Clustering Tuscaloosa, Montgomery, Brimingham Restaurants in Alabama

1. Introduction

1.1 Background

This is the capstone project of IBM Data Science Professional Certificate. In this project, I will use the Foursquare location data to explore venues in three city Tuscaloosa, Montgomery, Brimingham in Alabama, and use Zomato api to get all the restaurants in these three city, and then come up with a problem that I can use the Foursquare location data to solve.

1.2 Problem

I will explore these restaurants and try to answer the question: "Where is the appropirate restaurant to invest in these three city?".

1.3 Interest

Alabama, also known as the Yellowhammer State, the Cotton State, and the Heart of Dixie, is considered to be the thirtieth largest state by area. What the state has become today is a product of the economic hardship it suffered in the past. In terms of business development, Alabama has many reasons to be proud of. Alabama is a hub for medicine and automotive manufacturing, with recent successes ranging from Remington to Polaris. Companies such as Airbus, Hyundai, and Mercedes-Benz locating plants in the state demonstrate how pro-business Alabama can be for big companies. A new study reveals that Alabama is likewise an attractive destination for small business entrepreneurs and owners. So invest in a restaurant in this city is an attractive idea for anyone who want to invest.

2. Data

To solve the problem, i'll use the **Foursquare API** to explore venue surrounding each restaurants. Foursquare outlines these high-level venue categories with more sub-categories.

```
Arts & Entertainment (4d4b7104d754a06370d81259)
College & University (4d4b7105d754a06372d81259)
Event (4d4b7105d754a06373d81259)
Food (4d4b7105d754a06374d81259)
Nightlife Spot (4d4b7105d754a06376d81259)
Outdoors & Recreation (4d4b7105d754a06377d81259)
Professional & Other Places (4d4b7105d754a06375d81259)
Residence (4e67e38e036454776db1fb3a)
Shop & Service (4d4b7105d754a06378d81259)
Travel & Transport (4d4b7105d754a06379d81259)
```

We'll be querying the number of venues in each category in a 1000m radius around each restaurants. This radius was chosen because 1000m is a reasonable walking distance.

Then i'll use the **Zomato API** to get the restaurants data from each city, Tuscaloosa, Montgomery, Brimingham in Alabama, using their coordinate.

```
Tuscaloosa (33.2098,-87.5692)
Montgomery (32.3792, -86.3077)
Brimingham (33.5186, -86.8104)
```

In [1]:

```
import pandas as pd
import numpy as np
import folium
import json
import requests
from pandas.io.json import json_normalize
from geopy.geocoders import Nominatim
import pprint
import matplotlib.pyplot as plt
import seaborn as sns

CLIENT_ID = "FFEFW3ZL0BZ4UXSVKSI0XSGC3FGI30HCTRQ5GPZLSJUR0GSI"
CLIENT_SECRET = "NO3YYEJ35KLKMMAAEWLKRQOCEV0VG35RK5ENJX4NKR4DZAOG"
VERSION = "20180604"
#key zomatto
key = "509c6a5ac1f97d60b55b4dc3da482107"
```

Tuscaloosa restaurants

In [2]:

```
tuscaloosa_restaurants = pd.DataFrame()
tuscaloosa_coord = (33.2098,-87.5692)
pages = [0,20,40,60,80]
for page in pages:
    uris = "https://developers.zomato.com/api/v2.1/search?lat={}&lon={}&apikey=
{}&radius={}&start={}&count={}".format(tuscaloosa_coord[0],tuscaloosa_coord[1],k
ey,10000,page,20)
    request_ = requests.get(uris).json()
    csv_ = json_normalize(request_['restaurants'])
    tuscaloosa_restaurants = tuscaloosa_restaurants.append(csv_)
```

Clean the dataframe

In [3]:

```
#Reset the index of the dataframe, becuase the dataframe is the result of the ap
pend method
tuscaloosa_restaurants.reset_index(inplace=True, drop=True)
#The feature that i'll use
feature = ['restaurant.name','restaurant.cuisines','restaurant.average_cost_for_
two','restaurant.user_rating.aggregate_rating','restaurant.timings','restaurant.
all_reviews_count',
           'restaurant.user_rating.votes','restaurant.location.latitude','restau
rant.location.longitude']
#Only use the feature that i consider as more valuable
tuscaloosa_restaurants = tuscaloosa_restaurants[feature]
#Convert some of the features that supose to be fload
col_to_convert_to_float = ['restaurant.user_rating.aggregate_rating','restauran
t.user_rating.votes', 'restaurant.location.latitude', 'restaurant.location.longi
tude']
for col in col_to_convert_to_float:
    tuscaloosa_restaurants[col] = tuscaloosa_restaurants[col].astype(float)
#make a coordinate column
tuscaloosa_restaurants['coordinate'] = [(i,j) for i, j in zip(tuscaloosa_restaur
ants['restaurant.location.latitude'], tuscaloosa_restaurants['restaurant.locatio
n.longitude'])]
```

In [4]:

```
tuscaloosa_restaurants.head()
```

Out[4]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	City Cafe	Diner, Seafood, Southern	10	
1	Swen Chinese Restaurant	Chinese	25	
2	DePalma's Italian Cafe	Italian	40	
3	Cypress Inn	American, Seafood, Steak	40	
4	Mugshots Grill & Bar	American, Burger, Bar Food	10	

Montgomery restaurants

In [5]:

```
montgomery_restaurants = pd.DataFrame()
montgomery_coord = (32.3792, -86.3077)
pages = [0,20,40,60,80]
for page in pages:
    uris = "https://developers.zomato.com/api/v2.1/search?lat={}&lon={}&apikey=
{}&radius={}&start={}&count={}".format(montgomery_coord[0],montgomery_coord[1],k
ey,10000,page,20)
    request_ = requests.get(uris).json()
    csv_ = json_normalize(request_['restaurants'])
    montgomery_restaurants = montgomery_restaurants.append(csv_)
```

