'Inspection ID': 419760,
     'DBA Name': 'LAKE & PULASKI CHILD DEVELOPMENT CENTER',
     'AKA Name': 'LAKE & PULASKI CHILD DEVELOPMENT CENTER',
     'License #': 1928369.0,
     'Facility Type': 'Daycare Above and Under 2 Years',
     'Risk': 'Risk 1 (High)',
     'Address': '316 N PULASKI RD ',
     'City': 'CHICAGO',
     'State': 'IL',
     'Zip': 60624.0,
     'Inspection Date': '10/22/2010',
     'Inspection Type': 'Consultation',
     'Results': 'Fail',
```

In [28]:
```
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 "h
its" 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 [29]:
```
# 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 "h
its" 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_inde
x])
                df_frequent_violator['doc_count'] = doc_count
                df_frequent_violator['score'] = score
                df_top_frequent_violators = df_top_frequent_violators.append(df_fre
quent_violator)
```

In [30]:
```
df_top_frequent_violators
```

Out[30]:

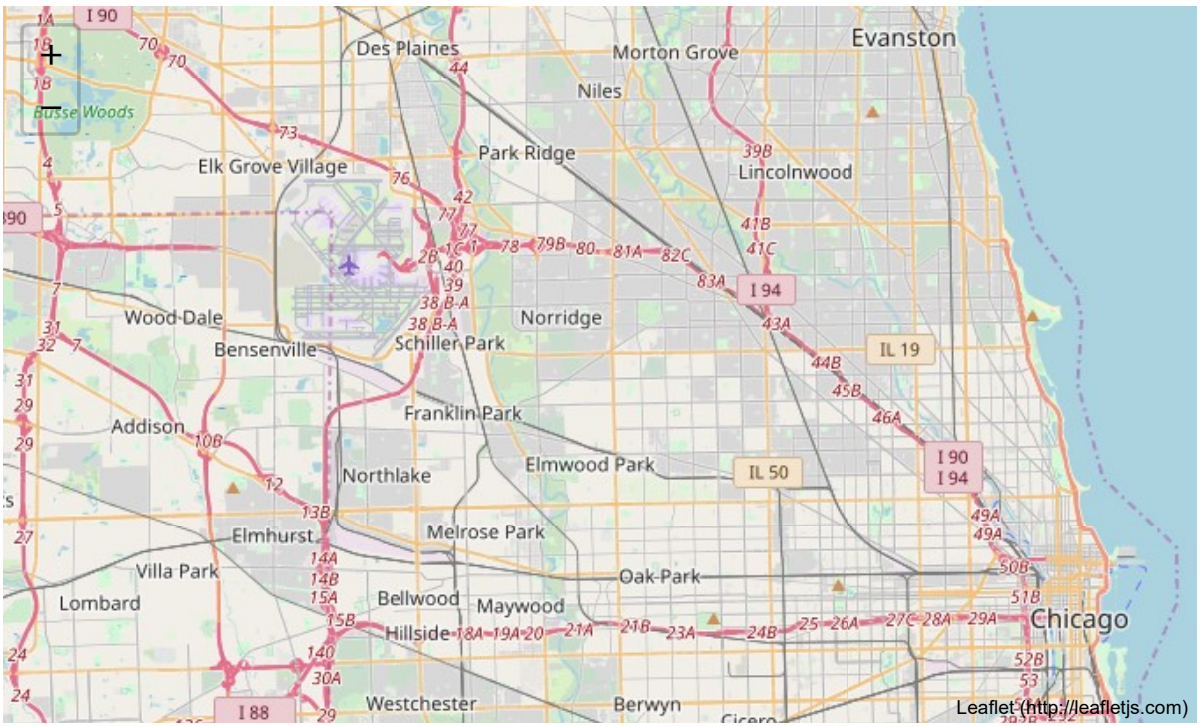
	Inspection ID	DBA Name	AKA Name	License #	Facility Type	Risk	Address	City
1	1319663	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	2215472.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO
2	1229713	BUSY BUMBLE BEE ACADEMY DAYCARE	BUSY BUMBLE BEE ACADEMY DAYCARE	3793.0	Daycare (2 - 6 Years)	Risk 1 (High)	6450 S COTTAGE GROVE AVE	CHICAGO
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
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
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
6	1229718	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
7	531382	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
8	531381	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
9	1229850	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
		BOTTLES TO BOOKS	BOTTLES TO BOOKS		Daycare Above	Risk	6014 S	

