Paris Tour Guide

Exploring the City of Light

---

1. Introduction

Paris is a popular tourist destination for all types of tourists and it is also called the City of Lights. The city attracts millions of tourists every year and it features grandiose monuments such as Arc de Triomphe, Eiffel Tower, and so on. The city has a romantic charm and it is filled with a plethora of activities that you can try out.

2. Business Problem

The intention of this project is to help tourists explore Paris depending on the experiences the neighborhoods in Paris has to offer. This project can later be extended in the future to generalize results, so that it is possible for tourists to get details about other interesting tourist destinations as well.

3. Data Description

The neighborhoods, boroughs, and venues are derived using the postal codes of Paris.

Data source: https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

The above source contains data related to all the neighborhoods in France. For the purpose of this project, only neighborhoods in Paris will be considered for now.

The source returns a JSON file that contains the following data,

- postal_code : Postal codes for France
- nom_comm : Neighbourhoods in France
- nom_dept : Boroughs(towns)
- geo_point_2d : latitude and longitude tuple of the Neighbourhoods

3.1 Foursquare API Usage

Foursquare API will be used to retrieve data related to venues in different neighborhoods. For each neighborhood, related venues and tourist attractions that are within the radius will be identified using the foursquare API.

The final dataframe created after processing the information obtained through the foursquare API are as follows,

- Neighbourhood

- Neighborhood latitude and Longitude
- Name of the venue
- Venue latitude and longitude
- Venue category

4. Implementation

Importing required Python Libraries

```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.colors as colors
import requests
import folium
from sklearn.cluster import KMeans
```

4.1 Data Collection and Preprocessing

Data is collected using a french government website. As the dataset contains details about France, it should be preprocessed and filtered to get the Paris data. Pandas is used to read data.

```
!wget -q -O 'france.json' https://www.data.gouv.fr/fr/datasets/r/e88c6fda-1d09-42a0-a069-606d3259114e

france_data = pd.read_json('france.json')
france_data.head()
```

	datasetid	recordid	fields	geometry	record_timestamp
0	correspondances-code-insee-code-postal	2bf36b38314b6c39dfbcd09225f97fa532b1fc45	{'code_comm': '645', 'nom_dept': 'ESSONNE', 's...	{'type': 'Point', 'coordinates': [2.2517129721...	2016-09-21T00:29:06.175+02:00
1	correspondances-code-insee-code-postal	7ee82e74e059b443df18bb79fc5a19b1f05e5a88	{'code_comm': '133', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [3.0529405055...	2016-09-21T00:29:06.175+02:00
2	correspondances-code-insee-code-postal	e2cd3186f07286705ed482a10b6aebd9de633c81	{'code_comm': '378', 'nom_dept': 'ESSONNE', 's...	{'type': 'Point', 'coordinates': [2.1971816504...	2016-09-21T00:29:06.175+02:00
3	correspondances-code-insee-code-postal	868bf03527a1d0a9defe5cf4e6fa0a730d725699	{'code_comm': '243', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [2.7097808131...	2016-09-21T00:29:06.175+02:00
4	correspondances-code-insee-code-postal	1bbcee92101fdb50f5f5fceb052681f2421ff961	{'code_comm': '414', 'nom_dept': 'SEINE-ET-MAR...	{'type': 'Point', 'coordinates': [3.2582355268...	2016-09-21T00:29:06.175+02:00

Initially, a dataframe is created using the data

```
france_dataframe = pd.DataFrame()
for field in france_data.fields:
    field_dict = field
    france_dataframe = france_dataframe.append(field_dict, ignore_index=True)
```

```
france_dataframe.head()
```

