Notebook 6: A Series of Comparison Maps for Key Metrics

In this section, we reselect many datasets in demographics, housing, and economy and combine them into one CSV so that we can use def function coding to create graph more efficiently. Then, we focus on making a function code so that we don't have to repeatedly make the mapps.

Section 0. Import All Modules and Set Up Notebook

In [1]:
```python
# Import all modules I will be using in this note book.

import pandas as pd
import geopandas as gpd
import contextily as ctx
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import numpy as np
```

```
/opt/conda/lib/python3.8/site-packages/geopandas/_compat.py:106: UserWarning: The Shapely GEOS version (3.8.1-CAPI-1.13.3) is i
ncompatible with the GEOS version PyGEOS was compiled with (3.9.0-CAPI-1.16.2). Conversions between both will be slow.
  warnings.warn(
```

Section 1. Prepare Basic Geo-Data

In this section, I will clean and prepare basic geo dataset for future use in this notebook. I will work with both SHP file and CSV file to create a list of county in the US with all geo information. Those data are used to be matched with census data and then map the findings.

In [2]:
```python
# Import the raw data that contains geo information. It is a SHP file.

countyborder = gpd.read_file('GeoData/02_Basemap_countyborder/cb_2018_us_county_500k.shp')
```

In [3]:
```python
# I want to take a look what it looks like.

countyborder.head()
```

Out[3]:

	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	GEOID	NAME	LSAD	ALAND	AWATER	geometry
0	21	007	00516850	0500000US21007	21007	Ballard	06	639387454	69473325	POLYGON ((-89.18137 37.04630, -89.17938 37.053...
1	21	017	00516855	0500000US21017	21017	Bourbon	06	750439351	4829777	POLYGON ((-84.44266 38.28324, -84.44114 38.283...
2	21	031	00516862	0500000US21031	21031	Butler	06	1103571974	13943044	POLYGON ((-86.94486 37.07341, -86.94346 37.074...
3	21	065	00516879	0500000US21065	21065	Estill	06	655509930	6516335	POLYGON ((-84.12662 37.64540, -84.12483 37.646...
4	21	069	00516881	0500000US21069	21069	Fleming	06	902727151	7182793	POLYGON ((-83.98428 38.44549, -83.98246 38.450...

In [4]:
```python
# Clean out the dataset by keeping the columns I need.

columns_to_keep4 = ['GEOID','geometry','NAME','STATEFP']
countyborder_trimmed1 = countyborder [columns_to_keep4]
countyborder_trimmed1.head()
```

Out[4]:

	GEOID	geometry	NAME	STATEFP
0	21007	POLYGON ((-89.18137 37.04630, -89.17938 37.053...	Ballard	21
1	21017	POLYGON ((-84.44266 38.28324, -84.44114 38.283...	Bourbon	21
2	21031	POLYGON ((-86.94486 37.07341, -86.94346 37.074...	Butler	21
3	21065	POLYGON ((-84.12662 37.64540, -84.12483 37.646...	Estill	21
4	21069	POLYGON ((-83.98428 38.44549, -83.98246 38.450...	Fleming	21

In [5]:
```python
# The geo data above misses the state name.
# So, I will import anly CSV data that contains the state info with the identifiers (STATEFP).

state_name = pd.read_csv('GeoData/07_Basemap_State_FIPS.csv',dtype={'STATEFP':str})
state_name.head(5)
```

Out[5]:

	STATEFP	Name
0	00	Northeast Region
1	00	New England Division
2	09	Connecticut
3	23	Maine
4	25	Massachusetts

In [6]:
```python
# I will merge those two geo dataset toeghter according to "STATEFP", the shared identifers

countyborder_trimmed2 = countyborder_trimmed1.merge(state_name,on ='STATEFP',how='left')

countyborder_trimmed2.head()
```

Out[6]:

	GEOID	geometry	NAME	STATEFP	Name
0	21007	POLYGON ((-89.18137 37.04630, -89.17938 37.053...	Ballard	21	Kentucky
1	21017	POLYGON ((-84.44266 38.28324, -84.44114 38.283...	Bourbon	21	Kentucky
2	21031	POLYGON ((-86.94486 37.07341, -86.94346 37.074...	Butler	21	Kentucky
3	21065	POLYGON ((-84.12662 37.64540, -84.12483 37.646...	Estill	21	Kentucky
4	21069	POLYGON ((-83.98428 38.44549, -83.98246 38.450...	Fleming	21	Kentucky

In [7]:
```python
# For better viewing, I create a new column that contains both the county name column and the state name column

countyborder_trimmed2 ['County_Name'] = countyborder_trimmed2['NAME'] + ','+' '+countyborder_trimmed2['Name']

countyborder_trimmed2 = countyborder_trimmed2.drop (['NAME','Name'],axis=1)
countyborder_trimmed2.head()
```

