Cities of Egypt.

IBM Data Science Professional Certificate -- Capstone Project

By: Abdullah M. Mustafa

1. Introduction:

Egypt is a big country with a population over 100 million with a total of 27 governorates. These governorates differ both culturally and economically. For Egypt, tourism is considered a main source of national income; however, not all governorates are considered attractive destinations for tourists. In this project, we aim to better understand the most popular venues across Egypt using the Foursquare API. The popular venues for the capital cities of each of the governorates are analyzed, and these cities are clustered to better understand the touristic attractions. We expect cities like Cairo, Luxor, and Hurgada to be popular destinations; on the other hand, poor governorates would be less popular. Our objective to enrich these poor cities to be more attractive.

Load neccessary libraries

In [1]:
```
import requests #request the data of some url
import json # manipulate jason files into python data strcutures
import pandas as pd #Tabular data manipulation in python
import numpy as np #numerical data manipulation in python
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
from sklearn.cluster import KMeans # A non-parametric clustering algorithm
import folium # Map visualisation package
import matplotlib.pyplot as plt # python plotting library
from matplotlib import cm, colors
from IPython.display import HTML, display
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

2. Getting Data:

To proceed with our problem, we first need the location data to feed to the Foursquare API. The data could be retrieved in JSON format from this simplemaps.com url. Out of multiple data columns, we are mainly interested in the capital of each governorates with its latitude and longitude.

In [2]:
```
r = requests.get('https://simplemaps.com/static/data/country-cities/eg/eg.json', allow_redirects=True)

with open('eg.json') as json_file:
    data = json.load(json_file)
```

In [3]:
```
df = pd.DataFrame(data)
df.head()
```

Out[3]:

	city	admin	country	population_proper	iso2	capital	lat	lng	population
0	Cairo	Al Qāhirah	Egypt	7734614	EG	primary	30.07708	31.285909	11893000
1	Alexandria	Al Iskandarīyah	Egypt	3811516	EG	admin	31.215645	29.955266	4165000
2	Al Jīzah	Al Jīzah	Egypt	2681863	EG	admin	30.008079	31.210931	2681863
3	Ismailia	Al Ismā'īlīyah	Egypt	284813	EG	admin	30.604272	32.272252	656135
4	Port Said	Būr Sa'īd	Egypt	500000	EG	admin	31.256541	32.284115	623864

We filter out the dataframe to extract only neccessary columns. We also convert datatypes of latitude and longitude to floats.

In [4]:
```
City = df['admin']
Latitude = df.lat.astype('float')
Longitude = df.lng.astype('float')
df_egypt = pd.DataFrame({'City':City,'Latitude': Latitude, 'Longitude': Longitude})
df_egypt = df_egypt.groupby('City').mean().reset_index()
df_egypt
```

Out[4]:

	City	Latitude	Longitude
0	Ad Daqahlīyah	31.036373	31.380691
1	Al Baḩr al Aḩmar	26.991034	33.877310
2	Al Buḩayrah	31.033452	30.446752
3	Al Fayyūm	29.309949	30.841804
4	Al Gharbīyah	30.788471	31.001921
5	Al Iskandarīyah	31.215645	29.955266
6	Al Ismā'īlīyah	30.604272	32.272252
7	Al Jīzah	30.008079	31.210931
8	Al Minyā	28.165388	30.777255
9	Al Minūfīyah	30.552581	31.009035
10	Al Qalyūbīyah	30.459065	31.178577
11	Al Qāhirah	30.077080	31.285909
12	Al Uqşur	25.695858	32.643592
13	Al Wādī al Jadīd	26.068280	29.133540
14	As Suways	29.973714	32.526267
15	Ash Sharqīyah	30.587676	31.501997
16	Aswān	24.093433	32.907038
17	Asyūţ	27.180956	31.183683
18	Banī Suwayf	29.074409	31.097848
19	Būr Sa'īd	31.256541	32.284115
20	Dumyāţ	31.416477	31.813316
21	Janūb Sīnā'	28.236381	33.625404
22	Kafr ash Shaykh	31.114304	30.940116
23	Maţrūḩ	30.793167	27.064948
24	Qinā	25.728768	32.640364
25	Shamāl Sīnā'	31.162909	33.788933
26	Sūhāj	26.447603	31.793197

Using Nominatim library, we extract the latitude and longitude of Egypt. This is neccessary for constructing a map around the country location.

In [5]:
```
address = 'Egypt'
geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Egypt are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Egypt are 26.2540493, 29.2675469.
```

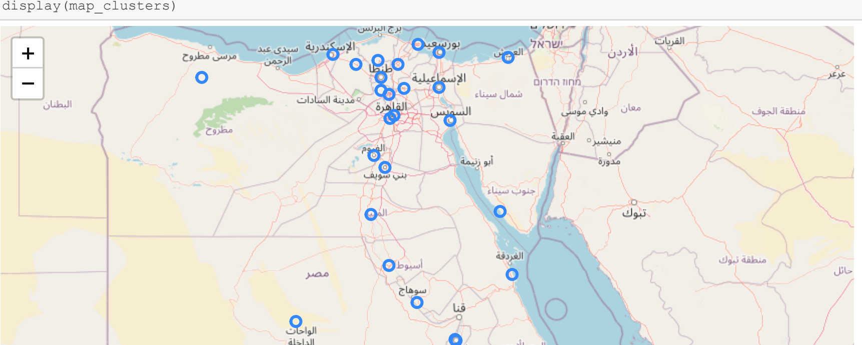
We can now use folium package to visulaize the country with different cities around it.

In [6]:
```
# create map
map_clusters = folium.Map(location=[latitude+1, longitude+2],zoom_start=6)