	code_arr	code_cant	code_comm	code_dept	code_reg	geo_point_2d	geo_shape	id_geofla	insee_com	nom_comm	nom_dept	nom_region	population	postal_code	statut	superficie	z_moyen
0	3	03	645	91	11	[48.750443119964764, 2.251712972144151]	{'type': 'Polygon', 'coordinates': [[[2.238024...	16275	91645	VERRIERES-LE-BUISSON	ESSONNE	ILE-DE-FRANCE	15.5	91370	Commune simple	999.0	121.0
1	3	20	133	77	11	[48.41256065214989, 3.052940505560729]	{'type': 'Polygon', 'coordinates': [[[3.076046...	31428	77133	COURCELLES-EN-BASSEE	SEINE-ET-MARNE	ILE-DE-FRANCE	0.2	77126	Commune simple	1082.0	88.0
2	1	09	378	91	11	[48.52726809075556, 2.19718165044305]	{'type': 'Polygon', 'coordinates': [[[2.203466...	30975	91378	MAUCHAMPS	ESSONNE	ILE-DE-FRANCE	0.3	91730	Commune simple	313.0	150.0
3	5	14	243	77	11	[48.87307018579678, 2.7097808131278462]	{'type': 'Polygon', 'coordinates': [[[2.727542...	17000	77243	LAGNY-SUR-MARNE	SEINE-ET-MARNE	ILE-DE-FRANCE	20.2	77400	Chef-lieu canton	579.0	71.0
4	3	25	414	77	11	[48.62891464105825, 3.258235526843923]	{'type': 'Polygon', 'coordinates': [[[3.294591...	34949	77414	SAINT-HILLIERS	SEINE-ET-MARNE	ILE-DE-FRANCE	0.4	77160	Commune simple	1907.0	158.0

Then the dataframe is filtered to get Paris data, and only the necessary columns for our analysis.

```
df = france_dataframe[['postal_code','nom_comm','nom_dept','geo_point_2d']]

df = df[df['nom_dept'].str.contains('PARIS')].reset_index(drop=True)
df.head()
```

	postal_code	nom_comm	nom_dept	geo_point_2d
0	75009	PARIS-9E-ARRONDISSEMENT	PARIS	[48.87689616237872, 2.337460241388529]
1	75002	PARIS-2E-ARRONDISSEMENT	PARIS	[48.86790337886785, 2.344107166658533]
2	75011	PARIS-11E-ARRONDISSEMENT	PARIS	[48.85941549762748, 2.378741060237548]
3	75008	PARIS-8E-ARRONDISSEMENT	PARIS	[48.87252726662346, 2.312582560420059]
4	75013	PARIS-13E-ARRONDISSEMENT	PARIS	[48.82871768452136, 2.362468228516128]

Final part of data preprocessing is to modify the Paris dataframe and separate Latitude and Longitudes into two columns

```
lat_lng = df['geo_point_2d'].astype('str')

# Process latitudes
lat = lat_lng.apply(lambda x: x.split(',')[0])
lat = lat.apply(lambda x: x.lstrip('['))

df_lat  = pd.DataFrame(lat.astype(float))
df_lat.columns=['Latitude']

# Process longitudes
lng = lat_lng.apply(lambda x: x.split(',')[1])
```

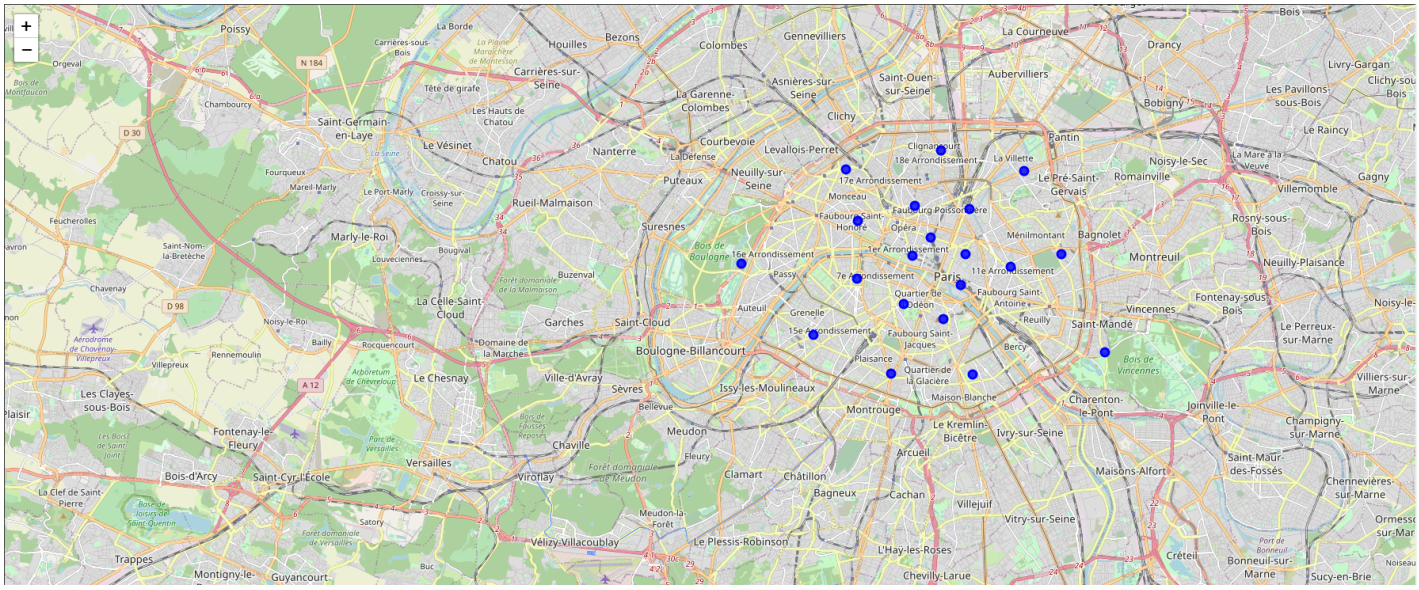
```
lng = lng.apply(lambda x: x.rstrip(']'))

df_lng = pd.DataFrame(lng.astype(float))
df_lng.columns=['Longitude']

# Combine columns
df = pd.concat([df.drop('geo_point_2d', axis=1), df_lat, df_lng], axis=1)
df.head()
```

	postal_code	nom_comm	nom_dept	Latitude	Longitude
0	75009	PARIS-9E-ARRONDISSEMENT	PARIS	48.876896	2.337460
1	75002	PARIS-2E-ARRONDISSEMENT	PARIS	48.867903	2.344107
2	75011	PARIS-11E-ARRONDISSEMENT	PARIS	48.859415	2.378741
3	75008	PARIS-8E-ARRONDISSEMENT	PARIS	48.872527	2.312583
4	75013	PARIS-13E-ARRONDISSEMENT	PARIS	48.828718	2.362468

We can visualize the map of Paris along with neighborhood data that we collected using the *Folium* package

```
import geocoder

# paris = geocode(address='Paris, France, FR')[0]
paris = geocoder.arcgis('Paris, France, FR')
paris_lat = paris.json['lat']
paris_lng = paris.json['lng']

map = folium.Map(location=[paris_lat, paris_lng], zoom_start=12)
map

# adding markers to map
for latitude, longitude, borough, town in zip(df['Latitude'], df['Longitude'], df['nom_comm'], df['no
    label = '{}, {}'.format(town, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [latitude, longitude],
        radius=5,
        popup=label,
        color='Blue',
        fill=True,
```

```
        fill_opacity=0.8
        ).add_to(map)

map
```



4.2 Configure the Foursquare API

The Foursquare API should be first configured to fetch the necessary data from the API

```
CLIENT_ID = 'NKH5LOIG3E1FIGHVASEZIY42KD5O0YHFKSIBXFWYP5BIYSTF'
CLIENT_SECRET = 'EJVUHICOHKIE0QQQTJ1JMLIS1HKWOPSXIOLXO0FO5LCWCUSZ'
VERSION = '20190101'
```

Next we will implement a function to fetch nearby venues from the Foursquare API by providing the neighborhoods, latitutes, longitudes, and radisu as parameters.

```
LIMIT=100

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

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

        # Define URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={}

        # GET request
```

```
            results = requests.get(url).json()["response"]['groups'][0]['items']

            # Append results to vunue list
            venues.append([(
                name,
                lat,
                lng,
                v['venue']['name'],
                v['venue']['categories'][0]['name']) for v in results])

    # Create Dataframe
    nearby_venues = pd.DataFrame([item for venue in venues for item in venue])
    nearby_venues.columns = ['Neighbourhood',
                  'Latitude',
                  'Longitude',
                  'Venue',
                  'Venue Category']

    return(nearby_venues)
```

We can use the above function to fetch nearby venues for each neighborhood in Paris

```
nearby_venues = fetchNearbyVenues(df['nom_comm'], df['Latitude'], df['Longitude'])
nearby_venues.head()
```

	Neighbourhood	Latitude	Longitude	Venue	Venue Category
0	PARIS-9E-ARRONDISSEMENT	48.876896	2.33746	Farine & O	Bakery
1	PARIS-9E-ARRONDISSEMENT	48.876896	2.33746	RAP	Gourmet Shop
2	PARIS-9E-ARRONDISSEMENT	48.876896	2.33746	Place Saint-Georges	Plaza
3	PARIS-9E-ARRONDISSEMENT	48.876896	2.33746	Le Bouclier de Bacchus	Wine Bar
4	PARIS-9E-ARRONDISSEMENT	48.876896	2.33746	La Compagnie du Café	Café

4.3 Exploring nearby venues in Paris

```
nearby_venues.groupby('Venue Category').max()
```

Venue Category	Neighbourhood	Latitude	Longitude	Venue
Afghan Restaurant	PARIS-11E-ARRONDISSEMENT	48.859415	2.378741	Afghanistan
African Restaurant	PARIS-9E-ARRONDISSEMENT	48.876896	2.361113	Wally Le Saharien
American Restaurant	PARIS-19E-ARRONDISSEMENT	48.892735	2.384694	Harper's
Antique Shop	PARIS-9E-ARRONDISSEMENT	48.876896	2.337460	Hôtel des Ventes Drouot
Argentinian Restaurant	PARIS-3E-ARRONDISSEMENT	48.863054	2.359361	Anahi
...	...	...	...	...
Wine Bar	PARIS-9E-ARRONDISSEMENT	48.892735	2.400820	Vingt Vins d'Art
Wine Shop	PARIS-3E-ARRONDISSEMENT	48.886869	2.400820	Trois Fois Vin
Women's Store	PARIS-2E-ARRONDISSEMENT	48.867903	2.344107	L'Appartement Sézane
Zoo	PARIS-12E-ARRONDISSEMENT	48.835156	2.419807	Parc zoologique de Paris
Zoo Exhibit	PARIS-12E-ARRONDISSEMENT	48.835156	2.419807	Grande Serre du Parc Zoologique de Paris

206 rows × 4 columns

In order to organize venues, we first perform One Hot Encoding. The encoded dataframe will later be used to identify different venue categories and to calculate the top 10 venues in each neighborhood

```
encoded = pd.get_dummies(nearby_venues[['Venue Category']], prefix="", prefix_sep="")
encoded.head()
```

	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	Argentinian Restaurant	Art Gallery	Art Museum	Arts & Crafts Store	A Restau
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	

5 rows × 206 columns

4.4 Analyzing the top venues in each Neighborhood

Initially the neighborhoods are added to the encoded dataframe

```
encoded['Neighbourhood'] = nearby_venues['Neighbourhood']

encoded_updates = [encoded.columns[-1]] + list(encoded.columns[:-1])
encoded = encoded[encoded_updates]
```

Next, the mean of venue categories are calculated for each neighborhood

```
df_paris = encoded.groupby('Neighbourhood').mean().reset_index()
```

Finally, the venues are sorted from the highers mean to the lowest. We consider the top ten venues for each neighborhood.

```
venue_count = 10

# Dataframe columns (neighborhood and the respective top 10 venues)
columns = ['Neighbourhood']
for index in np.arange(venue_count):
    try:
        columns.append('Venue {}'.format(index+1))
    except:
        columns.append('Venue {}'.format(index+1))

# Function to get the top 10 venues of a neighborhood
def getTopVenues(row, venue_count):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:venue_count]

# create a new dataframe for Paris
top_venues_in_neighborhood = pd.DataFrame(columns=columns)
top_venues_in_neighborhood['Neighbourhood'] = df_paris['Neighbourhood']

for index in np.arange(df_paris.shape[0]):
    top_venues_in_neighborhood.iloc[index, 1:] = getTopVenues(df_paris.iloc[index, :], venue_count)

top_venues_in_neighborhood.head()
```

	Neighbourhood	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
0	PARIS-10E-ARRONDISSEMENT	French Restaurant	Bistro	Café	Hotel	Coffee Shop	Japanese Restaurant	Indian Restaurant	Pizza Place	Asian Restaurant	Bar
1	PARIS-11E-ARRONDISSEMENT	Café	Restaurant	Asian Restaurant	Pastry Shop	Italian Restaurant	Wine Bar	Vietnamese Restaurant	French Restaurant	Bakery	Sandwich Place
2	PARIS-12E-ARRONDISSEMENT	Zoo Exhibit	Bistro	Monument / Landmark	Supermarket	Zoo	Argentinian Restaurant	Frozen Yogurt Shop	Fountain	Food Court	Food & Drink Shop
3	PARIS-13E-ARRONDISSEMENT	Vietnamese Restaurant	Asian Restaurant	Thai Restaurant	Chinese Restaurant	French Restaurant	Juice Bar	Park	Coffee Shop	Creperie	Plaza
4	PARIS-14E-ARRONDISSEMENT	French Restaurant	Hotel	Bakery	Plaza	Supermarket	Tea Room	Pizza Place	Italian Restaurant	Japanese Restaurant	Brasserie

5. Cluster Analysis using K means

In this final step, we will use the results of our implementation phase to cluster the neighborhood in to 5 clusters and analyze each cluster separately.

Paris dataframe will be first clustered into 5 clusters using K means clustering

```
k = 5

clusters = df_paris.drop('Neighbourhood', 1)

paris_k = KMeans(n_clusters=k, random_state=0).fit(clusters)
print(paris_k)
print("Cluster labels: ", paris_k.labels_)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
Cluster labels:  [0 0 4 2 1 0 3 1 0 0 0 0 0 0 0 0 1 1 0]
```

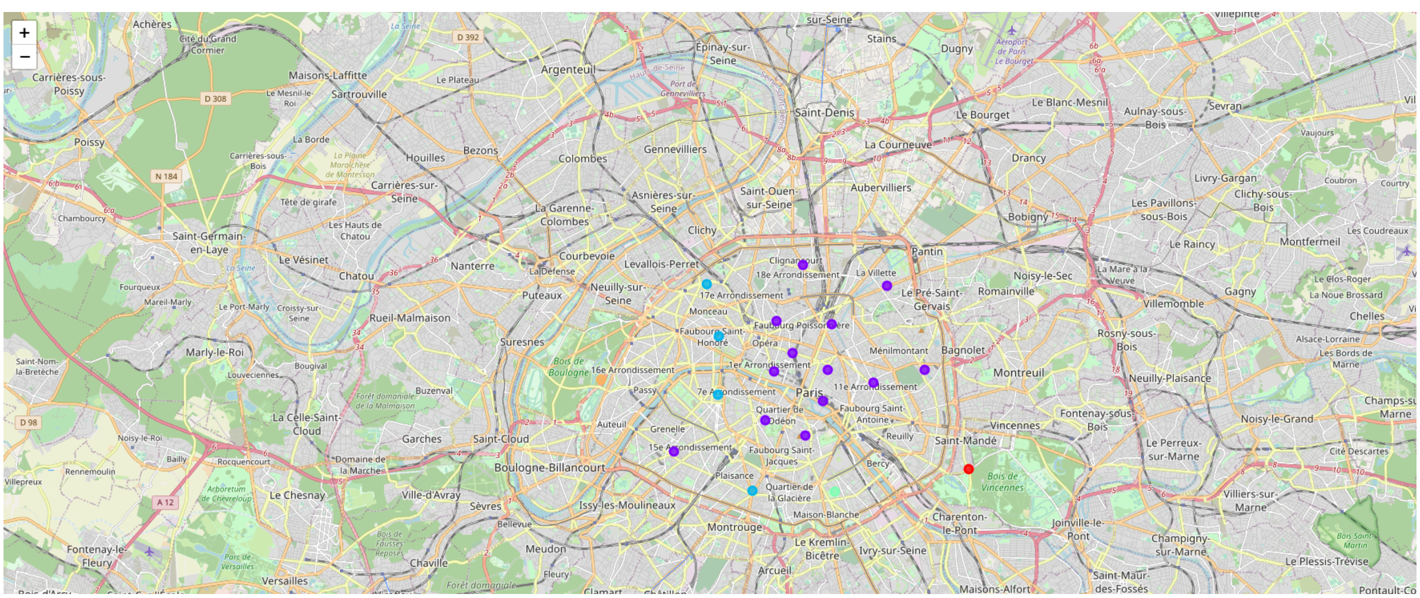
The cluster labels are then inserted into the Paris dataframe to construct the final complete dataframe

```
top_venues_in_neighborhood.insert(0, 'Cluster Labels', paris_k.labels_ +1)

paris_data = df
paris_data = paris_data.join(top_venues_in_neighborhood.set_index('Neighbourhood'), on='nom_comm')

paris_data.head()
```

	postal_code	nom_comm	nom_dept	Latitude	Longitude	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
0	75009	PARIS-9E-ARRONDISSEMENT	PARIS	48.876896	2.337460	1	French Restaurant	Hotel	Bistro	Japanese Restaurant	Restaurant	Wine Bar	Cocktail Bar	Lounge	Bakery	Gym / Fitness Center
1	75002	PARIS-2E-ARRONDISSEMENT	PARIS	48.867903	2.344107	1	French Restaurant	Cocktail Bar	Wine Bar	Bakery	Salad Place	Plaza	Coffee Shop	Hotel	Creperie	Italian Restaurant
2	75011	PARIS-11E-ARRONDISSEMENT	PARIS	48.859415	2.378741	1	Café	Restaurant	Asian Restaurant	Pastry Shop	Italian Restaurant	Wine Bar	Vietnamese Restaurant	French Restaurant	Bakery	Sandwich Place
3	75008	PARIS-8E-ARRONDISSEMENT	PARIS	48.872527	2.312583	2	French Restaurant	Hotel	Spa	Corsican Restaurant	Art Gallery	Cocktail Bar	Theater	Plaza	Resort	Park
4	75013	PARIS-13E-ARRONDISSEMENT	PARIS	48.828718	2.362468	3	Vietnamese Restaurant	Asian Restaurant	Thai Restaurant	Chinese Restaurant	French Restaurant	Juice Bar	Park	Coffee Shop	Creperie	Plaza

5.1 Visualizing the clustered Neighborhoods

```
# Get rid of NaN values
paris_data = paris_data.dropna(subset=['Cluster Labels'])

paris_cluster_map = folium.Map(location=[paris_lat, paris_lng], zoom_start=12)

x = np.arange(k)
ys = [i + x + (i*x)**2 for i in range(k)]

# Set colors
k_colors = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in k_colors]

# Add markers
for lat, lon, poi, cluster in zip(paris_data['Latitude'], paris_data['Longitude'], paris_data['nom_cc
    label = folium.Popup('Cluster ' + str(int(cluster) +1) + ' ' + str(poi) , parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)],
        fill_opacity=0.8
        ).add_to(paris_cluster_map)

paris_cluster_map
```



5.2 Exploring the clusters created

Cluster 1

```
paris_data.loc[paris_data['Cluster Labels'] == 1, paris_data.columns[[1] + list(range(5, paris_data.shape[1]))]]
```

	nom_comm	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
0	PARIS-9E-ARRONDISSEMENT	1	French Restaurant	Hotel	Bistro	Japanese Restaurant	Restaurant	Wine Bar	Cocktail Bar	Lounge	Bakery	Gym / Fitness Center
1	PARIS-2E-ARRONDISSEMENT	1	French Restaurant	Cocktail Bar	Wine Bar	Bakery	Salad Place	Plaza	Coffee Shop	Hotel	Creperie	Italian Restaurant
2	PARIS-11E-ARRONDISSEMENT	1	Café	Restaurant	Asian Restaurant	Pastry Shop	Italian Restaurant	Wine Bar	Vietnamese Restaurant	French Restaurant	Bakery	Sandwich Place
6	PARIS-3E-ARRONDISSEMENT	1	French Restaurant	Japanese Restaurant	Coffee Shop	Art Gallery	Gourmet Shop	Cocktail Bar	Bakery	Wine Bar	Italian Restaurant	Sandwich Place
7	PARIS-6E-ARRONDISSEMENT	1	French Restaurant	Chocolate Shop	Bakery	Plaza	Pastry Shop	Restaurant	Fountain	Theater	Italian Restaurant	Garden
8	PARIS-4E-ARRONDISSEMENT	1	French Restaurant	Clothing Store	Ice Cream Shop	Pastry Shop	Hotel	Park	Wine Bar	Gay Bar	Italian Restaurant	Pedestrian Plaza
9	PARIS-10E-ARRONDISSEMENT	1	French Restaurant	Bistro	Café	Hotel	Coffee Shop	Japanese Restaurant	Indian Restaurant	Pizza Place	Asian Restaurant	Bar
11	PARIS-5E-ARRONDISSEMENT	1	French Restaurant	Hotel	Italian Restaurant	Plaza	Bakery	Pub	Coffee Shop	Café	Bar	Historic Site
12	PARIS-19E-ARRONDISSEMENT	1	French Restaurant	Bar	Supermarket	Hotel	Sushi Restaurant	Beer Bar	Brewery	Seafood Restaurant	Bakery	Bistro
13	PARIS-20E-ARRONDISSEMENT	1	Plaza	Japanese Restaurant	Bakery	Bistro	French Restaurant	Café	Bar	Hotel	Italian Restaurant	Laundromat
15	PARIS-18E-ARRONDISSEMENT	1	French Restaurant	Bar	Italian Restaurant	Plaza	Café	Restaurant	Pizza Place	Bistro	Bakery	Convenience Store
17	PARIS-15E-ARRONDISSEMENT	1	Italian Restaurant	Hotel	French Restaurant	Park	Lebanese Restaurant	Brasserie	Coffee Shop	Restaurant	Indian Restaurant	Thai Restaurant
18	PARIS-1ER-ARRONDISSEMENT	1	French Restaurant	Japanese Restaurant	Plaza	Hotel	Restaurant	Italian Restaurant	Art Museum	Garden	Dessert Shop	Tea Room