Out[7]:

	GEOID	geometry	STATEFP	County_Name
0	21007	POLYGON ((-89.18137 37.04630, -89.17938 37.053...	21	Ballard, Kentucky
1	21017	POLYGON ((-84.44266 38.28324, -84.44114 38.283...	21	Bourbon, Kentucky
2	21031	POLYGON ((-86.94486 37.07341, -86.94346 37.074...	21	Butler, Kentucky
3	21065	POLYGON ((-84.12662 37.64540, -84.12483 37.646...	21	Estill, Kentucky
4	21069	POLYGON ((-83.98428 38.44549, -83.98246 38.450...	21	Fleming, Kentucky

In [8]:
```python
countyborder_trimmed2['Region'] = 'Non_Metro_the_contiguous_US'
```

In [9]:
```python
countyborder_trimmed2.head()
```

Out[9]:

	GEOID	geometry	STATEFP	County_Name	Region
0	21007	POLYGON ((-89.18137 37.04630, -89.17938 37.053...	21	Ballard, Kentucky	Non_Metro_the_contiguous_US
1	21017	POLYGON ((-84.44266 38.28324, -84.44114 38.283...	21	Bourbon, Kentucky	Non_Metro_the_contiguous_US
2	21031	POLYGON ((-86.94486 37.07341, -86.94346 37.074...	21	Butler, Kentucky	Non_Metro_the_contiguous_US
3	21065	POLYGON ((-84.12662 37.64540, -84.12483 37.646...	21	Estill, Kentucky	Non_Metro_the_contiguous_US
4	21069	POLYGON ((-83.98428 38.44549, -83.98246 38.450...	21	Fleming, Kentucky	Non_Metro_the_contiguous_US

In [10]:
```python
NYC_5County = ['36005','36047','36061','36081','36085']
NonNYC_Metro = ['09001','09005','09009','34003','34013','34017','34019','34021','34023',
                '34025','34027','34029','34031','34035','34037','34039','36027','36059',
                '36071','36079','36087','36103','36111','36119','42089','42103']
NonContiguous = ['72','02','15','66','69','78','60']
```

In [11]:
```python
def regionbyGEOID_NYC(name):
    countyborder_trimmed2.loc[countyborder_trimmed2['GEOID'] == name,'Region'] = 'NYC'

def regionbyGEOID_NonNYC_Metro(name):
    countyborder_trimmed2.loc[countyborder_trimmed2['GEOID'] == name,'Region'] = 'NonNYC_Metro'

def regionbyGEOID_NonContiguous(name):
    countyborder_trimmed2.loc[countyborder_trimmed2['STATEFP'] == name,'Region'] = 'Non_the_contiguous_US'
```

In [12]:
```python
for GEOID in NYC_5County:
    regionbyGEOID_NYC(GEOID)

NYC_5county = countyborder_trimmed2[countyborder_trimmed2.Region == 'NYC']
NYC_5county
```

Out[12]:

	GEOID	geometry	STATEFP	County_Name	Region
165	36047	POLYGON ((-74.04201 40.62605, -74.04199 40.626...	36	Kings, New York	NYC
169	36081	POLYGON ((-73.96262 40.73903, -73.96138 40.742...	36	Queens, New York	NYC
989	36061	MULTIPOLYGON (((-73.99950 40.70033, -73.99750 ...	36	New York, New York	NYC
2217	36085	MULTIPOLYGON (((-74.16170 40.64586, -74.16060 ...	36	Richmond, New York	NYC
2834	36005	MULTIPOLYGON (((-73.77336 40.85945, -73.77244 ...	36	Bronx, New York	NYC

In [13]:
```python
for GEOID in NonNYC_Metro:
    regionbyGEOID_NonNYC_Metro(GEOID)

NonNYC_Metro = countyborder_trimmed2[countyborder_trimmed2.Region == 'NonNYC_Metro']
NonNYC_Metro.head()
```

Out[13]:

	GEOID	geometry	STATEFP	County_Name	Region
56	09009	MULTIPOLYGON (((-72.76143 41.24233, -72.75973 ...	09	New Haven, Connecticut	NonNYC_Metro
153	34003	POLYGON ((-74.27066 41.02103, -74.25046 41.060...	34	Bergen, New Jersey	NonNYC_Metro
155	34013	POLYGON ((-74.37623 40.76275, -74.37389 40.762...	34	Essex, New Jersey	NonNYC_Metro
156	34023	POLYGON ((-74.63023 40.34313, -74.63047 40.344...	34	Middlesex, New Jersey	NonNYC_Metro
445	34019	POLYGON ((-75.19511 40.57969, -75.19466 40.581...	34	Hunterdon, New Jersey	NonNYC_Metro

