MELBOURNE CITY OPEN DATA PLAYGROUND

CLUE Residential Dwellings

Exploratory Data Analysis

Date	Author/Contributor	Change
16-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 = 'rm92-h5tq' # Melbourne CLUE Residential Dwellings
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]:

	84
name	Residential dwellings 2020
id	rm92-h5tq
parent_fxf	[44kh-ty54]
description	Data collected as part of the City of Melbourn...
data_upd_at	2021-11-02T22:25:14.000Z
pv_last_wk	9
pv_last_mth	37
pv_total	2189
download_count	563
categories	[environment, finance, housing & development]
domain_category	Property
domain_tags	[census of land use and employment, clue, clue...
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	Data is summarised from the City of Melbourne'...

[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 (10402, 10).
Below are the first 3 rows of this dataset:
```

Out[4]:

	0	1	2
census_year	2020	2020	2020
block_id	1	1	11
pbs_property_id	611394	611395	103957
bps_base_id	611394	611395	103957
street_name	545-557 Flinders Street MELBOURNE VIC 3000	561-581 Flinders Street MELBOURNE VIC 3000	517-537 Flinders Lane MELBOURNE VIC 3000
clue_small_area	Melbourne (CBD)	Melbourne (CBD)	Melbourne (CBD)
dwelling_type	Residential Apartments	Residential Apartments	Residential Apartments
dwelling_number	196	189	26
x_coordinate	144.9565145	144.9559094	144.9566569
y_coordinate	-37.82097941	-37.82108687	-37.81987147

[03] Data Pre-processing

Cast Data types before analysis

In [5]:
```
dataset[['census_year', 'dwelling_number']] = dataset[['census_year', 'dwelling_number']].astype(int)
dataset[['x_coordinate', 'y_coordinate']] = dataset[['x_coordinate', 'y_coordinate']].astype(float)
dataset = dataset.convert_dtypes() # convert remaining to string
dataset.dtypes
```

Out[5]:
```
census_year          Int32
block_id            string
pbs_property_id     string
bps_base_id         string
street_name         string
clue_small_area     string
dwelling_type       string
dwelling_number      Int32
x_coordinate       float64
y_coordinate       float64
dtype: object
```

Are there any missing values?

In [6]:
```
print(dataset.isnull().sum())
```

```
census_year        0
block_id           0
pbs_property_id    0
bps_base_id        0
street_name        0
clue_small_area    0
dwelling_type      0
dwelling_number    0
x_coordinate       2
y_coordinate       2
dtype: int64
```

In [7]:
```
dataset[dataset['x_coordinate'].isnull()]
```

Out[7]:

	census_year	block_id	pbs_property_id	bps_base_id	street_name	clue_small_area	dwelling_type	dwelling_number	x_coordinate	y_coordinate
4499	2020	432	506137	506137	20 Chetwynd Street WEST MELBOURNE VIC 3003	West Melbourne (Residential)	House/Townhouse	1	NaN	NaN
4500	2020	432	506138	506138	18 Chetwynd Street WEST MELBOURNE VIC 3003	West Melbourne (Residential)	House/Townhouse	1	NaN	NaN

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 [8]:
```
## 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 Dwellings by CLUE small area

In [9]:
```
groupbyfields = ['clue_small_area']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

# barchart

# map
```

Out[9]:

	clue_small_area	dwelling_number
		sum
0	Carlton	12802
1	Docklands	9730
2	East Melbourne	3426
3	Kensington	5476
4	Melbourne (CBD)	28519
5	Melbourne (Remainder)	1577
6	North Melbourne	8946
7	Parkville	2851
8	Port Melbourne	2
9	South Yarra	2752

Count of Dwellings by Dwelling Type

In [10]:
```
groupbyfields = ['dwelling_type']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

# barchart
```

Out[10]:

	dwelling_type	dwelling_number
		sum
0	House/Townhouse	10026
1	Residential Apartments	78921
2	Student Apartments	5636

Count of Dwellings by CLUE small area and Dwelling Type

In [11]:
```
groupbyfields = ['clue_small_area','dwelling_type']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

# map
```

Out[11]:

	clue_small_area	dwelling_type	dwelling_number
			sum
0	Carlton	House/Townhouse	1520
1	Carlton	Residential Apartments	7800
2	Carlton	Student Apartments	3482
3	Docklands	House/Townhouse	164
4	Docklands	Residential Apartments	9566
5	East Melbourne	House/Townhouse	635
6	East Melbourne	Residential Apartments	2758
7	East Melbourne	Student Apartments	33
8	Kensington	House/Townhouse	3388
9	Kensington	Residential Apartments	2088
10	Melbourne (CBD)	House/Townhouse	45
11	Melbourne (CBD)	Residential Apartments	27111
12	Melbourne (CBD)	Student Apartments	1363
13	Melbourne (Remainder)	House/Townhouse	1
14	Melbourne (Remainder)	Residential Apartments	1576
15	North Melbourne	House/Townhouse	2268
16	North Melbourne	Residential Apartments	5947
17	North Melbourne	Student Apartments	731
18	Parkville	House/Townhouse	791
19	Parkville	Residential Apartments	2033
20	Parkville	Student Apartments	27
21	Port Melbourne	Residential Apartments	2
22	South Yarra	House/Townhouse	561
23	South Yarra	Residential Apartments	2191
24	Southbank	Residential Apartments	13784
25	West Melbourne (Residential)	House/Townhouse	653
26	West Melbourne (Residential)	Residential Apartments	4065

Count of Dwellings by Block Id

In [12]:
```
groupbyfields = ['block_id']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

# vertical barchart

# map
```

Out[12]:

	block_id	dwelling_number
		sum
0	1	385
1	101	863
2	103	638
3	104	1093
4	105	1729
5	107	24
6	11	690
7	1101	120
8	1103	860
9	1104	421

Count, Min, Max, Sum of Dwellings by CLUE small area, Block Id and Dwelling Type

In [13]:
```
groupbyfields = ['clue_small_area','block_id','dwelling_type']
aggregatebyfields = {'dwelling_number': ["count","min","max","sum"]}

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

# map
```

Out[13]:

	clue_small_area	block_id	dwelling_type	dwelling_number			
				count	min	max	sum
0	Carlton	201	House/Townhouse	5	1	5	15
1	Carlton	201	Residential Apartments	9	1	50	235
2	Carlton	202	House/Townhouse	38	1	4	47
3	Carlton	203	House/Townhouse	50	1	2	53
4	Carlton	203	Residential Apartments	4	1	6	12
5	Carlton	203	Student Apartments	1	16	16	16
6	Carlton	204	House/Townhouse	11	1	1	11
7	Carlton	204	Student Apartments	2	3	553	556
8	Carlton	205	Student Apartments	1	84	84	84
9	Carlton	206	House/Townhouse	25	1	2	26

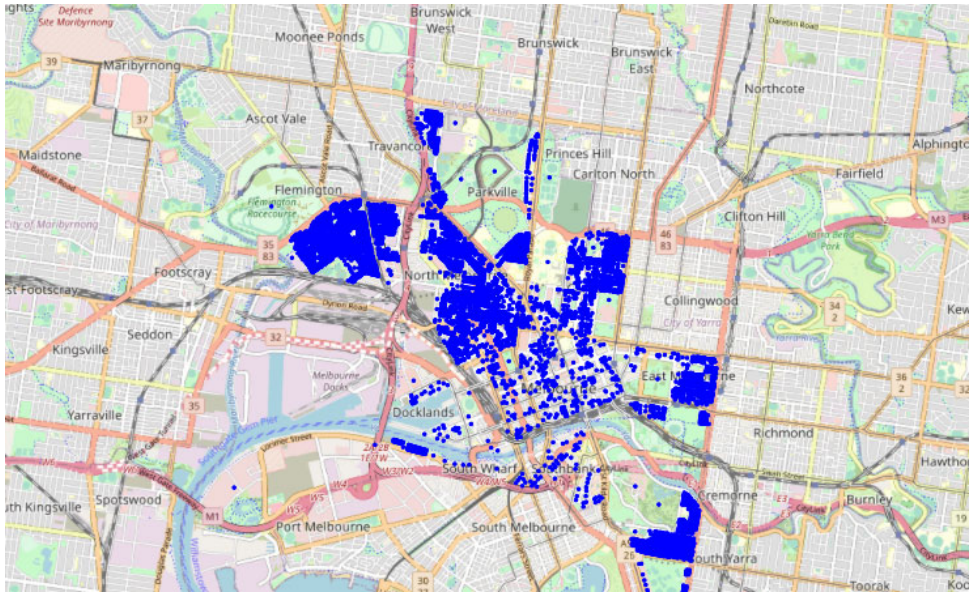
Plot Dwellings by Location on map

In [14]:
```
groupbyfields = ['clue_small_area','block_id','y_coordinate','x_coordinate']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

Out[14]:

	clue_small_area	block_id	y_coordinate	x_coordinate	dwelling_number
					sum
0	Carlton	201	-37.794301	144.965943	20
1	Carlton	201	-37.793904	144.965614	82
2	Carlton	201	-37.793832	144.966149	98
3	Carlton	201	-37.793622	144.966234	50
4	Carlton	202	-37.795680	144.966008	3
5	Carlton	202	-37.795633	144.965812	1
6	Carlton	202	-37.795624	144.965729	1
7	Carlton	202	-37.795616	144.965653	1
8	Carlton	202	-37.795609	144.965587	1
9	Carlton	202	-37.795603	144.966021	1

In [15]:
```
fig = px.scatter_mapbox(dwellingsByLocn, lat="y_coordinate", lon="x_coordinate",
                        hover_name="clue_small_area",
                        hover_data=["clue_small_area", "block_id"],
                        color_discrete_sequence=["blue"], zoom=12, height=600)
fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



Plot Dwelling Density by Block

In [16]:
```
groupbyfields = ['block_id','clue_small_area']
aggregatebyfields = {'dwelling_number': ["sum"]}

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

Out[16]:

	block_id	clue_area	dwelling_count
0	1	Melbourne (CBD)	385
1	101	West Melbourne (Residential)	863
2	103	Melbourne (CBD)	638
3	104	Melbourne (CBD)	1093
4	105	Melbourne (CBD)	1729
5	107	Melbourne (CBD)	24
6	11	Melbourne (CBD)	690
7	1101	Docklands	120
8	1103	Docklands	860
9	1104	Docklands	421

Get Block Polygon data in GeoJSON format

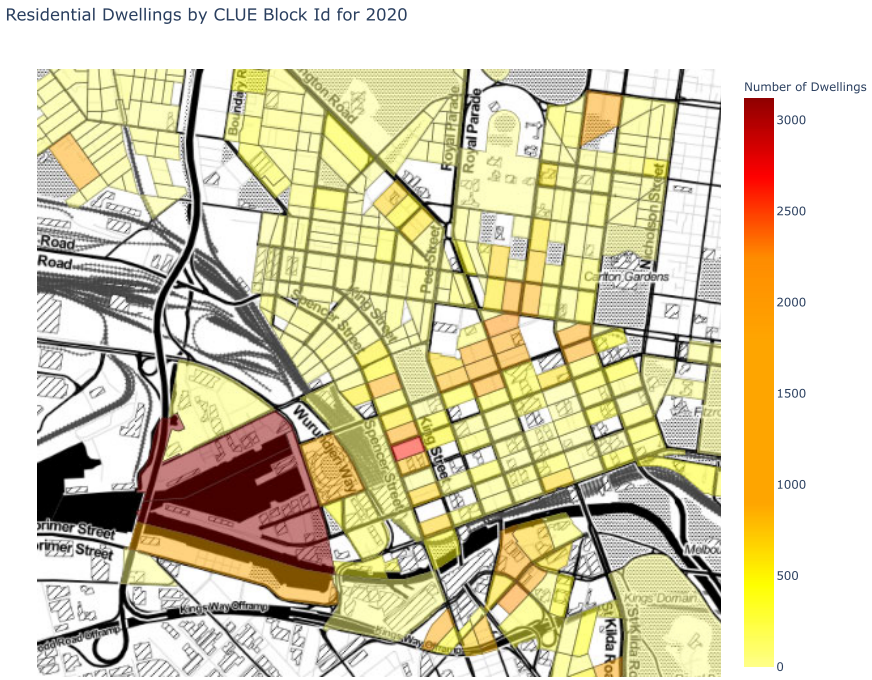
Load the CLUE Blocks in GeoJSON format and verify the location keys.

In [18]:
```
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[18]:
```
dict_keys(['block_id', 'clue_area'])
```

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

In [19]:
```
range_max = dwellingsByBlock['dwelling_count'].max()

fig = px.choropleth_mapbox(dwellingsByBlock, geojson=block, locations='block_id', color='dwelling_count',
                           color_continuous_scale=["#FFFF88", "yellow", "orange", "orange",
                                                   "orange", "darkorange", "red", "darkred"],
                           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,'dwelling_count':True},
                           labels={'dwelling_count':'Number of Dwellings','block_id':'CLUE Block Id'},
                           title='Residential Dwellings by CLUE Block Id for 2020',
                           width=950, height=800
                          )
fig.show()
```



In [ ]:
```
```