Clsuter 2

```
paris_data.loc[paris_data['Cluster Labels'] == 2, paris_data.columns[[1] + list(range(5, paris_data.shape[1]))]]
```

	nom_comm	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
3	PARIS-8E-ARRONDISSEMENT	2	French Restaurant	Hotel	Spa	Corsican Restaurant	Art Gallery	Cocktail Bar	Theater	Plaza	Resort	Park
14	PARIS-7E-ARRONDISSEMENT	2	Hotel	French Restaurant	Italian Restaurant	Café	History Museum	Bistro	Cocktail Bar	Art Museum	Plaza	Coffee Shop
16	PARIS-17E-ARRONDISSEMENT	2	French Restaurant	Hotel	Italian Restaurant	Japanese Restaurant	Bakery	Café	Plaza	Bistro	Restaurant	Breakfast Spot
19	PARIS-14E-ARRONDISSEMENT	2	French Restaurant	Hotel	Bakery	Plaza	Supermarket	Tea Room	Pizza Place	Italian Restaurant	Japanese Restaurant	Brasserie

Cluster 3

```
paris_data.loc[paris_data['Cluster Labels'] == 3, paris_data.columns[[1] + list(range(5, paris_data.shape[1]))]]
```

	nom_comm	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
4	PARIS-13E-ARRONDISSEMENT	3	Vietnamese Restaurant	Asian Restaurant	Thai Restaurant	Chinese Restaurant	French Restaurant	Juice Bar	Park	Coffee Shop	Creperie	Plaza

Cluster 4

```
paris_data.loc[paris_data['Cluster Labels'] == 4, paris_data.columns[[1] + list(range(5, paris_data.shape[1]))]]
```

	nom_comm	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
10	PARIS-16E-ARRONDISSEMENT	4	Plaza	Lake	French Restaurant	Park	Pool	Boat or Ferry	Art Museum	Bus Station	Bus Stop	Food

Cluster 5

```
paris_data.loc[paris_data['Cluster Labels'] == 5, paris_data.columns[[1] + list(range(5, paris_data.shape[1]))]]
```

	nom_comm	Cluster Labels	Venue 1	Venue 2	Venue 3	Venue 4	Venue 5	Venue 6	Venue 7	Venue 8	Venue 9	Venue 10
5	PARIS-12E-ARRONDISSEMENT	5	Zoo Exhibit	Bistro	Monument / Landmark	Supermarket	Zoo	Argentinian Restaurant	Frozen Yogurt Shop	Fountain	Food Court	Food & Drink Shop

The above clusters describe the Neighborhood that belong to the cluster and venues in the descending order of their popularity.

6. Conclusion

The intention of this study is to explore the neighborhoods in the city of Paris to guide its tourists with what the city has to offer. Based on the neighborhood, this provides the top 10 places that vistors of Paris can explore from popular landmarks to cafes.

Based on the ranking of the venue the tourist get the chance to experience the best the city of lights has to offer.