In [14]:
```python
for GEOID in NonContiguous:
    regionbyGEOID_NonContiguous(GEOID)

NonCountiguous = countyborder_trimmed2[countyborder_trimmed2.Region == 'Non_the_contiguous_US']
NonCountiguous.head()
```

Out[14]:

	GEOID	geometry	STATEFP	County_Name	Region
26	02016	MULTIPOLYGON (((179.48246 51.98283, 179.48656 ...	02	Aleutians West, Alaska	Non_the_contiguous_US
27	02130	MULTIPOLYGON (((-130.98311 55.36598, -130.9809...	02	Ketchikan Gateway, Alaska	Non_the_contiguous_US
28	02180	MULTIPOLYGON (((-161.31946 64.12363, -161.3183...	02	Nome, Alaska	Non_the_contiguous_US
29	02282	MULTIPOLYGON (((-139.51201 59.70289, -139.5095...	02	Yakutat, Alaska	Non_the_contiguous_US
86	15007	MULTIPOLYGON (((-159.78794 22.03010, -159.7864...	15	Kauai, Hawaii	Non_the_contiguous_US

The Follwing Dataset is Ready: List of All US Counties with Geo Info:

In [15]:
```python
# I don't need "STATEFP" and "CountyName" column anymore. Now I'm gonna drop it for clearning.

county_geodata_ready = countyborder_trimmed2.drop(['STATEFP','County_Name'],axis=1)

county_geodata_ready.head()
```

Out[15]:

	GEOID	geometry	Region
0	21007	POLYGON ((-89.18137 37.04630, -89.17938 37.053...	Non_Metro_the_contiguous_US
1	21017	POLYGON ((-84.44266 38.28324, -84.44114 38.283...	Non_Metro_the_contiguous_US
2	21031	POLYGON ((-86.94486 37.07341, -86.94346 37.074...	Non_Metro_the_contiguous_US
3	21065	POLYGON ((-84.12662 37.64540, -84.12483 37.646...	Non_Metro_the_contiguous_US
4	21069	POLYGON ((-83.98428 38.44549, -83.98246 38.450...	Non_Metro_the_contiguous_US

The Following Dataset is Ready: List of NYC Metro Counties with Geo Info:

In [16]:
```python
# I want to create a new dataframe that only contains the geo data for NYC_Metro.
# This is especially important when I am going to map out the findings just for NYC_Metro.

NonNYC_Metro_geodata_ready = county_geodata_ready[county_geodata_ready.Region == 'NonNYC_Metro']

NonNYC_Metro_geodata_ready = NonNYC_Metro_geodata_ready.reset_index(drop=True)

NonNYC_Metro_geodata_ready
```

Out[16]:

	GEOID	geometry	Region
0	09009	MULTIPOLYGON (((-72.76143 41.24233, -72.75973 ...	NonNYC_Metro
1	34003	POLYGON ((-74.27066 41.02103, -74.25046 41.060...	NonNYC_Metro
2	34013	POLYGON ((-74.37623 40.76275, -74.37389 40.762...	NonNYC_Metro
3	34023	POLYGON ((-74.63023 40.34313, -74.63047 40.344...	NonNYC_Metro
4	34019	POLYGON ((-75.19511 40.57969, -75.19466 40.581...	NonNYC_Metro
5	34021	POLYGON ((-74.94228 40.34089, -74.93228 40.339...	NonNYC_Metro
6	34025	POLYGON ((-74.61458 40.18238, -74.59963 40.186...	NonNYC_Metro
7	34029	POLYGON ((-74.55311 40.07913, -74.53347 40.087...	NonNYC_Metro
8	34035	POLYGON ((-74.79582 40.51527, -74.78903 40.512...	NonNYC_Metro
9	36103	MULTIPOLYGON (((-72.03683 41.24984, -72.03496 ...	NonNYC_Metro
10	36119	MULTIPOLYGON (((-73.77278 40.88460, -73.77231 ...	NonNYC_Metro
11	42103	POLYGON ((-75.35564 41.24112, -75.35050 41.244...	NonNYC_Metro

Section 2. Analyze Economic Changes between 2014 and 2018

In this section, I will be processing the economic changes between the two years. There are three economic metrics I will be using: GDP, Job Number, and Income. The data process is same to each of them. I will use CSV data and later paired with geo data.

