Geospatial Visualisations

This notebook will follow the concept of visualising information in a geospatial format. Essentially a heatmap of london with different prices. Presentation like the following image:

This would be an appealing method of conveying possible trends that align with the location of an area in particular and answer some of the questions in our brief in a more readable manner.

This would be done via the `Folium` library. This allows for geospatial visualisations and the creation of maps and handling different geospatial datastructures such as `GeoJsons` . A `GeoJson` is a special json file that can store different polygons that, when overlapped onto a world map, can form districts. They are lists of specific points which translate into shapes and hences these districts. In our use case we would need to either find or create a `GeoJson` of the London Outcodes we have collected.

Luckliy a user on github named *radoi90* managed to create a pre-made GeoJson file for this exact purpose.

In [2]:
```
import folium
import json
import geopandas as gpd
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
import numpy as np
```

In [3]:
```
path = ".\GeoSpatial\London.geojson"

df = gpd.read_file(path)
df = df.to_crs(epsg=4326) #Folium uses this form of Long and Lat as measurements so w
# crs = "Coordinate reference system"

df = df.rename(columns={"Name":"Postcode"})
df.head()
```

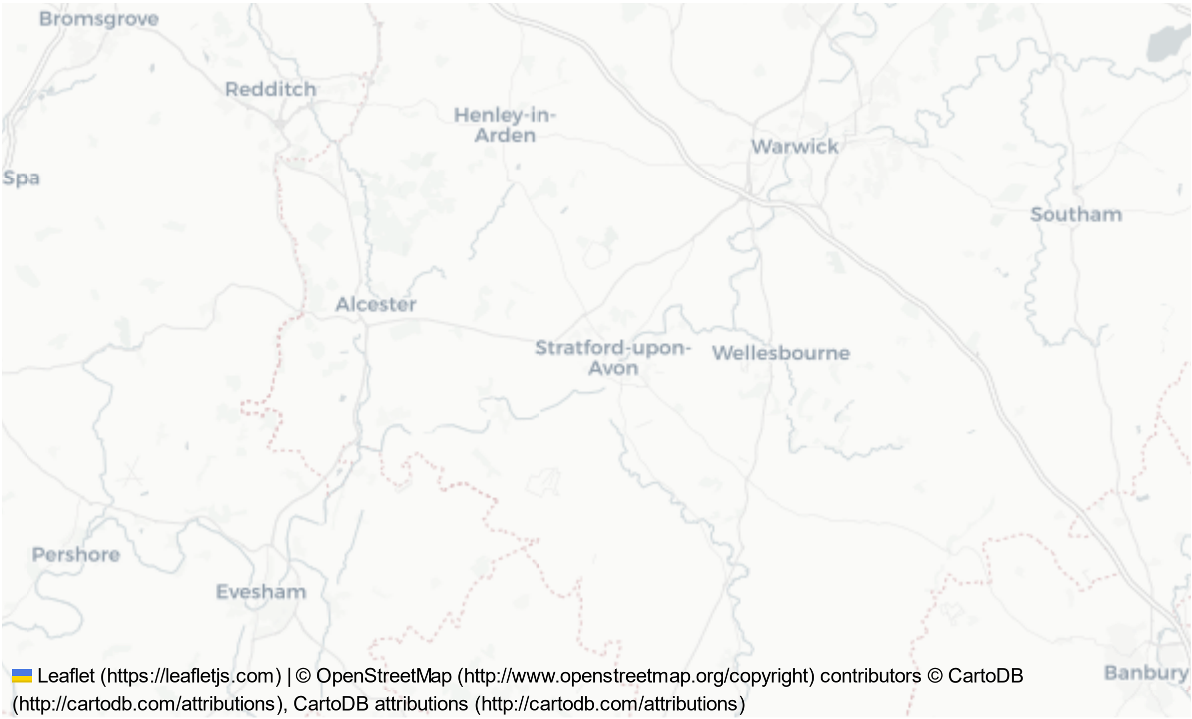
Out[3]:

	Postcode	Description	geometry
0	E2	E2 postcode district	POLYGON ((-0.04385 51.53179, -0.04324 51.53155...
1	E3	E3 postcode district	POLYGON ((-0.03401 51.53952, -0.03317 51.54040...
2	E4	E4 postcode district	POLYGON ((0.02200 51.63223, 0.02200 51.63209, ...
3	E5	E5 postcode district	POLYGON ((-0.03337 51.55989, -0.03248 51.55917...
4	E6	E6 postcode district	POLYGON ((0.04204 51.53823, 0.04217 51.53840, ...

In [4]:
```
with open(path, 'r') as f:
    data = json.load(f)
```

In [5]:
```
m = folium.Map(location=[51.5072, 0.1276],tiles='CartoDB positron',zoom_control=False
for _, r in df.iterrows():
    # Without simplifying the representation of each borough,
    # the map might not be displayed
    sim_geo = gpd.GeoSeries(r['geometry']).simplify(tolerance=0.0001)
    geo_j = sim_geo.to_json()
    geo_j = folium.GeoJson(data=geo_j,
                           style_function=lambda x: {'fillColor': 'orange'})
    folium.Popup(r['Postcode']).add_to(geo_j)
    geo_j.add_to(m)
m
```

Out[5]:



In [6]:
```
import pandas as pd
```

In [10]:
```
data = pd.read_csv("Final_data.csv")
data = data.drop([data.columns[0]],axis=1)
data.head()
```

Out[10]:

	asb	burglary	robbery	vehicle	violent	shoplifting	criminal_damage_and_arson	other_theft	dru
0	9.0	1.0	1.0	2.0	2.0	5.0	0.0	10.0	
1	3.0	10.0	5.0	4.0	9.0	0.0	2.0	12.0	
2	7.0	1.0	2.0	2.0	5.0	6.0	1.0	8.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	13.0	4.0	1.0	7.0	11.0	3.0	7.0	14.0	

In [11]:
```
Nan_postcodes = data[data["drugs"].isna()]["Postcode"]
Nan_postcodes.shape

valid_postcodes = data[~data["drugs"].isna()]
valid_postcodes.shape

valid_postcodes["Postcode"]
```

Out[11]:
```
0      EC1A
1      EC1M
2      EC1N
3      EC1P
4      EC1R
       ...
150     W10
151     W11
152     W12
153     W13
154     W14
Name: Postcode, Length: 153, dtype: object
```

In [12]:
```
Valid_postcodes = list(set(valid_postcodes["Postcode"]) & set(df["Postcode"]))
```

In [13]:
```
df = df.rename(columns={"Name":"Postcode"})
df = df.set_index("Postcode")
df.shape
```

Out[13]:
```
(177, 2)
```

In [14]:
```
valid_postcodes = valid_postcodes.set_index("Postcode")
valid_postcodes.shape
```

Out[14]:
```
(153, 18)
```

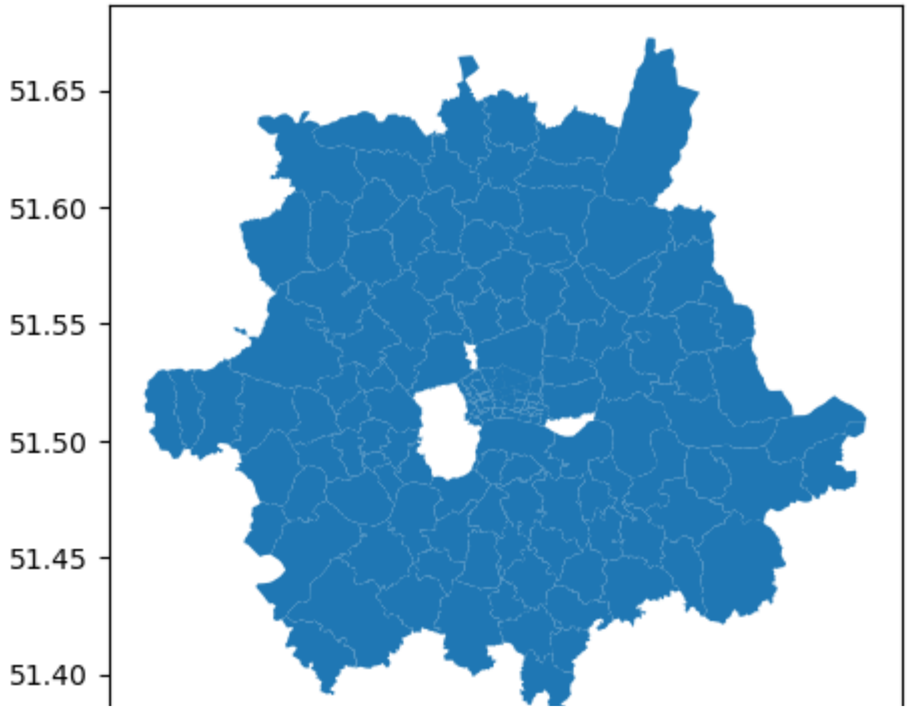
Geopandas Plotting

Using the Geopandas library, we can plot several different interesting visuals. The following section shows some examples provided that can be incorporated into the analysis.

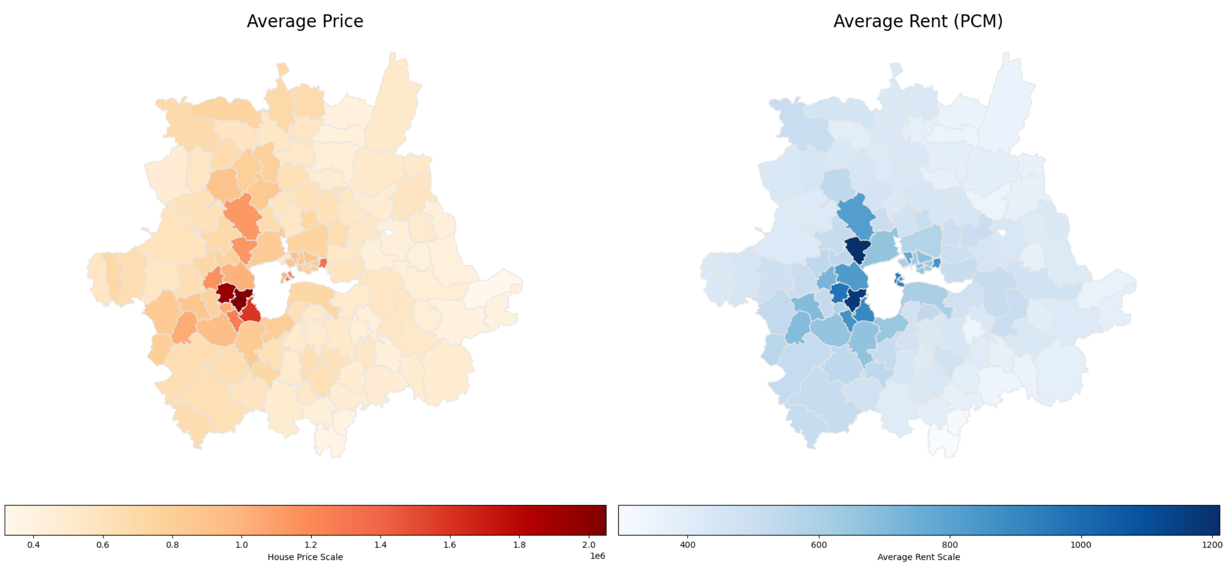
The visualisations below cover single values that each postcode district/area contains. The type of plot is known as a Chloropleth. Each district has a value and a colour is assigned on a scale to the value, similar to a heatmap. This method allots the visualisation of the geospatial distribution.

In [15]:
```
final_data = pd.merge(valid_postcodes,df,on="Postcode")

final_data = final_data.reset_index()

final_data = gpd.GeoDataFrame(final_data)

final_data.plot()
```

Out[15]:
```
<AxesSubplot:>
```



In [12]:
```
fig, ax = plt.subplots(nrows=1,ncols=2, figsize=(20,9),constrained_layout=True)
final_data.plot(column='average_price', cmap='OrRd', linewidth=1, ax=ax[0], edgecolor
final_data.plot(column='average_rent', cmap='Blues', linewidth=1, ax=ax[1], edgecolor

ax[0].axis('off')
ax[0].set_title("Average Price",size=20)

ax[1].axis('off')
ax[1].set_title("Average Rent (PCM)",size=20);
```



In [8]:
```
final_data["total_norm"] =(final_data["total"] - final_data["total"].min())/(final_da
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13000\17567686.py in <module>
----> 1 final_data["total_norm"] =(final_data["total"] - final_data["total"].min
())/(final_data["total"].max() - final_data["total"].min())
```

```
NameError: name 'final data' is not defined
```

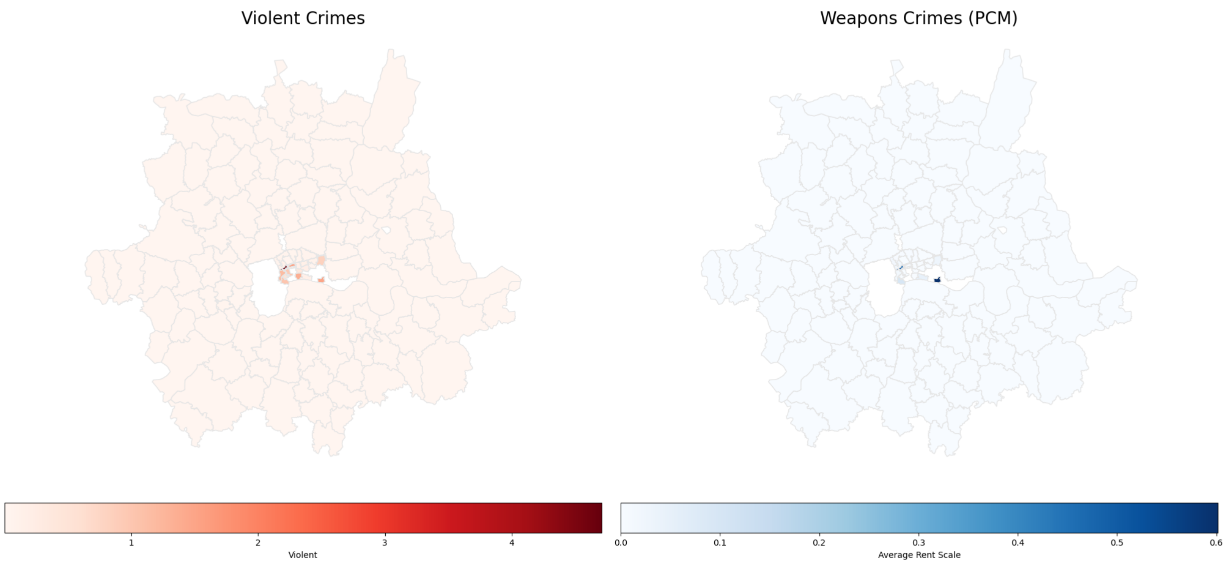
In [14]:
```
#final_data.head()
final_data_prices = final_data["average_price"]
final_data.head()
```

Out[14]:

	Postcode	asb	burglary	robbery	vehicle	violent	shoplifting	criminal_damage_and_arsoı
0	EC1A	0.287324	0.000000	0.028732	0.057465	0.201127	0.086197	0.02873:
1	EC1A	0.287324	0.000000	0.028732	0.057465	0.201127	0.086197	0.02873:
2	EC1M	0.282103	0.037614	0.094034	0.056421	0.150455	0.056421	0.03761
3	EC1M	0.282103	0.037614	0.094034	0.056421	0.150455	0.056421	0.03761
4	EC1N	0.436180	0.054523	0.018174	0.054523	0.145393	0.036348	0.09087

5 rows × 22 columns

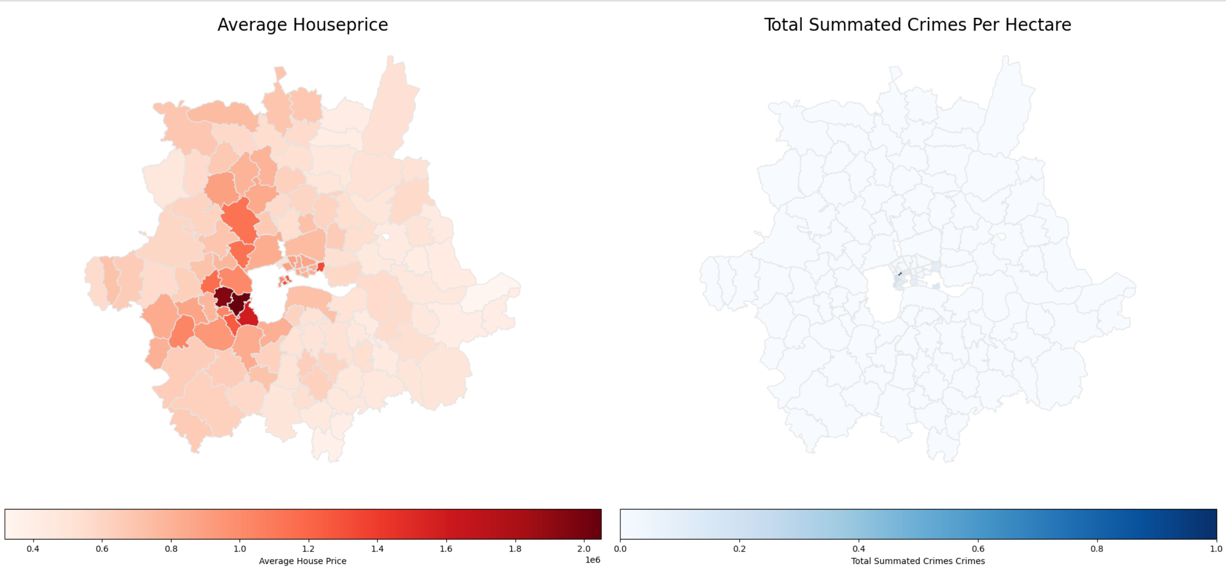
In [15]:
```
fig, ax = plt.subplots(nrows=1,ncols=2, figsize=(20,9),constrained_layout=True)
final_data.plot(column='violent', cmap='Reds', linewidth=1, ax=ax[0], edgecolor='0.9
final_data.plot(column='weapons', cmap='Blues', linewidth=1, ax=ax[1], edgecolor='0.9

ax[0].axis('off')
ax[0].set_title("Violent Crimes",size=20)

ax[1].axis('off')
ax[1].set_title("Weapons Crimes (PCM)",size=20);
```



Crime per Hectare - Average Houseprice

In [16]:

```
fig, ax = plt.subplots(nrows=1,ncols=2, figsize=(20,9),constrained_layout=True)
final_data.plot(column='average_price', cmap='Reds', linewidth=1, ax=ax[0], edgecolor
final_data.plot(column='total_norm', cmap='Blues', linewidth=1, ax=ax[1], edgecolor=

ax[0].axis('off')
ax[0].set_title("Average Houseprice",size=20)

ax[1].axis('off')
ax[1].set_title("Total Summated Crimes Per Hectare",size=20);
```



This comparison fails to provide any information surrounding the crime rate and price of housing. This could be due to the minimal amount of crimes done per hectare. One observation we can see is the central area of London containing somewhat higher figures than any other area. This is expected.

The areas with the lowest housing prices dont have the lowest crimerates according to the `per hectare` variables.

Folium Plotting

Although using these static visualisations is useful in observing the spread of values over a geographical area, it would be more useful to see the direct crime statistic plotted alongside the houseprice/rent. This can be done using `folium` where it provides a more interactive map library built upon javascript.

In [7]:
```
M = folium.Map(location=[51.5072, 0.0],tiles='CartoDB positron',
                zoom_start=10.5,
                zoom_control=False,
                scrollWheelZoom=False,
                dragging=True)#Put this here so map resets


for _, r in final_data.iterrows():
    # Without simplifying the representation of each borough,
    # the map might not be displayed
    sim_geo = gpd.GeoSeries(r['geometry']).simplify(tolerance=0.000000001)
    geo_j = sim_geo.to_json()
    geo_j = folium.GeoJson(data=geo_j,
                           style_function=lambda x: {'fillColor': 'orange',"weight":'
    folium.Popup(r['Postcode']).add_to(geo_j)
    geo_j.add_to(M)
M
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13000\3104442542.py in <module>
      6
      7
----> 8 for _, r in final_data.iterrows():
      9     # Without simplifying the representation of each borough,
     10     # the map might not be displayed

NameError: name 'final_data' is not defined
```

Controlling the colour of each district using a specific stylefunction mapper would be difficult as it requires looking at outdated documentation. One approach would be to directly calculate the chloropleth map using folium, meaning we have to write to a Geopandas using the cleaned data, then read it again and link it to a dataframe we have.

In [26]:
```
columns_drop
```

Out[26]:
```
Index(['asb', 'burglary', 'robbery', 'vehicle', 'violent', 'shoplifting',
       'criminal_damage_and_arson', 'other_theft', 'drugs', 'bike_theft',
       'theft_from_the_person', 'weapons', 'public_order', 'other', 'total',
       'error', 'average_price', 'average_rent', 'Description'],
      dtype='object')
```

In [25]:
```
columns_drop = final_data.columns[1:-2]
geo_export_df = final_data.drop(columns_drop,axis=1)
geo_export_df = geo_export_df.set_index("Postcode")
geo_export_df.to_file("postcode.geojson", driver='GeoJSON')
```

In [27]:
```
geo_export_df
```

Out[27]:

	geometry	total_norm
Postcode		
EC1A	POLYGON ((-0.09809 51.52013, -0.09783 51.52037...	0.026064
EC1A	POLYGON ((-0.09809 51.52013, -0.09783 51.52037...	0.026064
EC1M	POLYGON ((-0.10597 51.52024, -0.10656 51.52026...	0.022997
EC1M	POLYGON ((-0.10597 51.52024, -0.10656 51.52026...	0.022997
EC1N	POLYGON ((-0.10962 51.51735, -0.10970 51.51721...	0.021469
...	...	...
W10	POLYGON ((-0.21900 51.52754, -0.21893 51.52760...	0.000803
W11	POLYGON ((-0.20033 51.52071, -0.20002 51.52018...	0.000962
W12	POLYGON ((-0.23281 51.52062, -0.23261 51.52058...	0.000367
W13	POLYGON ((-0.33052 51.51312, -0.33046 51.51343...	0.000245
W14	POLYGON ((-0.22144 51.49868, -0.22170 51.49890...	0.000500

141 rows × 2 columns

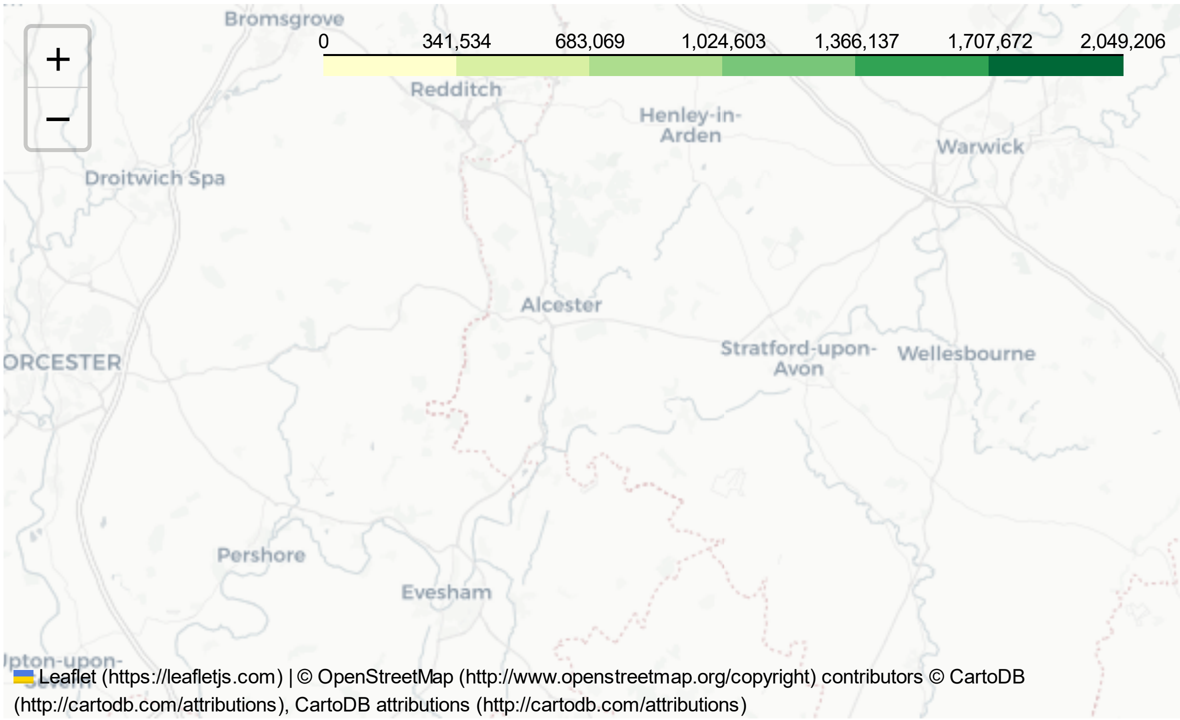
In [17]:

```
df_data = final_data.drop(["geometry","Description","error"],axis=1)
df_data.head()
df_data = df_data.fillna(0)
df_data.head()

df_data.columns[-4]

aliases_n= ["Anti-Social Behaviour:",
          "Burglary",
          'Robbery',
          'Vehicle Theft',
          'Violent Crimes',
          'Shoplifting',
          'criminal Damage/Arson',
          'Misc theft',
          'Drug Crime',
          'Bike theft',
          'Mugging',
          'Weapon Crime',
          'Public Order Crime']
```

Now we have both datasources compiled, with the ID for the postcode being the ID on the geoJson and the data with the geometry removed, we can compile the maps and obtain a [folium chloropleth map](#).

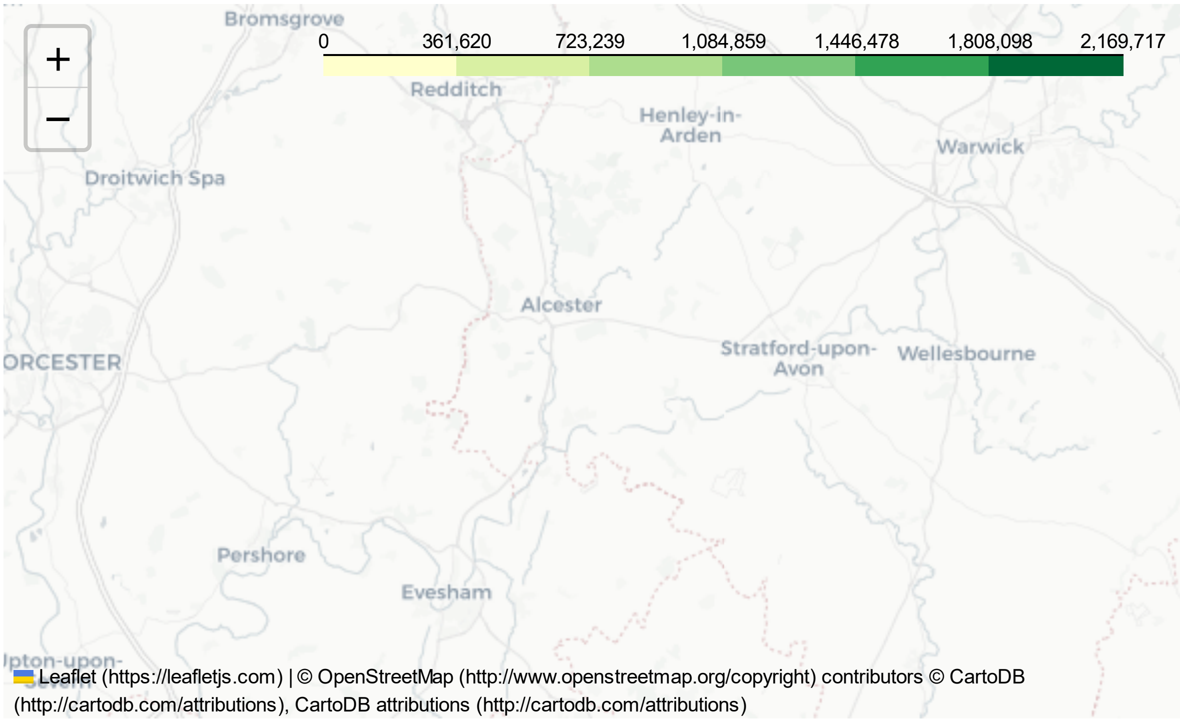
In [57]:
```
M = folium.Map(location=[51.5072, 0.0],tiles='cartodbpositron',
                zoom_start=10.4,
                zoom_control=True,
                scrollWheelZoom=False,
                dragging=True)#Put this here so map resets



folium.Choropleth(
    geo_data="postcode.geojson",
    name="choropleth",
    data=df_data,
    columns=["Postcode", "average_price"],
    key_on="feature.properties.Postcode",
    fill_color="YlGn",
    fill_opacity=0.7,
    line_opacity=0.2).add_to(M)

folium.features.GeoJson(
    data=final_data,
    name="choropleth",
    smooth_factor=0.01,
    style_function=lambda x: {'color':'black','fillColor':'transparent','weight':0.5}
    tooltip=folium.features.GeoJsonTooltip(
                    fields=list(df_data.columns[1:-5]),
                    aliases=aliases_n,
                    localize=True,
                    sticky=False,
                    labels=True,
                    style="""
                        background-color: #F0EFEF;
                        border: 2px solid black;
                        border-radius: 3px;
                        box-shadow: 3px;
                    """,
                    max_width=800),
    highlight_function=lambda x: {'weight':3,'fillColor':'grey'}).add_to(M)


M
```

Out[57]:


0
341,534
683,069
1,024,603
1,366,137
1,707,672
2,049,206
Bromsgrove
Redditch
Henley-in-Arden
Warwick
Droitwich Spa
Alcester
ORCESTER
Stratford-upon-Avon
Wellesbourne
Pershore
Evesham
Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

In [18]:
```
M = folium.Map(location=[51.5072, 0.0],tiles='cartodbpositron',
                zoom_start=10.4,
                zoom_control=True,
                scrollWheelZoom=False,
                dragging=True)#Put this here so map resets



folium.Choropleth(
    geo_data="postcode.geojson",
    name="choropleth",
    data=df_data,
    columns=["Postcode", "average_price"],
    key_on="feature.properties.Postcode",
    fill_color="YlGn",
    fill_opacity=0.7,
    line_opacity=0.2).add_to(M)

folium.features.GeoJson(
    data=final_data,
    name="choropleth",
    smooth_factor=0.01,
    style_function=lambda x: {'color':'black','fillColor':'transparent','weight':0.5}
    tooltip=folium.features.GeoJsonTooltip(
                        fields=[df_data.columns[-4]],
                        aliases=["Total Crime"],
                        localize=True,
                        sticky=False,
                        labels=True,
                        style="""
                            background-color: #F0EFEF;
                            border: 2px solid black;
                            border-radius: 3px;
                            box-shadow: 3px;
                        """,
                        max_width=800),
    highlight_function=lambda x: {'weight':3,'fillColor':'grey'}).add_to(M)


M
```

Out[18]:



The entire analysis above was done with metrics pertaining to `per-hectare` crimes. This means it may be more representable of relationships/correlations. Performing the same calculations again with the actual count value of crimes may be more representable. The question of:

"**Is there an association between rents and the level of crime in Greater London postcodes?**"

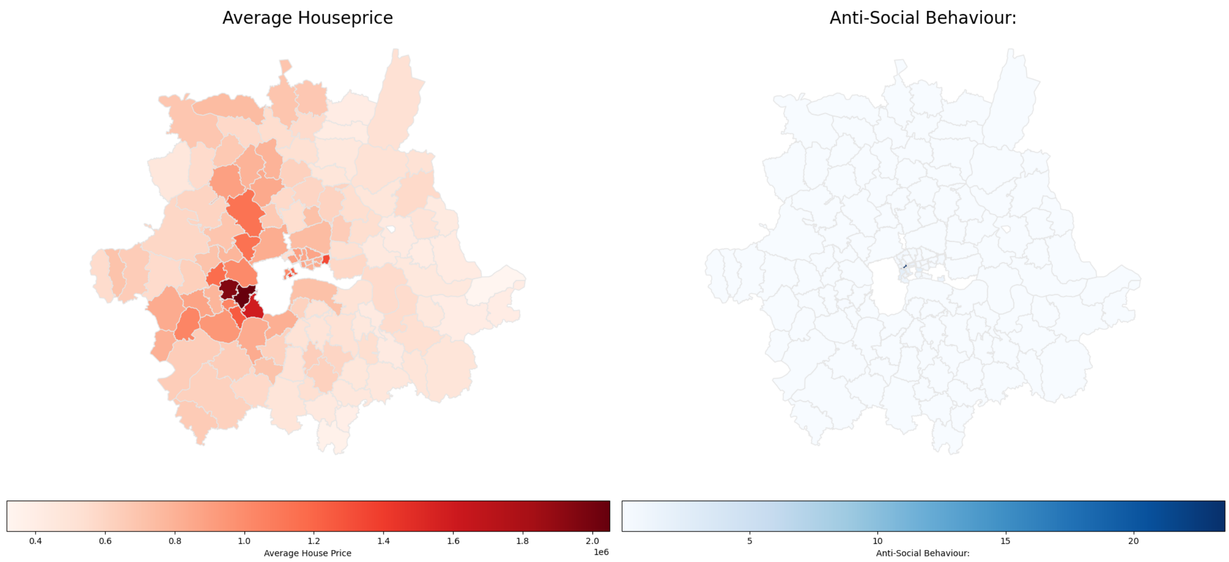
Seems to be inconclusive. Currently there doesnt seem to be a relationship visible in any of the areas shown in the geospatial projections. The initial geopandas plot of crimes `per-hectare` against the average `house price` shows almost no relationship. This could due to many external factors effecting peoples decision making process when choosing an area to live. It would make logical sense that the level of crime being higher may make an area more undesireable to live in, but the opposite occurence may be true. A lower income area may introduce higher levels of crimes. This cannot be stated as conclusive though as the relationship is not fully modelled in this analysis.

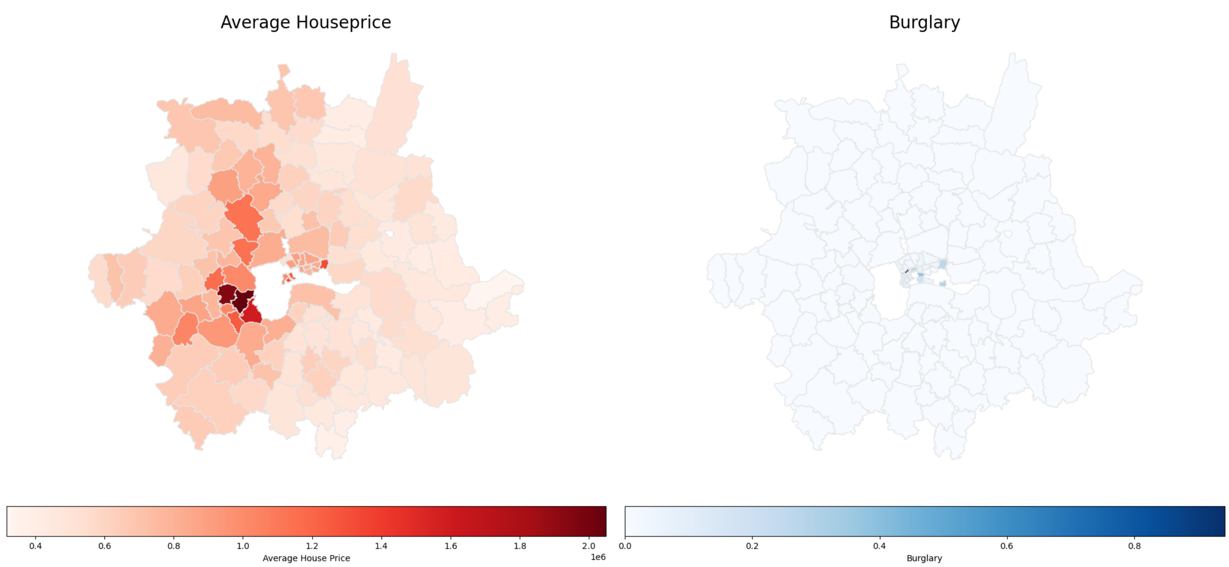
Crime count Analysis

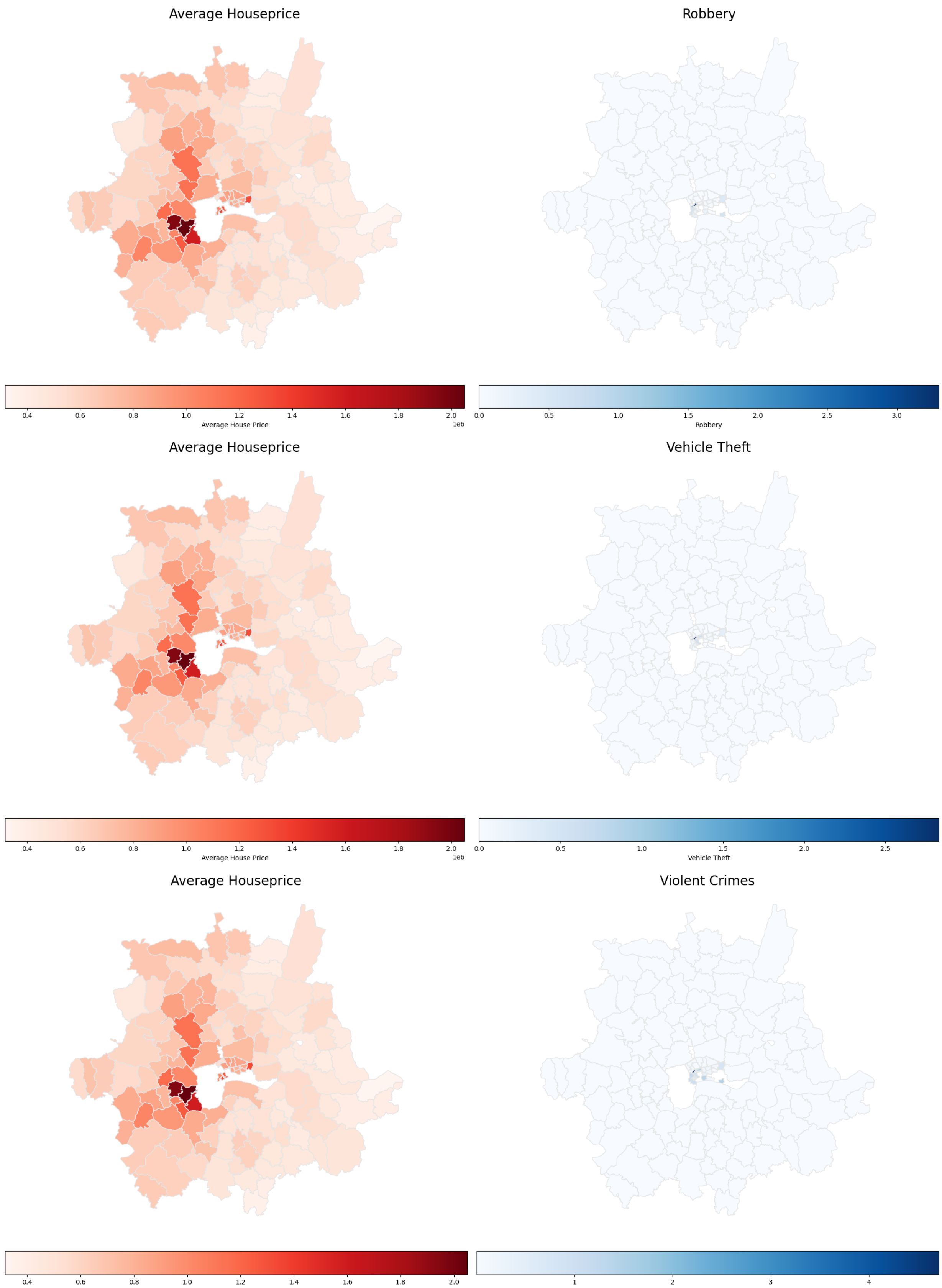
Following the above analysis, we will now plot every crime count alongside the average house price to see if there is any observable patterns. The total crime per hectare although useful as a summerative statistic, abstracts the smaller values and doesnt allow them to speak for themselves.

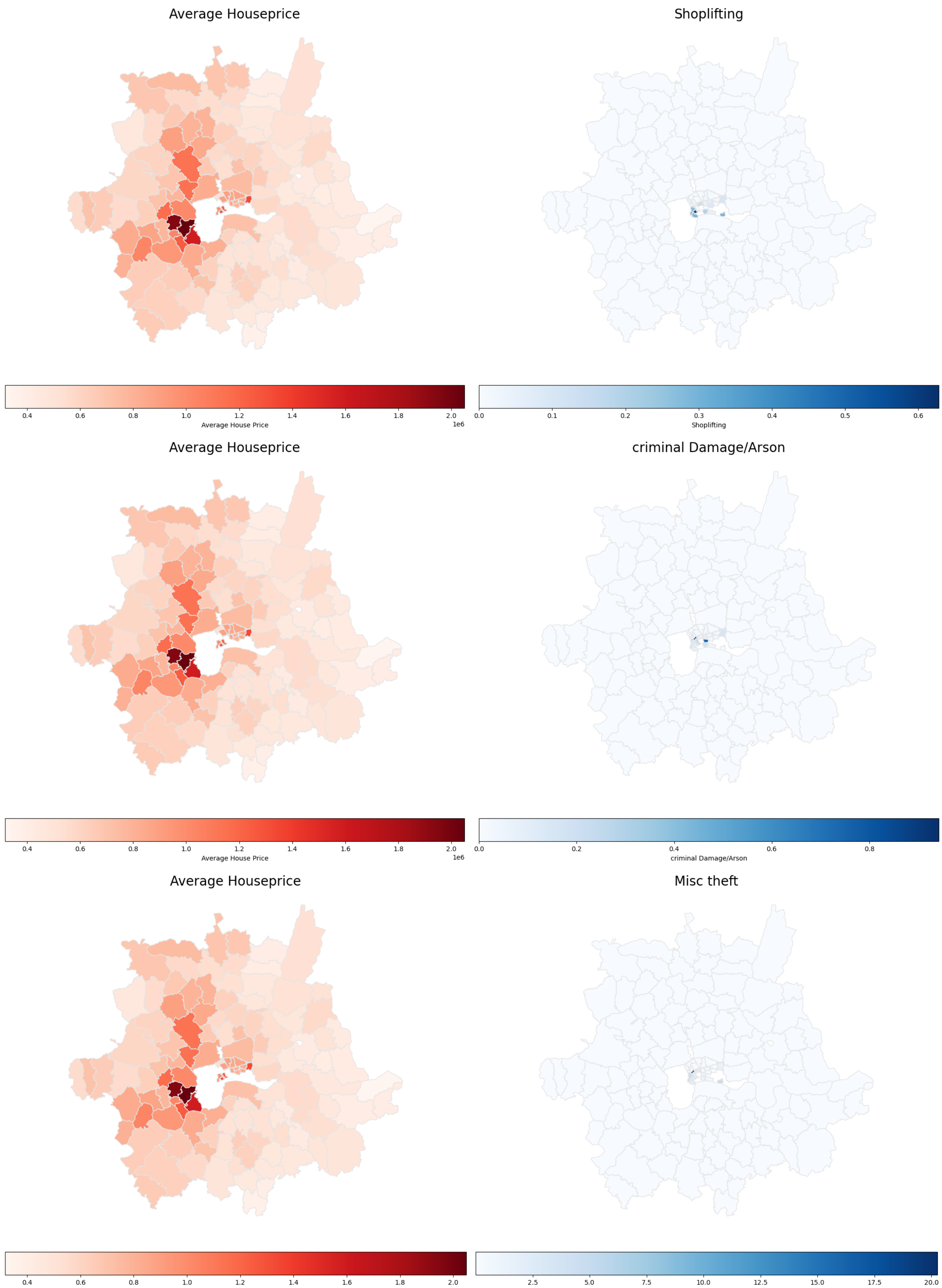
In [44]:
```
def compare_plot(column_n,alias):
    fig, ax = plt.subplots(nrows=1,ncols=2, figsize=(20,9),constrained_layout=True)
    final_data.plot(column='average_price', cmap='Reds', linewidth=1, ax=ax[0], edgec
    final_data.plot(column=column_n, cmap='Blues', linewidth=1, ax=ax[1], edgecolor=

    ax[0].axis('off')
    ax[0].set_title("Average Houseprice",size=20)

    ax[1].axis('off')
    ax[1].set_title(alias,size=20);
```

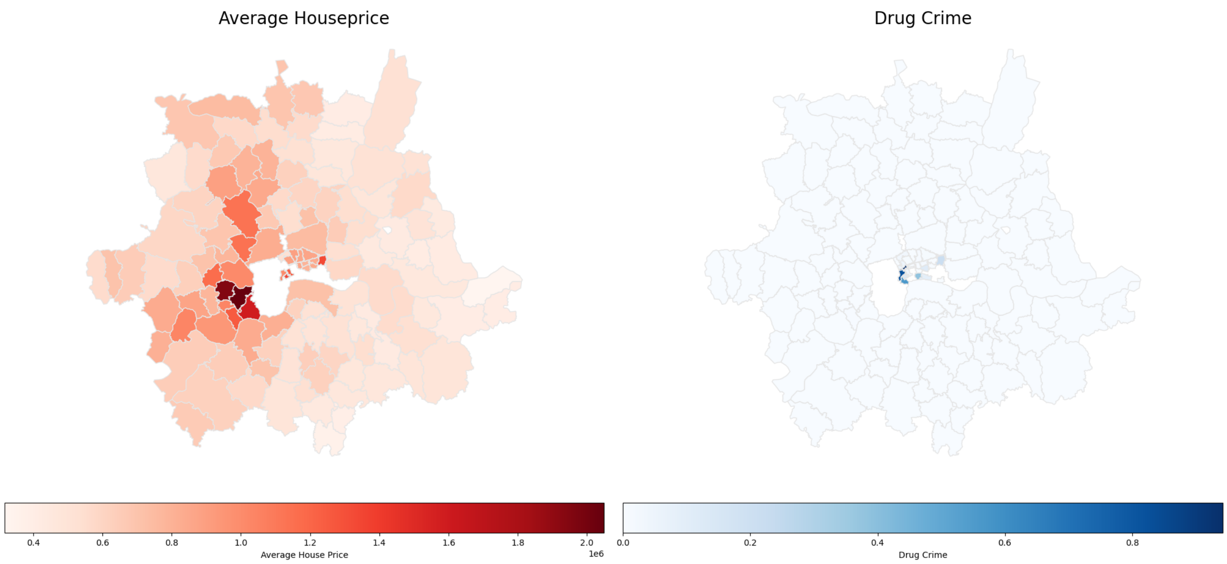
In [63]:
```
Columns_n = list(final_data.columns[1:-8])


for i in range(len(Columns_n)):
    compare_plot(Columns_n[i],aliases_n[i])
```

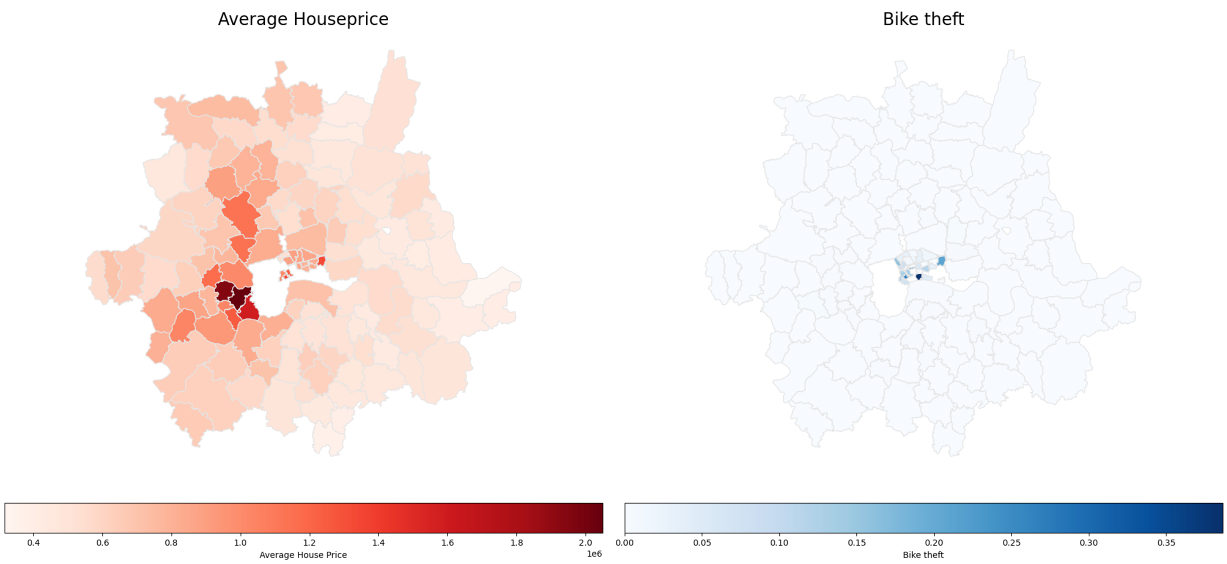


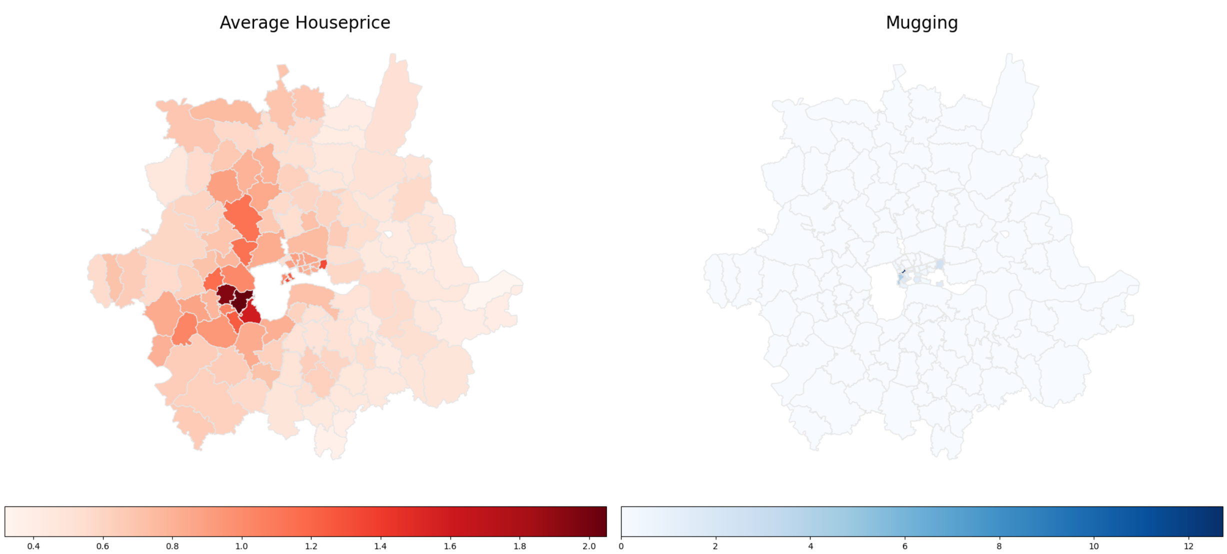


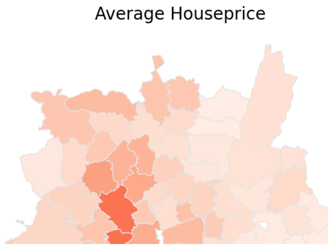

Average Houseprice
Robbery
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
1e6
Average House Price
0.0 0.5 1.0 1.5 2.0 2.5 3.0
Robbery
Average Houseprice
Vehicle Theft
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
1e6
Average House Price
0.0 0.5 1.0 1.5 2.0 2.5
Vehicle Theft
Average Houseprice
Violent Crimes
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
1e6
Average House Price
1 2 3 4
Violent Crimes


Average Houseprice
Shoplifting
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
0.0 0.1 0.2 0.3 0.4 0.5 0.6
Shoplifting
Average Houseprice
criminal Damage/Arson
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
0.0 0.2 0.4 0.6 0.8
criminal Damage/Arson
Average Houseprice
Misc theft
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Misc theft


Average Houseprice
Drug Crime
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
0.0 0.2 0.4 0.6 0.8
Drug Crime


Average Houseprice
Bike theft
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35
Bike theft


Average Houseprice
Mugging
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Average House Price
1e6
0 2 4 6 8 10 12
Mugging



Following the visualisations abo