In [31]:
```
# Lets print the number of violations for every DBA NAME

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

Out[31]:
```
BUSY BUMBLE BEE ACADEMY DAYCARE                              9
BOTTLES TO BOOKS LEARNING CENTER                             8
A CHILD'S WORLD EARLY LEARNING CENTER                        7
AMAZING GRACE DAYCARE CENTER                                 7
EARLY CHILDHOOD EDUCARE CENTER                               6
LINCOLN KING DAY CARE                                        6
JELLYBEAN LEARNING CENTER                                    6
DISCOVERY LEARNING ACADEMY, INC.                             6
KIDS R FIRST LEARNING ACADEMY                                6
FIRMAN COMMUNITY SERVICES                                    6
COMMONWEALTH DAYCARE CENTER                                  6
LITTLE KIDS VILLAGE LEARNING                                 6
THE WORLD IS YOUR'S CHILD CARE & LEARNING CENTER INC.        6
Little People's Day Care & Kindergarten, Inc.                6
ADA S MCKINLEY MAGGIE DRUMMON                                5
KENYATTA'S DAYCARE                                           5
EZZARD CHARLES DAYCARE CENTER                                5
LAKE & PULASKI CHILD DEVELOPMENT CENTER                      5
CENTRO INFANTIL                                              5
ANGELS                                                       5
GREATER INSTITUTE AME CHURCH                                 5
MONTESSORI ACDY. INFT/TOD. CNT                               5
GRANT DAY CARE INC                                           5
THE CRYSTAL PALACE EARLY LITERACY ZONE                       5
MOLADE' CHILD DEVELOPMENT CENTER                             5
THE EDSEL ALBERT AMMONS NURSER                               5
Name: DBA Name, dtype: int64
```

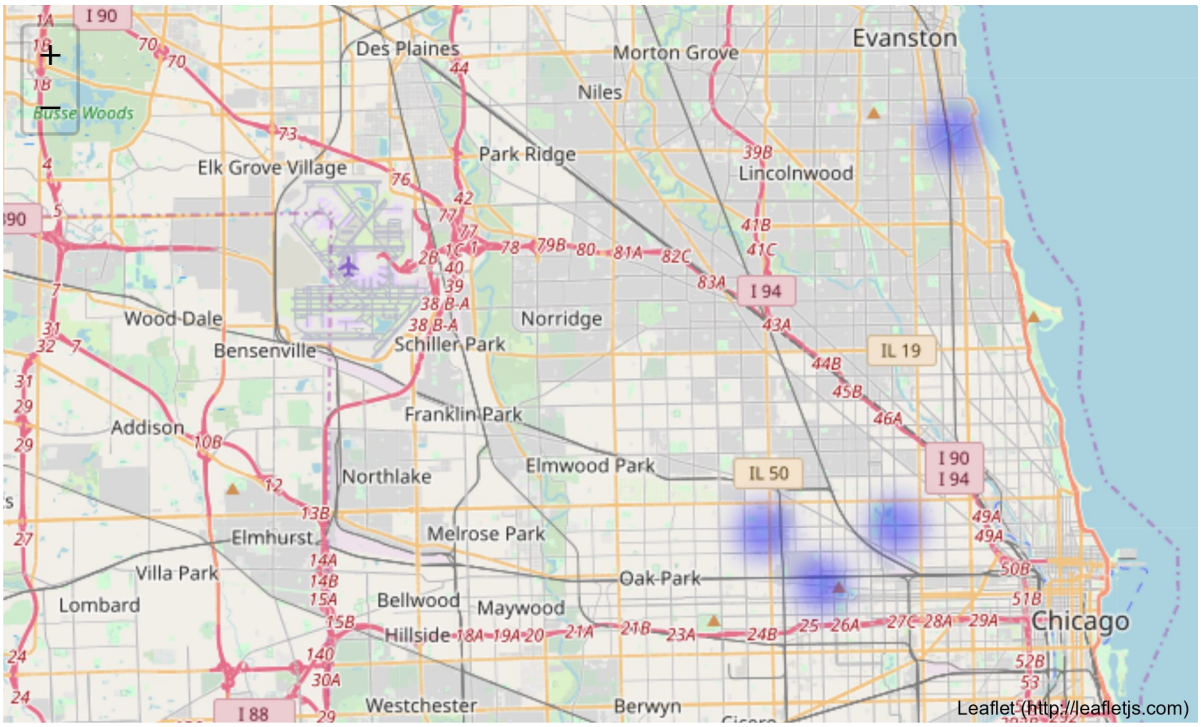
In [32]:
```
chicago_map = folium.Map([41.90293279, -87.70769386], zoom_start=11)
chicago_map
```

Out[32]:



In [33]:
```
# 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[33]:



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)**

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

... **Write your comparative analysis and commentary in this cell** ...

Requirement #2:

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

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

Requirement #4:

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