Introduction: Business Problem

Since the beginning of 2020, Jakarta and many other cities around the world have been under attack by an invisible army called ‘Novel Corona Virus’, also known as ‘Covid-19’. Every effort has been focusing on solving or minimizing problems, including Data Scientists. Data Scientists assessed the situations in places around the world, such as availability, amount, and geographical distribution (i.e. locations) of health infrastructures, such as virus testing centers and authorized hospitals to treat affected patients. In this article, we would like to present a simple analysis for determining strategic locations for the distribution of **masks and medical devices** for COVID-19 treatment, based on confirmed cases on May 28, 2020, and the red zone areas for**“new normal”**condition analysis.

Data

A few Identified factors that influence our decision are:

1. Covid-19 cases per district [“Riwayat File Covid-19 DKI Jakarta”](https://drive.google.com/file/d/1w5ovPYjXREfd7lz9o3GwAUQuqcDYhCC5/view)
2. Total population in DKI Jakarta 2020 [statistik.jakarta.go.id](http://statistik.jakarta.go.id/berapa-kepadatan-penduduk-dki-jakarta-saat-ini/)
3. 10 most population in DKI Jakarta 2020 per district [statistik.jakarta.go.id](http://statistik.jakarta.go.id/berapa-kepadatan-penduduk-dki-jakarta-saat-ini/)
4. Hospital for treatment covid-19 [megapolitan.kompas.com](https://megapolitan.kompas.com/read/2020/04/29/19325161/daftar-13-rumah-sakit-rujukan-covid-19-di-jakarta)

The following data sources are needed to extract/generate the required information:

1. Processed covid-19 positive case data collection on 28 May 2020 At 09.00.
2. The distribution of mask sales based on the population in the DKI Jakarta area.
3. The distribution of mask sales based on 5 districts with the most densely populated populations.
4. New datasets (to be created) from Hospital table that contains city, district, along with their latitudes and longitudes.

Let’s start the Project by importing necessary Python libraries.

**Import necessary libraries**

**import** **requests** *# library to handle requests*  
**import** **pandas** **as** **pd** *# library for data analsysis*  
**import** **numpy** **as** **np** *# library to handle data in a vectorized manner*  
**import** **random** *# library for random number generation*  
**from** **bs4** **import** BeautifulSoup *# library for web scrapping*   
  
*#!conda install -c conda-forge geocoder --yes*  
*#print ("install geocoder")*  
*#!conda install -c conda-forge/label/gcc7 geocoder --yes*  
*#print ("install geocoder2")*  
*#!conda install -c conda-forge/label/cf201901 geocoder --yes*  
*#print ("install geocoder3")*  
*#!conda install -c conda-forge/label/cf202003 geocoder --yes*  
*#print ("install geocoder4")*  
**import** **geocoder**  
  
*#!conda install -c conda-forge geopy --yes*   
**from** **geopy.geocoders** **import** Nominatim *# module to convert an address into latitude and longitude values*  
  
*# libraries for displaying images*  
**from** **IPython.display** **import** Image   
**from** **IPython.core.display** **import** HTML   
   
*# tranforming json file into a pandas dataframe library*  
**from** **pandas.io.json** **import** json\_normalize  
  
*#!conda install -c conda-forge folium=0.5.0 --yes*  
**import** **folium** *# plotting library*  
  
**from** **sklearn.cluster** **import** KMeans  
**import** **matplotlib.cm** **as** **cm**  
**import** **matplotlib.colors** **as** **colors**  
  
print('Folium installed')  
print('Libraries imported.')

Output:

Folium installed  
Libraries imported.

Make sure that we have created a Foursquare developer account and have our credentials handy.

CLIENT\_ID = 'XXXXXXXXXXXXX' *# your Foursquare ID*  
CLIENT\_SECRET = 'XXXXXXXXXXXX' *# your Foursquare Secret*  
VERSION = '20180604'  
LIMIT = 30  
print('Your credentails:')  
print('CLIENT\_ID: ' + CLIENT\_ID)  
print('CLIENT\_SECRET:' + CLIENT\_SECRET)

Output:

Your credentails:  
CLIENT\_ID: XXXXXXXXX  
CLIENT\_SECRET: XXXXXXXXX

Data

**Read and show all data used.**

Read and show data Covid-19 cases per district.

*# Read in the data Covid-19 cases per district (28 May,2020)*  
df\_cases = pd.read\_csv("https://raw.githubusercontent.com/cahyati/Coursera\_Capstone/master/Standar%20Kelurahan%20Data%20Corona%20(28%20MEI**%202020%**20Pukul%2009.00).csv")  
*# View the top rows of the dataset*  
df\_cases



Read and show the top 5 data rows from Covid-19 cases per district.

df\_cases.head()



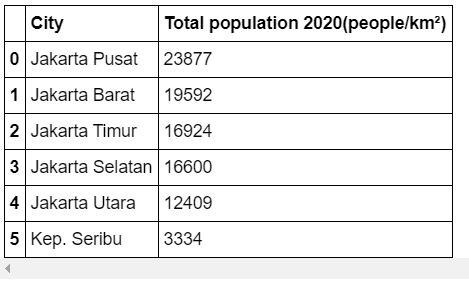
Read and show the bottom 5 data rows from Covid-19 cases per district

df\_cases.tail()



Read and show the total population data in DKI Jakarta 2020.

**import** **pandas** **as** **pd**  
*# Read in the data total population in DKI Jakarta 2020*  
df\_population = pd.read\_csv("https://raw.githubusercontent.com/cahyati/Coursera\_Capstone/master/population2020\_DKI\_Jakarta.csv")  
*# View the top rows of the dataset*  
df\_population

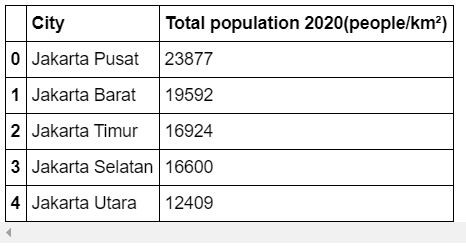


Total population in Jakarta.

df\_population.info()  
  
*# Get the number of total / confirmed POSITIVE cases in Jakarta per 28 May 2020*  
print ("Total Polulation :", df\_population['Total population 2020(people/km²)'].sum())<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 6 entries, 0 to 5  
Data columns (total 2 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 City 6 non-null object  
 1 Total population 2020(people/km²) 6 non-null int64   
dtypes: int64(1), object(1)  
memory usage: 224.0+ bytes  
Total Polulation : 92736

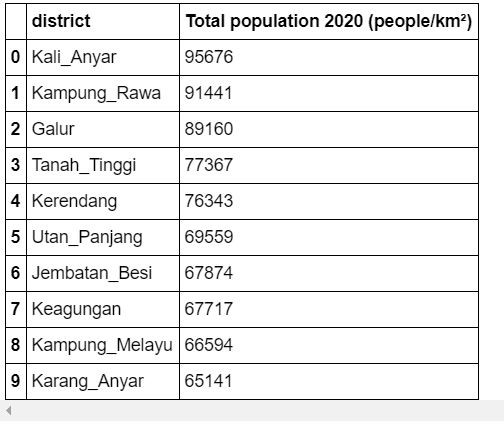
Read and show the top 5 data rows from total population in DKI Jakarta, 2020.

df\_population.head()



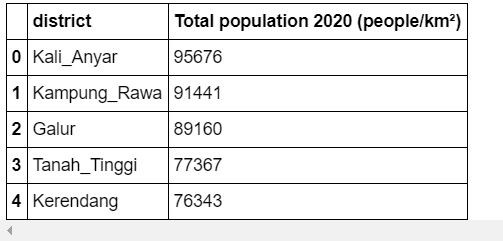
Read and show the data from 10 districts most pupulated in DKI Jakarta, 2020.

*# Read in the data 10 most pupulation in DKI Jakarta 2020 per district*  
df\_most\_population = pd.read\_csv("https://raw.githubusercontent.com/cahyati/Coursera\_Capstone/master/10\_kelurahan%20terpadat\_DKI\_Jakarta.csv")  
*# View the top rows of the dataset*  
df\_most\_population



Read and show the top 5 data rows from 10 most populated areas in DKI Jakarta, 2020 per district.

df\_most\_population.head()



According to the information update from Kompas.com (megapolitan.kompas.com), the following hospitals are the existing reference hospitals for Covid-19 testing in Jakarta area:

1. RSPI Sulianti Saroso, Jakarta Utara
2. RSUP Persahabatan, Jakarta Timur
3. RSPAD Gatot Soebroto, Jakarta Pusat
4. RSUP Fatmawati, Jakarta Selatan
5. RSU Bhayangkara, Jakarta Timur
6. RSAL Mintohardjo, Jakarta Pusat
7. RSUD Cengkareng, Jakarta Barat
8. RSUD Pasar Minggu, Jakarta Selatan
9. RSKD Duren Sawit, Jakarta Timur
10. RS Pelni, Jakarta Barat
11. RSUD Tarakan, Jakarta Pusat
12. RSUD Koja, Jakarta Utara
13. RSU Pertamina Jaya, Jakarta Pusat

Construct a Pandas data frame for subsequent data analysis.

Read and show Hospital data that provide treatment Covid-19.

*# Read in the data Hospital for treatment covid-19*  
df\_hospital = pd.read\_csv("https://raw.githubusercontent.com/cahyati/Coursera\_Capstone/master/Hospital**%20f**or%20treatment**%20c**ovid-19.csv")  
*# View the top rows of the dataset*  
df\_hospital



Read and show the top 5 data rows from Hospital data providing treatment Covid-19.

df\_hospital.head()



This sums up our data mining and data exploration section. In the following METHODOLOGY section, we will describe the process of how to do a ‘Visual’ approach to better understand our data using data science and data analytics tool kits.

Methodology

First, we create a new dataset of only positive cases from the Covid-19 Case table on May 28, 2020.

df\_cases.columns

Output:

Index(['ID\_KEL', 'ID\_KEL.1', 'Nama\_provinsi', 'nama\_kota', 'nama\_kecamatan',  
 'nama\_kelurahan', 'ODP', 'Proses Pemantauan', 'Selesai Pemantauan',  
 'PDP', 'Masih Dirawat', 'Pulang dan Sehat', 'POSITIF', 'Dirawat',  
 'Sembuh', 'Meninggal', 'Self Isolation', 'Keterangan'],  
 dtype='object')

Remove / drop irrelevant columns for this analysis.

df\_cases.drop(columns =["ID\_KEL","ID\_KEL.1", "Nama\_provinsi", "nama\_kecamatan", "ODP", "Proses Pemantauan", "Selesai Pemantauan", "PDP", "Masih Dirawat", "Pulang dan Sehat", "Dirawat", "Sembuh", "Meninggal", "Self Isolation", "Keterangan"], inplace=**True**)  
df\_cases.head()



indexNames = df\_cases[(df\_cases['nama\_kelurahan'] == 'BELUM DIKETAHUI') | (df\_cases['nama\_kota'] == 'LUAR DKI JAKARTA')].index  
df\_cases.drop(indexNames, inplace=**True**)  
df\_cases.head()



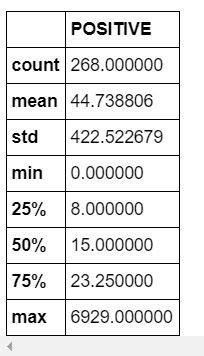
df\_cases.tail()



*# Rename columns name to English*  
df\_cases = df\_cases.rename(columns = {'nama\_kota':'CITY', 'nama\_kelurahan':'DISTRICT', 'POSITIF':'POSITIVE'})  
df\_cases



*# Get the number and the names of each municipality or city in Jakarta*  
df\_cases['CITY'].unique()  
print(df\_cases['CITY'].unique())  
np.array(['JAKARTA TIMUR', 'JAKARTA PUSAT', 'JAKARTA BARAT','JAKARTA SELATAN', 'JAKARTA UTARA', 'KAB.ADM.KEP.SERIBU'],dtype=object)  
*# Get the number of districts (i.e. counts) in Jakarta.*  
*# Get the mean number of positive cases of each district in the city and the standard deviation*  
df\_cases.describe()



Check if there are any missing or null values.

df\_cases.info()  
  
*# Get the number of total / confirmed POSITIVE cases in Jakarta per 28 May 2020*  
df\_cases['POSITIVE'].sum()  
  
print ("positive cases :", ((df\_cases['POSITIVE'].sum()) - 6929))  
  
*# Group the data by CITY*  
df\_cases\_grp = df\_cases.groupby(['CITY'])  
df\_cases\_grp  
  
df\_cases\_grp['POSITIVE'].sum()

Output :

<class 'pandas.core.frame.DataFrame'>  
Int64Index: 268 entries, 0 to 269  
Data columns (total 3 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 CITY 267 non-null object  
 1 DISTRICT 268 non-null object  
 2 POSITIVE 268 non-null int64   
dtypes: int64(1), object(2)  
memory usage: 8.4+ KB  
positive cases : 5061

Out[153]:

CITY  
JAKARTA BARAT 1122  
JAKARTA PUSAT 922  
JAKARTA SELATAN 888  
JAKARTA TIMUR 1162  
JAKARTA UTARA 953  
KAB.ADM.KEP.SERIBU 14  
Name: POSITIVE, dtype: int64

From all these processes: data mining, preparation, and exploration, the total number of Covid-19 confirmed positive cases in Jakarta is 5,061 per 28 May 2020, distributed across 6 main municipalities or cities in Jakarta, across 268 districts (or ‘Kelurahan’) out of just over 92.736 population of Jakarta.

East Jakarta (Jakarta Timur) has the highest number of total POSITIVE cases with 1162 confirmed positives. Just like any other city, each city/municipality has many neighborhoods that can be used to pinpoint the location of the new proposed Covid-19 testing center along with further analysis of the neighborhood using FourSquare API and Folium map visualization technique.

Need to get Latitude & Longitude of Jakarta city and the districts

To assist in the analysis, we will use the ‘’free services” provided by Open Cage Geocode (https://opencagedata.com/) to get the latitude and longitude of cities, districts, particular venues, or neighborhoods. We will start by opening an account and downloading the required dependencies for our analysis. Terms and condition applies. Please refer to their website for further details.

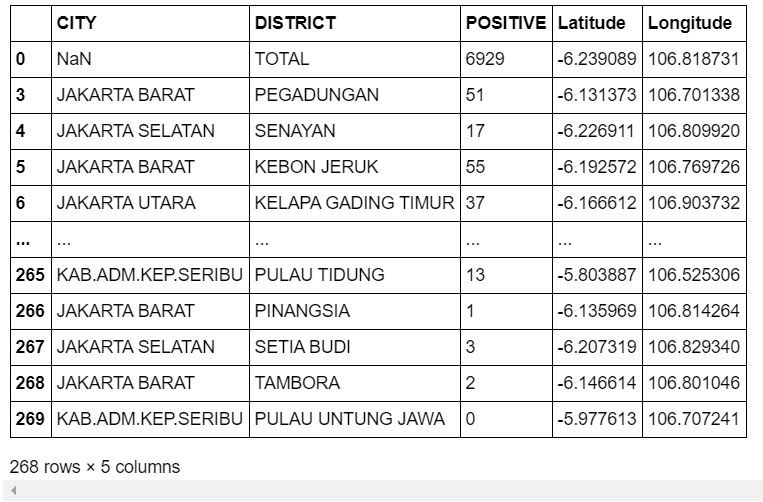
*# Import required package for obtaining Latitude and Longitude of each cities*  
*# Need to get Latitude & Longitude of Jakarta city and the districts*  
*# Get API key from the service provider (Open Cage Geocode)*  
*#!pip install opencage*  
**from** **opencage.geocoder** **import** OpenCageGeocode  
key = 'xxxxxxxxxxxxx'  
geocoder = OpenCageGeocode(key)  
query = 'Jakarta, Indonesia'   
results = geocoder.geocode(query)  
*# print (results)  
# Isolate only the Latitude & Longitude of Jakarta from the Json file*  
lat = results[0]['geometry']['lat']  
lng = results[0]['geometry']['lng']  
print ('The Latitude and Longitude of Jakarta is **{}** and **{}** reprectively.'.format(lat, lng))

Output :

The Latitude and Longitude of Jakarta is -6.1753942 and 106.827183 reprectively.

Similarly, we can use the API service from OpenCage Geocoder to obtain the latitude and longitude of all districts in Jakarta.

*# Get latitude and longitude of all districts*  
list\_lat = [] *# create empty lists for latitude*  
list\_long = [] *# create empty lists for longitude*  
**for** index, row **in** df\_cases.iterrows(): *# iterate over rows in dataframe*  
 District = row['DISTRICT']   
 query = str(District)+', Jakarta'  
 results = geocoder.geocode(query)   
 lat = results[0]['geometry']['lat']  
 long = results[0]['geometry']['lng']  
 list\_lat.append(lat)  
 list\_long.append(long)  
*# create new columns from lists*   
df\_cases['Latitude'] = list\_lat   
df\_cases['Longitude'] = list\_long  
df\_cases



df\_cases.head(10)



Get the latitude and longitude Hospital

Besides, we also need to get the latitude and longitude of all Covid-19 testing centers in Jakarta that we have checked from the source [www.kompas.com](http://www.kompas.com./).

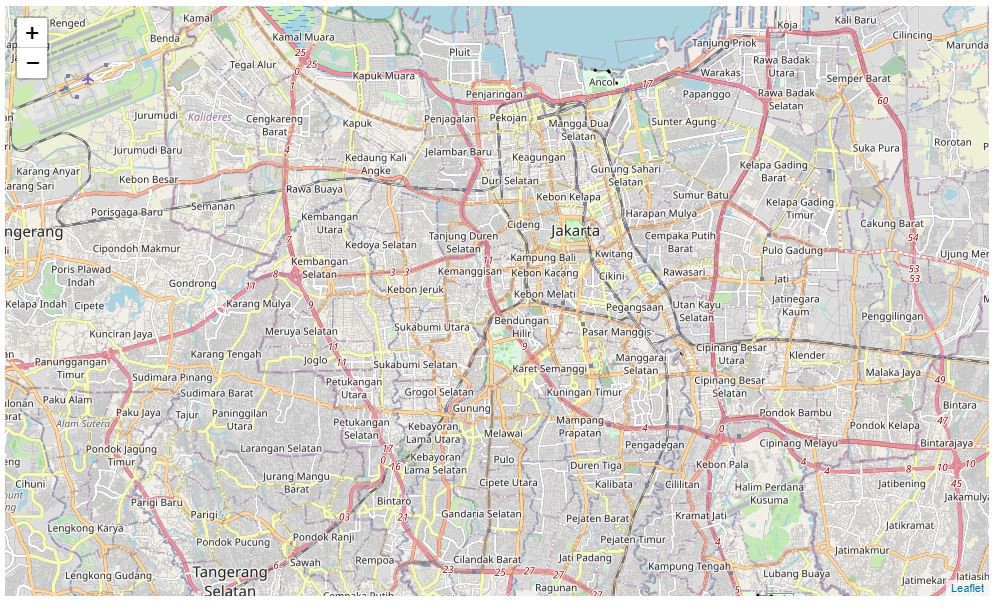
*# Get the latitude and longitude of all of the specialist hospitals*  
list2\_lat = [] *# create empty lists for latitude*  
list2\_long = [] *# create empty lists for longitude*  
**for** index, row **in** df\_hospital.iterrows(): *# iterate over rows in dataframe*  
 hosp = row['Hospital']  
 distr = row['District']   
 query = str(hosp) + ', ' + str(distr) + ', Jakarta'  
 results = geocoder.geocode(query)   
 lat = results[0]['geometry']['lat']  
 long = results[0]['geometry']['lng']  
 list2\_lat.append(lat)  
 list2\_long.append(long)  
*# create new columns from lists*   
df\_hospital['Latitude'] = list2\_lat   
df\_hospital['Longitude'] = list2\_long  
df\_hospital



We then need to know how to get a map of the city that we are interested in (i.e. Jakarta) to present our data to the stakeholders using a ‘Visualization’ approach.

We have downloaded all the required dependencies earlier in the report, and now we are ready to use the FOLIUM API service as described in the following section.

*# Define the map object and then display using the specified latitude and longitude values*  
map\_jkt = folium.Map(location=[-6.2, 106.8], zoom\_start=12)  
map\_jkt



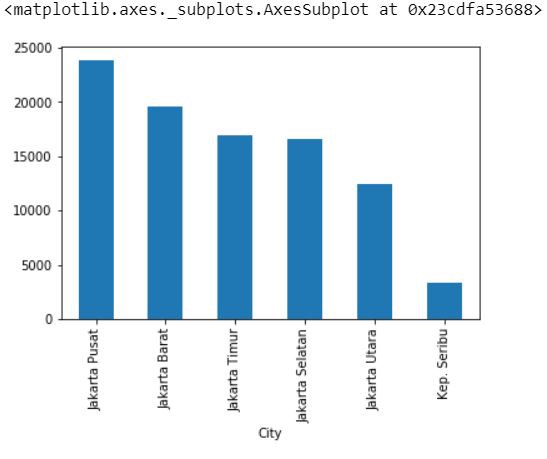
The map shows the main outer ring roads surrounding the city of Jakarta. It does NOT, however, show the official territorial boundary of the city concerning other administrative regions in the east, west, and south of Jakarta.

However, because the author is from Indonesia, we know roughly which neighborhood belongs to Jakarta and which does not. In this scenario, we want to propose a strategic locations (i.e. neighborhood) for the investing group within the Jakarta governmental area.

Results

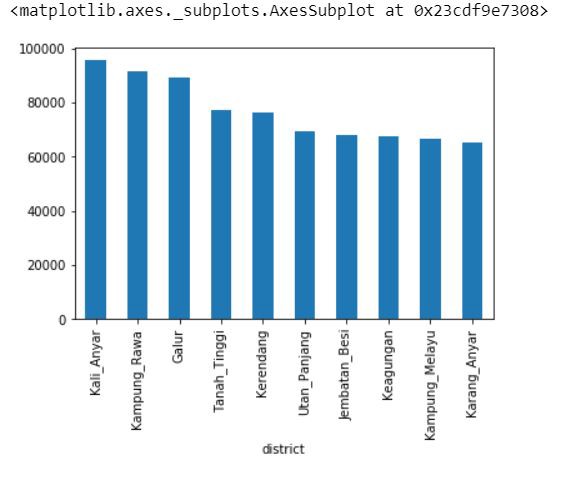
**The chart below show the population density in Jakarta.**

**import** **matplotlib** **as** **plt**  
  
df\_population.set\_index('City')['Total population 2020(people/km²)'].plot.bar()



**The chart below show the population density in Jakarta, per district**

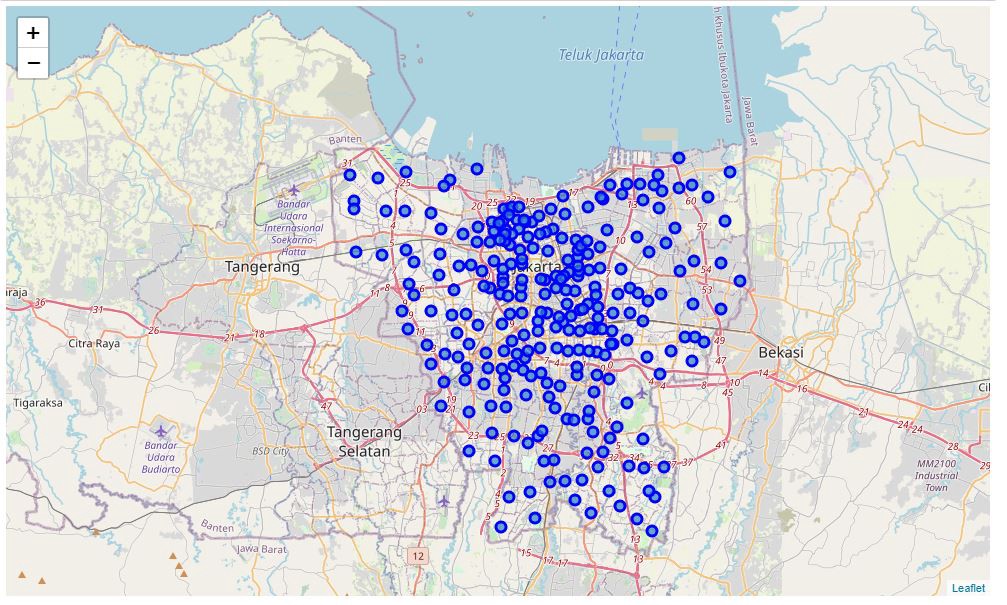
**import** **matplotlib** **as** **plt**  
  
df\_most\_population.set\_index('district')['Total population 2020 (people/km²)'].plot.bar()



**Based on the graph results shown that areas need the distribution of masks the most is Central Jakarta (Jakarta Pusat) with the most populated areas. Then 5 districts that mostly need for a distribution of masks are Kali Anyar, Kampung Rawa, Galur, Tanah Tinggi, and Kerendang**.

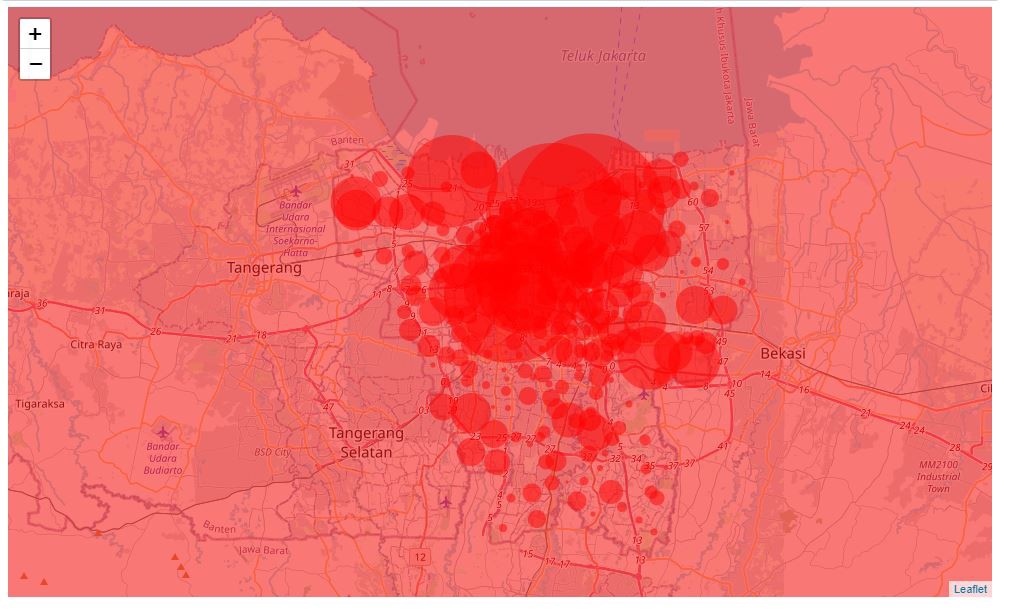
To better understand and estimate the territories or areas that are within the administrative government of Jakarta city, we need to plot all the districts that we have downloaded from the riwayat-file-covid-19-dki-jakarta-jakartagis.hub.arcgis.com site together with their latitude and longitude values. The following lines of Python code will execute the task using Folium API.

*# Construct a map of all districts neighborhood in Jakarta*   
map\_jkt = folium.Map(location=[-6.2, 106.8], zoom\_start=11)  
**for** lat, lng, label **in** zip(df\_cases['Latitude'], df\_cases['Longitude'], df\_cases['DISTRICT']):  
 label = folium.Popup(label, parse\_html=**True**)  
 folium.CircleMarker(  
 location=[lat, lng],  
 radius=5,  
 popup=label,  
 color='blue',  
 fill=**True**,  
 fill\_color='#3186cc',  
 fill\_opacity=0.7,  
 parse\_html=**False**).add\_to(map\_jkt)   
map\_jkt



As we can see from the above map, most of the districts are within the main outer ring roads surrounding the city, and others are situated outside the main ring roads. To solve in our business challenge, we need to show the extent and the distribution medical devices for treatment of COVID-19 positive case-patients within the city of Jakarta based on the number that we obtained from the government site. The following lines of Python code will achieve the task and present the data in a clear visual approach.

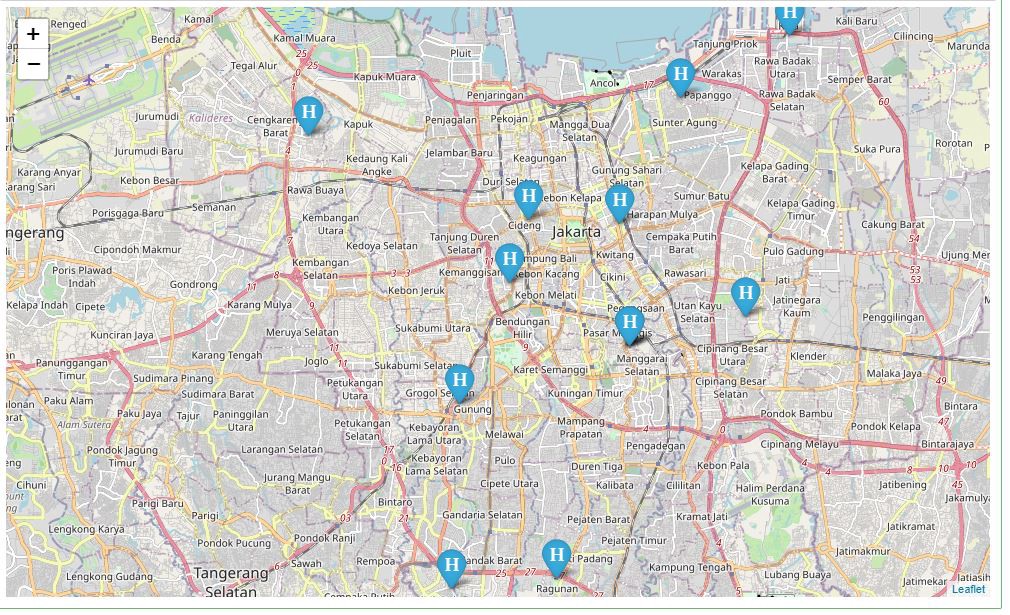
*# Plot a map of Covid-19 distribution in the city of Jakarta per May 28, 2020*  
map\_covid\_jkt = folium.Map(location=[-6.2, 106.8], zoom\_start=11)  
**for** lat,lon,area,size **in** zip(df\_cases['Latitude'],df\_cases['Longitude'],df\_cases['DISTRICT'],df\_cases['POSITIVE']):  
 folium.CircleMarker(  
 location=[lat, lon],  
 popup=area,  
 radius=size/2,  
 color='r',  
 opacity=0.5,  
 fill=**True**,  
 fill\_opacity=0.5,  
 fill\_color='red',  
 ).add\_to(map\_covid\_jkt)  
map\_covid\_jkt



This is a similar map plot that we can see from the government task force for Covid-19 cases in Jakarta. Their graph can be seen in this link: <https://corona.jakarta.go.id/id/peta-persebaran.> As we can see, most of the regions in Jakarta are now in the ‘RED’ zone, with the radius of the circle represent the relative extent of Covid-19 distribution in the City of Jakarta.

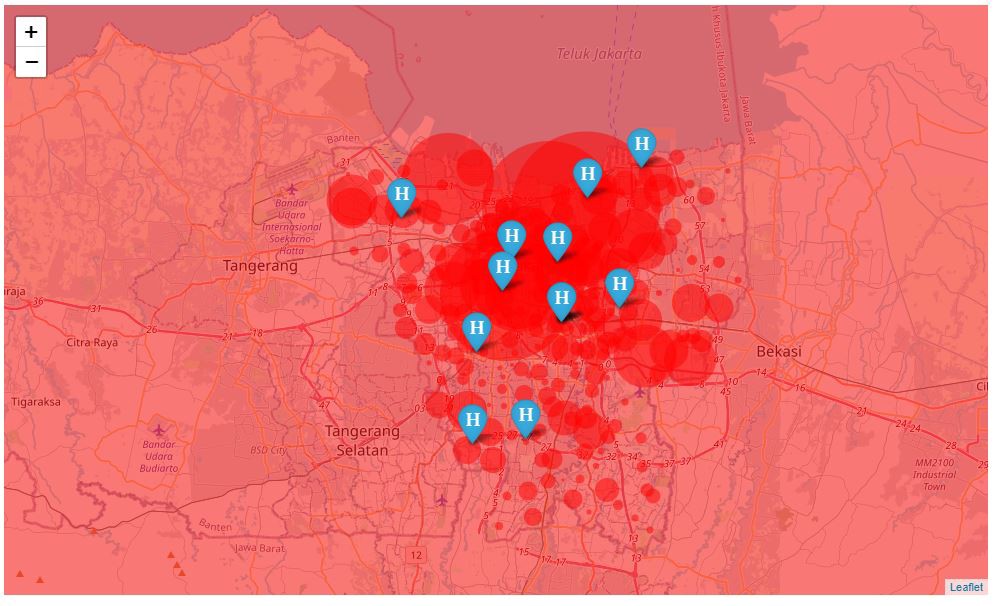
A better presentation of the data would be to use a ‘slider’ in the map that shows the growth of the circle day by day or simply an animation that shows the daily growth of Covid-19 cases in the city. An app developer might develop an App that alerts vehicles/road users that alerts that they are not allowed to pass the RED zone within the city. This App could save lives! The next set of problems that we need to solve is to show the location of existing and approved Covid-19 testing centers (or reference hospitals) and see how well they are distributed to each other within the city and in which regions of Jakarta. The following lines of Python code show how. We will first try to plot the hospitals WITHOUT the RED circles as that might cause distraction.

*# Construct a map of all existing Covid-19 testing hospitals in Jakarta*   
map\_hosp = folium.Map(location=[-6.2, 106.8], zoom\_start=12)  
**for** lat, lng, hosp **in** zip(df\_hospital['Latitude'], df\_hospital['Longitude'], df\_hospital['Hospital']):  
 label = folium.Popup(hosp, parse\_html=**True**)  
 folium.Marker(  
 location=[lat, lng],  
 popup=hosp,  
 icon=folium.Icon(color='blue', icon='header'),  
 ).add\_to(map\_hosp)   
map\_hosp



As you can see, the hospitals are quite sparsely distributed within each other except the two hospitals in the south are relatively close to each other (i.e. Fatmawati and Pasar Minggu hospitals). Let’s see how strategic they are in accomodating the extent of positive cases patients in the city. We can do this by overlaying the two data within a single map as shown in the following codes:

*# Plot a combined map of Covid-19 distribution in the city of Jakarta & current available testing centers*  
map\_covid\_hosp\_jkt = folium.Map(location=[-6.2, 106.8], zoom\_start=11)  
**for** lat,lon,area,size **in** zip(df\_cases['Latitude'],df\_cases['Longitude'],df\_cases['DISTRICT'],df\_cases['POSITIVE']):  
 folium.CircleMarker(  
 location=[lat, lon],  
 popup=area,  
 radius=size/2,  
 color='r',  
 opacity=0.5,  
 fill=**True**,  
 fill\_opacity=0.5,  
 fill\_color='red',  
 ).add\_to(map\_covid\_hosp\_jkt)  
*# Construct a map of all existing Covid-19 testing hospitals in Jakarta*   
**for** lat, lng, hosp **in** zip(df\_hospital['Latitude'], df\_hospital['Longitude'], df\_hospital['Hospital']):  
 label = folium.Popup(hosp, parse\_html=**True**)  
 folium.Marker(  
 location=[lat, lng],  
 popup=hosp,  
 icon=folium.Icon(color='blue', icon='header'),  
 ).add\_to(map\_covid\_hosp\_jkt)   
map\_covid\_hosp\_jkt



**We can see from the results of the distribution of COVID-19 cases and the location of hospitals, almost all hospitals require a lot of medical equipment for COVID-19 treatment. In addition to Fatmawati hospital and the Pasar Minggu hospital, the distribution of the COVID-19 case is not as extensive as other hospitals.**

Discussion

We will try to analyze locations in the red zone based on the location of the hospital in the middle of the red zone. We determine based on the location of the Tarakan Hospital, Central Jakarta.

Let’s begin by trying to get the top 100 venues that are within Tarakan Hospital neighborhood and are within a radius of 500 meters of our candidate Covid-19 testing center using FOURSQUARE API. First, let’s create the GET request URL. Name that URL, url.

prop\_neighborhood = pd.DataFrame({  
 'Hospital':['RSUD Tarakan']  
})neighborhood\_latitude\_list = [] # create empty lists for latitude  
neighborhood\_longitude\_list = [] # create empty lists for longitudefor index, row in prop\_neighborhood.iterrows(): # iterate over rows in dataframe  
 neigh = row['Hospital']   
 query = str(neigh) + ', Jakarta Pusat'  
 results = geocoder.geocode(query)   
 lat = results[0]['geometry']['lat']  
 long = results[0]['geometry']['lng']  
 neighborhood\_latitude\_list.append(lat)  
 neighborhood\_longitude\_list.append(long)  
# create new columns from lists   
prop\_neighborhood['Latitude'] = neighborhood\_latitude\_list   
prop\_neighborhood['Longitude'] = neighborhood\_longitude\_list  
prop\_neighborhood



# Define function that extracts the category of the venue based on the returned JSON file  
def get\_category\_type(row):  
 try:  
 categories\_list = row['categories']  
 except:  
 categories\_list = row['venue.categories']  
   
 if len(categories\_list) == 0:  
 return None  
 else:  
 return categories\_list[0]['name']neighborhood\_latitude = 0  
neighborhood\_longitude = 0address = 'RSUD Tarakan, Jakarta Pusat'geolocator = Nominatim(user\_agent="Hospital\_agent")  
location = geolocator.geocode(address)  
neighborhood\_latitude= location.latitude  
neighborhood\_longitude= location.longitude  
print("RSUD Tarakan: ", neighborhood\_latitude, ",",neighborhood\_longitude)

Output:

RSUD Tarakan: -6.17163765 , 106.81034620548138

Get URL for the API in Tarakan Hospital neighborhood.

LIMIT = 100 # limit of number of venues returned by Foursquare API  
radius = 500 # define radius in meter  
# create URL  
url = ‘[https://api.foursquare.com/v2/venues/explore?&client\_id={}&client\_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(](https://api.foursquare.com/v2/venues/explore?&client_id=%7b%7d&client_secret=%7b%7d&v=%7b%7d&ll=%7b%7d,%7b%7d&radius=%7b%7d&limit=%7b%7d%27.format()  
 CLIENT\_ID,   
 CLIENT\_SECRET,   
 VERSION,   
 neighborhood\_latitude,   
 neighborhood\_longitude,   
 radius,   
 LIMIT)  
# url — not printed for privacy

Next, let’s make a request using REQUEST library, and name our query results for Tarakan Hospital area, results.

# Send the GET request and examine the resutls  
results = requests.get(url).json()  
# results - not printed for shortening of the report

Next, we will use the above function (get\_category\_type) to extract information from the JSON file related to venues in the Tarakan Hospital neighborhood. The following line of code should do the trick:

venues = results['response']['groups'][0]['items']  
   
nearby\_venues = json\_normalize(venues) # flatten JSON# filter columns  
filtered\_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']  
nearby\_venues =nearby\_venues.loc[:, filtered\_columns]# filter the category for each row  
nearby\_venues['venue.categories'] = nearby\_venues.apply(get\_category\_type, axis=1)# clean columns  
nearby\_venues.columns = [col.split(".")[-1] for col in nearby\_venues.columns]  
nearby\_venues



# Check how many venues were returned by Foursquare?  
print('{} venues were returned by Foursquare for {} neighborhood.'.format(nearby\_venues.shape[0], prop\_neighborhood.iloc[0,0]))

Output:

28 venues were returned by Foursquare for RSUD Tarakan neighborhood.

Based on the results generated by the FOURSQUARE API, we can locate the business site around Tarakan hospital and identify affected business locations in the red zone.

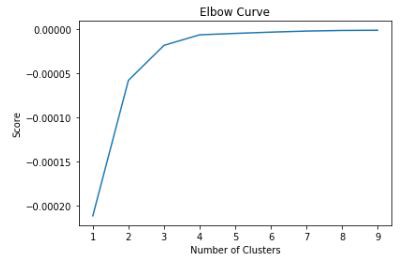
# Sort venues around Tarakan Hospital area  
df\_tarakan\_neigh = nearby\_venues.groupby('categories').count()  
df\_tarakan\_neigh.drop(columns =['lat', 'lng'], inplace=True)  
df\_tarakan\_neigh.sort\_values(by='name', ascending=False, inplace=True)  
df\_tarakan\_neigh1 = df\_tarakan\_neigh.iloc[0:14]  
df\_tarakan\_neigh2 = df\_tarakan\_neigh.iloc[14:]  
df\_tarakan\_neigh1.reset\_index()



The next set of challenges that we need to tackle is to gain slightly more insights (profile) of the Tarakan hospital area. To simplify our analysis, we will just use the Euclidian (distance-based) clustering technique which is part of the unsupervised machine learning technique. In particular, we will use K-means clustering.

To start, we need to decide the best K-value for our analysis. We will let the K-means clustering algorithm to calculate this for us. The following lines of code will carry out the task.

import matplotlib.pyplot as plt# Apply unsupervised Machine Learning clustering technique to the neighborhood data in Tarakan Hospital  
K\_clusters = range(1,10)  
kmeans = [KMeans(n\_clusters=i) for i in K\_clusters]  
Y\_axis = nearby\_venues[['lat']]  
X\_axis = nearby\_venues[['lng']]  
score = [kmeans[i].fit(Y\_axis).score(Y\_axis) for i in range(len(kmeans))]  
# Visualize  
plt.plot(K\_clusters, score)  
plt.xlabel('Number of Clusters')  
plt.ylabel('Score')  
plt.title('Elbow Curve')  
plt.show()



The X-axis of the plot shows various number of K-values that we can use for our clustering analysis. As we can see from the chart, the curve starts flattening out at K=3. Therefore, we will use a K=3 to cluster neighborhoods surrounding our proposed Covid-19 testing center. The following lines of code assign Cluster label to all venues that are within a 500-meter radius of our Covid-19 testing center in Tarakan Hospital area:

kmeans = KMeans(n\_clusters = 3, init ='k-means++')  
kmeans.fit(nearby\_venues[nearby\_venues.columns[2:4]]) # Compute k-means clustering.  
nearby\_venues['cluster\_label'] = kmeans.fit\_predict(nearby\_venues[nearby\_venues.columns[2:4]])  
centers = kmeans.cluster\_centers\_ # Coordinates of cluster centers.  
labels = kmeans.predict(nearby\_venues[nearby\_venues.columns[2:4]]) # Labels of each point  
nearby\_venues



# Check whether all the cluster labels exist in the data  
list(nearby\_venues['cluster\_label'].unique())  
[0, 2, 1]

Output:

[0, 2, 1]

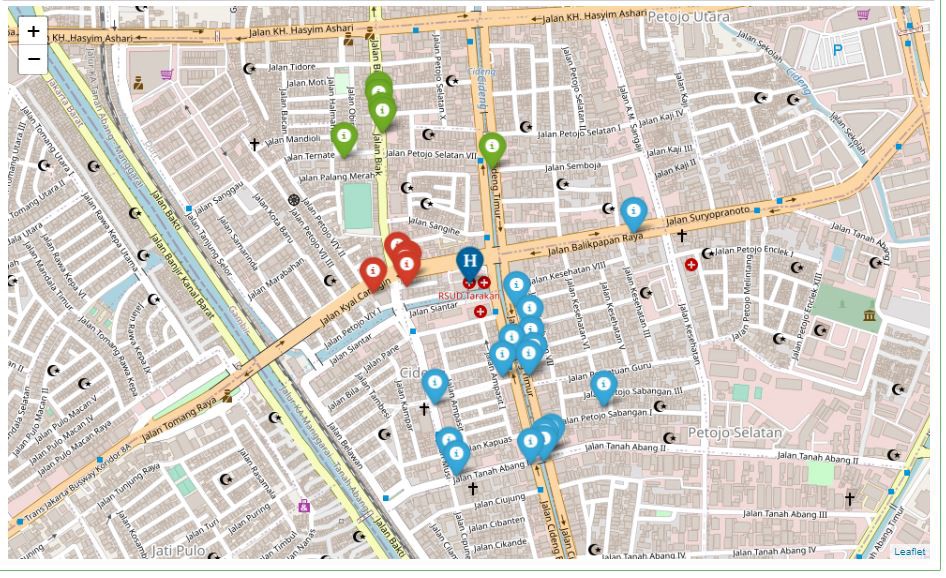
To better visualize the clustering of our neighborhood, we will need to create a custom function that we call ‘regioncolors’ that will assign a color to each area within a 500-meter radius of our proposed facility. The following line of code should help us with this task.