Clean the data with the same prosedure as Tuscaloosa data

In [6]:

```
montgomery_restaurants.reset_index(inplace=True, drop=True)
feature = ['restaurant.name','restaurant.cuisines','restaurant.average_cost_for_
two','restaurant.user_rating.aggregate_rating','restaurant.timings','restaurant.
all_reviews_count',
           'restaurant.user_rating.votes','restaurant.location.latitude','restau
rant.location.longitude']
montgomery_restaurants = montgomery_restaurants[feature]
col_to_convert_to_float = ['restaurant.user_rating.aggregate_rating','restauran
t.user_rating.votes', 'restaurant.location.latitude', 'restaurant.location.longi
tude']
for col in col_to_convert_to_float:
    montgomery_restaurants[col] = montgomery_restaurants[col].astype(float)
montgomery_restaurants['coordinate'] = [(i,j) for i, j in zip(montgomery_restaur
ants['restaurant.location.latitude'], montgomery_restaurants['restaurant.locatio
n.longitude'])]
```

In [7]:

```
montgomery_restaurants.head()
```

Out[7]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	Sa Za	Italian, Pizza	40	
1	Sushi Cafe	Japanese, Sushi	25	
2	Lek's Railroad Thai	Asian, Sushi, Thai	25	
3	Wishbone Cafe	Sandwich, Seafood, Cajun	25	
4	Dreamland BBQ Montgomery	BBQ, Burger, Sandwich	25	

Brimingham restaurants

In [8]:

```
brimingham_restaurants = pd.DataFrame()
pages = [0,20,40,60,80]
brimingham_coord = (33.5186,-86.8104)
for page in pages:
    uris = "https://developers.zomato.com/api/v2.1/search?lat={}&lon={}&apikey=
{}&radius={}&start={}&count={}".format(brimingham_coord[0],brimingham_coord[1],k
ey,10000,page,20)
    request_ = requests.get(uris).json()
    csv_ = json_normalize(request_['restaurants'])
    brimingham_restaurants = brimingham_restaurants.append(csv_)
```

Clean the data

In [9]:

```
brimingham_restaurants.reset_index(inplace=True, drop=True)
feature = ['restaurant.name','restaurant.cuisines','restaurant.average_cost_for_
two','restaurant.user_rating.aggregate_rating','restaurant.timings','restaurant.
all_reviews_count',
           'restaurant.user_rating.votes','restaurant.location.latitude','restau
rant.location.longitude']
brimingham_restaurants = brimingham_restaurants[feature]
col_to_convert_to_float = ['restaurant.user_rating.aggregate_rating','restauran
t.user_rating.votes', 'restaurant.location.latitude', 'restaurant.location.longi
tude']
for col in col_to_convert_to_float:
    brimingham_restaurants[col] = brimingham_restaurants[col].astype(float)
brimingham_restaurants['coordinate'] = [(i,j) for i, j in zip(brimingham_restaur
ants['restaurant.location.latitude'], brimingham_restaurants['restaurant.locatio
n.longitude'])]
```

In [10]:

```
brimingham_restaurants.head()
```

Out[10]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratii
0	Bottega Restaurant and Cafe	Italian, Cafe, Pizza, Mediterranean	40	
1	GianMarco's	Italian, Mediterranean, Southern	40	
2	Surin West	Asian, Sushi, Thai	25	
3	SAW's BBQ	BBQ, Sandwich	25	
4	Joe's Italian Pizza, Pasta & Caffe	Desserts, Italian, Pizza	25	

Get the venues for each city using Foursquare API

In [11]:

```
#The list category of venues
venue_cat = ['Arts_Entertainment', 'College_University', 'Event', 'Nightlife_Spo
t', 'Outdoors_Recreation',
            'Professional_n_Other_Places', 'Shop_Service', 'Travel_Transport',
'Residence']
#The Id for each venue category
venue_cat_id = ['4d4b7104d754a06370d81259','4d4b7105d754a06372d81259','4d4b7105d
754a06373d81259', '4d4b7105d754a06376d81259','4d4b7105d754a06377d81259','4d4b710
5d754a06375d81259','4d4b7105d754a06378d81259','4d4b7105d754a06379d81259', '4e67e
38e036454776db1fb3a']


ra = []

#three towns coordinates (tusaloosa, montgomary, brimingham)
coord = [(33.2098,-87.5692), (32.3792, -86.3077), (33.5186, -86.8104)]

#get the venue for each category and for each city, with radius 20000m
for (i,j) in coord:
    reqs1 = []
    for cat, cat_id in zip(venue_cat,venue_cat_id):
            url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&cl
ient_secret={}&v={}&ll={},{}&radius={}&categoryId={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            i,
            j,
            20000,
            cat_id)
            results = requests.get(url).json()
            reqs1.append(results)
    ra.append(reqs1)
```

Group the venues for each city

In [13]:

```
#convert the address from a list to a string
def concat(x):
    s = x[0]
    for st in x[1:]:
        s += ', '+st
    return s

def city_venues(venue_cat, result_respond):
    result = {}
    for venue, json in zip(venue_cat, result_respond):
        feature = ['categories', 'id', 'location.formattedAddress', 'location.la
t', 'location.lng', 'name']
        data1 = pd.DataFrame(columns = feature)
        for rest in json['response']['groups'][0]['items']:
                d = json_normalize(rest['venue'])
                d = d[feature]
                d['categories'] = [data[0]['name'] for data in d['categories']]
                d['location.formattedAddress'] = d['location.formattedAddress'].
apply(concat)
                data1 = data1.append(d)
        data1.reset_index(inplace=True, drop = True)
        data1['coordinate'] = [(i,j) for i,j in zip(data1['location.lat'], data1
['location.lng'])]
        result[venue] = data1
    return result

Tuscaloosa_venues = city_venues(venue_cat, ra[0])
Montgomery_venues = city_venues(venue_cat, ra[1])
Brimingham_venues = city_venues(venue_cat, ra[2])
```

Measure the distance for every restaurant to every venues for each city

First we difne the Haversine function to measure the distance between two coordinates

In [17]:

```
import math

def haversine(coord1, coord2):
    R = 6372800  # Earth radius in meters
    lat1, lon1 = coord1
    lat2, lon2 = coord2

    phi1, phi2 = math.radians(lat1), math.radians(lat2)
    dphi       = math.radians(lat2 - lat1)
    dlambda    = math.radians(lon2 - lon1)

    a = math.sin(dphi/2)**2 + \
        math.cos(phi1)*math.cos(phi2)*math.sin(dlambda/2)**2

    return 2*R*math.atan2(math.sqrt(a), math.sqrt(1 - a))
```

Then i'll measure the distance for every coordinate restaurant to each city venue from the result above then get the number of each venue category that still inside the 1000m radius and return in it as a list

