In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import descartes
import geopandas as gpd
from shapely.geometry import Point, Polygon
import csv
import bokeh
from bokeh.io import output_notebook, show
from bokeh.plotting import figure
import json
from bokeh.models import (CDSView, ColorBar, ColumnDataSource,
                          CustomJS, CustomJSFilter,
                          GeoJSONDataSource, HoverTool,
                          LinearColorMapper, Slider)
from bokeh.layouts import column, row, widgetbox
from bokeh.palettes import brewer
from bokeh.plotting import figure, save
from bokeh.palettes import RdYlBu11 as palette
from bokeh.models import LogColorMapper
```

In [35]:
```
# !pip install -U notebook-as-pdf
```

In [36]:
```
# !pyppeteer-install
```

Data Files upload

- Grocery stores csv and shp file
- Entropy Index csv from Vianny

In [2]:
```
#Affordldable Housing Locations
url='https://raw.githubusercontent.com/kathiavf16/DS4A-Team-10/main/data/affordable_hou
df=pd.read_csv(url)
```

In [3]:
```
#Grocery Store Locations
url='https://raw.githubusercontent.com/kathiavf16/DS4A-Team-10/main/data/grocery_store_
df_stores=pd.read_csv(url)
```

In [4]:
```
#Entropy Index csv
Entro=pd.read_csv('https://raw.githubusercontent.com/kathiavf16/DS4A-Team-10/main/data/

#made into geodataframe if needed
Entro_geo=gpd.GeoDataFrame(Entro)
```

In [5]:
```
# df_stores.head(5)
# df_stores.columns
```

In [6]:
```
#Check columns to see if there is x, y coordinates
df_stores.columns
```

Out[6]:
```
Index(['the_geom', 'cartodb_id', 'x', 'y', 'objectid', 'storename', 'address',
       'phone', 'present90', 'present95', 'present00', 'present05',
       'present08', 'notes', 'present09', 'present10', 'present11',
       'present12', 'present13', 'present14', 'present15', 'present16',
       'present17', 'x_coord', 'y_coord', 'ward', 'address_id', 'ssl',
       'zipcode', 'present18', 'gis_id'],
      dtype='object')
```

CSV --> plotable point example using Grocery Stores CSV

In [7]:
```
#convert xy coordinates to a point by adding a new column named geometry
geometry= [Point(xy) for xy in zip(df_stores['x'], df_stores['y'])]
geometry[:3]
```

Out[7]:
```
[<shapely.geometry.point.Point at 0x29b4f4d2130>,
 <shapely.geometry.point.Point at 0x29b4f4d24c0>,
 <shapely.geometry.point.Point at 0x29b4f4d2520>]
```

In [8]:
```
#convert to a geodata frame
df_stores_geo= gpd.GeoDataFrame(df_stores, geometry=geometry)
```

In [9]:
```
#df_stores_geo.head(5)
```

SHP file upload

In [10]:
```
Street_Map= gpd.read_file('Roadway_SubBlock.shp')
Census_tracts=gpd.read_file('Census_Tracts_in_2010.shp')
stores=gpd.read_file('Grocery_Store_Locations.shp')
```

In [11]:
```
#check that type is an object
Census_tracts.type
```

Out[11]:
```
0      Polygon
1      Polygon
2      Polygon
3      Polygon
4      Polygon
       ...
174    Polygon
175    Polygon
176    Polygon
177    Polygon
178    Polygon
Length: 179, dtype: object
```

Joins -- Spatial and Attribute

- Spatial join between food store locations (shp) and census tracts (shp) to see which stores fall in which census tract https://geopandas.org/reference/geopandas.sjoin.html

- Attribute join. This is joining a dataframe with a geodataframe aka table with shp file

In [12]:
```
#spatial join
FoodStore_within_Census_tracts= gpd.sjoin(stores,Census_tracts, op='within', how='inner
```

In [13]:
```
#check if all columns were added
FoodStore_within_Census_tracts.columns
pd.set_option('display.max_columns', None)
```

In [14]:
```
#attribute join using concat
#Joining the entropy index to each census tract
Entro_CensusTracts=pd.concat([Entro,Census_tracts], join='inner', axis=1)

#making it into a geodataframe to be mapped
Entro_CensusTracts_geo=gpd.GeoDataFrame(Entro_CensusTracts)
```

In [15]:
```
Entro_CensusTracts_geo.head(5)
```

Out[15]:

	GEOID	State	County	Urban	POP2010	OHU2010	GroupQuartersFlag	NUMGQTRS	PCT
0	11001000100	District of Columbia	District of Columbia	1	4890	2686	0	24	0
1	11001000201	District of Columbia	District of Columbia	1	3916	2	1	3908	0
2	11001000202	District of Columbia	District of Columbia	1	5425	1933	0	1135	0
3	11001000300	District of Columbia	District of Columbia	1	6233	2754	0	0	0
4	11001000400	District of Columbia	District of Columbia	1	1455	636	0	6	0

In [16]:
```
Entro_CensusTracts_geo.to_file('data/Entro_CensusTracts_2.shp', driver='ESRI Shapefile'
```

File Manipulation for mapping

In [17]:
```
# in the USDA file 1=exist 0= no
#the variable we choose here was access to car

one= Entro_CensusTracts_geo[Entro_CensusTracts_geo['LATractsVehicle_20'] == 1]
zero= Entro_CensusTracts_geo[Entro_CensusTracts_geo['LATractsVehicle_20'] == 0]

One_1=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_1And10'] == 1]
Zero_1=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_1And10'] == 0]


One_2=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_halfAnd10'] == 1]
Zero_2=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_halfAnd10'] == 0]

One_3=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_1And20'] == 1]
Zero_3=Entro_CensusTracts_geo[Entro_CensusTracts_geo['LILATracts_1And20'] == 0]

# LILATracts_1And10
# LILATracts_halfAnd10
# LILATracts_1And20
```

Exporting Files

In [18]:
```
#dataframe to a csv
FoodStore_within_Census_tracts.to_csv('data/FoodStores_CensusTracts.csv', index=False)
```

In [19]:
```
#to shp file
FoodStore_within_Census_tracts.to_file('data/FoodStore_within_Census_tracts.shp')
```

In [20]:
```
# to shp file and csv
Entro_CensusTracts_geo.to_file('data/Entro_CensusTracts.shp')
Entro_CensusTracts.to_csv('data/Entro_CensusTracts.csv', index=False)
```

In [21]:
```
Entro_c=gpd.read_file('data/Entro_CensusTracts.shp')
Entro_c.head(5)
```

Out[21]:

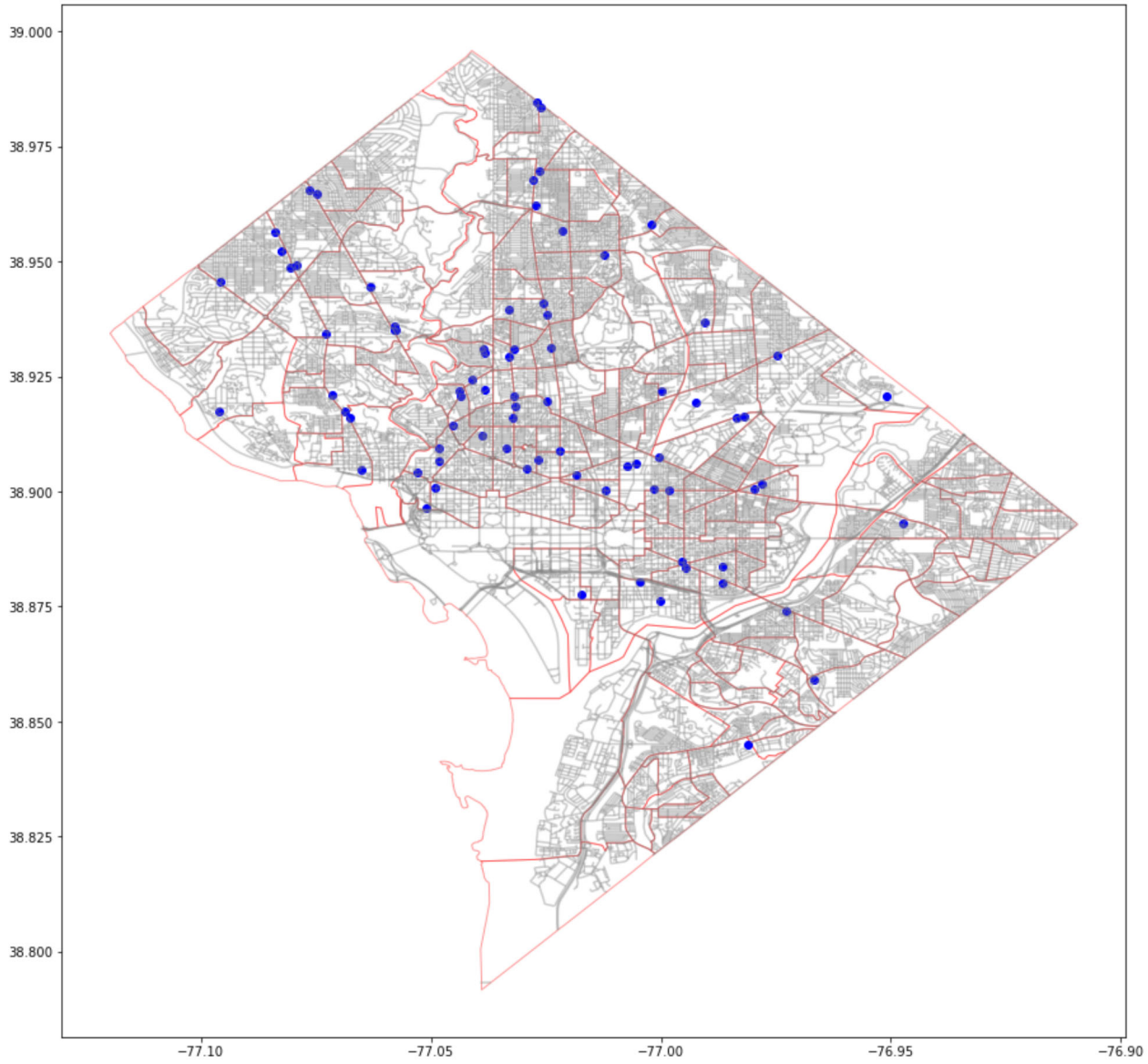
	GEOID	State	County	Urban	POP2010	OHU2010	GroupQuart	NUMGQTRS
0	11001000100	4800.865938637454	District of Columbia	4444	0	1	11001003400	250000
1	11001000201	2982.473558434313	District of Columbia	2940	0	2	11001003500	0

	GEOID	State	County	Urban	POP2010	OHU2010	GroupQuart	NUMGQTRS
2	11001000202	2293.465280386086	District of Columbia	4705	0	3	11001003600	205625
3	11001000300	2297.040643539269	District of Columbia	5497	0	4	11001003700	210821
4	11001000400	2708.295073802764	District of Columbia	1247	0	5	11001003800	250000

Maps

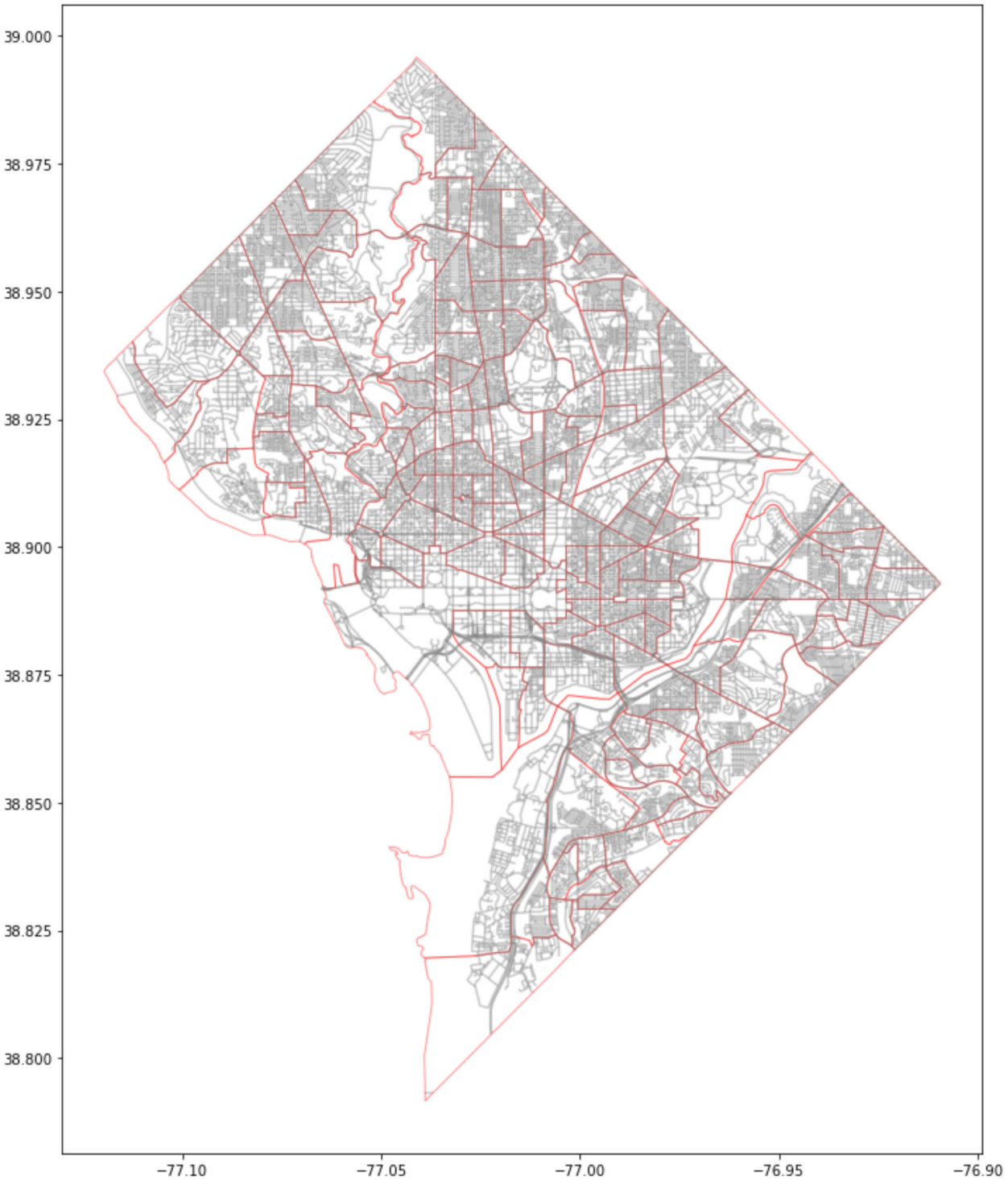
In [22]:
```
#Food Stores in each census tract
fig,ax=plt.subplots(figsize=(15,15))
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
Census_tracts.plot(ax=ax, alpha=0.4, edgecolor='red', facecolor='None')
df_stores_geo.plot(ax=ax, color='blue', marker='o')
```

Out[22]: <AxesSubplot:>



In [23]:
```
#Testing to see if the census entro file was working
fig,ax=plt.subplots(figsize=(15,15))
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
Entro_CensusTracts_geo.plot(ax=ax, alpha=0.4, edgecolor='red', facecolor='None')
```

Out[23]: <AxesSubplot:>



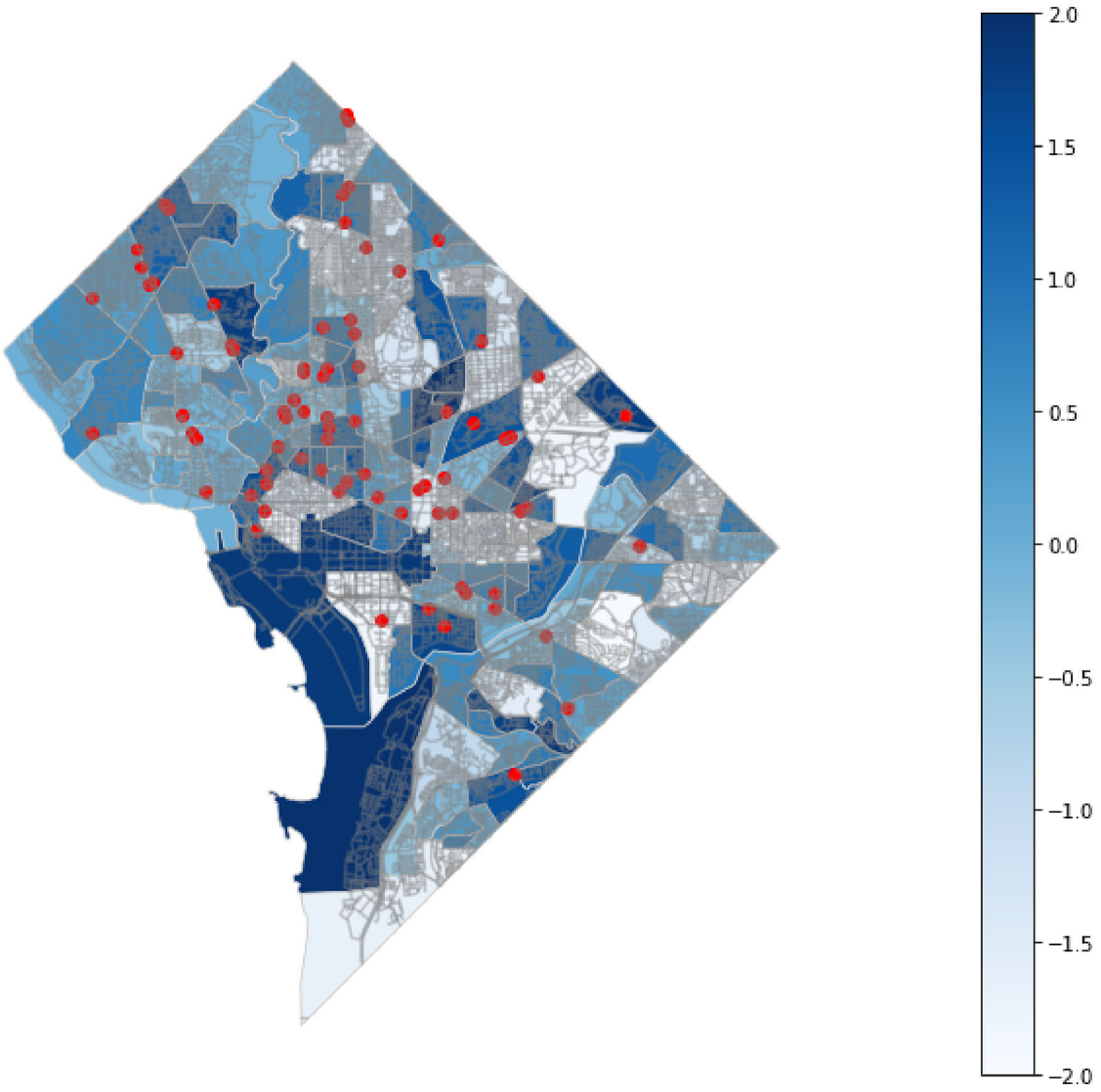
In [24]:

```
#Map example

variable= 'entropy_index'
vmin, vmax = -2, 2
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin, vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgeco
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')
```

```
#one.plot(ax=ax, alpha=0.4, color='purple')
# zero.plot(ax=ax, alpha=0.4, color='yellow')
```

Out[24]: <AxesSubplot:>



In [25]:
```
Entro_CensusTracts_geo.describe()
```

Out[25]:

	GEOID	Urban	POP2010	OHU2010	GroupQuartersFlag	NUMGQTRS	PCTGQTRS	LIL/
count	1.790000e+02	179.0	179.000000	179.00000	179.00000	179.000000	179.000000	
mean	1.100101e+10	1.0	3361.581006	1489.98324	0.01676	223.581006	0.057989	
std	3.275207e+03	0.0	1301.246536	729.89484	0.12873	656.490052	0.147274	
min	1.100100e+10	1.0	33.000000	2.00000	0.00000	0.000000	0.000000	
25%	1.100100e+10	1.0	2462.000000	1020.50000	0.00000	6.000000	0.001847	
50%	1.100101e+10	1.0	3072.000000	1342.00000	0.00000	26.000000	0.008745	
75%	1.100101e+10	1.0	3974.000000	1806.00000	0.00000	112.500000	0.035588	
max	1.100101e+10	1.0	7436.000000	4795.00000	1.00000	5079.000000	0.997957	

In [26]:
```
Entro_CensusTracts_geo.columns.unique
```

Out[26]:
```
<bound method Index.unique of Index(['GEOID', 'State', 'County', 'Urban', 'POP2010', 'OH
U2010',
       'GroupQuartersFlag', 'NUMGQTRS', 'PCTGQTRS', 'LILATracts_1And10',
       ...
       'FAGI_MED_1', 'FAGI_TOT_2', 'FAGI_MED_2', 'FAGI_TOT_3', 'FAGI_MED_3',
       'FAGI_TOT_4', 'FAGI_MED_4', 'FAGI_TOT_5', 'FAGI_MED_5', 'geometry'],
      dtype='object', length=234)>
```

In [27]:
```
list(Entro_CensusTracts_geo.columns)
```

Out[27]:
```
['GEOID',
 'State',
 'County',
 'Urban',
 'POP2010',
 'OHU2010',
 'GroupQuartersFlag',
 'NUMGQTRS',
 'PCTGQTRS',
 'LILATracts_1And10',
 'LILATracts_halfAnd10',
 'LILATracts_1And20',
 'LILATracts_Vehicle',
 'HUNVFlag',
 'LowIncomeTracts',
 'PovertyRate',
 'MedianFamilyIncome',
 'LA1and10',
 'LAhalfand10',
 'LA1and20',
 'LATracts_half',
 'LATracts1',
 'LATracts10',
 'LATracts20',
 'LATractsVehicle_20',
 'LAPOP1_10',
 'LAPOP05_10',
 'LAPOP1_20',
 'LALOWI1_10',
 'LALOWI05_10',
 'LALOWI1_20',
 'lapophalf',
 'lapophalfshare',
 'lalowihalf',
 'lalowihalfshare',
 'lakidshalf',
 'lakidshalfshare',
 'laseniorshalf',
 'laseniorshalfshare',
 'lawhitehalf',
 'lawhitehalfshare',
 'lablackhalf',
 'lablackhalfshare',
 'laasianhalf',
 'laasianhalfshare',
 'lanhopihalf',
 'lanhopihalfshare',
 'laaianhalf',
 'laaianhalfshare',
```

```
'laomultirhalf',
'laomultirhalfshare',
'lahisphalf',
'lahisphalfshare',
'lahunvhalf',
'lahunvhalfshare',
'lasnaphalf',
'lasnaphalfshare',
'lapop1',
'lapop1share',
'lalowi1',
'lalowi1share',
'lakids1',
'lakids1share',
'laseniors1',
'laseniors1share',
'lawhite1',
'lawhite1share',
'lablack1',
'lablack1share',
'laasian1',
'laasian1share',
'lanhopi1',
'lanhopi1share',
'laaian1',
'laaian1share',
'laomultir1',
'laomultir1share',
'lahisp1',
'lahisp1share',
'lahunv1',
'lahunv1share',
'lasnap1',
'lasnap1share',
'lapop10',
'lapop10share',
'lalowi10',
'lalowi10share',
'lakids10',
'lakids10share',
'laseniors10',
'laseniors10share',
'lawhite10',
'lawhite10share',
'lablack10',
'lablack10share',
'laasian10',
'laasian10share',
'lanhopi10',
'lanhopi10share',
'laaian10',
'laaian10share',
'laomultir10',
'laomultir10share',
'lahisp10',
'lahisp10share',
'lahunv10',
'lahunv10share',
'lasnap10',
'lasnap10share',
'lapop20',
'lapop20share',
'lalowi20',
'lalowi20share',
'lakids20',
```

```
'lakids20share',
'laseniors20',
'laseniors20share',
'lawhite20',
'lawhite20share',
'lablack20',
'lablack20share',
'laasian20',
'laasian20share',
'lanhopi20',
'lanhopi20share',
'laaian20',
'laaian20share',
'laomultir20',
'laomultir20share',
'lahisp20',
'lahisp20share',
'lahunv20',
'lahunv20share',
'lasnap20',
'lasnap20share',
'TractLOWI',
'TractKids',
'TractSeniors',
'TractWhite_x',
'TractBlack_x',
'TractAsian_x',
'TractNHOPI_x',
'TractAIAN_x',
'TractOMultir_x',
'TractHispanic_x',
'TractHUNV',
'TractSNAP',
'TractWhite_y',
'TractBlack_y',
'TractAsian_y',
'TractNHOPI_y',
'TractAIAN_y',
'TractOMultir_y',
'TractHispanic_y',
'totalpop',
'prop_white',
'prop_black',
'prop_asian',
'prop_hisp',
'prop_nhopi',
'prop_aian',
'prop_multir',
'totalprop',
'entropy_index',
'Shape_Length',
'Shape_Area',
'OBJECTID',
'TRACT',
'GEOID',
'P0010001',
'P0010002',
'P0010003',
'P0010004',
'P0010005',
'P0010006',
'P0010007',
'P0010008',
'OP000001',
'OP000002',
```

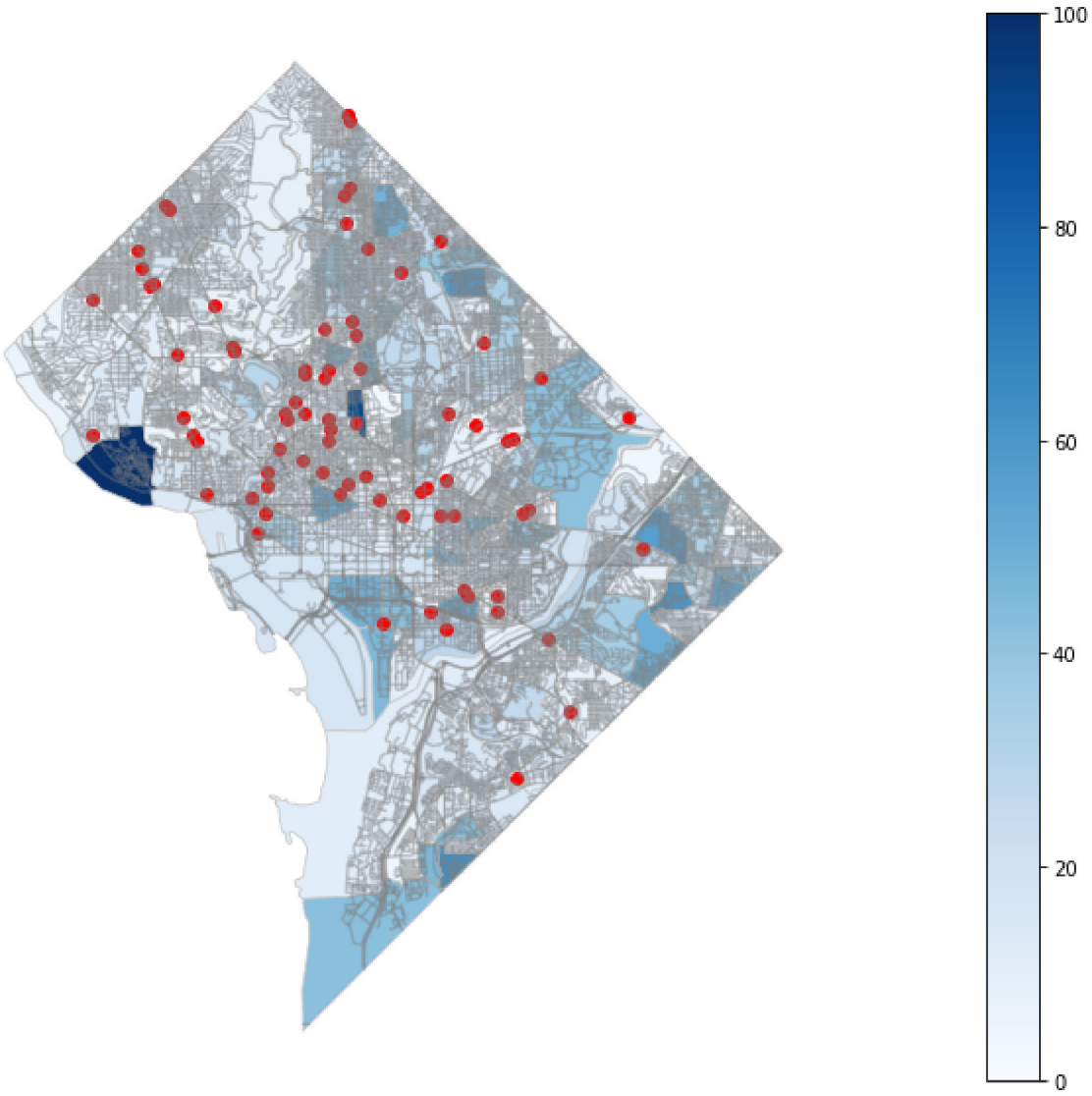
```
'OP000003',
'OP000004',
'P0020002',
'P0020005',
'P0020006',
'P0020007',
'P0020008',
'P0020009',
'P0020010',
'OP00005',
'OP00006',
'OP00007',
'OP00008',
'P0030001',
'P0030003',
'P0030004',
'P0030005',
'P0030006',
'P0030007',
'P0030008',
'OP00009',
'OP00010',
'OP00011',
'OP00012',
'P0040002',
'P0040005',
'P0040006',
'P0040007',
'P0040008',
'P0040009',
'P0040010',
'OP000013',
'OP000014',
'OP000015',
'OP000016',
'H0010001',
'H0010002',
'H0010003',
'ACRES',
'SQ_MILES',
'Shape_Leng',
'Shape_Area',
'FAGI_TOTAL',
'FAGI_MEDIA',
'FAGI_TOT_1',
'FAGI_MED_1',
'FAGI_TOT_2',
'FAGI_MED_2',
'FAGI_TOT_3',
'FAGI_MED_3',
'FAGI_TOT_4',
'FAGI_MED_4',
'FAGI_TOT_5',
'FAGI_MED_5',
'geometry']
```

In [28]:

```
variable= 'PovertyRate'
vmin, vmax = 0, 100
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin, vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgeco
```

```
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')
```

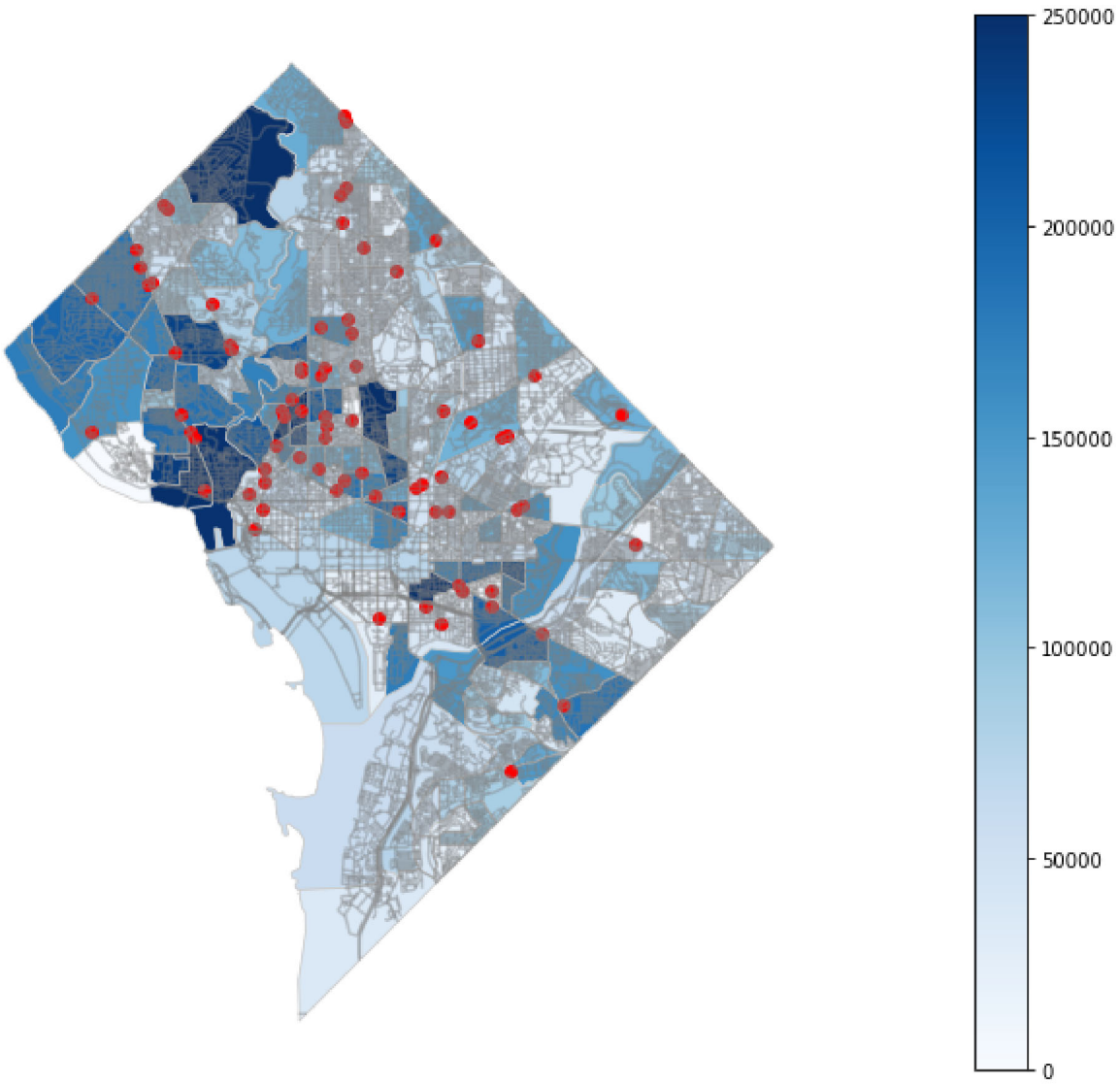
Out[28]: <AxesSubplot:>



In [29]:

```
variable= 'MedianFamilyIncome'
vmin, vmax = 0, 250000
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin, vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgeco
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')
```

Out[29]: <AxesSubplot:>

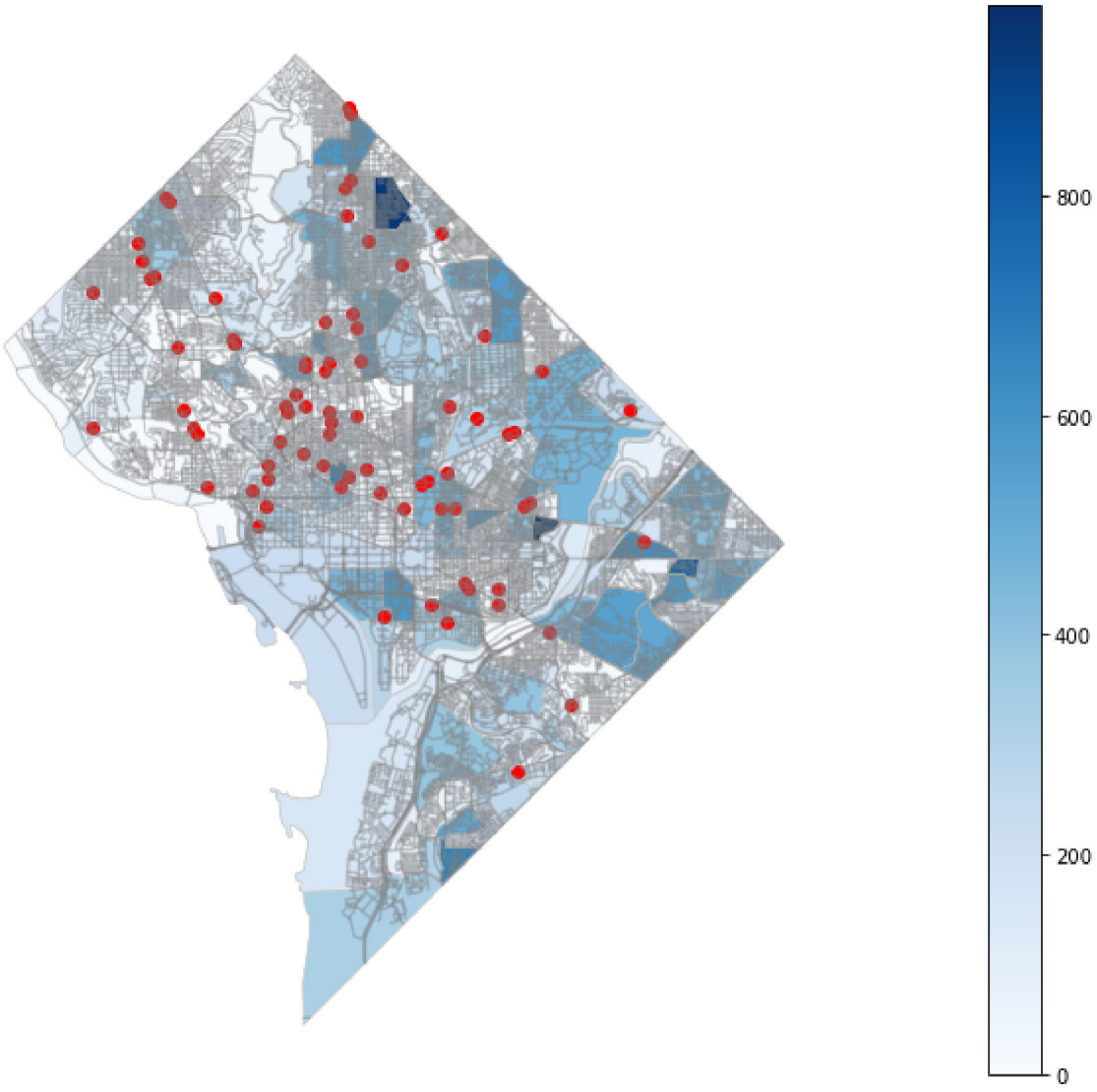


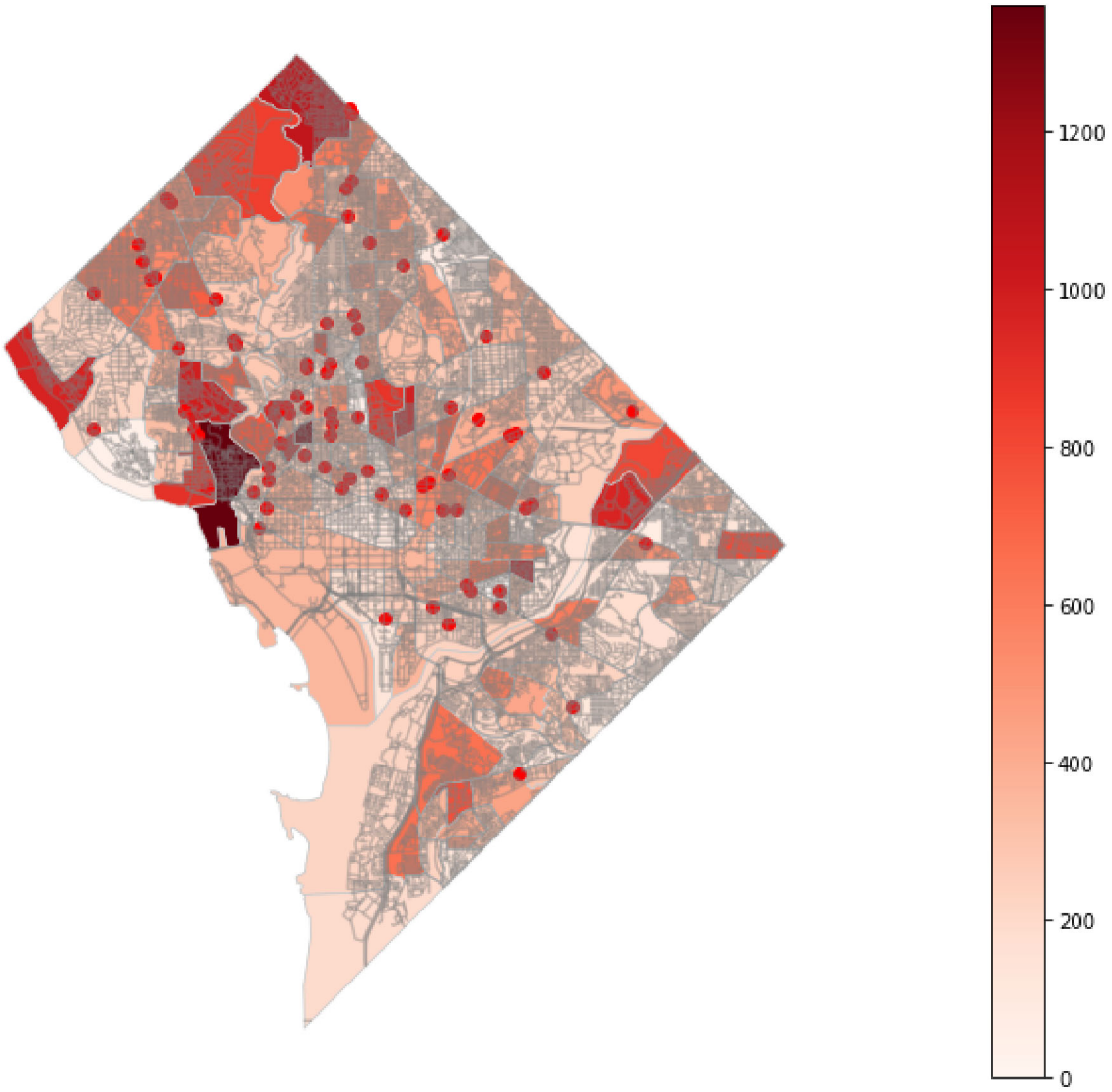
In [30]:

```
variable= 'TractSNAP'
vmin, vmax = 0, 973
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin, vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgeco
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')


va= 'TractSeniors'
vmin1, vmax1 = 0, 1359
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Reds', norm=plt.Normalize(vmin=vmin1, vmax=vmax1))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=va, cmap='Reds', linewidth=0.8, ax=ax, edgecolor='0.
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')
```

Out[30]: <AxesSubplot:>





In [31]:

```
va= 'TractSeniors'
vmin, vmax = 0, 1359
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Reds', norm=plt.Normalize(vmin=vmin2, vmax=vmax2))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=va, cmap='Reds', linewidth=0.8, ax=ax, edgecolor='0.
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-fe3b903d1f5c> in <module>
      3 fig, ax = plt.subplots(1, figsize=(30, 10))
      4 ax.axis('off')
----> 5 sm = plt.cm.ScalarMappable(cmap='Reds', norm=plt.Normalize(vmin=vmin2, vmax=vma
x2))
      6 sm.set_array([])
      7 fig.colorbar(sm)

NameError: name 'vmin2' is not defined
```

In [ ]:
```
variable= 'entropy_index'
vmin, vmax = -2, 2
fig, ax = plt.subplots(1, figsize=(30, 10))
ax.axis('off')
sm = plt.cm.ScalarMappable(cmap='Blues', norm=plt.Normalize(vmin=vmin, vmax=vmax))
sm.set_array([])
fig.colorbar(sm)
Entro_CensusTracts_geo.plot(column=variable, cmap='Blues', linewidth=0.8, ax=ax, edgeco
Street_Map.plot(ax=ax, alpha=0.4, color='grey')
stores.plot(ax=ax, color='red', marker='o')

One_2.plot(ax=ax, alpha=0.4, color='yellow')
# zero.plot(ax=ax, alpha=0.4, color='yellow')
```

More Mapping

In [ ]:
```
CRS=roads.crs
stores=stores.to_crs(crs=CRS)
```

In [ ]:
```
CRS=roads.crs
stores=stores.to_crs(crs=CRS)
```

In [ ]:
```
output_notebook()
```

In [ ]:
```
roads= gpd.read_file('Roadway_SubBlock.shp')
Census_tracts=gpd.read_file('Census_Tracts_in_2010.shp')
stores=gpd.read_file('Grocery_Store_Locations.shp')
Entro_census=gpd.read_file('data/Entro_CensusTracts.shp')
Food_stores=gpd.read_file('data/FoodStore_within_Census_tracts.shp')
```

In [ ]:
```
def getPointCoords(row, geom, coord_type):
    """Calculates coordinates ('x' or 'y') of a Point geometry"""
    if coord_type == 'x':
        return row[geom].x
    elif coord_type == 'y':
        return row[geom].y
```

In [ ]:
```
Food_stores['x'] = Food_stores.apply(getPointCoords, geom='geometry', coord_type='x', a
```

In [ ]:
```
Food_stores['y'] = Food_stores.apply(getPointCoords, geom='geometry', coord_type='y', a
```

In [ ]:
```
# Food_stores.head()
```

In [ ]:
```
def getPolyCoords(row, geom, coord_type):
    exterior = row[geom].exterior

    if coord_type == 'x':
        # Get the x coordinates of the exterior
        return list( exterior.coords.xy[0] )
    elif coord_type == 'y':
        # Get the y coordinates of the exterior
        return list( exterior.coords.xy[1] )
```

In [ ]:
```
Entro_CensusTracts_geo['x']=Entro_CensusTracts_geo.apply(getPolyCoords, geom='geometry'
Entro_CensusTracts_geo['y']=Entro_CensusTracts_geo.apply(getPolyCoords, geom='geometry'
```

In [ ]:
```
CRS=roads.crs
stores=stores.to_crs(crs=CRS)
```

In [ ]:
```
Entro_CensusTracts_geo.head(5)
```

In [ ]:
```
Food_stores_df = Food_stores.drop('geometry', axis=1).copy()
Entro_CensusTracts_geo_df=Entro_CensusTracts_geo.drop('geometry', axis=1).copy()
```

In [ ]:
```
Entro_CensusTracts_geo_df.head(5)
```

In [ ]:
```
from bokeh.models import ColumnDataSource
```

In [ ]:
```
psource = ColumnDataSource(Food_stores_df)
gsource=ColumnDataSource(Entro_CensusTracts_geo_df)
```

In [ ]:
```
color_mapper = LogColorMapper(palette="Viridis256")
```

In [ ]:
```
my_hover=HoverTool()
my_hover.tooltips = [('entropy index','@entropy_index')]
```

In [ ]:
```
p = figure(title="test")

p.patches('x','y', source=gsource,
          fill_color={'field':'entropy_index','transform':color_mapper},
```

```
        fill_alpha=1.0, line_color="black", line_width=1)
p.circle('x', 'y', source=psource, color='red', size=10)
p.add_tools(my_hover)
p.legend.location='top_left'
color_bar = ColorBar(color_mapper = color_mapper,
                     label_standoff = 8,
                     width = 500, height = 20,
                     border_line_color = None,
                     location = (0,0),
                     orientation = 'horizontal')
p.add_layout(color_bar, 'below')

show(p)
```

In [ ]:

```
Entro_census.head(5)
```

In [ ]: