# The Study-Gym

Capstone Project – The Battle of Neighborhood

Zheng Jiahui

May 2020

# Introduction

## 1. The Background

### a) The Country

Singapore, or officially the Republic of Singapore, is a sovereign city-state and island country in maritime Southeast Asia.

Planning Areas, also known as DGP areas or DGP zones, are the main urban planning and census divisions of Singapore delineated by the Urban Redevelopment Authority. There are a total of 55 of these areas, organised into five regions.Planning Areas are further subdivided into subzones for statistical purposes.

### b) The Education System

Singapore's education system has been described as "world-leading" and in 2010 was among those picked out for commendation by the Conservative former UK Education Secretary Michael Gove. According to PISA, an influential worldwide study on educational systems, Singapore has the highest performance in international education and tops in global rankings.In January 2020, Singapore students made up half of the perfect scorers in the IB examinations worldwide. However, the Singaporean culture of Kiasuism spurs parents provide additional support for their children, for fear of their child “losing out” to others. Therefore, parents invest a lot in their children's education, as a good university degree guarantees a higher-paying job.

Besides formal education (local or international schools), 70% of parents also send their children for tuition and 6 in 10 secondary school students pay for private tuition (Straits Times, 2015). The number of tuition centres in Singapore has steadily grow from a list of 700 MOE-registered tuition agencies in 2012, to 800 in 2013 and 840 in 2014, and the tuition industry was worth 1.1 billion in 2018.

Much of the demand for tuition comes from those preparing for these exams. Peer pressure and competition between parents and students also serve to intensify the demand for tuition.

## 2. The Question

### a) The Client

A friend of mine wants to set up a private tuition center, which would target at students of age 12-18 years old, who are currently studying O-Levels/A-Levels or IGCSE/IB syllabus at local schools in Singapore.

However, due to the fierce competition, she has come up with a new idea: she would like to open Study-Gyms , which is a combination of a study room and a gym, in which students from the school could study, get help from tutors, read books and do physical exercises.

Recently, she approached me, asking me to help her find out **where she should open such business ventures.**

### b) The Factors

the factors affecting the suitability of a location are:

1) Population, esp. population of teenagers who are 12-18 years old:
The larger the population of students, the larger the market, as more teenagers would need private tuition services;

2) Public transport networks e.g. metro/MRT/buses, etc. :
The well-developed public transport network enables students who live further sway to come to tuition center conveniently;

3) Number of Schools in the area:
Higher concentration of schools means a large amount of students available.

4) Number of existing Bookstores and Gyms:
Higher number means more intense competition

## Methodology

### 1. The Aim

**There are two aims for this project:**

1) Help my friend to find out suitable locations for her Study-Gym;

2) ) Meanwhile, apply what I have learned from IBM Data Science Professional Certificate Courses as much as possible.

### 2. The Method

**a) The Steps**

1) Find out the planning areas which has both the highest density of Student Population from 12-18 years old, who are studying either in Secondary Schools or Junior Colleges (or equivalent), and the highest number of schools (Secondary and JC Levels);

2) Rank the schools located in the planning areas from 1) in terms of  the number of existing gyms and bookstores.

3) Come up with the list of Schools, whose neighborhoods are the most venues for the business venture.

**b) The Tools**

The skills and tools used in this project are : Webpage Scraping by BeautifulSoup and Requets, Data Wrangling and Data Pre-processing by python pandas and numpy, Data Visualization by matplotlip and folium, Data Analysis, Geocoding by Nominatim, Finding Nearby Venues by Foursquare API, One-Hot Encoding, K-mean Clustering by sklearn.

Other libraries and packages used include seaborn, json, etc. .

The entire project was run on Watson Studio Jupyter Notebook.

## Data Acquisition

### 1. Data Source

1) **The Planning Area of Singapore**, which provides information on area of each Planning Area (2014): https://en.wikipedia.org/wiki/Planning_Areas_of_Singapore:

2) **The Planning Area and Subzones**, which provides information on overall population for each planning area and subzone: https://www.citypopulation.de/en/singapore/admin/

*3)* ***The Demographic Information,*** Table 135 Resident Students Aged 5 Years and Over by Planning Area and Level of Education Attending: https://data.gov.sg/dataset/resident-students-aged-5-years-and-over-by-planning-area-and-level-of-education-attending-2015?resource_id=43762b5d-334f-4161-a4eb-d72d6f598d60

4) **Secondary Schools in Singapore**: https://en.wikipedia.org/wiki/List_of_secondary_schools_in_Singapore

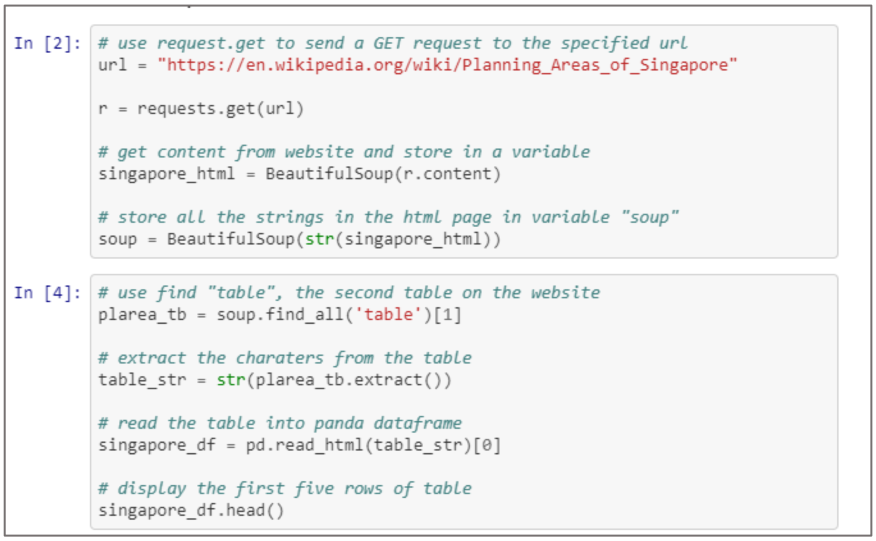
5) **Junior Colleges in Singapore**: https://en.wikipedia.org/wiki/List_of_schools_in_Singapore

6) **Schools with Postal Code and Level of Education**: https://data.gov.sg/dataset/school-directory-and-information

7) **Coordinates for Planning Area, Subzones and Schools**: Manually prepared after getting Coordinates through Nominatim.

## 2. The Code

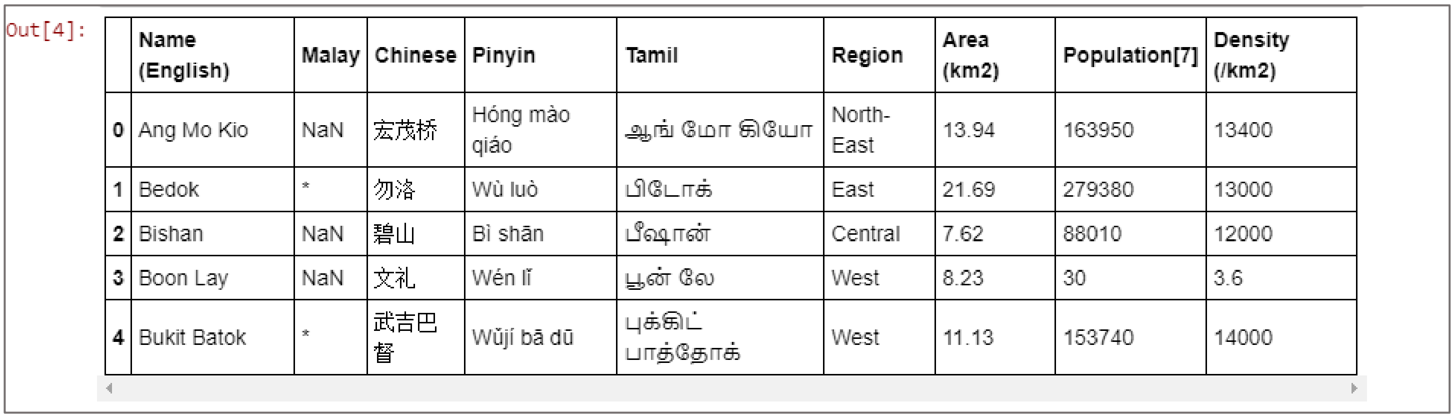
### 1) Web Scraping and Data Wrangling (Example)



```
In [2]: # use request.get to send a GET request to the specified url
        url = "https://en.wikipedia.org/wiki/Planning_Areas_of_Singapore"

        r = requests.get(url)

        # get content from website and store in a variable
        singapore_html = BeautifulSoup(r.content)

        # store all the strings in the html page in variable "soup"
        soup = BeautifulSoup(str(singapore_html))

In [4]: # use find "table", the second table on the website
        plarea_tb = soup.find_all('table')[1]

        # extract the charaters from the table
        table_str = str(plarea_tb.extract())

        # read the table into panda dataframe
        singapore_df = pd.read_html(table_str)[0]

        # display the first five rows of table
        singapore_df.head()
```

**requests**: a Python module that allows you to send HTTP requests. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

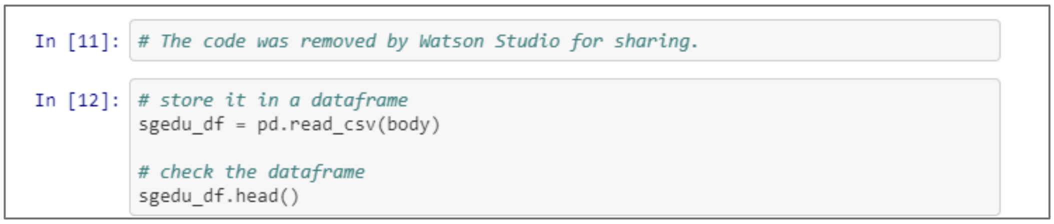
**BeautifulSoup**: a Python library for pulling data out of HTML and XML files.



Out[4]:

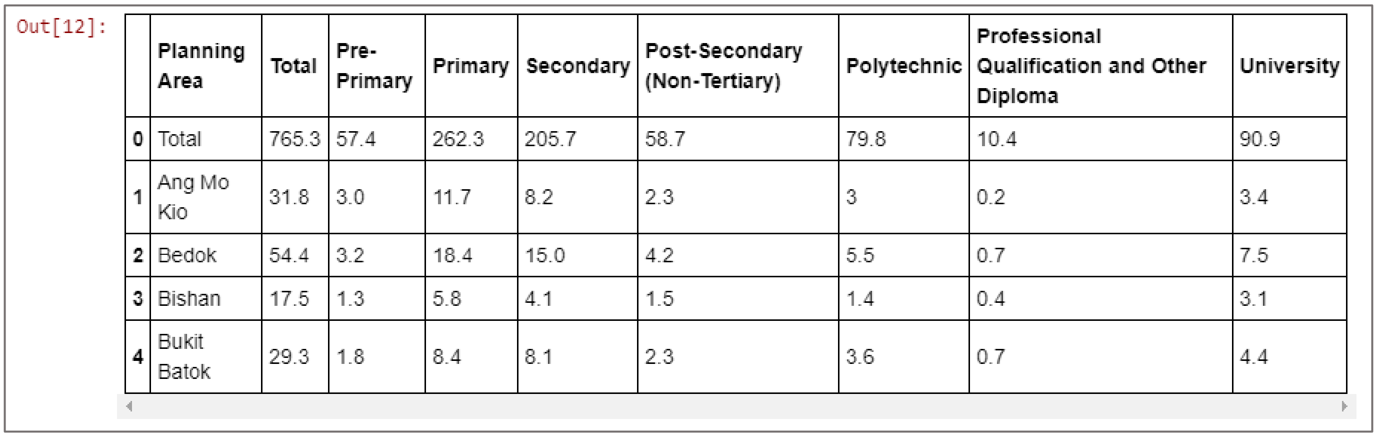
| | Name (English) | Malay | Chinese | Pinyin | Tamil | Region | Area (km2) | Population[7] | Density (/km2) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Ang Mo Kio | NaN | 宏茂桥 | Hóng mào qiáo | ஆங் மோ கியோ | North-East | 13.94 | 163950 | 13400 |
| 1 | Bedok | * | 勿洛 | Wù luò | பிடோக் | East | 21.69 | 279380 | 13000 |
| 2 | Bishan | NaN | 碧山 | Bì shān | பீஷான் | Central | 7.62 | 88010 | 12000 |
| 3 | Boon Lay | NaN | 文礼 | Wén lǐ | பூன் லே | West | 8.23 | 30 | 3.6 |
| 4 | Bukit Batok | * | 武吉巴督 | Wǔjí bā dū | புக்கிட் பாத்தோக் | West | 11.13 | 153740 | 14000 |

### 2) Direct from local csv file uploaded to Watson Studio (Example)



```
In [11]: # The code was removed by Watson Studio for sharing.

In [12]: # store it in a dataframe
         sgedu_df = pd.read_csv(body)

         # check the dataframe
         sgedu_df.head()
```

Watson Studio has inbuilt codes to get the uploaded csv file. However the code can't be shown as it is considered as sensitive.



Out[12]:

| | Planning Area | Total | Pre-Primary | Primary | Secondary | Post-Secondary (Non-Tertiary) | Polytechnic | Professional Qualification and Other Diploma | University |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Total | 765.3 | 57.4 | 262.3 | 205.7 | 58.7 | 79.8 | 10.4 | 90.9 |
| 1 | Ang Mo Kio | 31.8 | 3.0 | 11.7 | 8.2 | 2.3 | 3 | 0.2 | 3.4 |
| 2 | Bedok | 54.4 | 3.2 | 18.4 | 15.0 | 4.2 | 5.5 | 0.7 | 7.5 |
| 3 | Bishan | 17.5 | 1.3 | 5.8 | 4.1 | 1.5 | 1.4 | 0.4 | 3.1 |
| 4 | Bukit Batok | 29.3 | 1.8 | 8.4 | 8.1 | 2.3 | 3.6 | 0.7 | 4.4 |

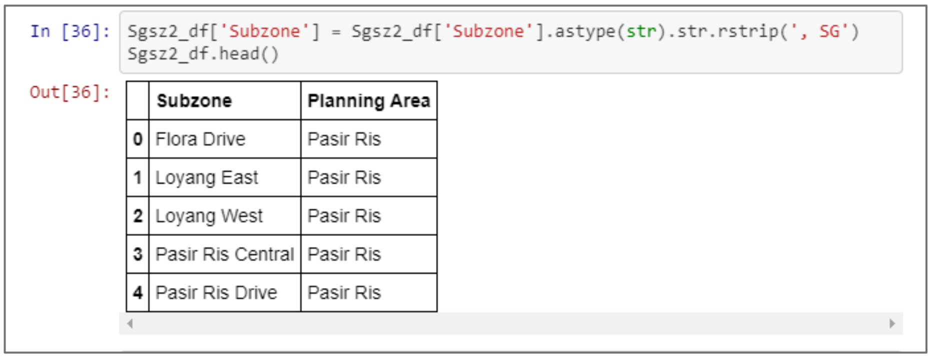
## Data Preparation

### 1. Codes

#### 1) Basic Processing

| Common pandas commands used for modification of Dataframes | | |
|---|---|---|
| pd. read_csv()<br>pd.merge()<br>pd.concat()<br>df[ ] =<br>df.columns =<br>df.shape<br>df.columns | df.drop()<br>df.rename()<br>df.to_numeric ()<br>df.loc ()<br>df.iloc()<br>df.sort_values()<br>df.head () | df.tail ()<br>df.to_csv ()<br>df.reset_index()<br>df.set_index<br>df.replace()<br>df.groupby()<br>df.set_value() |

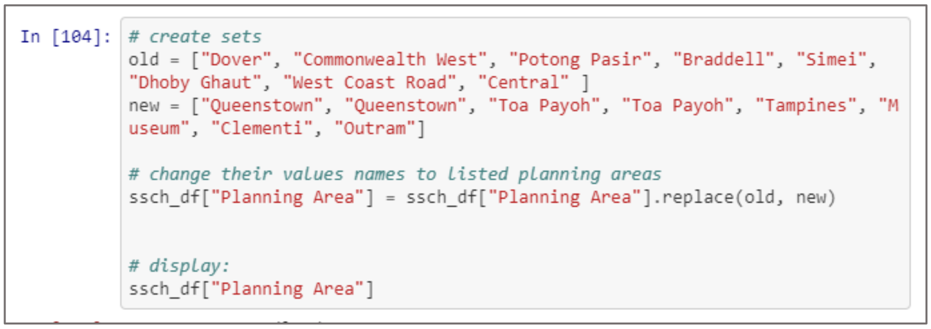
#### 2) Examples



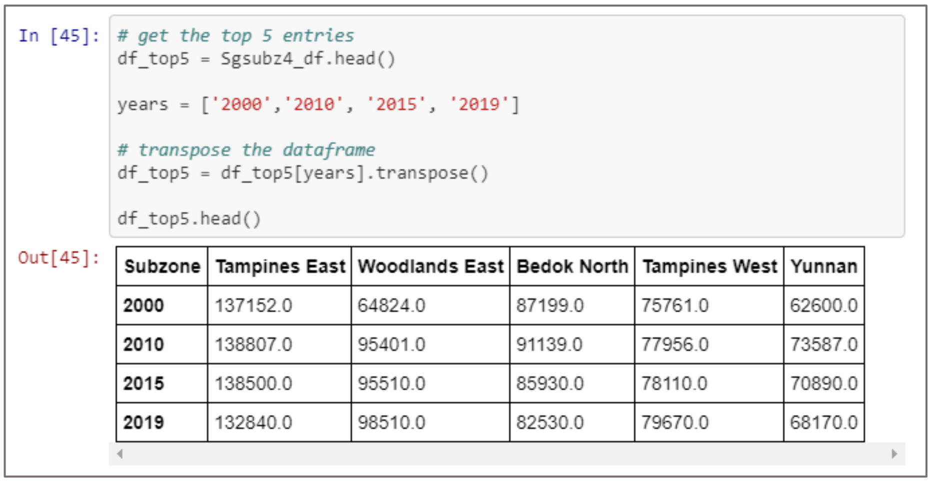
```
In [36]: Sgsz2_df['Subzone'] = Sgsz2_df['Subzone'].astype(str).str.rstrip(', SG')
         Sgsz2_df.head()
```

Out[36]:

| | Subzone | Planning Area |
|---|---|---|
| 0 | Flora Drive | Pasir Ris |
| 1 | Loyang East | Pasir Ris |
| 2 | Loyang West | Pasir Ris |
| 3 | Pasir Ris Central | Pasir Ris |
| 4 | Pasir Ris Drive | Pasir Ris |



```
In [104]: # create sets
          old = ["Dover", "Commonwealth West", "Potong Pasir", "Braddell", "Simei",
          "Dhoby Ghaut", "West Coast Road", "Central" ]
          new = ["Queenstown", "Queenstown", "Toa Payoh", "Toa Payoh", "Tampines", "M
          useum", "Clementi", "Outram"]

          # change their values names to listed planning areas
          ssch_df["Planning Area"] = ssch_df["Planning Area"].replace(old, new)


          # display:
          ssch_df["Planning Area"]
```



```
In [45]: # get the top 5 entries
         df_top5 = Sgsubz4_df.head()

         years = ['2000','2010', '2015', '2019']

         # transpose the dataframe
         df_top5 = df_top5[years].transpose()

         df_top5.head()
```

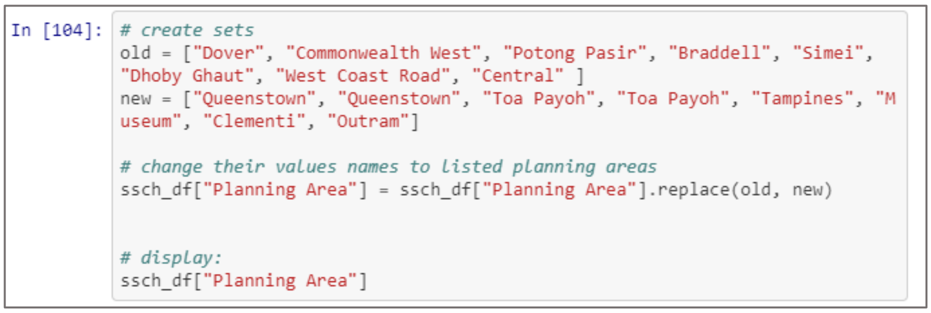
Out[45]:

| Subzone | Tampines East | Woodlands East | Bedok North | Tampines West | Yunnan |
|---|---|---|---|---|---|
| 2000 | 137152.0 | 64824.0 | 87199.0 | 75761.0 | 62600.0 |
| 2010 | 138807.0 | 95401.0 | 91139.0 | 77956.0 | 73587.0 |
| 2015 | 138500.0 | 95510.0 | 85930.0 | 78110.0 | 70890.0 |
| 2019 | 132840.0 | 98510.0 | 82530.0 | 79670.0 | 68170.0 |

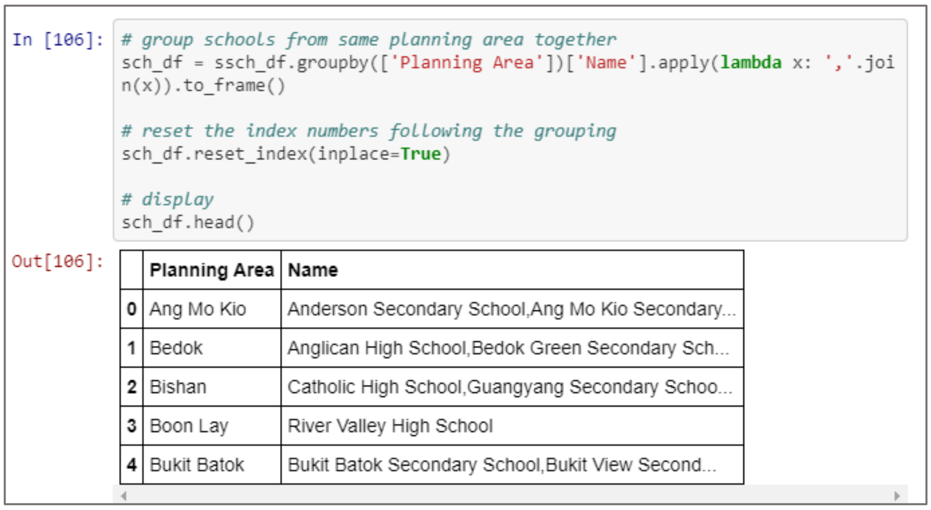
# Data Preparation

## 1. Codes

### 3) Other Examples



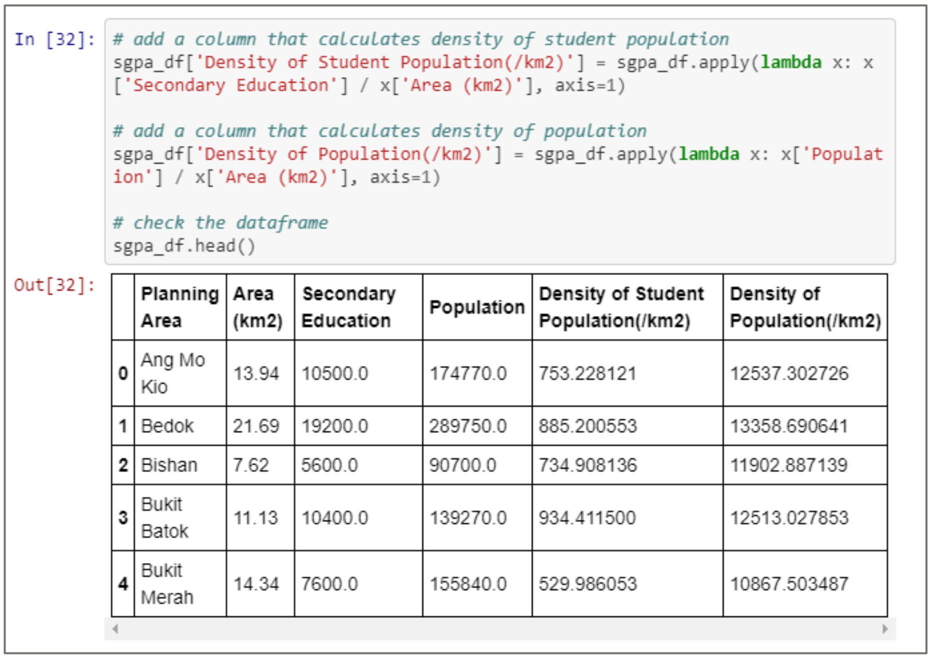
```
In [104]: # create sets
          old = ["Dover", "Commonwealth West", "Potong Pasir", "Braddell", "Simei",
          "Dhoby Ghaut", "West Coast Road", "Central" ]
          new = ["Queenstown", "Queenstown", "Toa Payoh", "Toa Payoh", "Tampines", "M
          useum", "Clementi", "Outram"]

          # change their values names to listed planning areas
          ssch_df["Planning Area"] = ssch_df["Planning Area"].replace(old, new)


          # display:
          ssch_df["Planning Area"]
```



```
In [106]: # group schools from same planning area together
          sch_df = ssch_df.groupby(['Planning Area'])['Name'].apply(lambda x: ','.joi
          n(x)).to_frame()

          # reset the index numbers following the grouping
          sch_df.reset_index(inplace=True)

          # display
          sch_df.head()
```

Out[106]:

|   | Planning Area | Name |
|---|---|---|
| 0 | Ang Mo Kio | Anderson Secondary School,Ang Mo Kio Secondary... |
| 1 | Bedok | Anglican High School,Bedok Green Secondary Sch... |
| 2 | Bishan | Catholic High School,Guangyang Secondary Schoo... |
| 3 | Boon Lay | River Valley High School |
| 4 | Bukit Batok | Bukit Batok Secondary School,Bukit View Second... |



```
In [32]: # add a column that calculates density of student population
         sgpa_df['Density of Student Population(/km2)'] = sgpa_df.apply(lambda x: x
         ['Secondary Education'] / x['Area (km2)'], axis=1)

         # add a column that calculates density of population
         sgpa_df['Density of Population(/km2)'] = sgpa_df.apply(lambda x: x['Populat
         ion'] / x['Area (km2)'], axis=1)

         # check the dataframe
         sgpa_df.head()
```

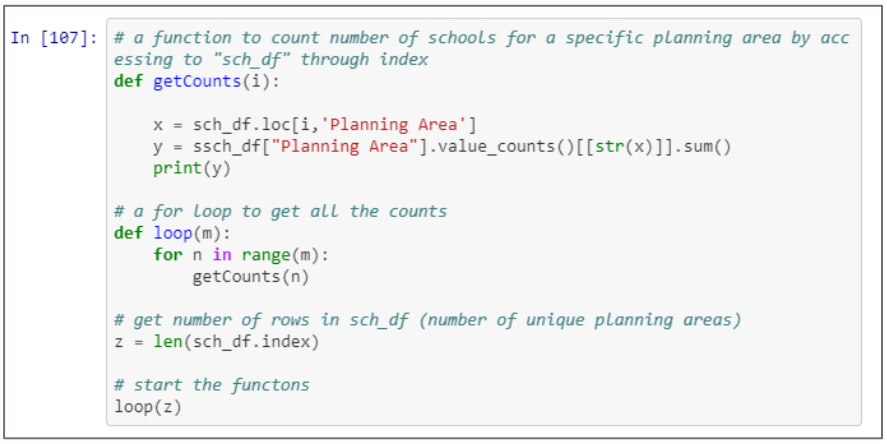
Out[32]:

|   | Planning Area | Area (km2) | Secondary Education | Population | Density of Student Population(/km2) | Density of Population(/km2) |
|---|---|---|---|---|---|---|
| 0 | Ang Mo Kio | 13.94 | 10500.0 | 174770.0 | 753.228121 | 12537.302726 |
| 1 | Bedok | 21.69 | 19200.0 | 289750.0 | 885.200553 | 13358.690641 |
| 2 | Bishan | 7.62 | 5600.0 | 90700.0 | 734.908136 | 11902.887139 |
| 3 | Bukit Batok | 11.13 | 10400.0 | 139270.0 | 934.411500 | 12513.027853 |
| 4 | Bukit Merah | 14.34 | 7600.0 | 155840.0 | 529.986053 | 10867.503487 |

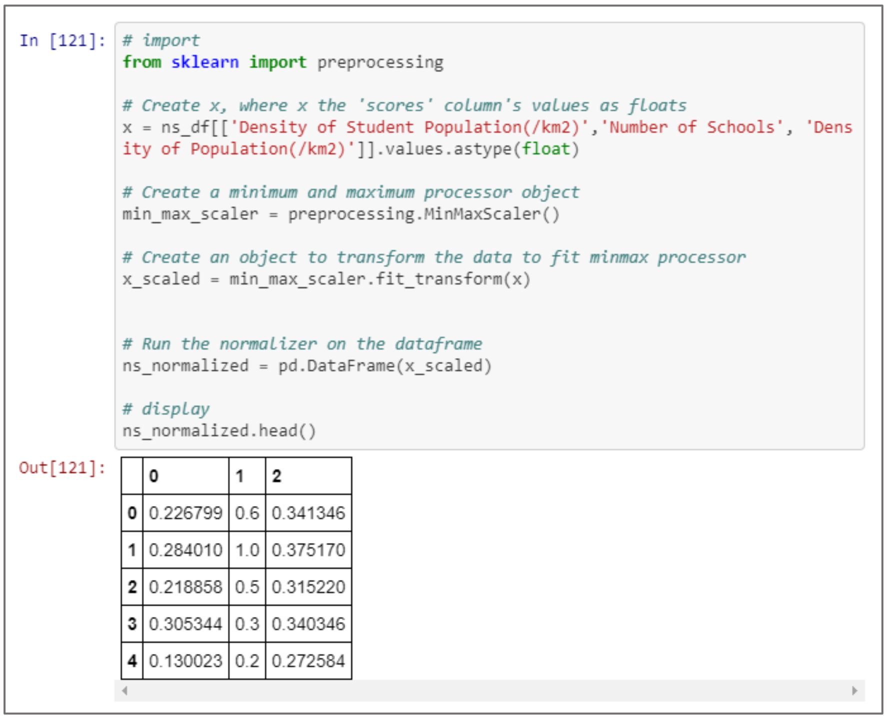
## Data Preparation

### 1. Codes

#### 3) Other Examples



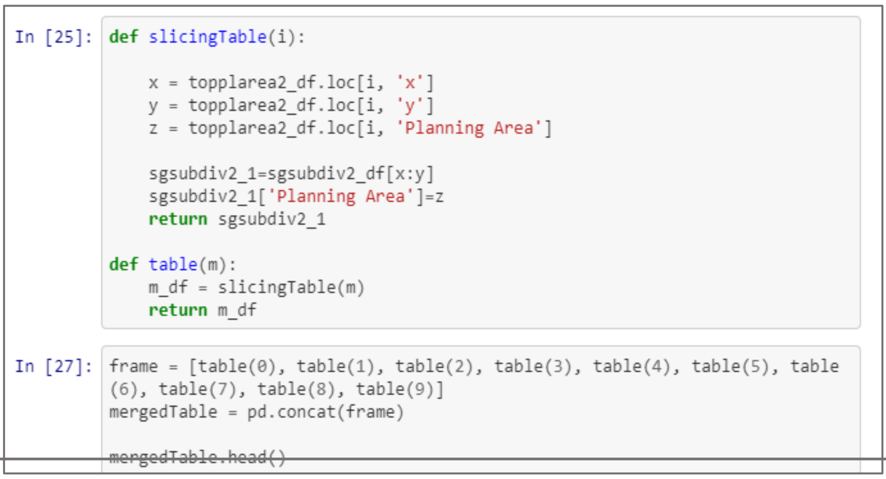
```
In [107]: # a function to count number of schools for a specific planning area by acc
          essing to "sch_df" through index
          def getCounts(i):

              x = sch_df.loc[i,'Planning Area']
              y = ssch_df["Planning Area"].value_counts()[[str(x)]].sum()
              print(y)

          # a for loop to get all the counts
          def loop(m):
              for n in range(m):
                  getCounts(n)

          # get number of rows in sch_df (number of unique planning areas)
          z = len(sch_df.index)

          # start the functons
          loop(z)
```



```
In [121]: # import
          from sklearn import preprocessing

          # Create x, where x the 'scores' column's values as floats
          x = ns_df[['Density of Student Population(/km2)','Number of Schools', 'Dens
          ity of Population(/km2)']].values.astype(float)

          # Create a minimum and maximum processor object
          min_max_scaler = preprocessing.MinMaxScaler()

          # Create an object to transform the data to fit minmax processor
          x_scaled = min_max_scaler.fit_transform(x)


          # Run the normalizer on the dataframe
          ns_normalized = pd.DataFrame(x_scaled)

          # display
          ns_normalized.head()
```

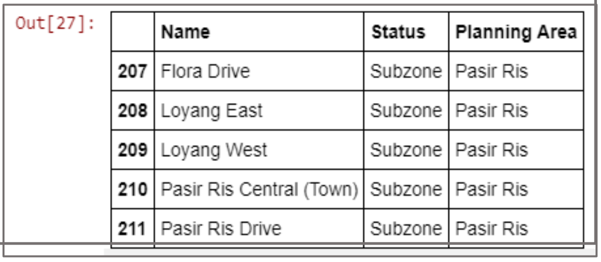
Out[121]:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0.226799 | 0.6 | 0.341346 |
| 1 | 0.284010 | 1.0 | 0.375170 |
| 2 | 0.218858 | 0.5 | 0.315220 |
| 3 | 0.305344 | 0.3 | 0.340346 |
| 4 | 0.130023 | 0.2 | 0.272584 |



```
In [25]: def slicingTable(i):

             x = topplarea2_df.loc[i, 'x']
             y = topplarea2_df.loc[i, 'y']
             z = topplarea2_df.loc[i, 'Planning Area']

             sgsubdiv2_1=sgsubdiv2_df[x:y]
             sgsubdiv2_1['Planning Area']=z
             return sgsubdiv2_1

         def table(m):
             m_df = slicingTable(m)
             return m_df
```

```
In [27]: frame = [table(0), table(1), table(2), table(3), table(4), table(5), table
         (6), table(7), table(8), table(9)]
         mergedTable = pd.concat(frame)

         mergedTable.head()
```

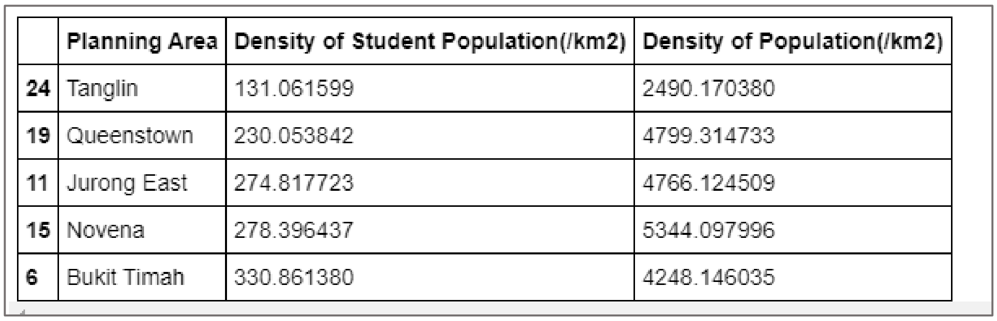


Out[27]:

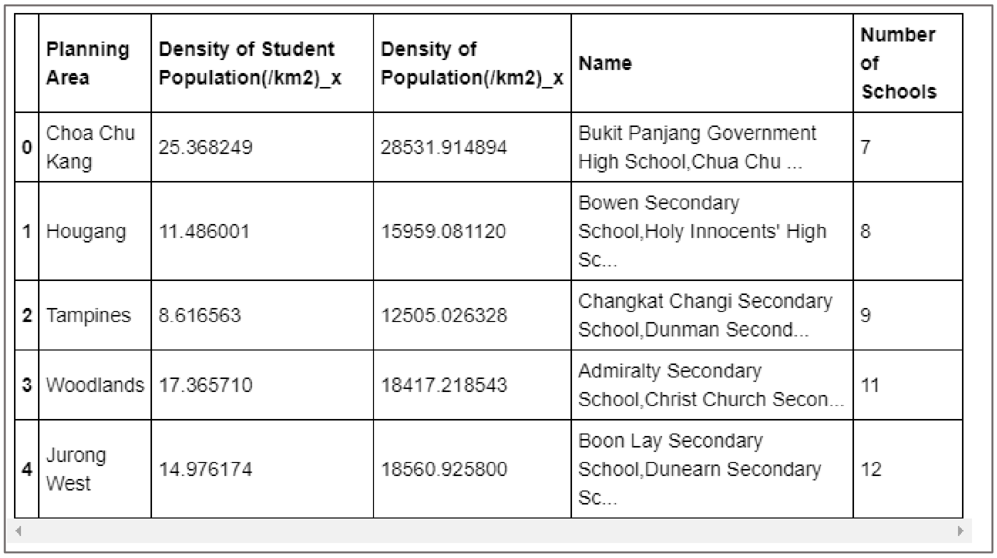
| | Name | Status | Planning Area |
|---|---|---|---|
| 207 | Flora Drive | Subzone | Pasir Ris |
| 208 | Loyang East | Subzone | Pasir Ris |
| 209 | Loyang West | Subzone | Pasir Ris |
| 210 | Pasir Ris Central (Town) | Subzone | Pasir Ris |
| 211 | Pasir Ris Drive | Subzone | Pasir Ris |

# Data Preparation

## 2. Output



| | Planning Area | Density of Student Population(/km2) | Density of Population(/km2) |
|---|---|---|---|
| **24** | Tanglin | 131.061599 | 2490.170380 |
| **19** | Queenstown | 230.053842 | 4799.314733 |
| **11** | Jurong East | 274.817723 | 4766.124509 |
| **15** | Novena | 278.396437 | 5344.097996 |
| **6** | Bukit Timah | 330.861380 | 4248.146035 |

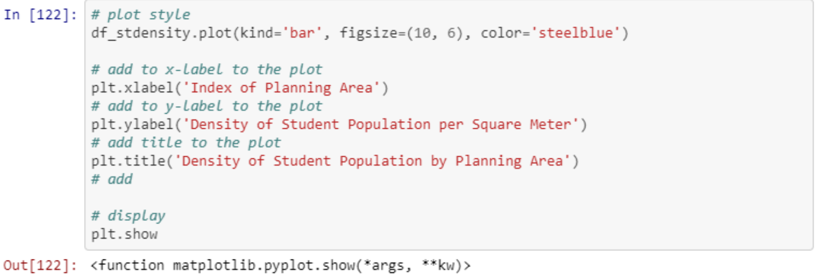


| | Planning Area | Density of Student Population(/km2)_x | Density of Population(/km2)_x | Name | Number of Schools |
|---|---|---|---|---|---|
| **0** | Choa Chu Kang | 25.368249 | 28531.914894 | Bukit Panjang Government High School,Chua Chu ... | 7 |
| **1** | Hougang | 11.486001 | 15959.081120 | Bowen Secondary School,Holy Innocents' High Sc... | 8 |
| **2** | Tampines | 8.616563 | 12505.026328 | Changkat Changi Secondary School,Dunman Second... | 9 |
| **3** | Woodlands | 17.365710 | 18417.218543 | Admiralty Secondary School,Christ Church Secon... | 11 |
| **4** | Jurong West | 14.976174 | 18560.925800 | Boon Lay Secondary School,Dunearn Secondary Sc... | 12 |

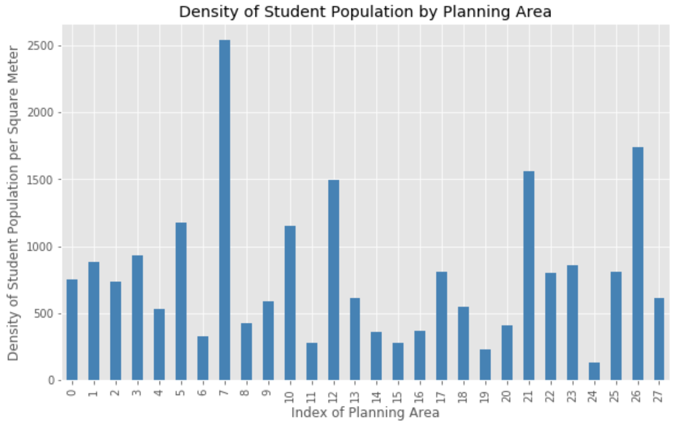
# Data Visualization With Bar Charts

## 1. Bar Charts – Population Density

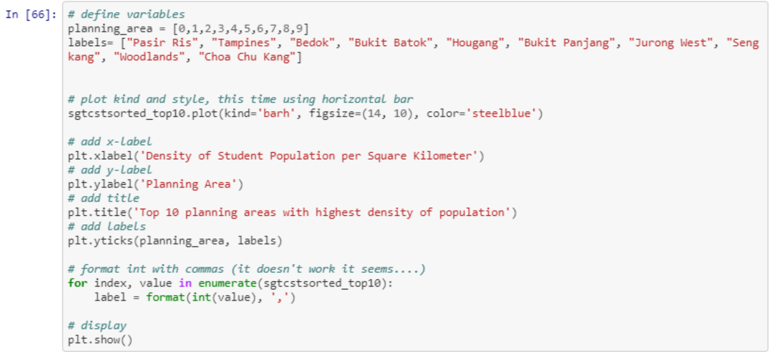
### a) Density of Student Population vs. Planning Area (index)



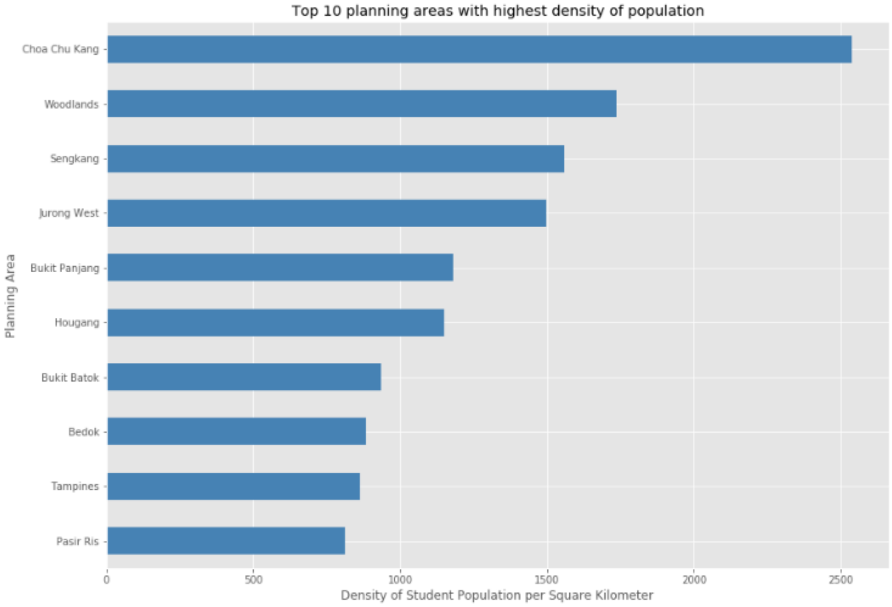
```
In [122]: # plot style
          df_stdensity.plot(kind='bar', figsize=(10, 6), color='steelblue')

          # add to x-label to the plot
          plt.xlabel('Index of Planning Area')
          # add to y-label to the plot
          plt.ylabel('Density of Student Population per Square Meter')
          # add title to the plot
          plt.title('Density of Student Population by Planning Area')
          # add

          # display
          plt.show

Out[122]: <function matplotlib.pyplot.show(*args, **kw)>
```



## 2. Bar Charts – Top Ten Population Density

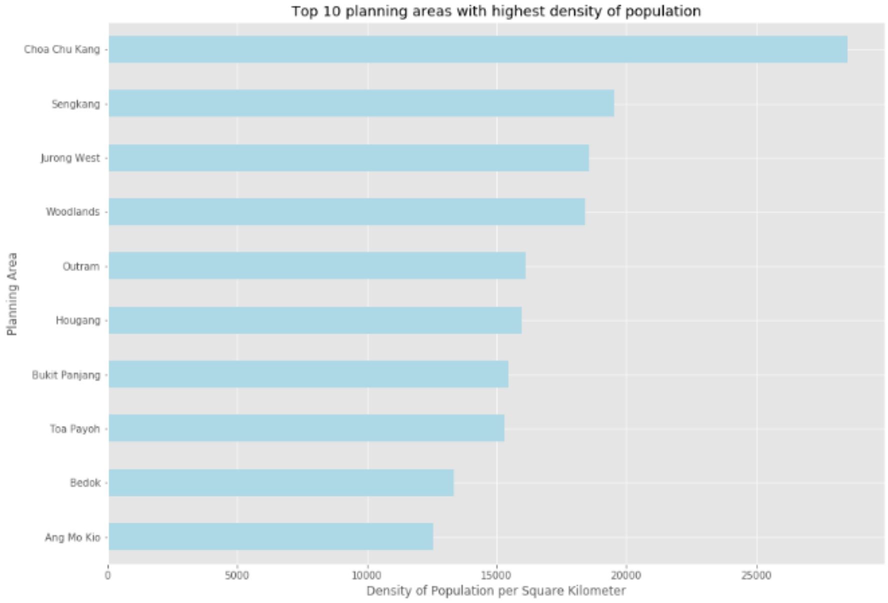


```
In [66]: # define variables
         planning_area = [0,1,2,3,4,5,6,7,8,9]
         labels= ["Pasir Ris", "Tampines", "Bedok", "Bukit Batok", "Hougang", "Bukit Panjang", "Jurong West", "Seng
         kang", "Woodlands", "Choa Chu Kang"]


         # plot kind and style, this time using horizontal bar
         sgtcstsorted_top10.plot(kind='barh', figsize=(14, 10), color='steelblue')

         # add x-label
         plt.xlabel('Density of Student Population per Square Kilometer')
         # add y-label
         plt.ylabel('Planning Area')
         # add title
         plt.title('Top 10 planning areas with highest density of population')
         # add labels
         plt.yticks(planning_area, labels)

         # format int with commas (it doesn't work it seems....)
         for index, value in enumerate(sgtcstsorted_top10):
             label = format(int(value), ',')

         # display
         plt.show()
```



The top ten planning areas with the highest density of population of students attending Secondary Schools and Junior Colleges (or equivalent) are:

***Choa Chu Kang***
***Woodlands***
***Sengkang***
***Jurong West***
***Bukit Panjang***
***Hougang***
***Bukit Batok***
***Betok***
***Tampines***
***Pasir Ris***

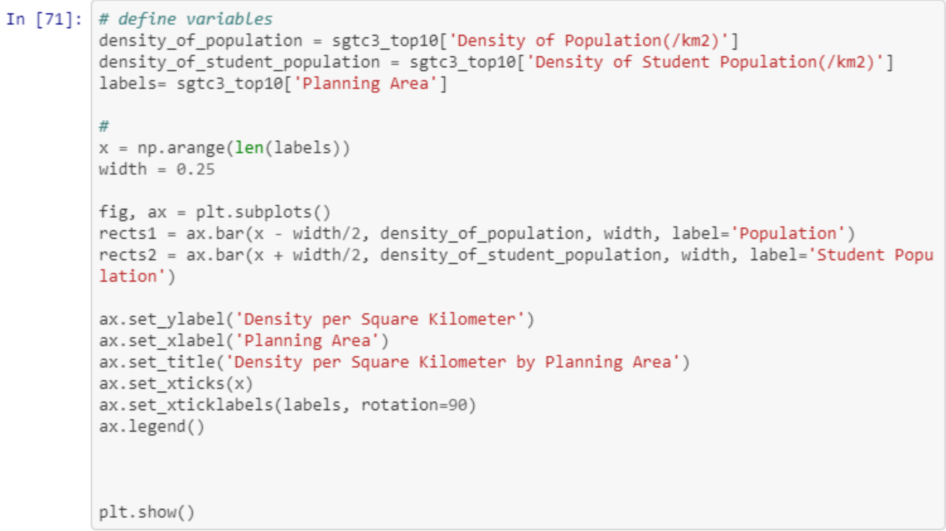


The top ten planning areas with the highest density of population are:

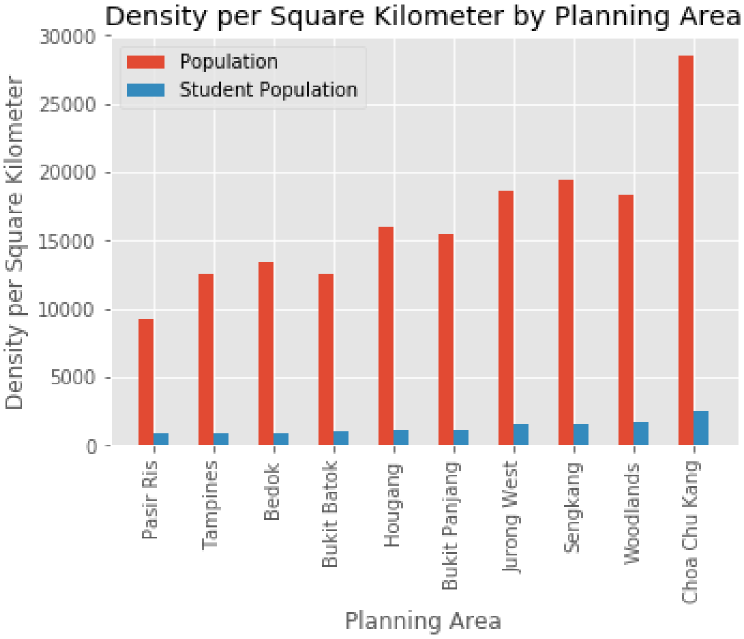
***Choa Chu Kang***
***Sengkang***
***Jurong West***
***Woodlands***
***Outram***
***Hougang***
***Bukit Panjang***
***Toa Payoh***
***Bedok***
***Ang Mo Kio***

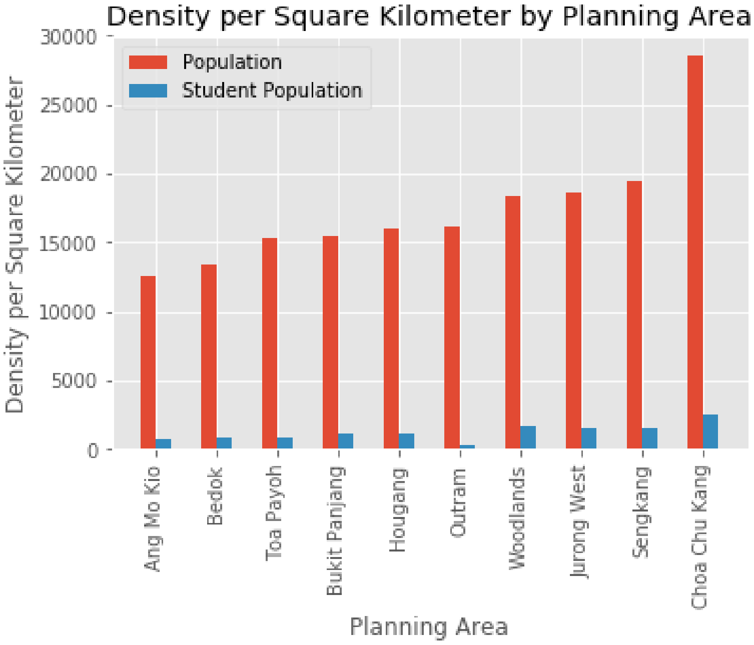
## 3. Bar Charts – Comparison

**The Code**:



```
In [71]: # define variables
         density_of_population = sgtc3_top10['Density of Population(/km2)']
         density_of_student_population = sgtc3_top10['Density of Student Population(/km2)']
         labels= sgtc3_top10['Planning Area']

         #
         x = np.arange(len(labels))
         width = 0.25

         fig, ax = plt.subplots()
         rects1 = ax.bar(x - width/2, density_of_population, width, label='Population')
         rects2 = ax.bar(x + width/2, density_of_student_population, width, label='Student Popu
         lation')

         ax.set_ylabel('Density per Square Kilometer')
         ax.set_xlabel('Planning Area')
         ax.set_title('Density per Square Kilometer by Planning Area')
         ax.set_xticks(x)
         ax.set_xticklabels(labels, rotation=90)
         ax.legend()


         plt.show()
```





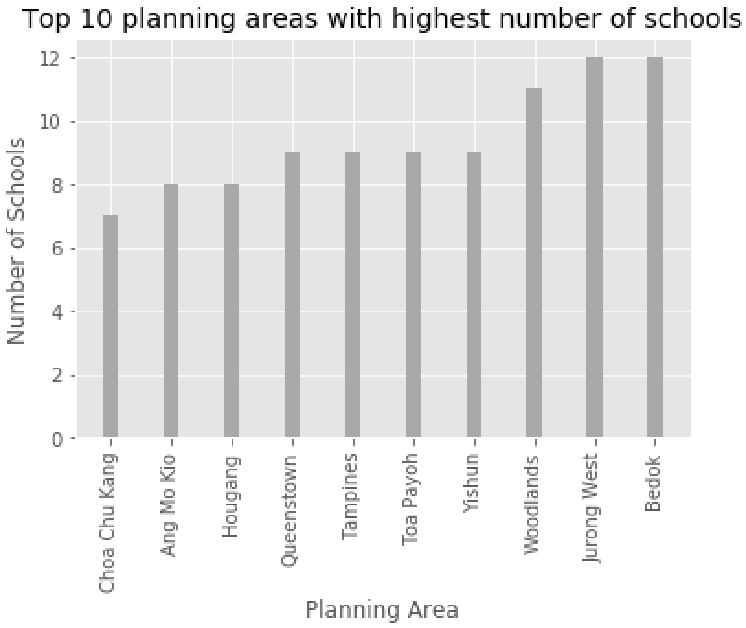
**Analysis**

From the above bar chart we could see that the planning areas with highest density of student population does not necessarily mean that they also have the highest population density.

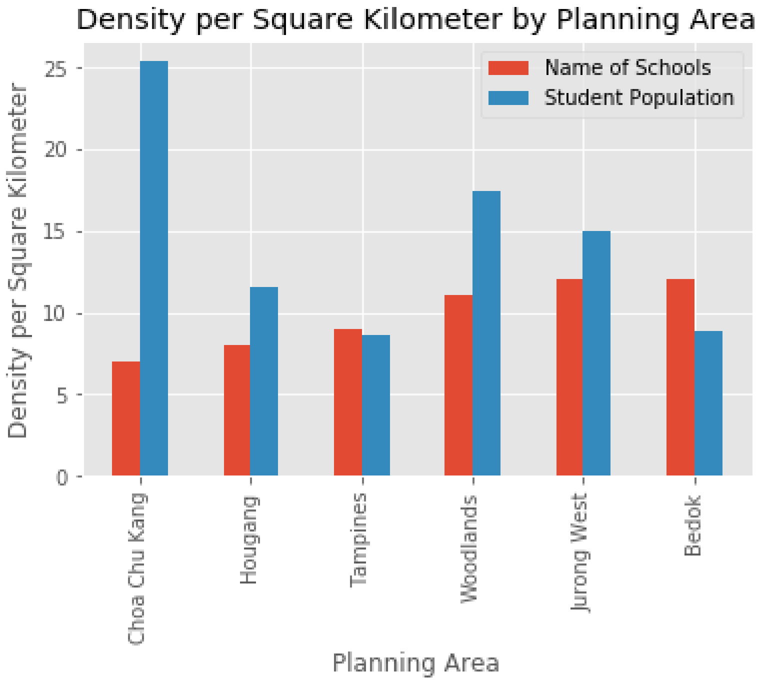
The discrepancy may imply: for planning areas like Bukit Panjang, Jurong West, and Woodlands, the percentage of students in secondary education is higher. Whereas for planning areas like Toa Payoh, and Outram Park, the percentage of students in secondary education is lower.

A conjecture thus arises: does the mismatch the result of number of schools in the specific planning areas

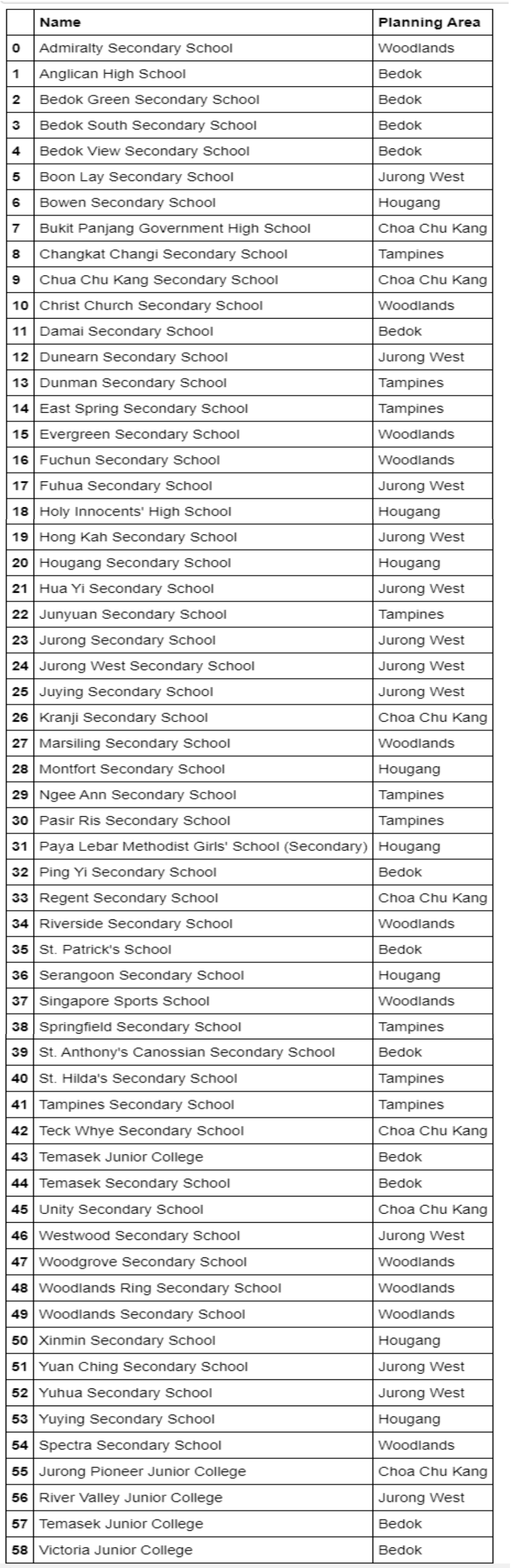
## 4. Bar Charts – Number of Schools



Top 10 Planning Areas with highest number of Secondary Schools/Junior Colleges(or equivalent)



The five planning areas with the highest density of Student (12 – 18 ys ) population and also the highest number of schools.



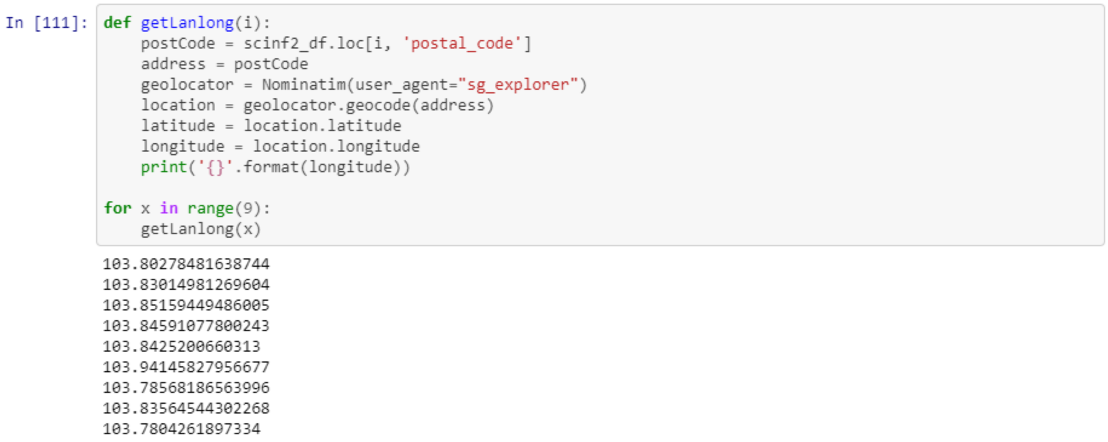
| | Name | Planning Area |
|---|---|---|
| 0 | Admiralty Secondary School | Woodlands |
| 1 | Anglican High School | Bedok |
| 2 | Bedok Green Secondary School | Bedok |
| 3 | Bedok South Secondary School | Bedok |
| 4 | Bedok View Secondary School | Bedok |
| 5 | Boon Lay Secondary School | Jurong West |
| 6 | Bowen Secondary School | Hougang |
| 7 | Bukit Panjang Government High School | Choa Chu Kang |
| 8 | Changkat Changi Secondary School | Tampines |
| 9 | Chua Chu Kang Secondary School | Choa Chu Kang |
| 10 | Christ Church Secondary School | Woodlands |
| 11 | Damai Secondary School | Bedok |
| 12 | Dunearn Secondary School | Jurong West |
| 13 | Dunman Secondary School | Tampines |
| 14 | East Spring Secondary School | Tampines |
| 15 | Evergreen Secondary School | Woodlands |
| 16 | Fuchun Secondary School | Woodlands |
| 17 | Fuhua Secondary School | Jurong West |
| 18 | Holy Innocents' High School | Hougang |
| 19 | Hong Kah Secondary School | Jurong West |
| 20 | Hougang Secondary School | Hougang |
| 21 | Hua Yi Secondary School | Jurong West |
| 22 | Junyuan Secondary School | Tampines |
| 23 | Jurong Secondary School | Jurong West |
| 24 | Jurong West Secondary School | Jurong West |
| 25 | Juying Secondary School | Jurong West |
| 26 | Kranji Secondary School | Choa Chu Kang |
| 27 | Marsiling Secondary School | Woodlands |
| 28 | Montfort Secondary School | Hougang |
| 29 | Ngee Ann Secondary School | Tampines |
| 30 | Pasir Ris Secondary School | Tampines |
| 31 | Paya Lebar Methodist Girls' School (Secondary) | Hougang |
| 32 | Ping Yi Secondary School | Bedok |
| 33 | Regent Secondary School | Choa Chu Kang |
| 34 | Riverside Secondary School | Woodlands |
| 35 | St. Patrick's School | Bedok |
| 36 | Serangoon Secondary School | Hougang |
| 37 | Singapore Sports School | Woodlands |
| 38 | Springfield Secondary School | Tampines |
| 39 | St. Anthony's Canossian Secondary School | Bedok |
| 40 | St. Hilda's Secondary School | Tampines |
| 41 | Tampines Secondary School | Tampines |
| 42 | Teck Whye Secondary School | Choa Chu Kang |
| 43 | Temasek Junior College | Bedok |
| 44 | Temasek Secondary School | Bedok |
| 45 | Unity Secondary School | Choa Chu Kang |
| 46 | Westwood Secondary School | Jurong West |
| 47 | Woodgrove Secondary School | Woodlands |
| 48 | Woodlands Ring Secondary School | Woodlands |
| 49 | Woodlands Secondary School | Woodlands |
| 50 | Xinmin Secondary School | Hougang |
| 51 | Yuan Ching Secondary School | Jurong West |
| 52 | Yuhua Secondary School | Jurong West |
| 53 | Yuying Secondary School | Hougang |
| 54 | Spectra Secondary School | Woodlands |
| 55 | Jurong Pioneer Junior College | Choa Chu Kang |
| 56 | River Valley Junior College | Jurong West |
| 57 | Temasek Junior College | Bedok |
| 58 | Victoria Junior College | Bedok |

List of 59 Schools in the top 5 Planning Areas:

***Choa Chu Kang, Hougang, Tampines, Woodlands, Jurong West and Bedok***

## Geocoding with Nominatim

### 1. The Codes

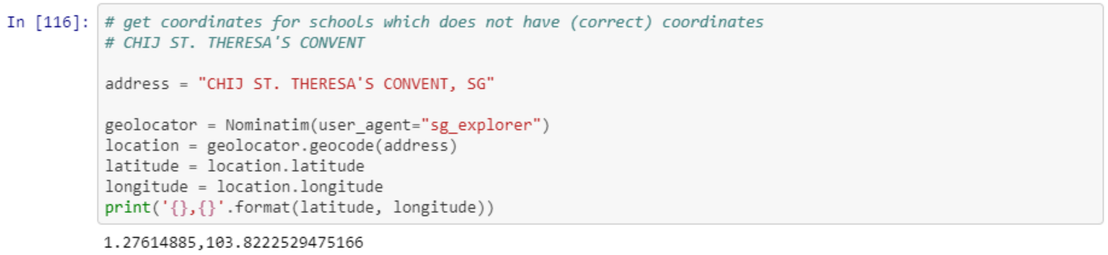


```
In [111]: def getLanlong(i):
              postCode = scinf2_df.loc[i, 'postal_code']
              address = postCode
              geolocator = Nominatim(user_agent="sg_explorer")
              location = geolocator.geocode(address)
              latitude = location.latitude
              longitude = location.longitude
              print('{}'.format(longitude))

          for x in range(9):
              getLanlong(x)

          103.80278481638744
          103.83014981269604
          103.85159449486005
          103.84591077800243
          103.8425200660313
          103.94145827956677
          103.78568186563996
          103.83564544302268
          103.7804261897334
```

I wrote a function to loop through the table to get coordinates for each postal codes of schools.

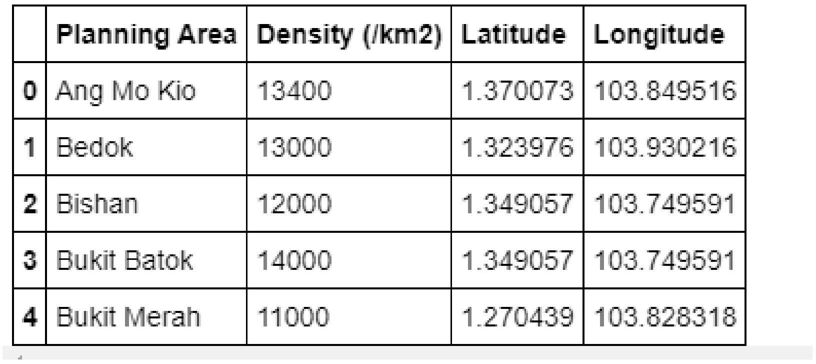
However, there are some errors with a few coordinates and I got them through simple nominatim codes.

Finally I combined all the data for latitudes and longitudes offsite in csv form with excel.



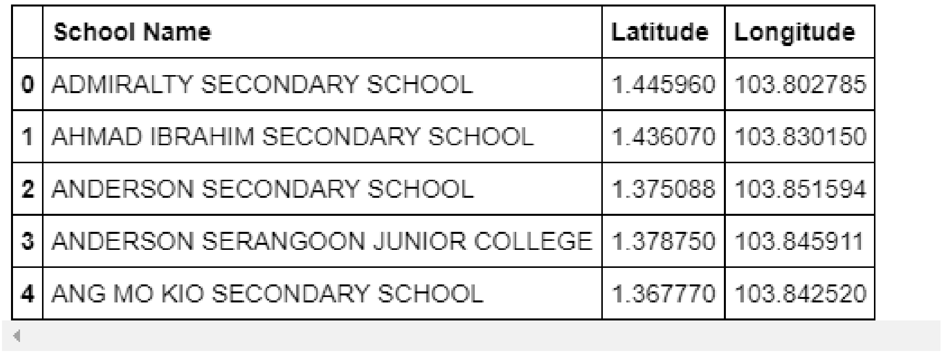
```
In [116]: # get coordinates for schools which does not have (correct) coordinates
          # CHIJ ST. THERESA'S CONVENT

          address = "CHIJ ST. THERESA'S CONVENT, SG"

          geolocator = Nominatim(user_agent="sg_explorer")
          location = geolocator.geocode(address)
          latitude = location.latitude
          longitude = location.longitude
          print('{},{}'.format(latitude, longitude))

          1.27614885,103.8222529475166
```

### 2. The Output



|   | Planning Area | Density (/km2) | Latitude | Longitude |
|---|---|---|---|---|
| 0 | Ang Mo Kio | 13400 | 1.370073 | 103.849516 |
| 1 | Bedok | 13000 | 1.323976 | 103.930216 |
| 2 | Bishan | 12000 | 1.349057 | 103.749591 |
| 3 | Bukit Batok | 14000 | 1.349057 | 103.749591 |
| 4 | Bukit Merah | 11000 | 1.270439 | 103.828318 |

Latitudes and Longitudes for Planning Areas in Singapore



|   | School Name | Latitude | Longitude |
|---|---|---|---|
| 0 | ADMIRALTY SECONDARY SCHOOL | 1.445960 | 103.802785 |
| 1 | AHMAD IBRAHIM SECONDARY SCHOOL | 1.436070 | 103.830150 |
| 2 | ANDERSON SECONDARY SCHOOL | 1.375088 | 103.851594 |
| 3 | ANDERSON SERANGOON JUNIOR COLLEGE | 1.378750 | 103.845911 |
| 4 | ANG MO KIO SECONDARY SCHOOL | 1.367770 | 103.842520 |

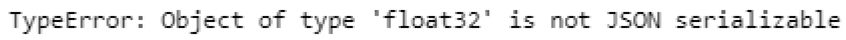
Latitudes and Longitudes for Secondary Schools and Junior Colleges (or equivalent) in Singapore

# Data Visualization (Maps) with Folium

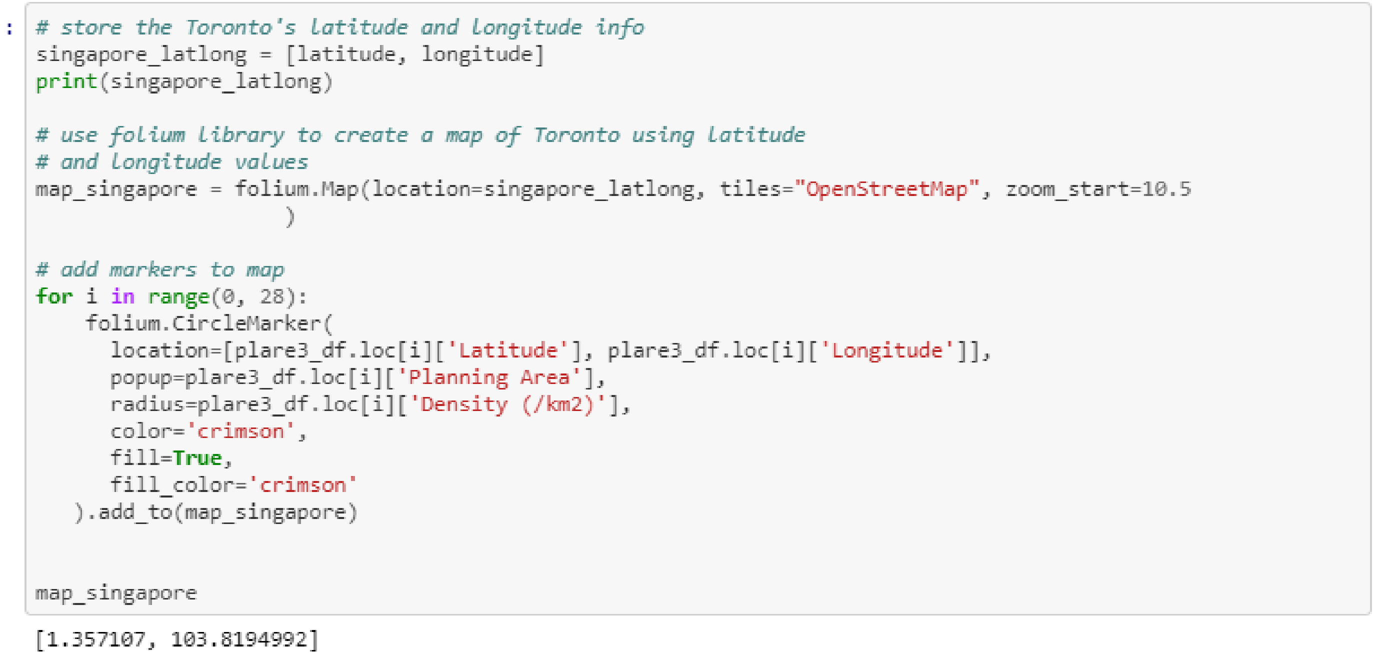
## 1. The (Failure) of Codes

I had quite a failure with this part. While I had no problem creating simple marked maps with folium, when I tried to make marked maps with different circle radii, here is the code for further investigation.

I would also like to make a cholopleth map but I failed to find the relevant Json file. This could be a future exercise.



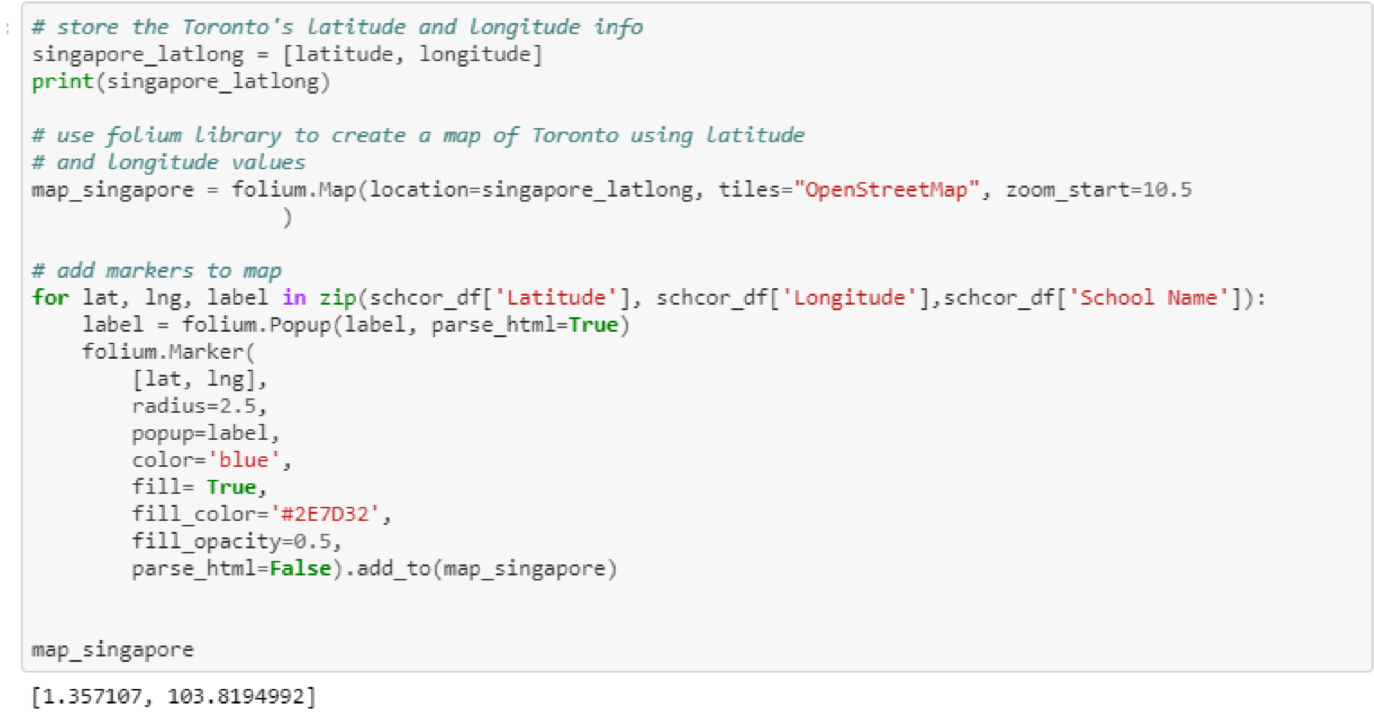
```
TypeError: Object of type 'float32' is not JSON serializable
```



```
# store the Toronto's latitude and longitude info
singapore_latlong = [latitude, longitude]
print(singapore_latlong)

# use folium library to create a map of Toronto using latitude
# and longitude values
map_singapore = folium.Map(location=singapore_latlong, tiles="OpenStreetMap", zoom_start=10.5
                    )

# add markers to map
for i in range(0, 28):
    folium.CircleMarker(
      location=[plare3_df.loc[i]['Latitude'], plare3_df.loc[i]['Longitude']],
      popup=plare3_df.loc[i]['Planning Area'],
      radius=plare3_df.loc[i]['Density (/km2)'],
      color='crimson',
      fill=True,
      fill_color='crimson'
   ).add_to(map_singapore)


map_singapore
```

```
[1.357107, 103.8194992]
```

The successful Code

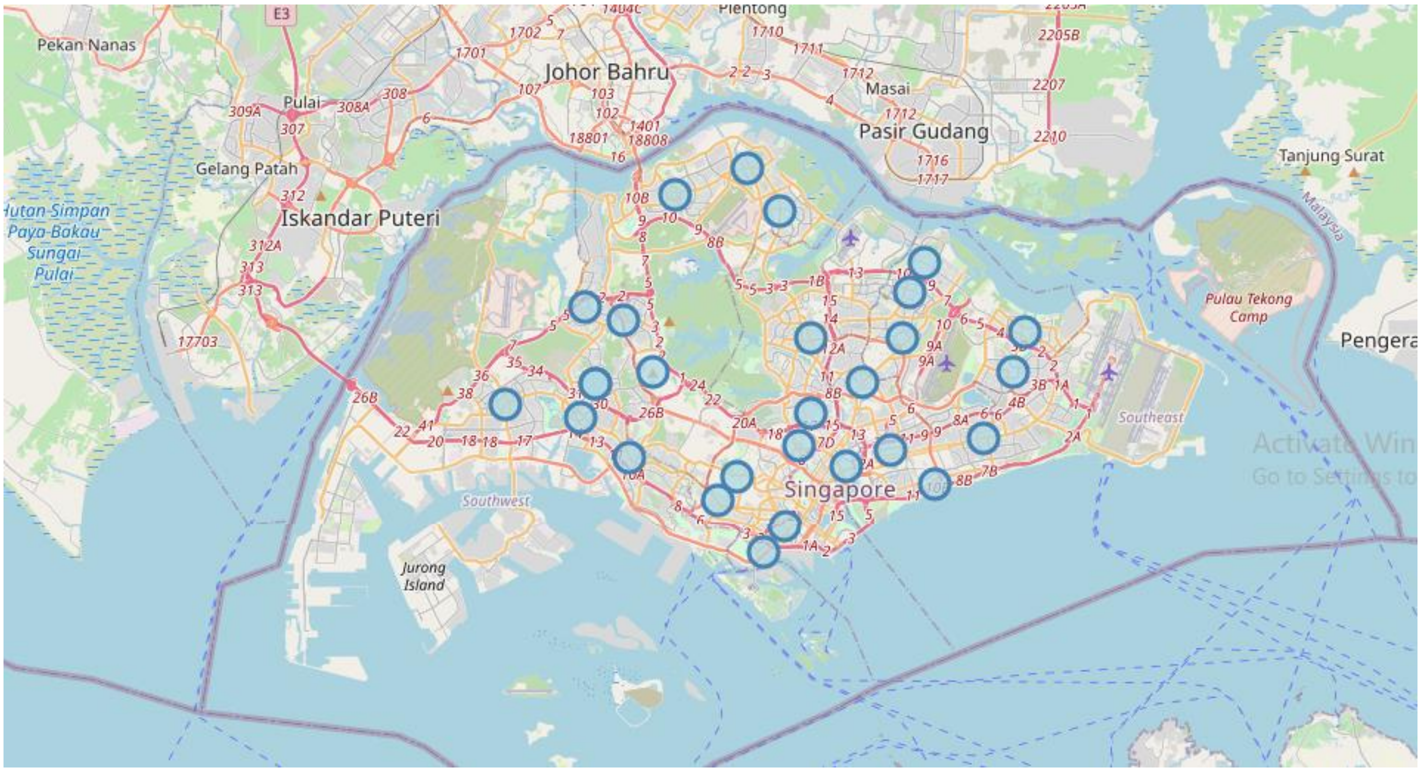


```
# store the Toronto's latitude and longitude info
singapore_latlong = [latitude, longitude]
print(singapore_latlong)

# use folium library to create a map of Toronto using latitude
# and longitude values
map_singapore = folium.Map(location=singapore_latlong, tiles="OpenStreetMap", zoom_start=10.5
                    )

# add markers to map
for lat, lng, label in zip(schcor_df['Latitude'], schcor_df['Longitude'],schcor_df['School Name']):
    label = folium.Popup(label, parse_html=True)
    folium.Marker(
        [lat, lng],
        radius=2.5,
        popup=label,
        color='blue',
        fill= True,
        fill_color='#2E7D32',
        fill_opacity=0.5,
        parse_html=False).add_to(map_singapore)


map_singapore
```

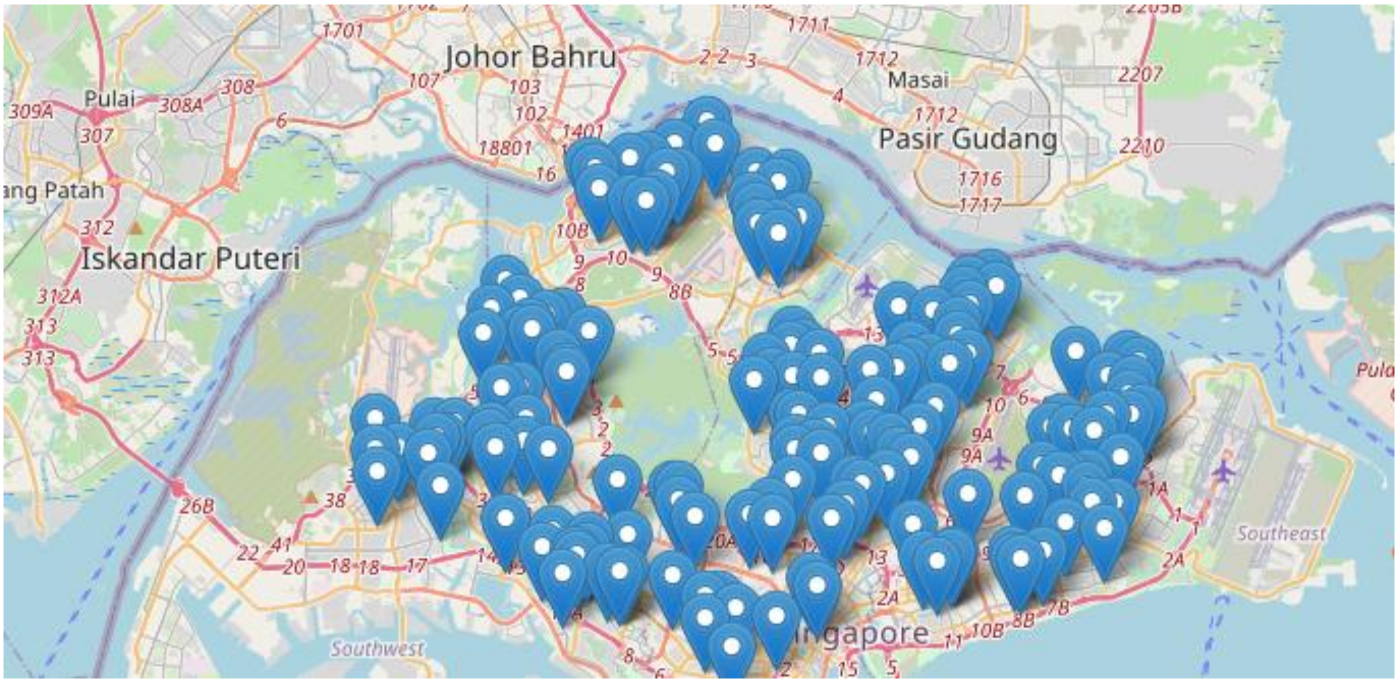
```
[1.357107, 103.8194992]
```

## Data Visualization (Maps) with Folium

### 2. The Output



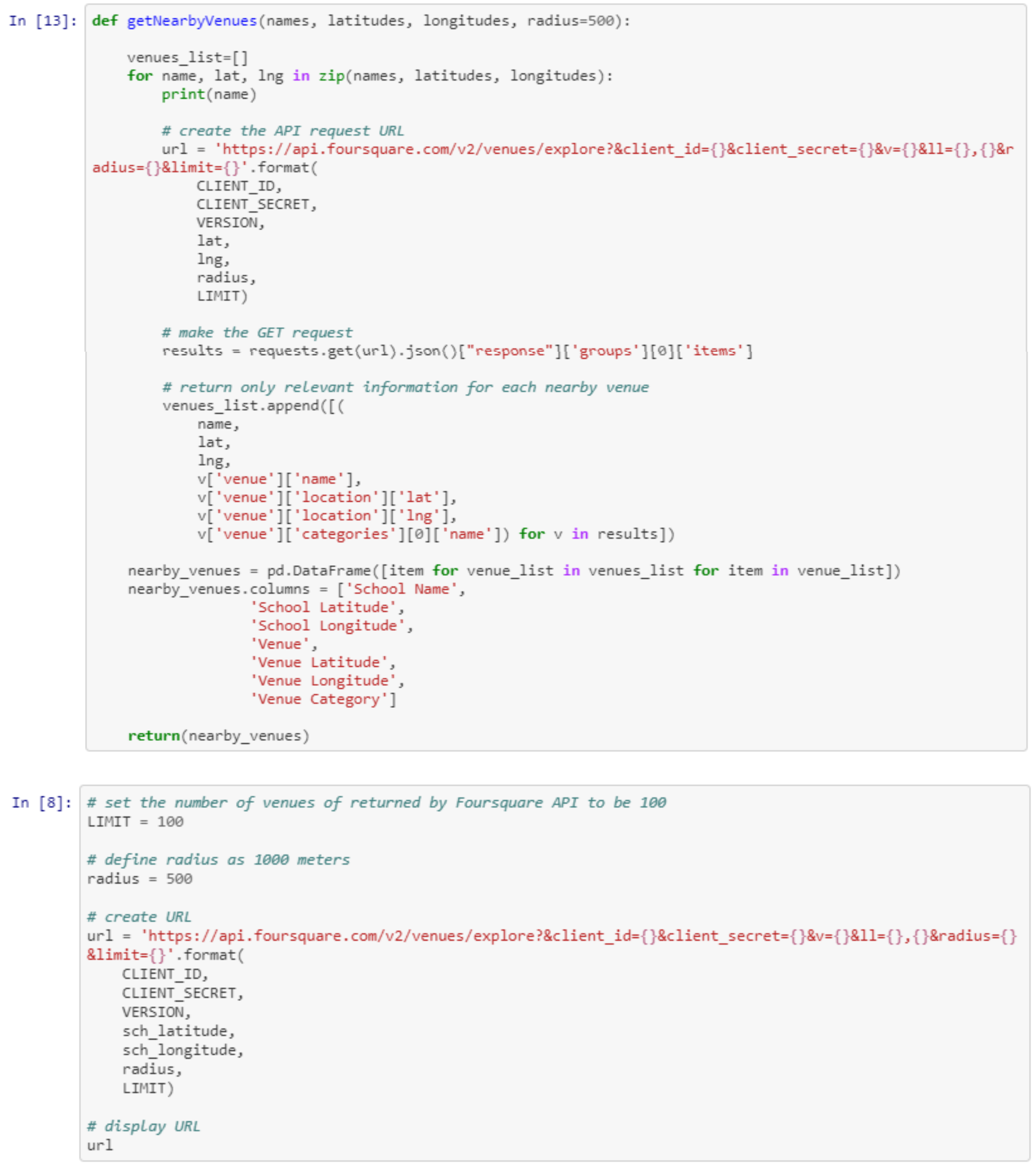
Singapore’s 27 Planning Areas with sizable number of Students.



Secondary Schools and Junior Colleges (or equivalent) in Singapore

# Venues with Foursquare

## 1. The Code



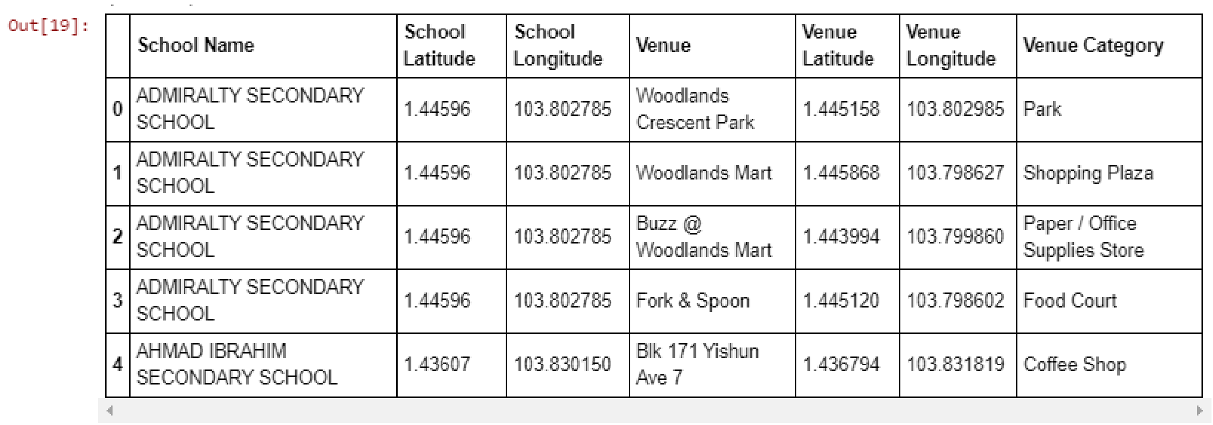
```
In [13]: def getNearbyVenues(names, latitudes, longitudes, radius=500):

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

                 # create the API request URL
                 url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&r
         adius={}&limit={}'.format(
                     CLIENT_ID,
                     CLIENT_SECRET,
                     VERSION,
                     lat,
                     lng,
                     radius,
                     LIMIT)

                 # 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 = ['School Name',
                           'School Latitude',
                           'School Longitude',
                           'Venue',
                           'Venue Latitude',
                           'Venue Longitude',
                           'Venue Category']

             return(nearby_venues)
```

```
In [8]: # set the number of venues of returned by Foursquare API to be 100
        LIMIT = 100

        # define radius as 1000 meters
        radius = 500

        # create URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}
        &limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            sch_latitude,
            sch_longitude,
            radius,
            LIMIT)

        # display URL
        url
```

Foursquare is an API that allows developers and users to explore venues near certain locations.

Here I searched for venues near each school in the dataset.

I set the radius to be 500 meters and the limit of venues returned to be 100.

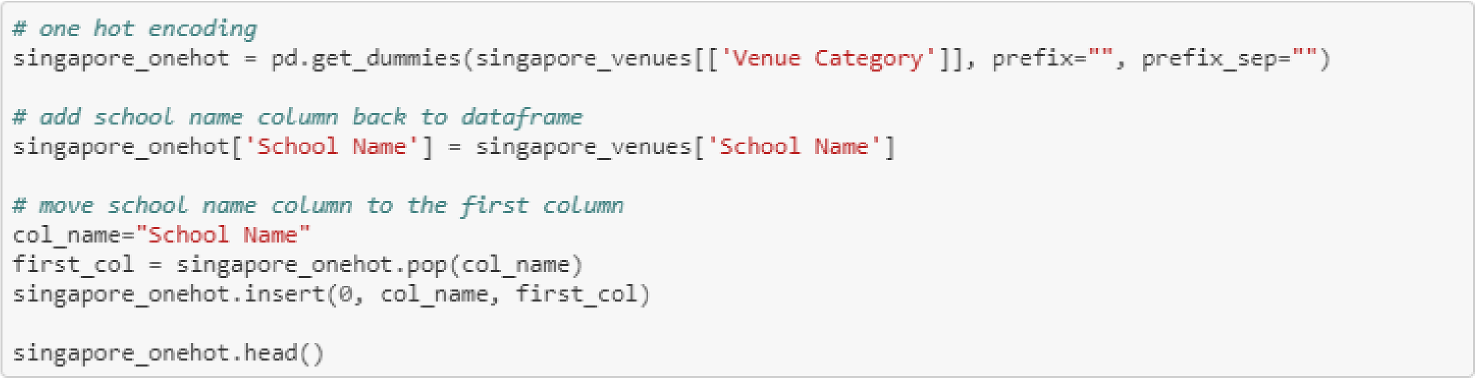


Out[19]:

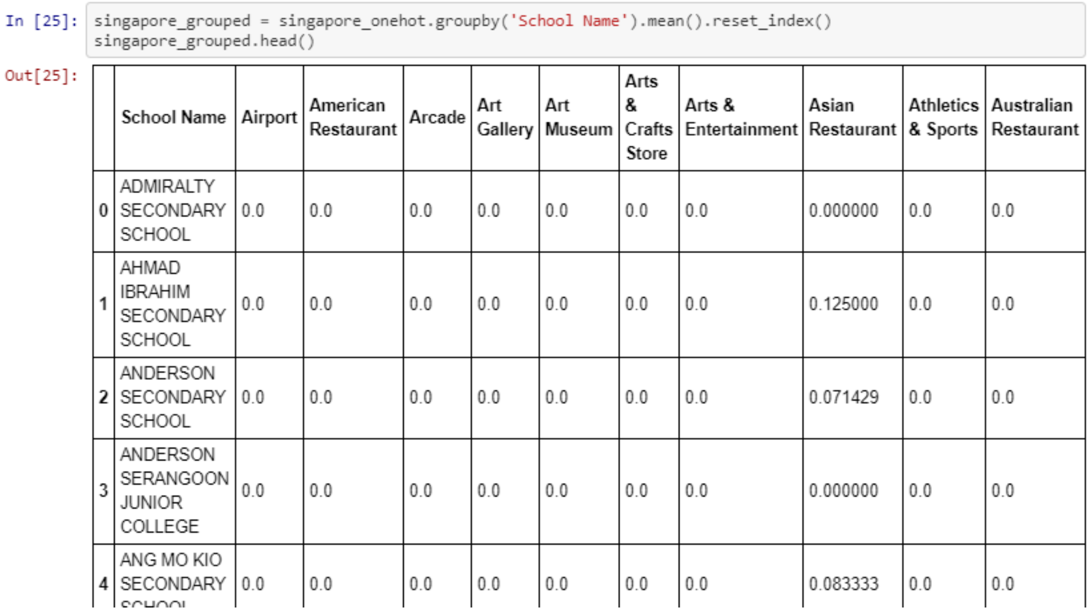
| | School Name | School Latitude | School Longitude | Venue | Venue Latitude | Venue Longitude | Venue Category |
|---|---|---|---|---|---|---|---|
| 0 | ADMIRALTY SECONDARY SCHOOL | 1.44596 | 103.802785 | Woodlands Crescent Park | 1.445158 | 103.802985 | Park |
| 1 | ADMIRALTY SECONDARY SCHOOL | 1.44596 | 103.802785 | Woodlands Mart | 1.445868 | 103.798627 | Shopping Plaza |
| 2 | ADMIRALTY SECONDARY SCHOOL | 1.44596 | 103.802785 | Buzz @ Woodlands Mart | 1.443994 | 103.799860 | Paper / Office Supplies Store |
| 3 | ADMIRALTY SECONDARY SCHOOL | 1.44596 | 103.802785 | Fork & Spoon | 1.445120 | 103.798602 | Food Court |
| 4 | AHMAD IBRAHIM SECONDARY SCHOOL | 1.43607 | 103.830150 | Blk 171 Yishun Ave 7 | 1.436794 | 103.831819 | Coffee Shop |

The Foursquare API returned in total 2535 venues for 162 Schools

## One hot encoding



```
# one hot encoding
singapore_onehot = pd.get_dummies(singapore_venues[['Venue Category']], prefix="", prefix_sep="")

# add school name column back to dataframe
singapore_onehot['School Name'] = singapore_venues['School Name']

# move school name column to the first column
col_name="School Name"
first_col = singapore_onehot.pop(col_name)
singapore_onehot.insert(0, col_name, first_col)

singapore_onehot.head()
```

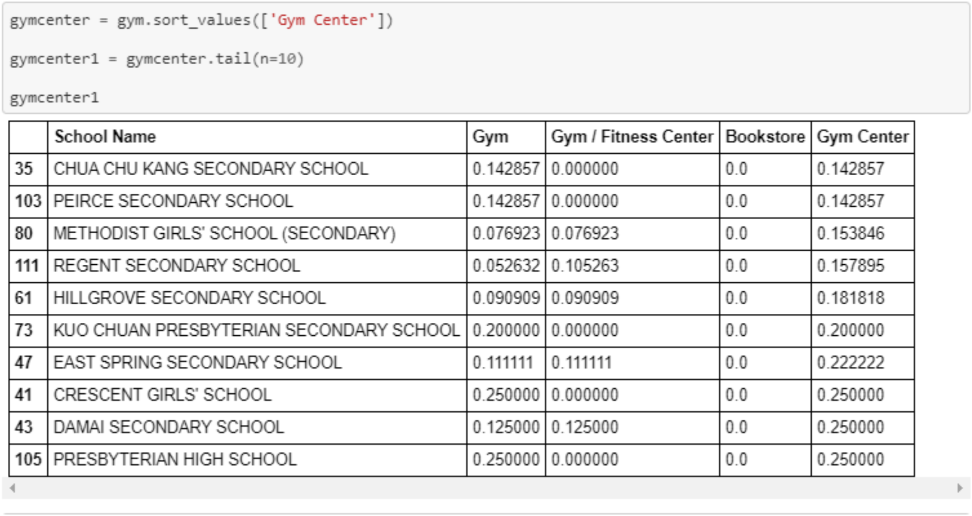


```
In [25]: singapore_grouped = singapore_onehot.groupby('School Name').mean().reset_index()
         singapore_grouped.head()
```

Out[25]:

| | School Name | Airport | American Restaurant | Arcade | Art Gallery | Art Museum | Arts & Crafts Store | Arts & Entertainment | Asian Restaurant | Athletics & Sports | Australian Restaurant |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ADMIRALTY SECONDARY SCHOOL | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 |
| 1 | AHMAD IBRAHIM SECONDARY SCHOOL | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.125000 | 0.0 | 0.0 |
| 2 | ANDERSON SECONDARY SCHOOL | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.071429 | 0.0 | 0.0 |
| 3 | ANDERSON SERANGOON JUNIOR COLLEGE | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 |
| 4 | ANG MO KIO SECONDARY SCHOOL | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.083333 | 0.0 | 0.0 |

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. For the K-means Clustering Algorithm, all unique items under Venue Category are one-hot encoded.



```
gymcenter = gym.sort_values(['Gym Center'])

gymcenter1 = gymcenter.tail(n=10)

gymcenter1
```

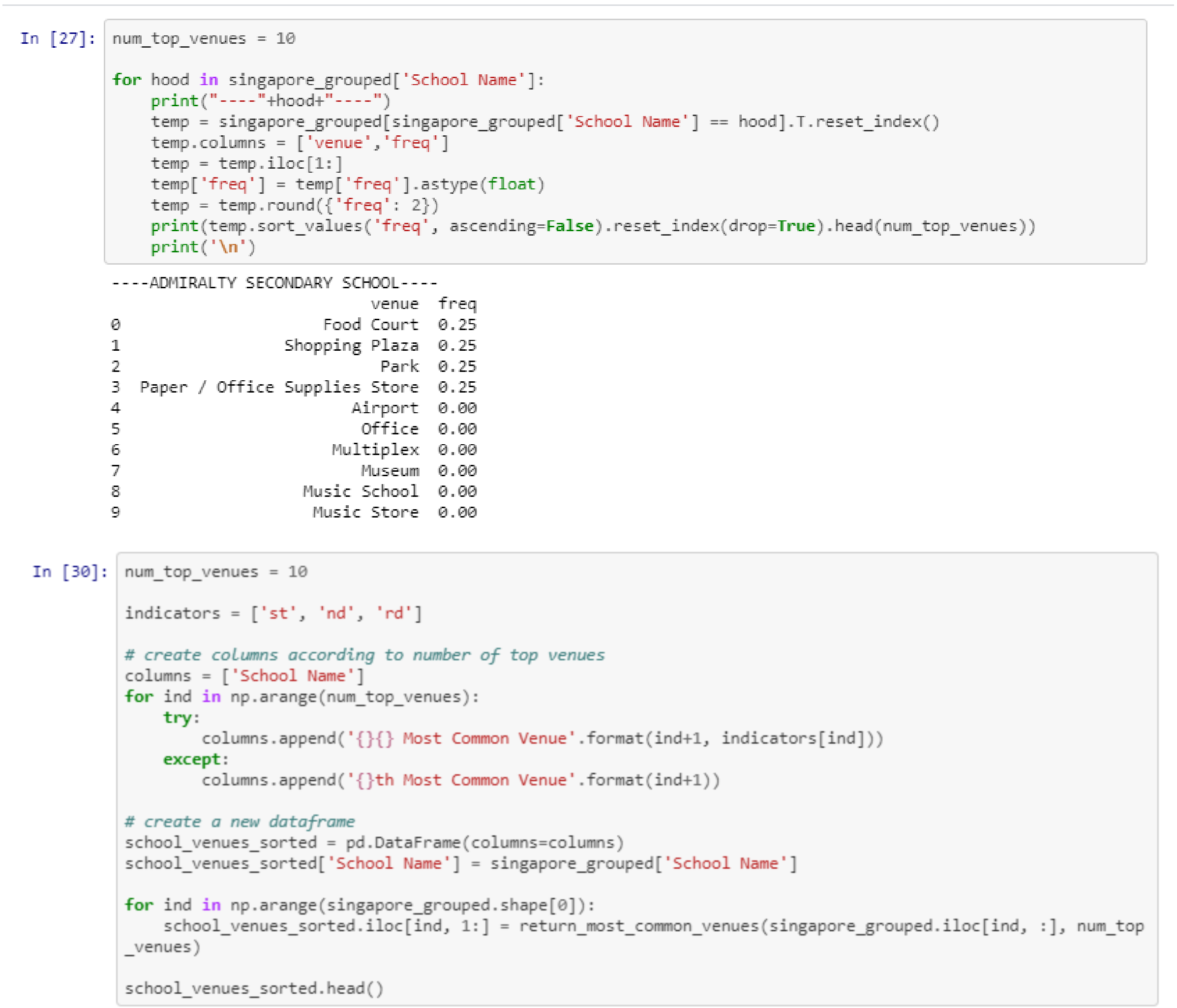
| | School Name | Gym | Gym / Fitness Center | Bookstore | Gym Center |
|---|---|---|---|---|---|
| 35 | CHUA CHU KANG SECONDARY SCHOOL | 0.142857 | 0.000000 | 0.0 | 0.142857 |
| 103 | PEIRCE SECONDARY SCHOOL | 0.142857 | 0.000000 | 0.0 | 0.142857 |
| 80 | METHODIST GIRLS' SCHOOL (SECONDARY) | 0.076923 | 0.076923 | 0.0 | 0.153846 |
| 111 | REGENT SECONDARY SCHOOL | 0.052632 | 0.105263 | 0.0 | 0.157895 |
| 61 | HILLGROVE SECONDARY SCHOOL | 0.090909 | 0.090909 | 0.0 | 0.181818 |
| 73 | KUO CHUAN PRESBYTERIAN SECONDARY SCHOOL | 0.200000 | 0.000000 | 0.0 | 0.200000 |
| 47 | EAST SPRING SECONDARY SCHOOL | 0.111111 | 0.111111 | 0.0 | 0.222222 |
| 41 | CRESCENT GIRLS' SCHOOL | 0.250000 | 0.000000 | 0.0 | 0.250000 |
| 43 | DAMAI SECONDARY SCHOOL | 0.125000 | 0.125000 | 0.0 | 0.250000 |
| 105 | PRESBYTERIAN HIGH SCHOOL | 0.250000 | 0.000000 | 0.0 | 0.250000 |

By using the table generated from One Hot Encoding, I sorted out the Schools in which the frequency of numbers for gym/fitness center, and for book stores are the highest 10.

By removing these schools from the list of 59 Schools, I get a new list of 45 Schools. These would my suggestions to my friend.

# Top Ten Venues

## 1. The Code

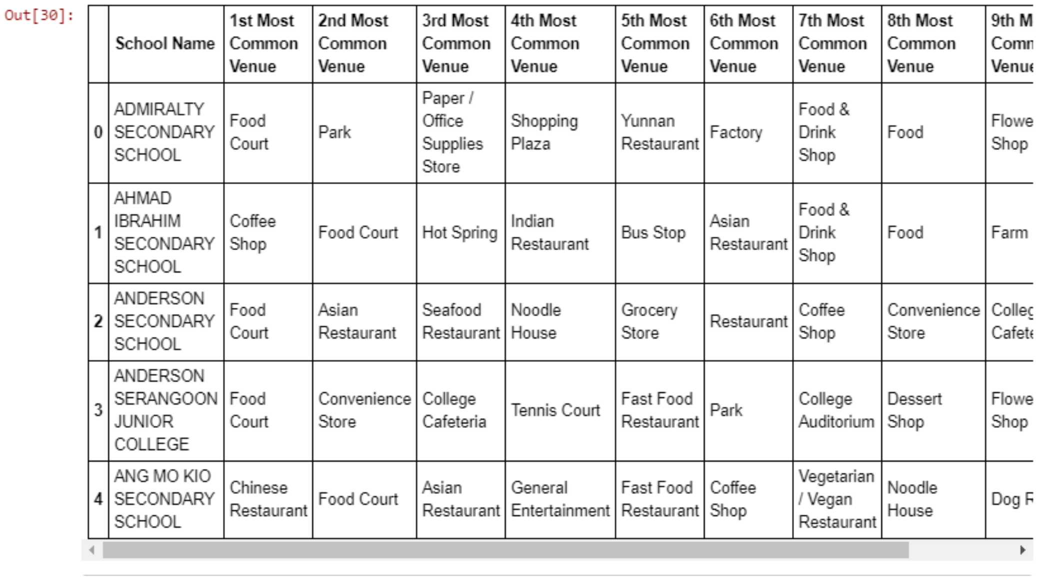


In [27]:
```
num_top_venues = 10

for hood in singapore_grouped['School Name']:
    print("----"+hood+"----")
    temp = singapore_grouped[singapore_grouped['School Name'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
    print('\n')
```

```
----ADMIRALTY SECONDARY SCHOOL----
                        venue  freq
0                  Food Court  0.25
1              Shopping Plaza  0.25
2                        Park  0.25
3  Paper / Office Supplies Store  0.25
4                     Airport  0.00
5                      Office  0.00
6                  Multiplex  0.00
7                      Museum  0.00
8                Music School  0.00
9                 Music Store  0.00
```

In [30]:
```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['School Name']
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
school_venues_sorted = pd.DataFrame(columns=columns)
school_venues_sorted['School Name'] = singapore_grouped['School Name']

for ind in np.arange(singapore_grouped.shape[0]):
    school_venues_sorted.iloc[ind, 1:] = return_most_common_venues(singapore_grouped.iloc[ind, :], num_top
_venues)

school_venues_sorted.head()
```

## 2. The Output



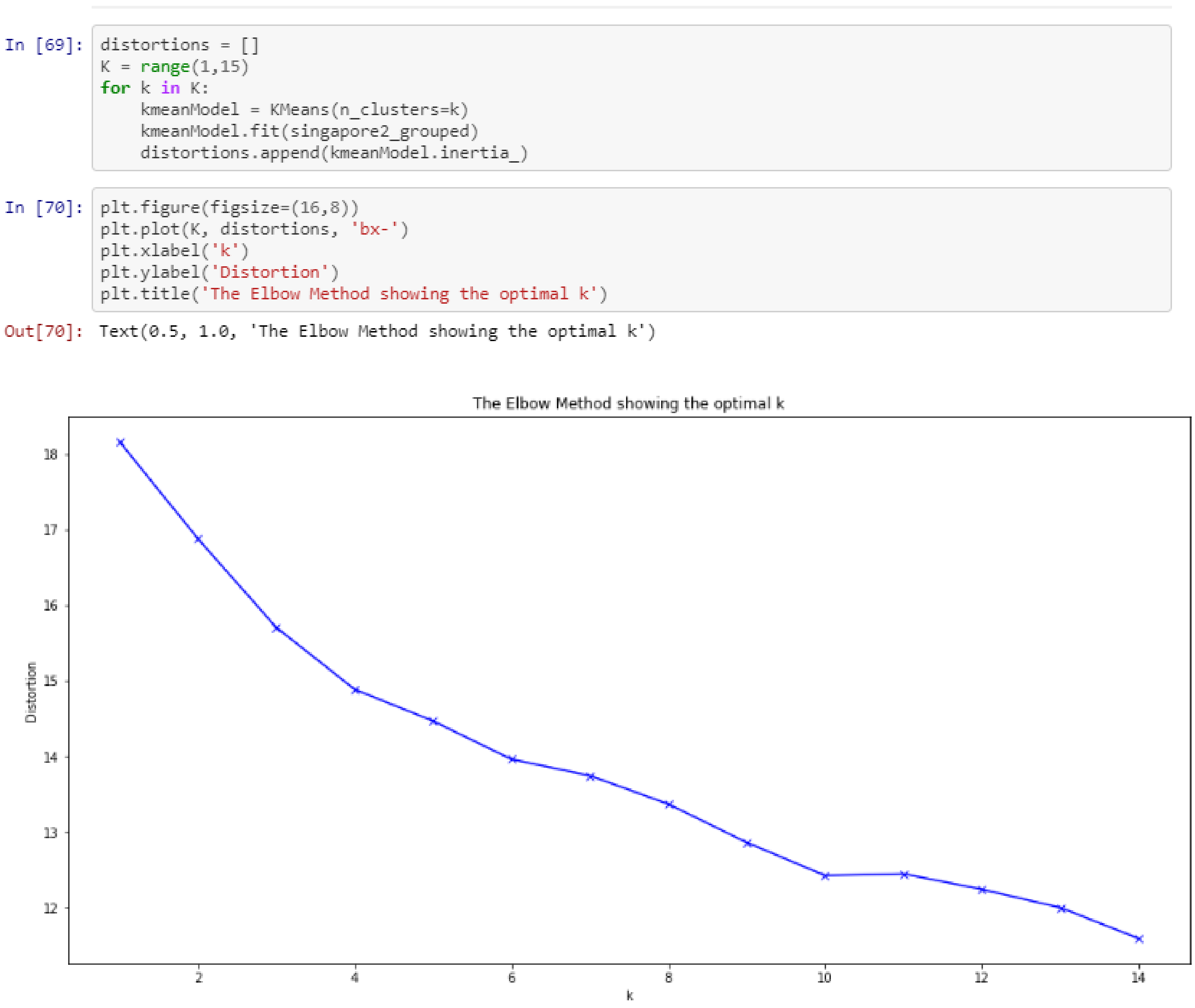
Out[30]:

| | School Name | 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 M Comn Venu |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ADMIRALTY SECONDARY SCHOOL | Food Court | Park | Paper / Office Supplies Store | Shopping Plaza | Yunnan Restaurant | Factory | Food & Drink Shop | Food | Flowe Shop |
| 1 | AHMAD IBRAHIM SECONDARY SCHOOL | Coffee Shop | Food Court | Hot Spring | Indian Restaurant | Bus Stop | Asian Restaurant | Food & Drink Shop | Food | Farm |
| 2 | ANDERSON SECONDARY SCHOOL | Food Court | Asian Restaurant | Seafood Restaurant | Noodle House | Grocery Store | Restaurant | Coffee Shop | Convenience Store | Colleg Cafete |
| 3 | ANDERSON SERANGOON JUNIOR COLLEGE | Food Court | Convenience Store | College Cafeteria | Tennis Court | Fast Food Restaurant | Park | College Auditorium | Dessert Shop | Flowe Shop |
| 4 | ANG MO KIO SECONDARY SCHOOL | Chinese Restaurant | Food Court | Asian Restaurant | General Entertainment | Fast Food Restaurant | Coffee Shop | Vegetarian / Vegan Restaurant | Noodle House | Dog F |

Due to high variety in the venues, only the top 10 common venues are selected and a new DataFrame is made, which is used to train the K-means Clustering Algorithm.

## K-Means Clustering

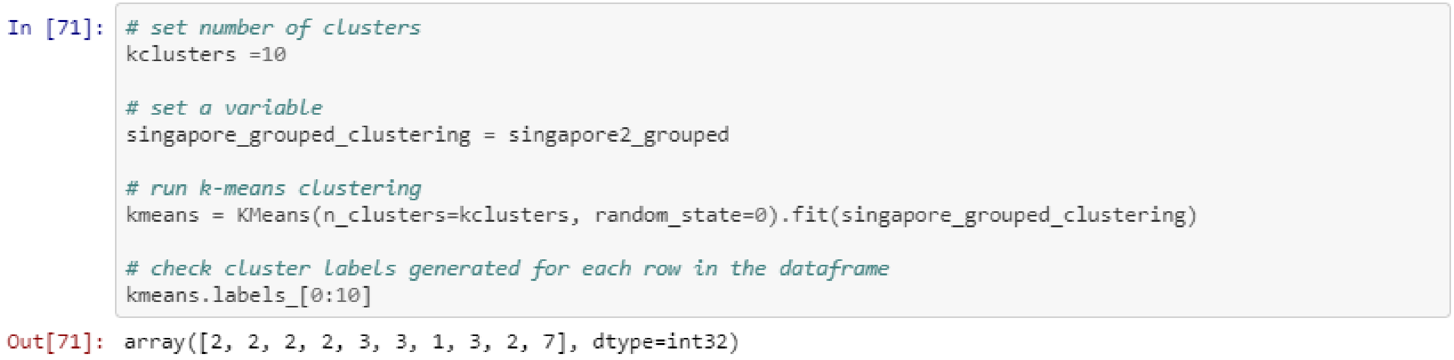
### 1. The Elbow Method – Opitimal K Value



```
In [69]: distortions = []
         K = range(1,15)
         for k in K:
             kmeanModel = KMeans(n_clusters=k)
             kmeanModel.fit(singapore2_grouped)
             distortions.append(kmeanModel.inertia_)

In [70]: plt.figure(figsize=(16,8))
         plt.plot(K, distortions, 'bx-')
         plt.xlabel('k')
         plt.ylabel('Distortion')
         plt.title('The Elbow Method showing the optimal k')

Out[70]: Text(0.5, 1.0, 'The Elbow Method showing the optimal k')
```

The Elbow is at k =10

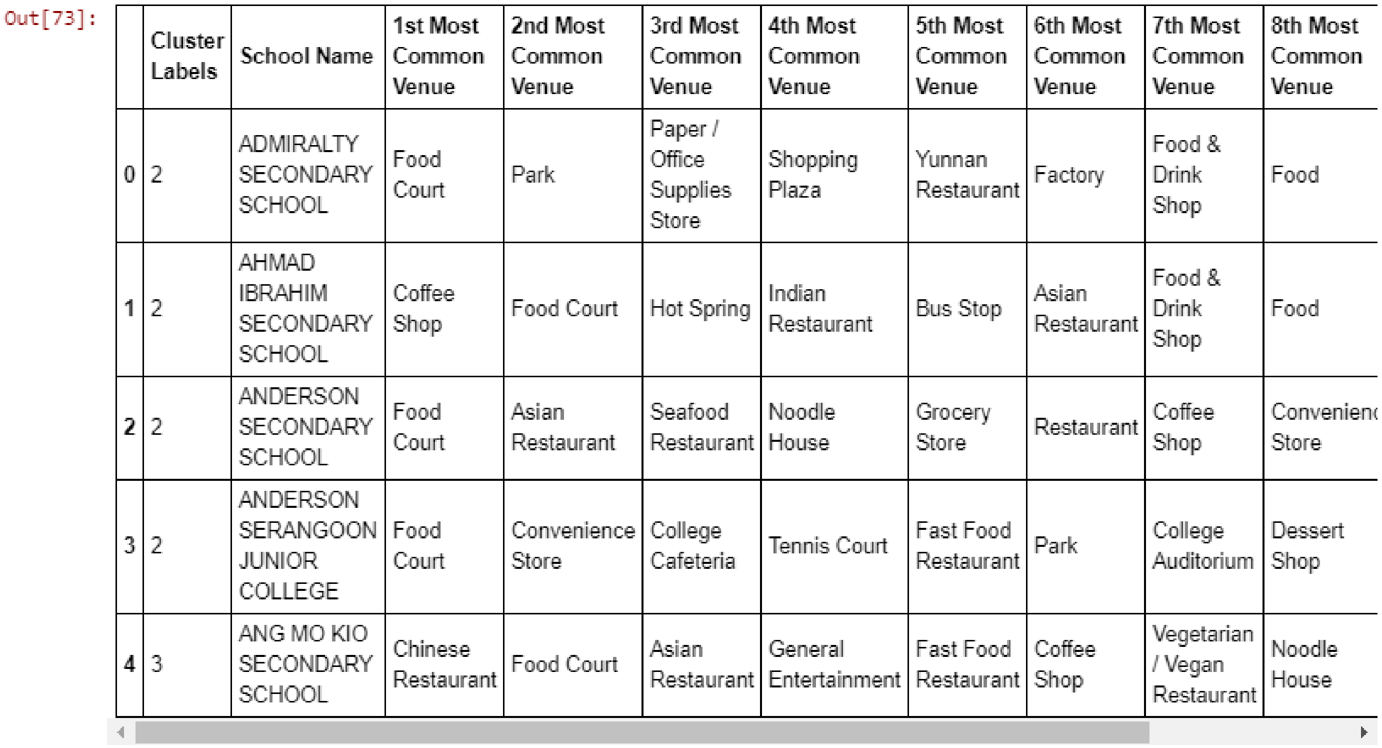
### 2. Clustering



```
In [71]: # set number of clusters
         kclusters =10

         # set a variable
         singapore_grouped_clustering = singapore2_grouped

         # run k-means clustering
         kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(singapore_grouped_clustering)

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

Out[71]: array([2, 2, 2, 2, 3, 3, 1, 3, 2, 7], dtype=int32)
```

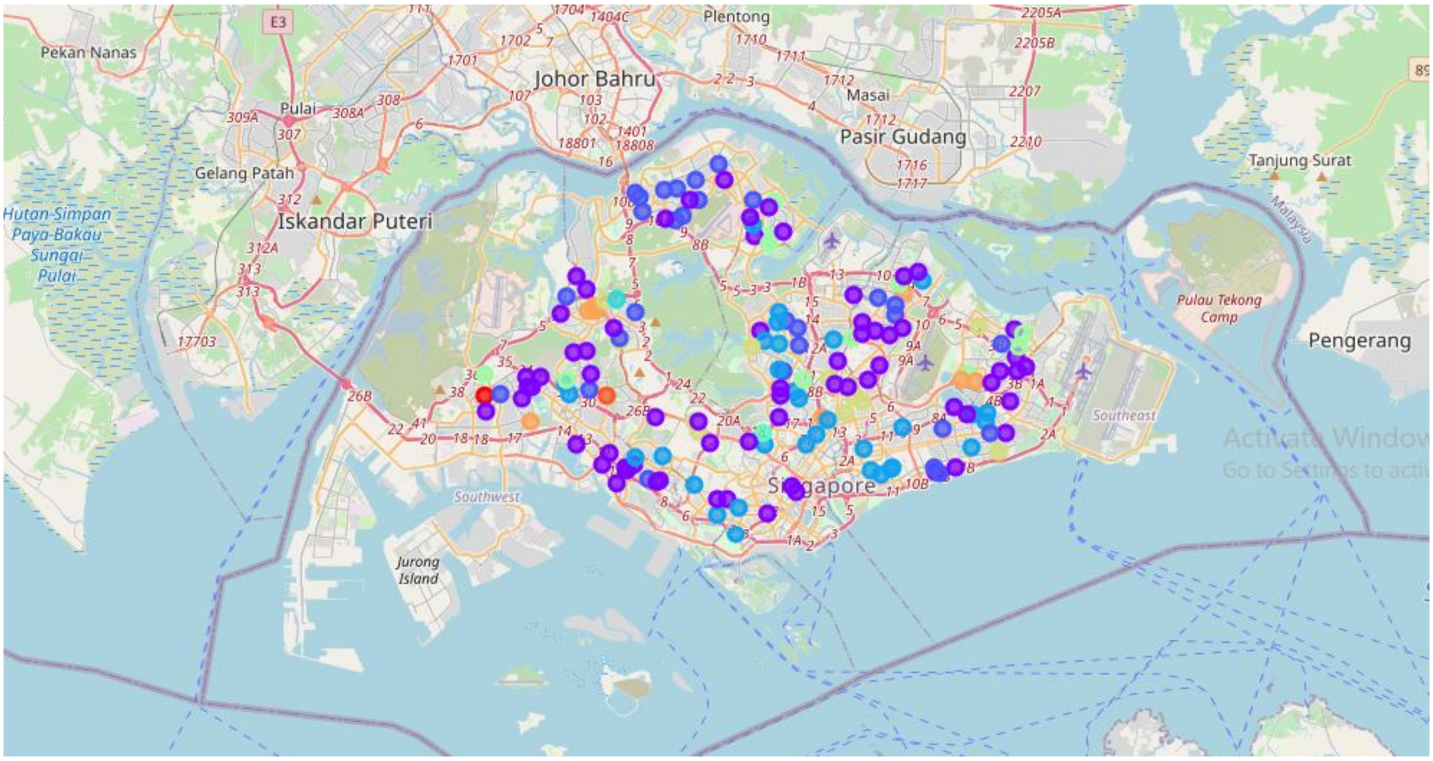
## K-Means Clustering

### 1. The Output



Out[73]:

| | Cluster Labels | School Name | 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | ADMIRALTY SECONDARY SCHOOL | Food Court | Park | Paper / Office Supplies Store | Shopping Plaza | Yunnan Restaurant | Factory | Food & Drink Shop | Food |
| 1 | 2 | AHMAD IBRAHIM SECONDARY SCHOOL | Coffee Shop | Food Court | Hot Spring | Indian Restaurant | Bus Stop | Asian Restaurant | Food & Drink Shop | Food |
| 2 | 2 | ANDERSON SECONDARY SCHOOL | Food Court | Asian Restaurant | Seafood Restaurant | Noodle House | Grocery Store | Restaurant | Coffee Shop | Convenienc Store |
| 3 | 2 | ANDERSON SERANGOON JUNIOR COLLEGE | Food Court | Convenience Store | College Cafeteria | Tennis Court | Fast Food Restaurant | Park | College Auditorium | Dessert Shop |
| 4 | 3 | ANG MO KIO SECONDARY SCHOOL | Chinese Restaurant | Food Court | Asian Restaurant | General Entertainment | Fast Food Restaurant | Coffee Shop | Vegetarian / Vegan Restaurant | Noodle House |



# Conclusion

## 1. The Suggestion

According to the research, I would say my friend has come up with a quite interesting idea by combing study room/tuition with gyms

The one hot encoding shows that the neighborhoods of schools with the highest frequency of gyms would have low frequency of bookstores, which implies that a study gym would most likely to fill in an empty space.

So I would suggest my friend to open this Study Gym near the 45 schools.

However, my research is still not so refined yet; new insights would be generated with further research.

## 2. Dicussion and Improvement

The methodology could be further improved in the following ways.

1) The listed 59 schools should have been used for venues and one hot encoding
Since I was only looking at the suitability of the 59 schools after step 1, I could have just used the data for these schools instead of the whole list of 162 schools, this would essentially reduce the time needed and make the work more efficient.

2) The frequency of tuition centers should be used instead of bookstores
The most relevant venue categories would be tuition centers, however, Foursquare API and one hot encoding did not provide that category, bookstores was used instead

3) Density of population in Subzones should be used
Planning Area are further divided into subzones, which is more accurate due to its smaller size.