In [18]:

```
def distance(coord, city_venues):
    a = []
    for cat, dataframe in city_venues.items():
            nearest_distance = [haversine(coord, i) for i in dataframe['coordina
te']]
            a.append(nearest_distance)
    for i in a:
        i = np.array(i)
    a = pd.DataFrame(a).transpose()
    a = list(a[a<=1000].notnull().sum())
    return a
```

Using that function, i'll append the result to each city restaurant dataframe

In [19]:

```
lists1 = np.array([distance(coord, Tuscaloosa_venues) for coord in tuscaloosa_re
staurants['coordinate']])
new_dataframe1 = pd.DataFrame(lists1, columns = venue_cat)
tuscaloosa_restaurants = pd.concat([tuscaloosa_restaurants, new_dataframe1], axi
s = 1)

lists2 = np.array([distance(coord, Montgomery_venues) for coord in montgomery_re
staurants['coordinate']])
new_dataframe2 = pd.DataFrame(lists2, columns = venue_cat)
montgomery_restaurants = pd.concat([montgomery_restaurants, new_dataframe2], axi
s = 1)

lists3 = np.array([distance(coord, Brimingham_venues) for coord in brimingham_re
staurants['coordinate']])
new_dataframe3 = pd.DataFrame(lists3, columns = venue_cat)
brimingham_restaurants = pd.concat([brimingham_restaurants, new_dataframe3], axi
s = 1)
```

In [20]:

```
tuscaloosa_restaurants.head()
```

Out[20]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	City Cafe	Diner, Seafood, Southern	10	
1	Swen Chinese Restaurant	Chinese	25	
2	DePalma's Italian Cafe	Italian	40	
3	Cypress Inn	American, Seafood, Steak	40	
4	Mugshots Grill & Bar	American, Burger, Bar Food	10	

In [21]:

```
montgomery_restaurants.head()
```

Out[21]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	Sa Za	Italian, Pizza	40	
1	Sushi Cafe	Japanese, Sushi	25	
2	Lek's Railroad Thai	Asian, Sushi, Thai	25	
3	Wishbone Cafe	Sandwich, Seafood, Cajun	25	
4	Dreamland BBQ Montgomery	BBQ, Burger, Sandwich	25	

In [22]:

```
brimingham_restaurants.head()
```

Out[22]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	Bottega Restaurant and Cafe	Italian, Cafe, Pizza, Mediterranean	40	
1	GianMarco's	Italian, Mediterranean, Southern	40	
2	Surin West	Asian, Sushi, Thai	25	
3	SAW's BBQ	BBQ, Sandwich	25	
4	Joe's Italian Pizza, Pasta & Caffe	Desserts, Italian, Pizza	25	

After I got the number of for each category venue for each restaurant, for each city. I'll merge the three city restaurant dataframe

In [23]:

```
restaurants = pd.concat([tuscaloosa_restaurants, montgomery_restaurants, briming
ham_restaurants], axis = 0).reset_index(drop=True)
```

In [52]:

```
restaurants.head()
```

Out[52]:

	restaurant.name	restaurant.cuisines	restaurant.average_cost_for_two	restaurant.user_ratir
0	City Cafe	Diner, Seafood, Southern	10	
1	Swen Chinese Restaurant	Chinese	25	
2	DePalma's Italian Cafe	Italian	40	
3	Cypress Inn	American, Seafood, Steak	40	
4	Mugshots Grill & Bar	American, Burger, Bar Food	10	

In [25]:

```
restaurants.to_csv("restaurants.csv")
```

3. Methodology

3.1 Explore Dataset

Visualize the Restaurants venue

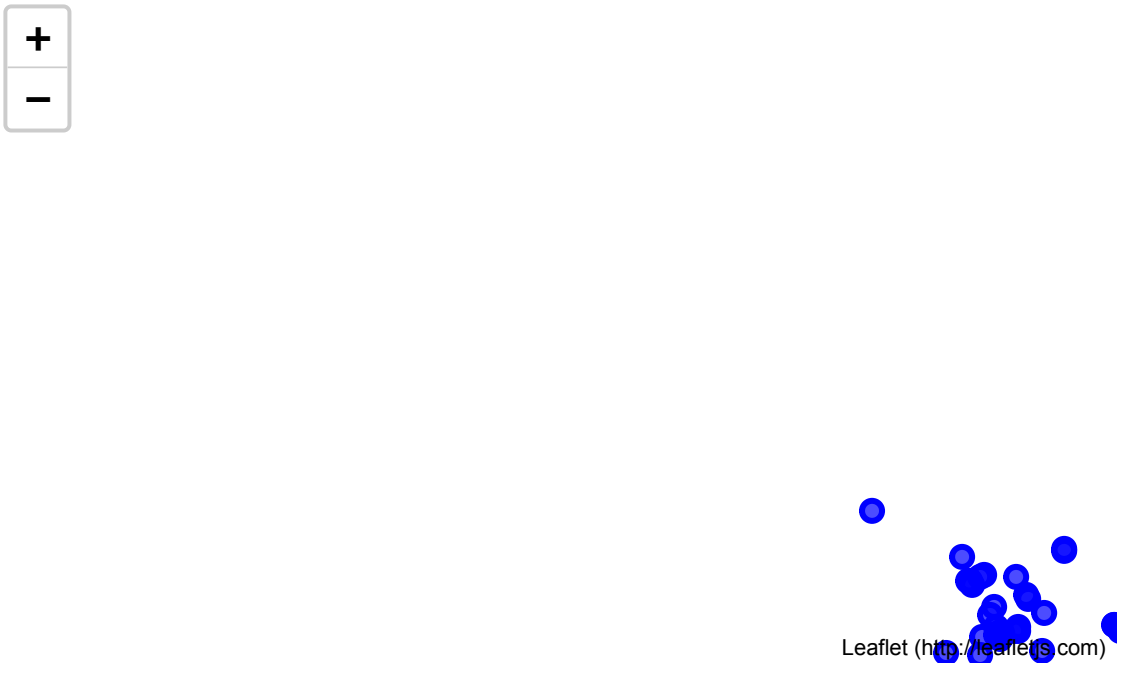
In [26]:

```
loc = [33.5186, -86.8104]
map_ = folium.Map(loc, zoom_start = 11)

for lat, lon, poi in zip(restaurants['restaurant.location.latitude'], restaurant
s['restaurant.location.longitude'], restaurants['restaurant.name']):
    label = folium.Popup(poi, parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color= 'blue',
        fill_opacity=0.7).add_to(map_)

map_
```

Out[26]:



3.2 Pre-processing Data

In [27]:

```
X = restaurants.drop(['restaurant.name', 'restaurant.cuisines', 'restaurant.timings', 'coordinate','restaurant.location.latitude','restaurant.location.longitude'],axis = 1)
X.head()
```

Out[27]:

	restaurant.average_cost_for_two	restaurant.user_rating.aggregate_rating	restaurant.all_revi
0	10	3.8	
1	25	3.6	
2	40	3.8	
3	40	3.5	
4	10	3.6	

In [28]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X)
X = scaler.transform(X)
X
```

Out[28]:

```
array([[-0.87596737, -0.23189218, -0.1432726 , ..., -0.54664725,
         0.29243116, -0.73186535],
       [ 0.13606168, -0.71500088, -0.16758356, ...,  0.6525789 ,
        -0.66635953, -0.01199779],
       [ 1.14809073, -0.23189218,  1.19383029, ...,  0.05296582,
         2.68940789, -0.01199779],
       ...,
       [-0.87596737,  0.00966217,  1.48556182, ..., -0.54664725,
        -0.66635953, -0.73186535],
       [ 0.81074772,  0.49277087,  1.29107413, ...,  0.35277236,
        -0.66635953, -0.73186535],
       [ 0.13606168,  0.00966217,  1.14520836, ..., -0.54664725,
        -0.66635953, -0.73186535]])
```

3.3 Clustering

I'm going to cluster the restaurants using K-Means Clustering

Find the optimum number of cluster

Write a loop that calculates and saves the WCSS for any number of clusters from 1 up to 10 (or more if you wish).

In [30]:

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans= KMeans(i)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [31]:

```
wcss
```

Out[31]:

```
[3900.0,
 2836.110773754155,
 2361.044930344215,
 2047.0038774927964,
 1774.2128498381935,
 1579.4911747315568,
 1391.3167756583084,
 1274.3282279880764,
 1175.8833981844757,
 1096.360248664638]
```

Using Kneed in selecting the Elbow Point

In [32]:

```
import sys
sys.path.append('..')

from kneed import KneeLocator

K = range(1,11)
kn = KneeLocator(list(K), wcss, S = 1.0, curve = 'convex', direction = 'decreasi
ng')
kn.knee
```

Out[32]:

```
4
```

From the result, the optimum number of clusters is 4, so i'm going to cluster the restaurants in to 4 cluster

In [33]:

```
clusterNumber = 4
k_means = KMeans(init = "k-means++", n_clusters = clusterNumber, n_init = 12, ra
ndom_state=3425)
k_means.fit(X)
class_label = k_means.labels_
restaurants['class'] = class_label
```

4. Results

4.1 Visualize Clusters

In [34]:

```
import matplotlib.cm as cm
import matplotlib.colors as colors

# create map
map_clusters = folium.Map(loc, zoom_start = 10)

# set color scheme for the clusters
x = np.arange(clusterNumber)
ys = [i + x + (i*x)**2 for i in range(clusterNumber)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]
labels = restaurants['restaurant.name']

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(restaurants['restaurant.location.latitude'], r
estaurants['restaurant.location.longitude'], labels, restaurants['class']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)


map_clusters
```

Out[34]:

4.2 Explore each Clusters

In [35]:

```
restaurants[restaurants['class'] == 0].describe()
```

Out[35]:

	restaurant.average_cost_for_two	restaurant.user_rating.aggregate_rating	restaurant.all_
count	30.000000	30.000000	
mean	17.166667	3.623333	
std	10.144077	0.355919	
min	10.000000	3.200000	
25%	10.000000	3.400000	
50%	10.000000	3.500000	
75%	25.000000	3.600000	
max	40.000000	4.500000	

In [36]:

```
restaurants[restaurants['class'] == 1].describe()
```

Out[36]:

	restaurant.average_cost_for_two	restaurant.user_rating.aggregate_rating	restaurant.all_
count	90.000000	90.000000	
mean	32.500000	4.264444	
std	16.663389	0.281773	
min	10.000000	3.500000	
25%	25.000000	4.100000	
50%	25.000000	4.300000	
75%	40.000000	4.400000	
max	70.000000	4.900000	

In [37]:

```
restaurants[restaurants['class'] == 2].describe()
```

Out[37]:

	restaurant.average_cost_for_two	restaurant.user_rating.aggregate_rating	restaurant.all_
count	144.000000	144.000000	
mean	17.361111	3.751389	
std	9.822631	0.332817	
min	0.000000	2.900000	
25%	10.000000	3.400000	
50%	10.000000	3.800000	
75%	25.000000	4.000000	
max	40.000000	4.600000	

In [38]:

```
restaurants[restaurants['class'] == 3].describe()
```

Out[38]:

	restaurant.average_cost_for_two	restaurant.user_rating.aggregate_rating	restaurant.all_
count	36.000000	36.000000	
mean	26.527778	3.780556	
std	17.105531	0.443892	
min	10.000000	3.300000	
25%	10.000000	3.400000	
50%	25.000000	3.600000	
75%	40.000000	4.200000	
max	70.000000	4.700000	

By looking at the stats descrption, in the cluster 1 the restaurants seems more expensive than the other clusters, and the number of rating votes is higher than the other clusters and also the rating is really high.

But if we looking for the good location strategy for restaurant to invest, we can consider cluster 0,just by looking at the stat desc, we can see that cluster 0 have more variety of venues around their restaurants.

In [41]:

```
cluster_0 = restaurants[restaurants['class'] == 0]
cluster_1 = restaurants[restaurants['class'] == 1]
cluster_2 = restaurants[restaurants['class'] == 2]
cluster_3 = restaurants[restaurants['class'] == 3]
clusters = [cluster_0,cluster_1,cluster_2,cluster_3]
```

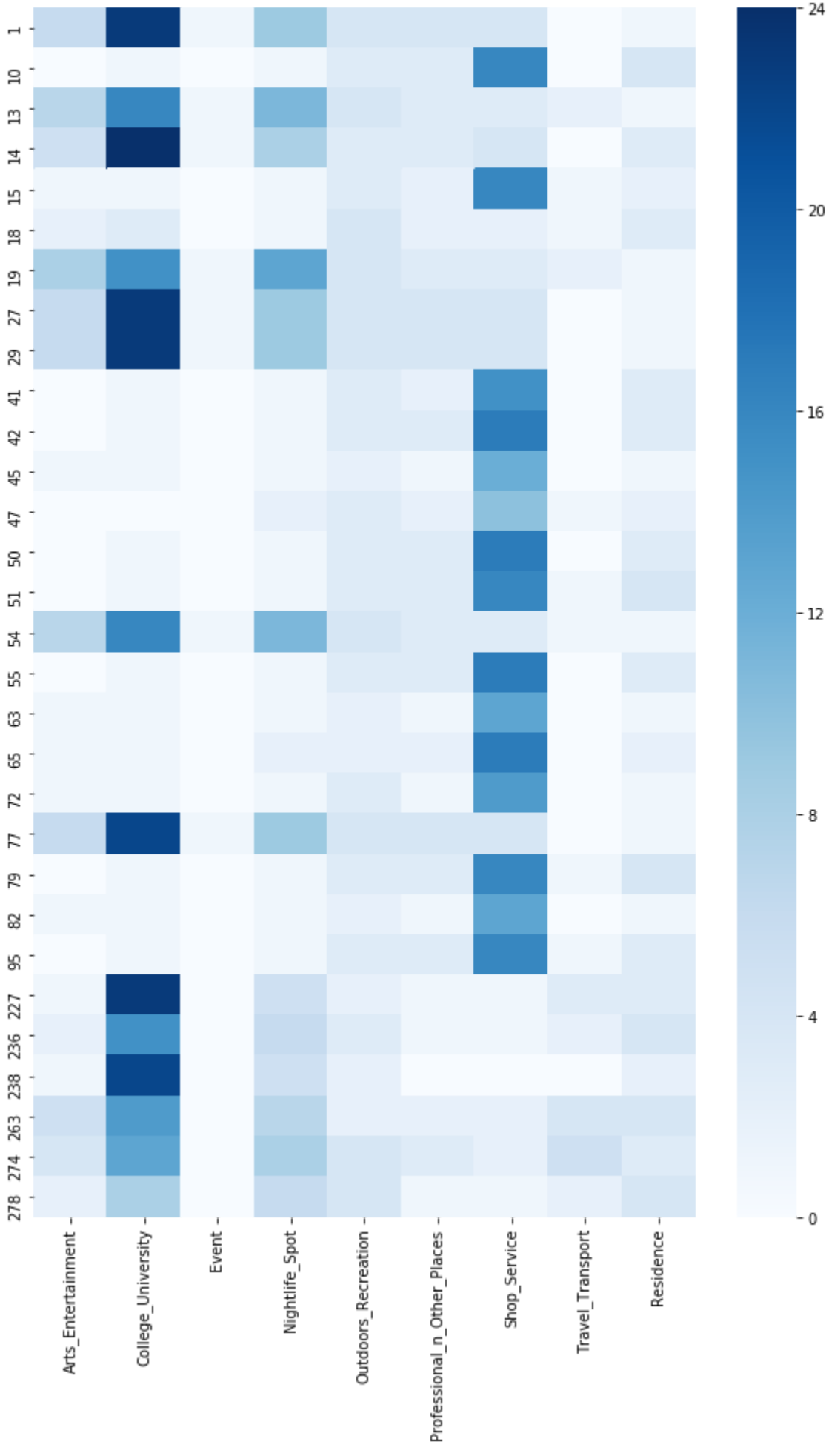
We can use Heatmap to visualize the number of venue for each category for each restuarants in every cluster

In [42]:

```
plt.figure(figsize= (10,15))
sns.heatmap(clusters[0].loc[:,'Arts_Entertainment':"Residence"],cmap = 'Blues')
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e949a2208>
```


1
10
13
14
15
18
19
27
29
41
42
45
47
50
51
54
55
63
65
72
77
79
82
95
227
236
238
263
274
278
Arts_Entertainment
College_University
Event
Nightlife_Spot
Outdoors_Recreation
Professional_n_Other_Places
Shop_Service
Travel_Transport
Residence
24
20
16
12
8
4
0

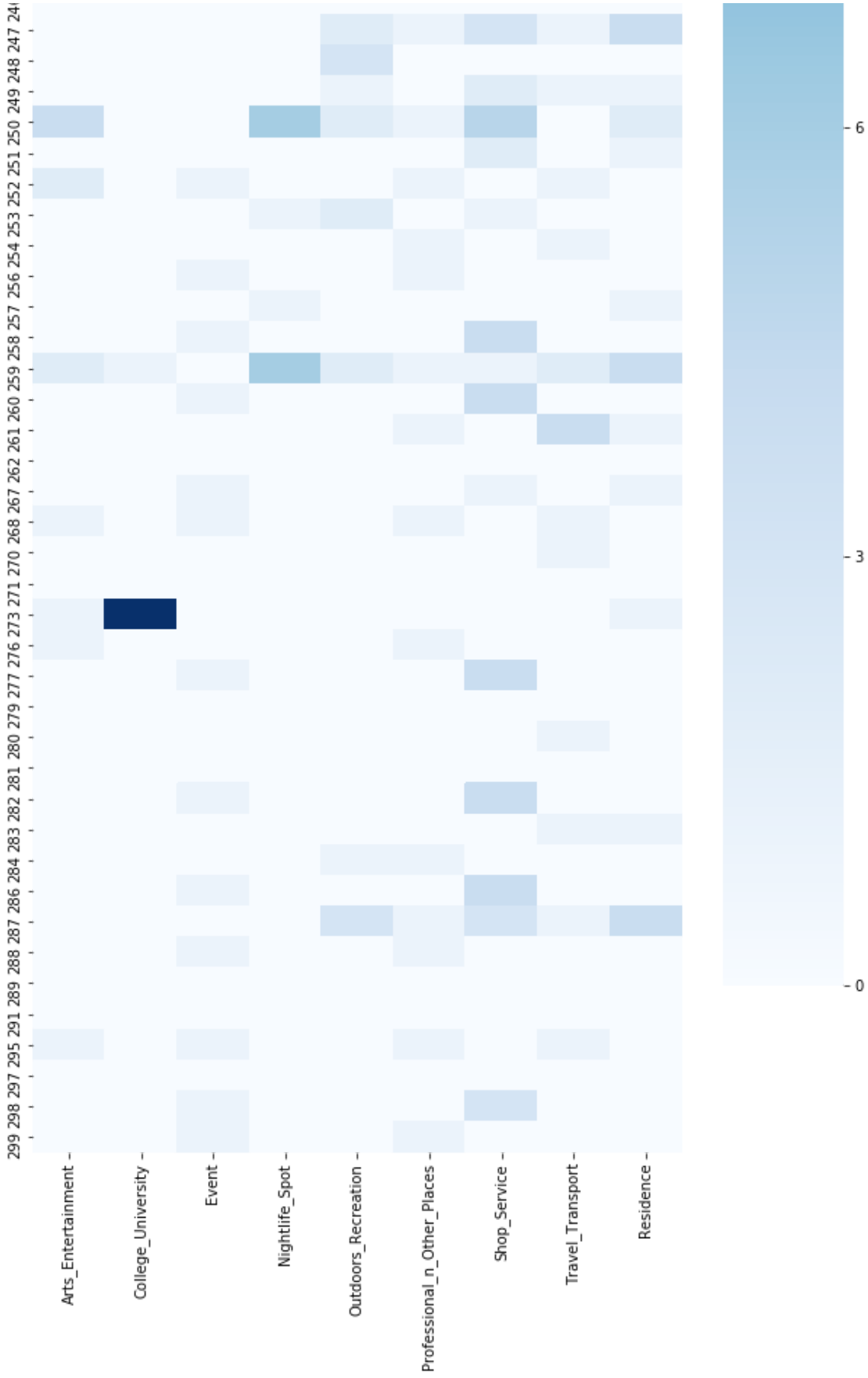
In [43]:

```
plt.figure(figsize= (10,35))
sns.heatmap(clusters[1].loc[:,'Arts_Entertainment':'Residence'],cmap = 'Blues')
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e8f0600b8>
```


3
11
101
103
105
106
107
109
111
117
118
129
138
200
201
202
203
204
205
206
207
208
209
210
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
228
229
230
232
233
234
237
239
240
242
243
244
245
6
15
12
9

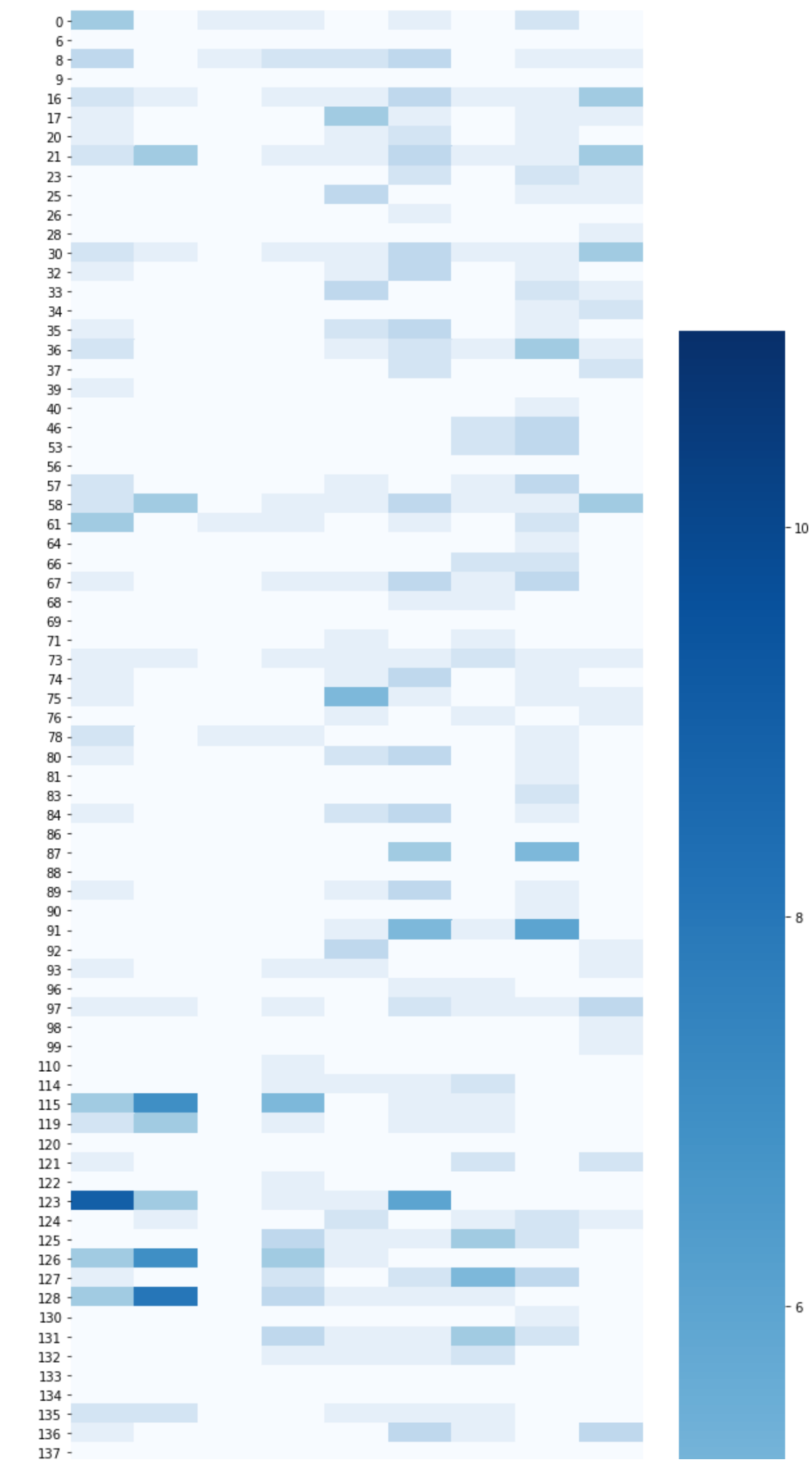

Arts_Entertainment
College_University
Event
Nightlife_Spot
Outdoors_Recreation
Professional_n_Other_Places
Shop_Service
Travel_Transport
Residence
6
3
0

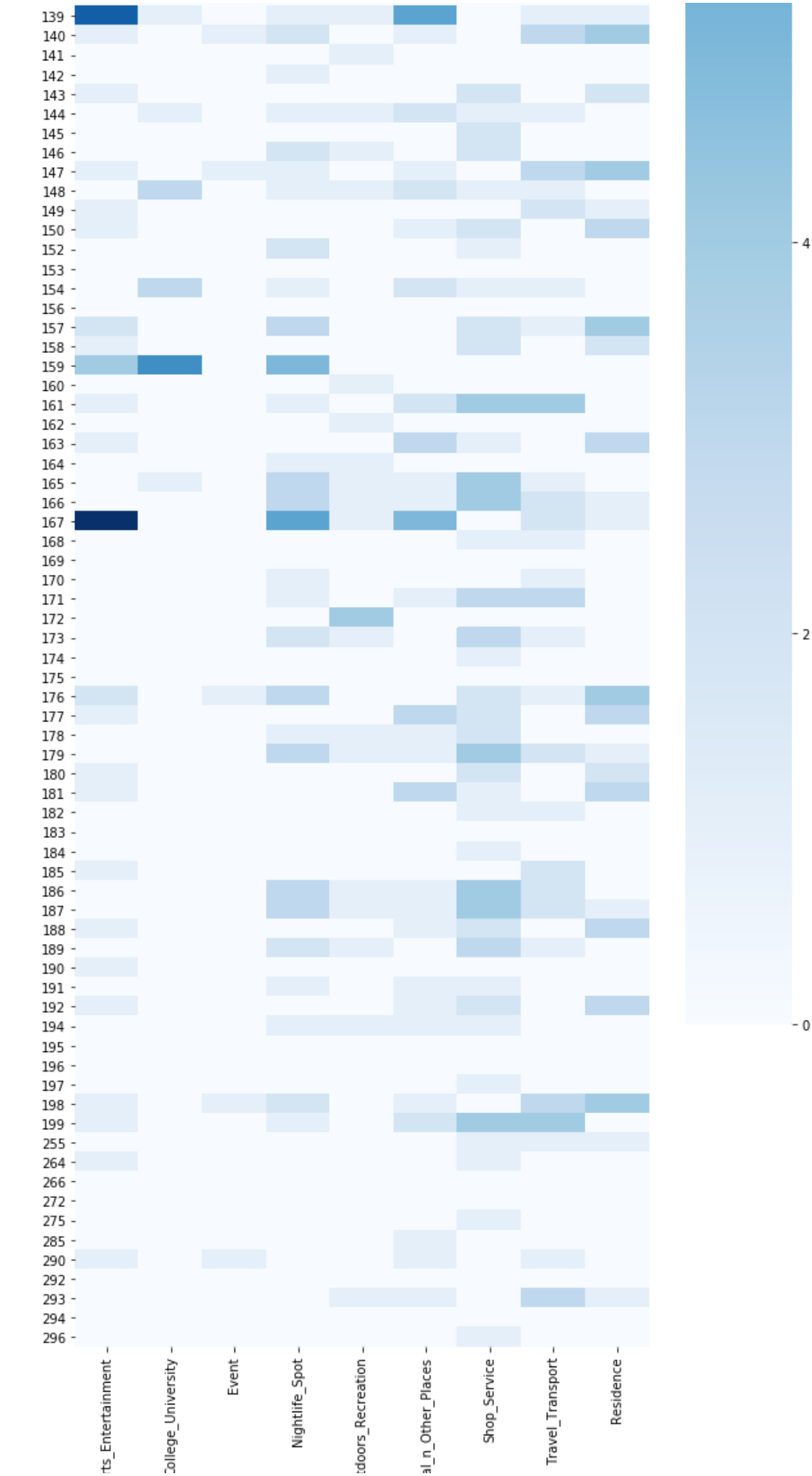
In [47]:

```
plt.figure(figsize= (10,40))
sns.heatmap(clusters[2].loc[:,'Arts_Entertainment':"Residence"],cmap = 'Blues')
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e8e8a4780>
```


0
6
8
9
16
17
20
21
23
25
26
28
30
32
33
34
35
36
37
39
40
46
53
56
57
58
61
64
66
67
68
69
71
73
74
75
76
78
80
81
83
84
86
87
88
89
90
91
92
93
96
97
98
99
110
114
115
119
120
121
122
123
124
125
126
127
128
130
131
132
133
134
135
136
137
10
8
6

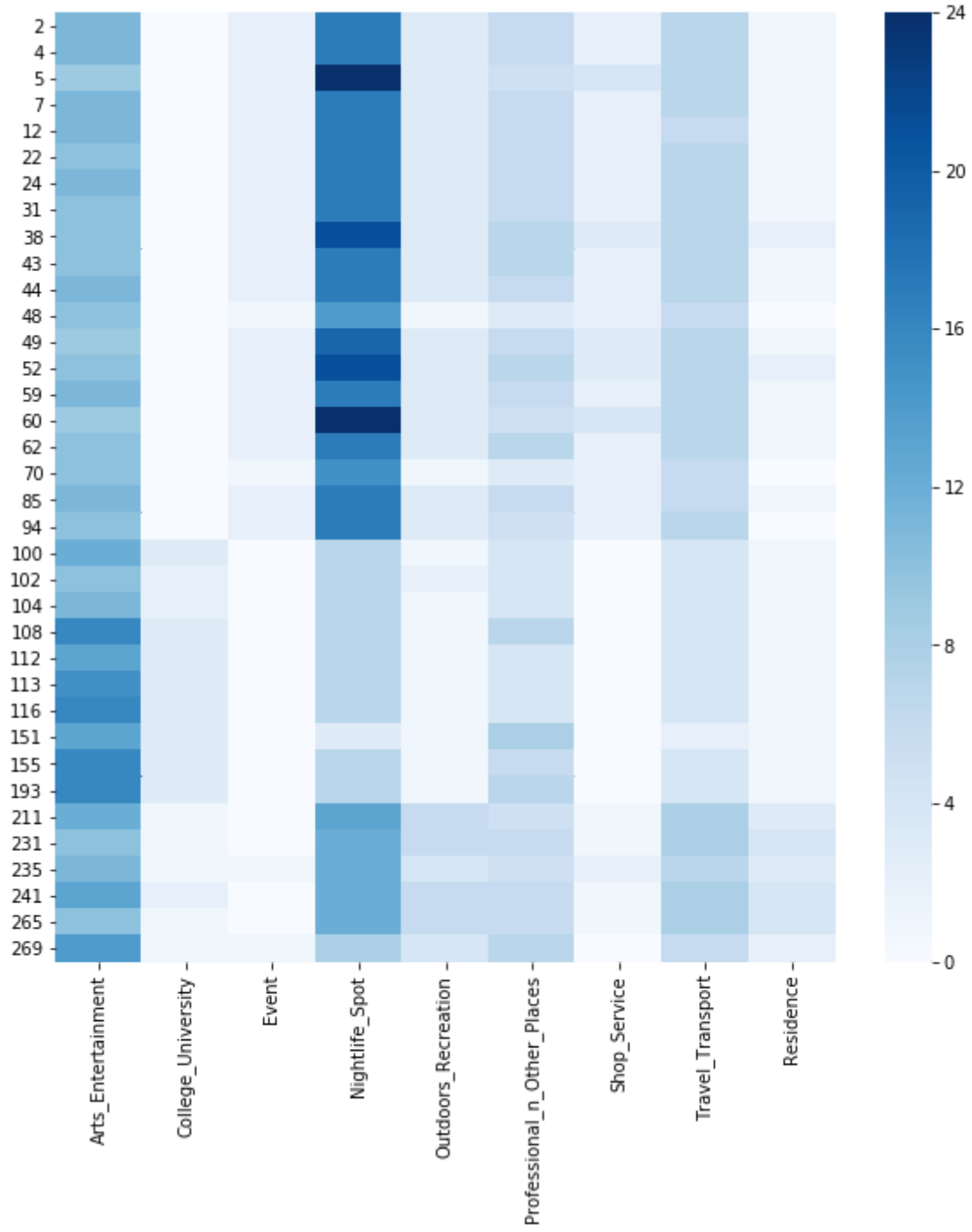

139
140
141
142
143
144
145
146
147
148
149
150
152
153
154
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
194
195
196
197
198
199
255
264
266
272
275
285
290
292
293
294
296
4
2
0
ts_Entertainment
ollege_University
Event
Nightlife_Spot
doors_Recreation
al n_Other_Places
Shop_Service
Travel_Transport
Residence

In [46]:

```
plt.figure(figsize= (10,10))
sns.heatmap(clusters[3].loc[:,'Arts_Entertainment':"Residence"],cmap = 'Blues')
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e8ee4e860>
```



4.3 Answer the question

From the Heatmaps we can see that, in cluster 0, cluster 3 their restaurants have more venues around them than the other clusters, and it seems the cluster 3 is a better choice if we want to invest our money in to one of the restaurants, based on the location strategy.

Visualize all the restaurants and the venues

First, i'm going to concat every city venues datas in two one dataframe, and then i'm going to plot every venue using their coordinate, and beside that i'm also going to plot all the restaurants with their clusters

In [48]:

```
def venues_dataframe(dataframes):
    venues = pd.DataFrame()
    for i in dataframes.keys():
        venues = pd.concat([dataframes[i], venues], axis = 0)
    venues.reset_index(inplace=True, drop=True)
    return venues
d1 = venues_dataframe(Tuscaloosa_venues)
d2 = venues_dataframe(Montgomery_venues)
d3 = venues_dataframe(Brimingham_venues)
venues = pd.concat([d1,d2,d3], axis = 0).reset_index(drop=True)
```

In [49]:

```
venues.head()
```

Out[49]:

	categories	id	location.formattedAddress	location.lat	location.lng
0	Residential Building (Apartment / Condo)	4c66de7919f3c9b6ff20a1ff	1100 Hargrove Rd E, Tuscaloosa, AL 35405, Unit...	33.189918	-87.521484
1	Residential Building (Apartment / Condo)	4c0c04e57e3fc9282731f682	grace street (University blvd.), Tuscaloosa, A...	33.212656	-87.554469
2	Residential Building (Apartment / Condo)	4c5b243904f9be9ab78df360	1900 Rice Mine Rd N, Tuscaloosa, AL 35406, Uni...	33.222982	-87.562209
3	Residential Building (Apartment / Condo)	56468afd498ea0ff87ff9e60	1418 10th Ave, Tuscaloosa, AL 35401, United St...	33.200005	-87.551736
4	Residential Building (Apartment / Condo)	4e0e3171d4c0f6d6b3f7b912	1100 17th St (17th St), Tuscaloosa, AL 35401, ...	33.197602	-87.552679

In [50]:

```
venues.to_csv("venues.csv")
```

In [51]:

```
venu_labels = venues['categories'] + ', ' + venues['name'].astype(str)
for lat, lon, poi in zip(venues['location.lat'], venues['location.lng'], venu_la
bels):
    label = folium.Popup(poi, parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color='grey',
        fill=True,
        fill_color= 'grey',
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[51]:

5. Discussion

For this project, k-means works properly. we use the most common venues in neighborhood, cluster the neighborhoods into 6 clusters. The result is good for our problem.

Although the result is accurate, we can also notice that the clusters is not precise enough. Adding more features like population and average income would be helpful.

6. Conclusion

1. Cluster 0 : just by looking at the stat desc and the heatmap, we can see that cluster 0 have more variety of venues around their restaurants and the number of the venues is high, their price is cheaper than in other clusters, and the number of user voting is high,. If we want to invest our money in restaurants that have their customer are mostly college students, then we can consider this cluster as a good place to invest.
2. Cluster 1 : in cluster 1, the restaurants have higher rating with average over 4 than other clusters, have more reviews and the user voting is also higher than any other clusters, and the price for two is more expencive than the other restaurants in other cluster, but unfortunately the location strategy is not good because the number of venue around their restaurant is smaller then the other clusters. If we want to consider to invest our money to restaurants that have many customer, high rating, more expensive cost, than this place is a good place to invest.
3. Cluster 2 : in cluster 2, the price for two cheaper then the other clusters with average rating 3. Based on the heatmap, cluster 2 have variety of venues around their restaurants. But for the loaction strategy, this cluster is better than cluster 1.
4. Cluster 3: From the Heatmaps, in cluster 3, their restaurants have more variety venues around them, and the number of each venue is higher than any other clusters, espcecially their Arts and Entertainment venue. By looking at the stat description, their price for two is consider expensive but their number of customer voting is higher than any other restaurants in other clusters and their rate is not bad, that means they have a lot of customer. so, the cluster 3 is a better choice if we want to invest our money in to one of their restaurants, based on the location strategy.

In [ ]: