MELBOURNE CITY OPEN DATA PLAYGROUND

CLUE Cafe, restaurant, bistro seats

Exploratory Data Analysis

Date	Author/Contributor	Change
18-Nov-2021	Steven Tuften	Initial Draft

ATTRIBUTIONS

Jupyter Notebook derivative of data exploration notebook and d2i_tools.py created by Albert Hon in T2 2021.

Package/Library Imports

In [1]:
```
import os
import time
from urllib.request import urlopen
import json
from datetime import datetime
import numpy as np
import pandas as pd
from sodapy import Socrata
import geopandas
import plotly.express as px
from shapely.geometry import Polygon, Point
from d2i_tools import *
import warnings
warnings.simplefilter("ignore")
```

Constants

In [2]:
```
dataset_id = 'dyqx-cfn5' # Melbourne CLUE Cafe, restaurant, bistro seats
geoJSON_Id = 'aia8-ryiq' # Melbourne CLUE Block polygons in GeoJSON format

apptoken = os.environ.get("SODAPY_APPTOKEN") # Anonymous app token
domain = "data.melbourne.vic.gov.au"
client = Socrata(domain, apptoken) # Open Dataset connection
```

```
WARNING:root:Requests made without an app_token will be subject to strict throttling limits.
```

[01] Retrieve dataset Metadata

In [3]:
```
metadata_df = loadClientDatasetsMetadata(client)
print('Selected metadata for the dataset of interest')
metadata_df[metadata_df.id.isin([dataset_id])].T
```

```
Selected metadata for the dataset of interest
```

Out[3]:

	60
name	Cafe, restaurant, bistro seats 2020
id	dyqx-cfn5
parent_fxf	[xt2y-tnn9]
description	Data collected as part of the City of Melbourn...
data_upd_at	2021-11-02T22:16:33.000Z
pv_last_wk	18
pv_last_mth	98
pv_total	3357
download_count	649
categories	[politics, economy, housing & development]
domain_category	Business
domain_tags	[beverage, business, census of land use and em...
domain_metadata	[{'key': 'Quality_Known-Issues', 'value': 'Non...
Quality_What's-included	Full dataset has been included
Quality_Update-frequency	Every two years
Quality_Reliability-level	Reliable and timely
How-to-use_Linked-to	NaN
Data-management_Source-data-update-frequency	Every two years
Quality_Known-Issues	None
How-to-use_Further-information	NaN
Quality_Data-quality-statement	A team of up to 6 surveyors conducts a field s...

[02] Display first few rows

In [4]:
```
dataresource = client.get_all(dataset_id)
dataset = pd.DataFrame.from_dict(dataresource)
print(f'The shape of dataset is {dataset.shape}.')
print('Below are the first 3 rows of this dataset:')
dataset.head(3).T
```

```
The shape of dataset is (3236, 13).
Below are the first 3 rows of this dataset:
```

Out[4]:

	0	1	2
census_year	2020	2020	2020
block_id	1	1	1
property_id	611394	611394	611394
base_property_id	611394	611394	611394
street_address	545-557 Flinders Street MELBOURNE VIC 3000	545-557 Flinders Street MELBOURNE VIC 3000	545-557 Flinders Street MELBOURNE VIC 3000
clue_small_area	Melbourne (CBD)	Melbourne (CBD)	Melbourne (CBD)
trading_name	551 Flinders Street MELBOURNE VIC 3000	551 Flinders Street MELBOURNE VIC 3000	553 Flinders Street MELBOURNE VIC 3000
industry_anzsic4_code	4511	4511	4512
industry_anzsic4_description	Cafes and Restaurants	Cafes and Restaurants	Takeaway Food Services
seating_type	Seats - Indoor	Seats - Outdoor	Seats - Indoor
number_of_seats	60	6	12
x_coordinate	144.9565145	144.9565145	144.9565145
y_coordinate	-37.82097941	-37.82097941	-37.82097941

[03] Data Pre-processing

Cast Data types before analysis

In [6]:
```
integer_columns = ['census_year', 'block_id', 'property_id', 'base_property_id', 'industry_anzsic4_code', 'number_of_seats']
fp_columns = ['x_coordinate', 'y_coordinate']

dataset[integer_columns] = dataset[integer_columns].astype(int)
dataset[fp_columns] = dataset[fp_columns].astype(float)
dataset = dataset.convert_dtypes() # convert remaining to string
dataset.dtypes
```

Out[6]:
```
census_year                    Int32
block_id                       Int32
property_id                    Int32
base_property_id               Int32
street_address                string
clue_small_area               string
trading_name                  string
industry_anzsic4_code          Int32
industry_anzsic4_description  string
seating_type                  string
number_of_seats                Int32
x_coordinate                 float64
y_coordinate                 float64
dtype: object
```

Are there any missing values?

In [7]:
```
print(dataset.isnull().sum())
```

```
census_year                   0
block_id                      0
property_id                   0
base_property_id              0
street_address                0
clue_small_area               0
trading_name                  0
industry_anzsic4_code         0
industry_anzsic4_description  0
seating_type                  0
number_of_seats               0
x_coordinate                  0
y_coordinate                  0
dtype: int64
```

In [8]:
```
dataset[dataset['x_coordinate'].isnull()]
```

Out[8]:

census_year	block_id	property_id	base_property_id	street_address	clue_small_area	trading_name	industry_anzsic4_code	industry_anzsic4_description	seating_type	number_of_seats	x_coordinate	y_coordinate

Drop rows with no latitude or longitude?

We will not be using the latitude and longitude at property level so we can leave these two rows in the dataset.

In [9]:
```
## If we wanted to drop these rows we would use the following two commands.

#dataset = dataset.dropna(axis=0)
#print(dataset.isnull().sum())
```

[04] Analyse data in Aggregate

Count of Number of Seats by CLUE small area

In [11]:
```
groupbyfields = ['clue_small_area']
aggregatebyfields = {'number_of_seats': ["sum"]}

maxByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
maxByBlock.head(10)
```

Out[11]:

	clue_small_area	number_of_seats
		sum
0	Carlton	15177
1	Docklands	21585
2	East Melbourne	7181
3	Kensington	5709
4	Melbourne (CBD)	88974
5	Melbourne (Remainder)	8767
6	North Melbourne	4499
7	Parkville	3695
8	Port Melbourne	1251
9	South Yarra	810

Count of Number of Seats by Seating Type

In [12]:
```
groupbyfields = ['seating_type']
aggregatebyfields = {'number_of_seats': ["sum"]}

maxByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
maxByBlock.head(10)

# barchart
```

Out[12]:

	seating_type	number_of_seats
		sum
0	Seats - Indoor	150405
1	Seats - Outdoor	31399

Count of Seats by CLUE small area and Seating Type

In [13]:
```
groupbyfields = ['clue_small_area','seating_type']
aggregatebyfields = {'number_of_seats': ["sum"]}

maxByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
maxByBlock.head(40)

# map
```

Out[13]:

	clue_small_area	seating_type	number_of_seats
			sum
0	Carlton	Seats - Indoor	11868
1	Carlton	Seats - Outdoor	3309
2	Docklands	Seats - Indoor	17094
3	Docklands	Seats - Outdoor	4491
4	East Melbourne	Seats - Indoor	6362
5	East Melbourne	Seats - Outdoor	819
6	Kensington	Seats - Indoor	5152
7	Kensington	Seats - Outdoor	557
8	Melbourne (CBD)	Seats - Indoor	75528
9	Melbourne (CBD)	Seats - Outdoor	13446
10	Melbourne (Remainder)	Seats - Indoor	7309
11	Melbourne (Remainder)	Seats - Outdoor	1458
12	North Melbourne	Seats - Indoor	3821
13	North Melbourne	Seats - Outdoor	678
14	Parkville	Seats - Indoor	2447
15	Parkville	Seats - Outdoor	1248
16	Port Melbourne	Seats - Indoor	821
17	Port Melbourne	Seats - Outdoor	430
18	South Yarra	Seats - Indoor	670
19	South Yarra	Seats - Outdoor	140
20	Southbank	Seats - Indoor	16879
21	Southbank	Seats - Outdoor	4336
22	West Melbourne (Industrial)	Seats - Indoor	60
23	West Melbourne (Industrial)	Seats - Outdoor	8
24	West Melbourne (Residential)	Seats - Indoor	2394
25	West Melbourne (Residential)	Seats - Outdoor	479

Count of Seats by Block Id

In [15]:
```
groupbyfields = ['block_id']
aggregatebyfields = {'number_of_seats': ["sum"]}

maxByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
maxByBlock.head(10)
```

Out[15]:

	block_id	number_of_seats
		sum
0	1	150
1	2	198
2	4	505
3	6	2743
4	11	1050
5	12	482
6	13	671
7	14	1093
8	15	1719
9	16	495

Count, Min, Max, Sum of Seats by CLUE small area, Block Id and Seating Type

In [16]:
```
groupbyfields = ['clue_small_area','block_id','seating_type']
aggregatebyfields = {'number_of_seats': ["count","min","max","sum"]}

maxByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
maxByBlock.head(10)
```

Out[16]:

	clue_small_area	block_id	seating_type	number_of_seats			
				count	min	max	sum
0	Carlton	203	Seats - Indoor	2	42	45	87
1	Carlton	203	Seats - Outdoor	1	6	6	6
2	Carlton	204	Seats - Indoor	2	50	70	120
3	Carlton	204	Seats - Outdoor	1	50	50	50
4	Carlton	205	Seats - Indoor	4	17	60	149
5	Carlton	205	Seats - Outdoor	3	8	36	68
6	Carlton	206	Seats - Indoor	5	20	120	335
7	Carlton	206	Seats - Outdoor	5	16	56	191
8	Carlton	207	Seats - Indoor	8	35	200	709
9	Carlton	207	Seats - Outdoor	7	6	29	143

Plot Seats by Location on map

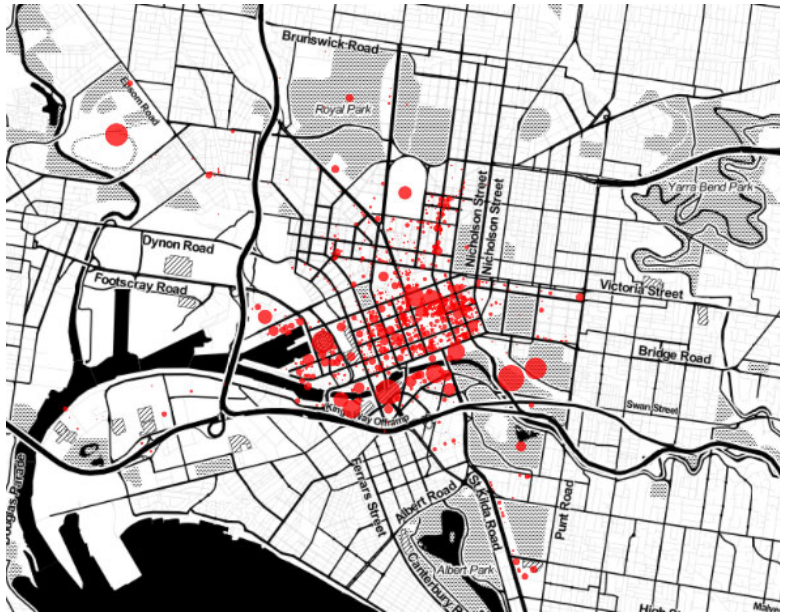
In [29]:
```
groupbyfields = ['clue_small_area','block_id','y_coordinate','x_coordinate']
aggregatebyfields = {'number_of_seats': ["sum"]}

seatsByLocn = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
seatsByLocn.columns = seatsByLocn.columns.map(''.join) # flatten column header
seatsByLocn.rename(columns={'clue_small_area': 'clue_area'}, inplace=True) #rename to match GeoJSON extract
seatsByLocn.rename(columns={'number_of_seatssum': 'number_of_seats'}, inplace=True) #rename to match GeoJSON extract

seatsByLocn['number_of_seats'] = seatsByLocn['number_of_seats'].astype(int)
seatsByLocn.head(10)
```

Out[29]:

	clue_area	block_id	y_coordinate	x_coordinate	number_of_seats
0	Carlton	203	-37.796707	144.965534	51
1	Carlton	203	-37.796680	144.964900	42
2	Carlton	204	-37.797833	144.965174	50
3	Carlton	204	-37.797255	144.965754	120
4	Carlton	205	-37.799470	144.964893	96
5	Carlton	205	-37.799001	144.964765	80
6	Carlton	205	-37.798721	144.965257	41
7	Carlton	206	-37.800457	144.966558	51
8	Carlton	206	-37.800191	144.966716	140
9	Carlton	206	-37.800046	144.966741	115

In [43]:
```
fig = px.scatter_mapbox(seatsByLocn, lat="y_coordinate", lon="x_coordinate", size="number_of_seats",
                        mapbox_style="stamen-toner", #"carto-positron",
                        zoom=12.5,
                        center = {"lat": -37.813, "lon": 144.945},
                        opacity=0.75,
                        hover_name="clue_area",
                        hover_data={"block_id":True,"number_of_seats":True,"x_coordinate":False,"y_coordinate":False},
                        color_discrete_sequence=['red'],
                        labels={'number_of_seats':'Number of Seats','block_id':'CLUE Block Id'},
                        title='Venue Seats by Location for 2020',
                        width=950, height=800)
fig.show()
```

Venue Seats by Location for 2020



Plot Seating Density by Block

In [44]:
```
groupbyfields = ['block_id','clue_small_area']
aggregatebyfields = {'number_of_seats': ["sum"]}

seatingDensityByBlock = pd.DataFrame(dataset.groupby(groupbyfields, as_index=False).agg(aggregatebyfields))
seatingDensityByBlock.columns = seatingDensityByBlock.columns.map(''.join) # flatten column header
seatingDensityByBlock.rename(columns={'clue_small_area': 'clue_area'}, inplace=True) #rename to match GeoJSON extract
seatingDensityByBlock.rename(columns={'number_of_seatssum': 'number_of_seats'}, inplace=True) #rename to match GeoJSON extract
seatingDensityByBlock.head(10)
```

Out[44]:

	block_id	clue_area	number_of_seats
0	1	Melbourne (CBD)	150
1	2	Melbourne (CBD)	198
2	4	Melbourne (CBD)	505
3	6	Melbourne (CBD)	2743
4	11	Melbourne (CBD)	1050
5	12	Melbourne (CBD)	482
6	13	Melbourne (CBD)	671
7	14	Melbourne (CBD)	1093
8	15	Melbourne (CBD)	1719
9	16	Melbourne (CBD)	495

Get Block Polygon data in GeoJSON format

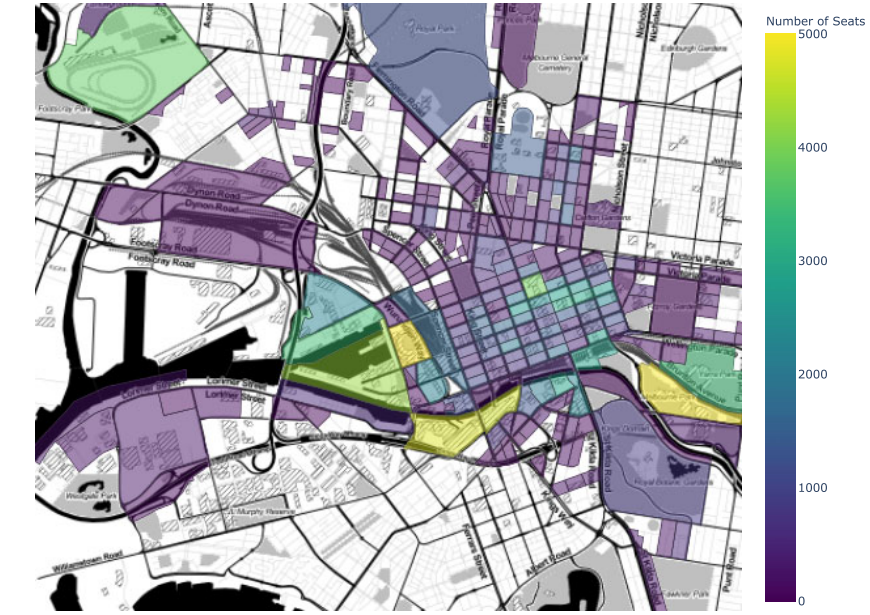
Load the CLUE Blocks in GeoJSON format and verify the location keys.

In [45]:
```
GeoJSONURL = 'https://'+domain+'/api/geospatial/'+geoJSON_Id+'?method=export&format=GeoJSON'
with urlopen(GeoJSONURL) as response:
    block = json.load(response)

block["features"][0].keys()
block["features"][0]['properties'].keys()
```

Out[45]: dict_keys(['block_id', 'clue_area'])

Illustrate Residential Dwelling Density using a Chloropleth Map using Block regions defined by the GeoJSON data

In [50]:
```
range_max = seatingDensityByBlock['number_of_seats'].max()

fig = px.choropleth_mapbox(seatingDensityByBlock, geojson=block, locations='block_id', color='number_of_seats',
                           color_continuous_scale= "Viridis",
                           range_color=(0, range_max),
                           featureidkey="properties.block_id",
                           mapbox_style="stamen-toner", #"carto-positron",
                           zoom=12.5,
                           center = {"lat": -37.813, "lon": 144.945},
                           opacity=0.5,
                           hover_name='clue_area',
                           hover_data={'block_id':True,'number_of_seats':True},
                           labels={'number_of_seats':'Number of Seats','block_id':'CLUE Block Id'},
                           title='Seating Density by CLUE Block Id for 2020',
                           width=950, height=800
                          )
fig.show()
```



In [ ]:
```

```