def regioncolors(counter):  
 if counter['cluster\_label'] == 0:  
 return 'green'  
 elif counter['cluster\_label'] == 1:  
 return 'blue'  
 elif counter['cluster\_label'] == 2:  
 return 'red'  
 else:  
 return 'error'nearby\_venues["color"] = nearby\_venues.apply(regioncolors, axis=1)  
nearby\_venues



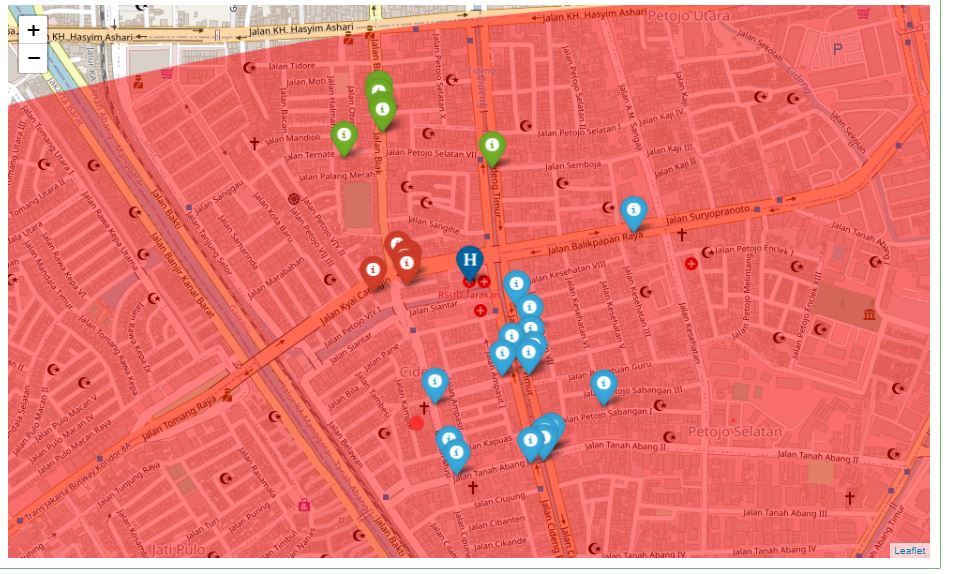
At this stage, we have assigned cluster labels to all of our neighborhood venues, and we have assigned unique colors to each cluster. Next, we can then visualize our clustering analysis to a Folium map to see how all of these venues are geographically distributed within the 500-meter radius that we specified surrounding the proposed facility.

# Construct a map of neighborhood venues in Tarakan Hospital, Cetral Jakarta   
map\_Tarakan = folium.Map(location=[-6.17163765, 106.81034620548138], zoom\_start=16)  
for lat, lng, cat, col in zip(nearby\_venues['lat'], nearby\_venues['lng'], nearby\_venues['categories'], nearby\_venues['color']):  
 label = folium.Popup(cat, parse\_html=True)  
 folium.Marker(  
 location=[lat, lng],  
 popup=cat,  
 icon=folium.Icon(color=col),  
 ).add\_to(map\_Tarakan)   
for lat, lng, neigh in zip(prop\_neighborhood['Latitude'], prop\_neighborhood['Longitude'], prop\_neighborhood['Hospital']):  
 label = folium.Popup(neigh, parse\_html=True)  
 folium.Marker(  
 location=[lat, lng],  
 popup=neigh,  
 icon=folium.Icon(color='darkblue', icon='header'),  
 ).add\_to(map\_Tarakan)  
   
map\_Tarakan



Then we compiled a map of the results of this business location with a map of the distribution of COVID-19 cases.

# Construct a map of neighborhood venues in Tarakan Hospital, Cetral Jakarta   
map\_Redzone = folium.Map(location=[-6.17163765, 106.81034620548138], zoom\_start=16)  
for lat, lng, cat, col in zip(nearby\_venues['lat'], nearby\_venues['lng'], nearby\_venues['categories'], nearby\_venues['color']):  
 label = folium.Popup(cat, parse\_html=True)  
 folium.Marker(  
 location=[lat, lng],  
 popup=cat,  
 icon=folium.Icon(color=col),  
 ).add\_to(map\_Redzone)   
for lat, lng, neigh in zip(prop\_neighborhood['Latitude'], prop\_neighborhood['Longitude'], prop\_neighborhood['Hospital']):  
 label = folium.Popup(neigh, parse\_html=True)  
 folium.Marker(  
 location=[lat, lng],  
 popup=neigh,  
 icon=folium.Icon(color='darkblue', icon='header'),  
 ).add\_to(map\_Redzone)  
for lat,lon,area,size in zip(df\_cases['Latitude'],df\_cases['Longitude'],df\_cases['DISTRICT'],df\_cases['POSITIVE']):  
 folium.CircleMarker(  
 location=[lat, lon],  
 popup=area,  
 radius=size/2,  
 color='r',  
 opacity=0.5,  
 fill=True,  
 fill\_opacity=0.5,  
 fill\_color='red',  
 ).add\_to(map\_Redzone)  
   
map\_Redzone



**The result of analysis is the location of the business which is in the Tarakan hospital neighborhood and is within a radius of 500 meters. Then, we also get the most congested cluster if businesses apply normal conditions in the red zone, potentially increasing cases of contracting the COVID-19 virus within the area.**

Results and Discussion

The project aims to provide information to local people who must be alerted to go out of the house from the distribution of the COVID-19 case in Jakarta. It also aims to provide information on areas that are most needed for a lot of mask distribution, according to population density in the area.

Further, it provides information on which hospitals that need the most medical equipments for COVID-19 treatment, possibly even additional medical personnels (doctors and nurses). It also provides information on the business neighborhood which shall implement Covid-19 health protocol with a high discipline when “new normal” comes.

Conclusion

This project helps mask sellers to understand potential distribution areas according to population density in Jakarta. It also helps the distribution of medical devices for corona care to hospitals that are estimated to have a large number of patients or even helps analyzing which hospitals need additional medical personnel (doctors and nurses).

It will also provide awareness to help business owners who run businesses surrounding the adjacent clusters to be better informed, with the density of people within the business neighborhood