# set color scheme for the clusters


# add markers to the map
markers_colors = []
for lat, lon, city in zip(df_egypt['Latitude'], df_egypt['Longitude'], df_egypt['City']):
    label = folium.Popup(city, parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label).add_to(map_clusters)

display(map_clusters)
```



We need to provide some user data to use the Foursquare API.

In [23]:
```
CLIENT_ID = '' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
VERSION = '' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID:
CLIENT_SECRET:
```

We need to estimate the radius of the area to explore. To do so, we measure the distances between cities across Egypt using the latitude and longitude data.

In [24]:
```
#!pip install pyproj
import pyproj

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

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

#X,Y = lonlat_to_xy(df_egypt['Longitude'],df_egypt['Latitude'])
result = map(lambda lng, lat: lonlat_to_xy(lng,lat), df_egypt['Longitude'], df_egypt['Latitude'])
XY = list(result)

from sklearn.metrics.pairwise import euclidean_distances
dist = euclidean_distances(XY)
np.fill_diagonal(dist, np.inf)

sort_dist = np.sort(np.matrix.flatten(dist))
sort_dist[:20]
```

Out[24]:
```
array([ 3804.07602273,  3804.07602273, 10846.40666268, 10846.40666268,
        19866.46150831, 19866.46150831, 26921.86416682, 26921.86416682,
        35199.32797382, 35199.32797382, 37157.89897978, 37157.89897978,
        37658.14113314, 37658.14113314, 41441.06230978, 41441.06230978,
        44185.13674601, 44185.13674601, 44904.38291043, 44904.38291043])
```

The shortest distace was about 4 km, followed by 10 km, and then 20 km. Ignoring these first 3 entries, we chose to set the exploring area radius to 25 km. This would be an acceptable value for the size of the city.

We can now use the Foursquare API to extract the most popular venues at each location. To better explore the city, we set the search diameter to 25 Km, and limit the number of top venues to 100. For each venue, we extract its name, location, and category. Different cities can be compared based on the popularity of each category.

In [9]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=25000):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},
{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            100)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['City',
                  'Country Latitude',
                  'Country Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [26]:
```
# type your answer here
egypt_venues = getNearbyVenues(names=df_egypt['City'],
                                   latitudes=df_egypt['Latitude'],
                                   longitudes=df_egypt['Longitude']
                                  )
egypt_venues.sample(10)
```

Out[26]:

	City	Country Latitude	Country Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
659	Al Uqşur	25.695858	32.643592	Old Souq (السوق القديم)	25.701367	32.642098	Flea Market
154	Al Buḩayrah	31.033452	30.446752	4 Season Cafe	31.034454	30.455369	Café
348	Al Ismā'īlīyah	30.604272	32.272252	Suez Canal	30.455172	32.354836	Canal
763	Banī Suwayf	29.074409	31.097848	Lamera Cafe	29.066536	31.108039	Café
264	Al Iskandarīyah	31.215645	29.955266	El Sheikh Wafiq (الشيخ وفيق)	31.203909	29.875343	Dessert Shop
357	Al Jīzah	30.008079	31.210931	Cairo Opera House (دار الأوبرا المصرية)	30.043268	31.222719	Opera House
771	Būr Sa'īd	31.256541	32.284115	Central Perk	31.266501	32.312315	Café
730	Aswān	24.093433	32.907038	Aswan Reservoir (خزان اسوان)	24.035147	32.871729	Reservoir
309	Al Iskandarīyah	31.215645	29.955266	Bitash (البيطاش)	31.115870	29.794117	Neighborhood
32	Ad Daqahlīyah	31.036373	31.380691	Costa Coffee	31.046025	31.355804	Coffee Shop

In [11]:
```
egypt_venues.shape
```

Out[11]:
```
(913, 7)
```

We retrieved the popular venues across the country. We could get about 913 venues.

3. Methodology:

To better understand the popularity of given cities, we need to cluster these cities according to the popular venues in each city. Though there exist quite a few clustering algorithms; K-means clustering is an intuitive and a powerul clustering algorithm.

To apply clustering, We need to process the data to into a numerical format. This can be done in two steps:

One-hot encoding of features:

We perform One-hot encoding of venue category column such that we intoduce extra 913 features one for each category. These features will get a value of 1 if the city has this category, and a value of 0 if the category doesn't exist within the city.

In [12]:
```
# one hot encoding
egypt_onehot = pd.get_dummies(egypt_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
egypt_onehot['City'] = egypt_venues['City']
# move neighborhood column to the first column
fixed_columns = ['City'] + list(set(egypt_onehot.columns) - set(['City']))

egypt_onehot = egypt_onehot[fixed_columns]

egypt_onehot.head()
```

Out[12]:

	City	Buffet	Hookah Bar	Bus Station	Social Club	Island	Modern European Restaurant	Soccer Stadium	Campground	Flea Market	...	Aquarium	Kebab Restaurant	Garden
0	Ad Daqahlīyah	0	0	0	0	0	0	0	0	0	...	0	0	0
1	Ad Daqahlīyah	0	0	0	0	0	0	0	0	0	...	0	0	0
2	Ad Daqahlīyah	0	0	0	0	0	0	0	0	0	...	0	0	0
3	Ad Daqahlīyah	0	0	0	0	0	0	0	0	0	...	0	0	0
4	Ad Daqahlīyah	0	0	0	0	0	0	0	0	0	...	0	0	0

5 rows × 148 columns

Grouping across cities:

We now can group the category features for different cities, and take their mean. This new value represents the populatity of a given category relative to all other categories across the city.

In [36]:
```
egypt_grouped = egypt_onehot.groupby('City').mean().reset_index()
print("Shape of the data: ", egypt_grouped.shape)
egypt_grouped.sample(10)
```

```
Shape of the data:  (27, 148)
```

Out[36]:

	City	Buffet	Hookah Bar	Bus Station	Social Club	Island	Modern European Restaurant	Soccer Stadium	Campground	Flea Market	...	Aquarium	Kebab Restaurant	Garden
24	Qinā	0.0	0.019231	0.00	0.0	0.0	0.0	0.0	0.0	0.038462	...	0.0	0.000000	0.0
17	Asyūţ	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
21	Janūb Sīnā'	0.0	0.000000	0.25	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
13	Al Wādī al Jadīd	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
14	As Suways	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
10	Al Qalyūbīyah	0.0	0.034483	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
16	Aswān	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
26	Sūhāj	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0
4	Al Gharbīyah	0.0	0.035088	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.017544	0.0
22	Kafr ash Shaykh	0.0	0.000000	0.00	0.0	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.000000	0.0

10 rows × 148 columns

The results include 148 categories across 27 governorates.

Popular Venues per city:

We use these results to obtain the popular categories in each governorate by sorting the popularity across each city.

In [14]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

In [39]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['City']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
egypt_venues_sorted = pd.DataFrame(columns=columns)
egypt_venues_sorted['City'] = egypt_grouped['City']

for ind in np.arange(egypt_grouped.shape[0]):
    egypt_venues_sorted.iloc[ind, 1:] = return_most_common_venues(egypt_grouped.iloc[ind, :], num_top_v
enues)

egypt_venues_sorted.sample(10)
```

Out[39]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
7	Al Jīzah	Hotel	Café	Sports Club	Middle Eastern Restaurant	Historic Site	Egyptian Restaurant	Italian Restaurant	Lounge	Pastry Shop	Lebanese Restaurant
20	Dumyāţ	Café	Fast Food Restaurant	Beach	Restaurant	Italian Restaurant	Pedestrian Plaza	BBQ Joint	Flea Market	Shopping Mall	Plaza
26	Sūhāj	Café	Dessert Shop	Waterfront	Fast Food Restaurant	Airport	Train Station	Bookstore	Pedestrian Plaza	Ice Cream Shop	Supermarket
5	Al Iskandarīyah	Coffee Shop	Café	Beach	Hotel	Juice Bar	Sports Club	Seafood Restaurant	Ice Cream Shop	Shopping Mall	Fast Food Restaurant
4	Al Gharbīyah	Café	Restaurant	Coffee Shop	Fast Food Restaurant	Juice Bar	Hookah Bar	Fried Chicken Joint	Train Station	Pizza Place	Lebanese Restaurant
24	Qinā	Historic Site	Hotel	Middle Eastern Restaurant	History Museum	Café	Flea Market	Fast Food Restaurant	Italian Restaurant	Pub	Hostel
9	Al Minūfīyah	Café	Middle Eastern Restaurant	Restaurant	Hookah Bar	Coffee Shop	Sports Club	Ice Cream Shop	Snack Place	Egyptian Restaurant	Plaza
22	Kafr ash Shaykh	Café	Fast Food Restaurant	Coffee Shop	Seafood Restaurant	Art Museum	Pier	BBQ Joint	Opera House	Hotel Bar	Airport
18	Banī Suwayf	Café	Dessert Shop	Fried Chicken Joint	Restaurant	Hotel	Airport	Pier	BBQ Joint	Opera House	Hotel Bar
8	Al Minyā	Café	Hotel	Restaurant	Dessert Shop	Fast Food Restaurant	Coffee Shop	Train Station	Sports Club	Waterfront	Shopping Mall

Given the nature of the users of Foursquare API, it seems that "Cafe" would be the most popular venue across most cities. Yet, we care more about landmarks around the city, an improvement of our current approach is to include additional datasets that provides info about landmarks and tourist attractions.

Clustering the data

We can use the egypt_grouped features to cluster the different cities. The K-means clustering algorithm uses euclidian distance based on these features. Thus, cities that share similiar categories would be grouped together. K-means is non-parametric except for the number of clusters. 4 clusters seemed to provide reasonable results.

In [40]:
```
# set number of clusters
kclusters = 4

egypt_grouped_clustering = egypt_grouped.drop('City', 1)
egypt_grouped_clustering.head()
# run k-means clustering
kmeans = KMeans(n_clusters=kclusters,n_init=1000).fit(egypt_grouped_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[40]:
```
array([1, 0, 1, 1, 1, 0, 1, 0, 1, 1])
```

4. Results:

Now, we could identify the cluster of each city along with the set of the most popular venue categories.

In [41]:
```
# add clustering labels
egypt_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

df_egypt_clust = df_egypt.merge(egypt_venues_sorted, on='City')

df_egypt_clust.sample(10) # check the last columns!
```

Out[41]:

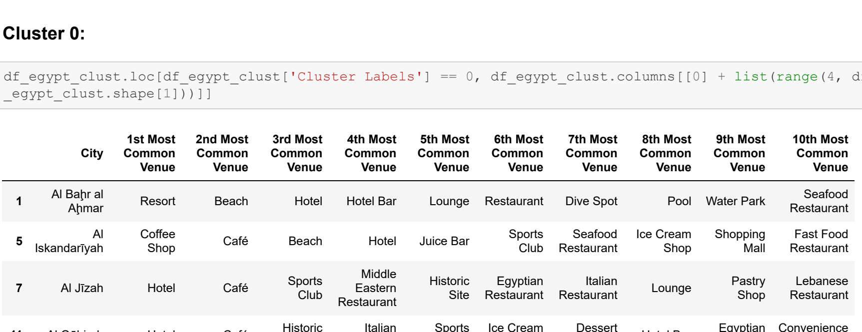
	City	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
14	As Suways	29.973714	32.526267	0	Seafood Restaurant	Toll Plaza	Waterfront	Café	Historic Site	Italian Restaurant	Restaurant	Fast Food Restaurant
17	Asyūţ	27.180956	31.183683	1	Café	Fast Food Restaurant	Restaurant	Fried Chicken Joint	Nightclub	Lounge	Airport	Seafood Restaurant
24	Qinā	25.728768	32.640364	0	Historic Site	Hotel	Middle Eastern Restaurant	History Museum	Café	Flea Market	Fast Food Restaurant	Italian Restaurant
6	Al Ismā'īlīyah	30.604272	32.272252	1	Café	Seafood Restaurant	Beach	Dessert Shop	Canal	Italian Restaurant	Fried Chicken Joint	Burger Joint
1	Al Baḩr al Aḩmar	26.991034	33.877310	0	Resort	Beach	Hotel	Hotel Bar	Lounge	Restaurant	Dive Spot	Pool
10	Al Qalyūbīyah	30.459065	31.178577	1	Café	Fried Chicken Joint	Coffee Shop	Middle Eastern Restaurant	Restaurant	Sports Club	Ice Cream Shop	Waterfront
25	Shamāl Sīnā'	31.162909	33.788933	1	Diner	Beach	Plaza	Restaurant	Café	Seafood Restaurant	Art Museum	Pier
13	Al Wādī al Jadīd	26.068280	29.133540	2	Border Crossing	Performing Arts Venue	Botanical Garden	Juice Bar	Pier	BBQ Joint	Opera House	Hotel Bar
26	Sūhāj	26.447603	31.793197	1	Café	Dessert Shop	Waterfront	Fast Food Restaurant	Airport	Train Station	Bookstore	Pedestrian Plaza
21	Janūb Sīnā'	28.236381	33.625404	0	Rest Area	Bus Station	Bay	Indian Restaurant	Hotel Bar	Botanical Garden	Juice Bar	Pier

In [47]:
```
# create map
map_clusters = folium.Map(location=[latitude+1, longitude+2],zoom_start=6)

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

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df_egypt_clust['Latitude'], df_egypt_clust['Longitude'], df_egypt_clu
st['City'], df_egypt_clust['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), 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=0.7).add_to(map_clusters)

display(map_clusters)
```



We can notice two main clusters of cities, with the other two clusters having only a single city. We can better investigate the clusters by showing the common features among each.

Cluster 0:

In [43]:
```
df_egypt_clust.loc[df_egypt_clust['Cluster Labels'] == 0, df_egypt_clust.columns[[0] + list(range(4, df
_egypt_clust.shape[1]))]]
```

Out[43]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
1	Al Baḩr al Aḩmar	Resort	Beach	Hotel	Hotel Bar	Lounge	Restaurant	Dive Spot	Pool	Water Park	Seafood Restaurant
5	Al Iskandarīyah	Coffee Shop	Café	Beach	Hotel	Juice Bar	Sports Club	Seafood Restaurant	Ice Cream Shop	Shopping Mall	Fast Food Restaurant
7	Al Jīzah	Hotel	Café	Sports Club	Middle Eastern Restaurant	Historic Site	Egyptian Restaurant	Italian Restaurant	Lounge	Pastry Shop	Lebanese Restaurant
11	Al Qāhirah	Hotel	Café	Historic Site	Italian Restaurant	Sports Club	Ice Cream Shop	Dessert Shop	Hotel Bar	Egyptian Restaurant	Convenience Store
12	Al Uqşur	Historic Site	Hotel	Middle Eastern Restaurant	Fast Food Restaurant	History Museum	Café	Flea Market	Boat or Ferry	Italian Restaurant	Pub
14	As Suways	Seafood Restaurant	Toll Plaza	Waterfront	Café	Historic Site	Italian Restaurant	Restaurant	Fast Food Restaurant	Middle Eastern Restaurant	Fried Chicken Joint
16	Aswān	Hotel	Historic Site	Resort	Reservoir	Coffee Shop	Perfume Shop	Egyptian Restaurant	Lake	Airport	Fried Chicken Joint
21	Janūb Sīnā'	Rest Area	Bus Station	Bay	Indian Restaurant	Hotel Bar	Botanical Garden	Juice Bar	Pier	BBQ Joint	Opera House
24	Qinā	Historic Site	Hotel	Middle Eastern Restaurant	History Museum	Café	Flea Market	Fast Food Restaurant	Italian Restaurant	Pub	Hostel

Cities within this cluster are considered the main tourist attractions. This includes coastal cities like Alexandria, Hurgada, and Sharm-Elsheikh, metropolian cities like Cairo, as well as touristic cities like Luxor, Giza, and Aswan. The common theme among these cities are hotels, cafes, and historic sites.

Cluster 1:

In [44]:
```
df_egypt_clust.loc[df_egypt_clust['Cluster Labels'] == 1, df_egypt_clust.columns[[0] + list(range(4, df
_egypt_clust.shape[1]))]]
```

Out[44]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Ad Daqahlīyah	Café	Coffee Shop	Dessert Shop	Kebab Restaurant	Juice Bar	Fast Food Restaurant	Bookstore	Clothing Store	Gaming Cafe	Gym / Fitness Center
2	Al Buḩayrah	Coffee Shop	Train Station	Café	Soccer Stadium	Art Museum	Juice Bar	Pier	BBQ Joint	Opera House	Hotel Bar
3	Al Fayyūm	Café	Scenic Lookout	Garden Center	Hotel	Lake	Art Museum	Pier	BBQ Joint	Opera House	Hotel Bar
4	Al Gharbīyah	Café	Restaurant	Coffee Shop	Fast Food Restaurant	Juice Bar	Hookah Bar	Fried Chicken Joint	Train Station	Pizza Place	Lebanese Restaurant
6	Al Ismā'īlīyah	Café	Seafood Restaurant	Beach	Dessert Shop	Canal	Italian Restaurant	Fried Chicken Joint	Burger Joint	Coffee Shop	Pizza Place
8	Al Minyā	Café	Hotel	Restaurant	Dessert Shop	Fast Food Restaurant	Coffee Shop	Train Station	Sports Club	Waterfront	Shopping Mall
9	Al Minūfīyah	Café	Middle Eastern Restaurant	Restaurant	Hookah Bar	Coffee Shop	Sports Club	Ice Cream Shop	Snack Place	Egyptian Restaurant	Plaza
10	Al Qalyūbīyah	Café	Fried Chicken Joint	Coffee Shop	Middle Eastern Restaurant	Restaurant	Sports Club	Ice Cream Shop	Waterfront	Snack Place	Egyptian Restaurant
15	Ash Sharqīyah	Café	Fast Food Restaurant	Pizza Place	Plaza	Steakhouse	Juice Bar	Bakery	BBQ Joint	Restaurant	Mobile Phone Shop
17	Asyūţ	Café	Fast Food Restaurant	Restaurant	Fried Chicken Joint	Nightclub	Lounge	Airport	Seafood Restaurant	Hotel Bar	Pier
18	Banī Suwayf	Café	Dessert Shop	Fried Chicken Joint	Restaurant	Hotel	Airport	Pier	BBQ Joint	Opera House	Hotel Bar
19	Būr Sa'īd	Café	Seafood Restaurant	Hotel	Fried Chicken Joint	Beach	Restaurant	Fast Food Restaurant	Waterfront	Pizza Place	Italian Restaurant
20	Dumyāţ	Café	Fast Food Restaurant	Beach	Restaurant	Italian Restaurant	Pedestrian Plaza	BBQ Joint	Flea Market	Shopping Mall	Plaza
22	Kafr ash Shaykh	Café	Fast Food Restaurant	Coffee Shop	Seafood Restaurant	Art Museum	Pier	BBQ Joint	Opera House	Hotel Bar	Airport
25	Shamāl Sīnā'	Diner	Beach	Plaza	Restaurant	Café	Seafood Restaurant	Art Museum	Pier	BBQ Joint	Opera House
26	Sūhāj	Café	Dessert Shop	Waterfront	Fast Food Restaurant	Airport	Train Station	Bookstore	Pedestrian Plaza	Ice Cream Shop	Supermarket

This cluster included cities that are less popular, more crowded, and overall less attractive for tourist activities. The main theme for this cluster is cafes and restuarants.

Cluster 2 & 3:

In [45]:
```
df_egypt_clust.loc[df_egypt_clust['Cluster Labels'] == 2, df_egypt_clust.columns[[0] + list(range(4, df
_egypt_clust.shape[1]))]]
```

Out[45]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
13	Al Wādī al Jadīd	Border Crossing	Performing Arts Venue	Botanical Garden	Juice Bar	Pier	BBQ Joint	Opera House	Hotel Bar	Airport	Art Museum

In [46]:
```
df_egypt_clust.loc[df_egypt_clust['Cluster Labels'] == 3, df_egypt_clust.columns[[0] + list(range(4, df
_egypt_clust.shape[1]))]]
```

Out[46]:

	City	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
23	Maţrūḩ	Campground	Airport Terminal	Cafeteria	Supermarket	Pedestrian Plaza	Ice Cream Shop	Lake	Convenience Store	Burger Joint	Temple

Each of these clusters has a single city due to the unique features within each. These two cities are less populated ones due to desertic conditions.

Discussion

The main purpose of this project is to examine the reasons for popularity of some Egyptian cities among tourists over other cities. Using the Foursquare API data of the popular venues, we could cluster the cities in a very natural clusters that seperates tourist-attractive cities from other cities.

World-wide, Egypt is famous for the its pharonic culture, as well as its beaches over the red and mediterranean seas. Thus, tourists come to Egypt for these two main purposes. However, these attractions don't represent the authentic culture of Egypt which can be actually found in cities of cluster 1.

Unlike cluster 0 cities, other clusters cities lack well-equipped hotels to host tourists, as well as suitable programs to explore the city. Cluster 1 cities represent the agricultre nature of Egypt and the humble life of its people. Similiarly, cluster 2 city "Al Wādī al Jadīd" can present a true bedouin experience which is also part of Egyptian culture. And cluster 3 city "Maţrūḩ" is a beautiful coastal city with many landmarks, and great scenery.

Conclusion

In this mini project, we explored the capital cities across Egypt to evaluate their popularity among tourists. Using the location data of each city, the Foursquare API provided the most popular venues in each city within a given radius. Knowing the popular venues, we managed to use K-means clustering to seperate the cities into two disntict clusters according to popularity among tourist. Tourist-attractive cities have hotels, cafes, historic sites, and beaches as main attractions. We can use the knowledge from these cluster to empower the less popular cities to promote the authentic Egyptian culture.

To improve on the current result, a more thorough data would be needed. Many of the attractions across Egypt won't make it to the Foursquare database due to its low popularity in Egypt. Data about local shops and attractions could be collected from some domestic APIs, and surveys.