In [17]:
```python
County_Factors_Raw = pd.read_csv('Data/2014vs2018.csv',
                                  dtype={'id':str})
County_Factors_Raw.head()
```

Out[17]:

	id	Geographic Area Name	Outflow_2018	Inflow_2018	Commuter_2014	Commuter_2018	Population_2014	Population_2018	Age_2014	Age_2018	...	H
0	0500000US34003	Bergen County, New Jersey	20899	21226	10216	13888	920456	929999	41.4	41.8	...	
1	0500000US36027	Dutchess County, New York	5118	9964	4555	5408	297388	293894	40.8	42.0	...	
2	0500000US34013	Essex County, New Jersey	20566	19458	18249	24738	789616	793555	36.7	37.5	...	
3	0500000US09001	Fairfield County, Connecticut	15281	19508	28070	33893	934215	944348	39.6	40.3	...	
4	0500000US34017	Hudson County, New Jersey	25448	20962	7691	8982	654878	668631	34.5	35.1	...	

In [18]:
```python
County_Factors_Raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       26 non-null     object
 1   Geographic Area Name     26 non-null     object
 2   Outflow_2018             26 non-null     int64
 3   Inflow_2018              26 non-null     int64
 4   Commuter_2014            26 non-null     int64
 5   Commuter_2018            26 non-null     int64
 6   Population_2014          26 non-null     int64
 7   Population_2018          26 non-null     int64
 8   Age_2014                 26 non-null     float64
 9   Age_2018                 26 non-null     float64
 10  Education_2014           26 non-null     int64
 11  Education_2018           26 non-null     int64
 12  Homeworker_2014          26 non-null     int64
 13  Homeworker_2018          26 non-null     int64
 14  GDP_2014                 26 non-null     int64
 15  GDP_2018                 26 non-null     int64
 16  Job_2014                 26 non-null     int64
 17  Job_2018                 26 non-null     int64
 18  Income_2014              26 non-null     int64
 19  Income_2018              26 non-null     int64
 20  OwnedUnits_2014          26 non-null     int64
 21  OwnedUnits_2018          26 non-null     int64
 22  HouseValue_2014          26 non-null     int64
 23  HouseValue_2018          26 non-null     int64
 24  Rent_2014                26 non-null     int64
 25  Rent_2018                26 non-null     int64
 26  RentalUnits_2014         26 non-null     int64
 27  RentalUnits_2018         26 non-null     int64
 28  OwnedAffordability_2014  26 non-null     float64
 29  OwnedAffordability_2018  26 non-null     float64
 30  RentalAffordability_2014  26 non-null     float64
 31  RentalAffordability_2018  26 non-null     float64
dtypes: float64(6), int64(24), object(2)
memory usage: 6.6+ KB
```

In [19]:
```python
County_Factors_Raw ['id'] = County_Factors_Raw['id'].str.strip().str[-5:]
```

In [20]:
```python
County_Factors_Raw.rename(columns={'id':'GEOID','Geographic Area Name':'County'},
                                   inplace=True)
```

In [21]:
```python
County_Factors_Raw.head()
```

Out[21]:

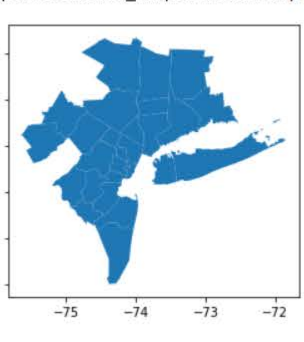
	GEOID	County	Outflow_2018	Inflow_2018	Commuter_2014	Commuter_2018	Population_2014	Population_2018	Age_2014	Age_2018	...	HouseValue_
0	34003	Bergen County, New Jersey	20899	21226	10216	13888	920456	929999	41.4	41.8	...	443
1	36027	Dutchess County, New York	5118	9964	4555	5408	297388	293894	40.8	42.0	...	282
2	34013	Essex County, New Jersey	20566	19458	18249	24738	789616	793555	36.7	37.5	...	358
3	09001	Fairfield County, Connecticut	15281	19508	28070	33893	934215	944348	39.6	40.3	...	422
4	34017	Hudson County, New Jersey	25448	20962	7691	8982	654878	668631	34.5	35.1	...	337

5 rows × 32 columns

In [22]:
```python
County_Factors_Merge = NonNYC_Metro_geodata_ready.merge(County_Factors_Raw,
                                                        on='GEOID',
                                                        how='left')
```

In [23]:
```python
County_Factors_Merge.plot()
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb10c052640>



In [24]:
```python
Factory_Analysis1 = County_Factors_Merge
```

In [25]:
```python
list(Factory_Analysis1)
```

Out[25]:
```
['GEOID',
 'geometry',
 'Region',
 'County',
 'Outflow_2018',
 'Inflow_2018',
 'Commuter_2014',
 'Commuter_2018',
 'Population_2014',
 'Population_2018',
 'Age_2014',
 'Age_2018',
 'Education_2014',
 'Education_2018',
 'Homeworker_2014',
 'Homeworker_2018',
 'GDP_2014',
 'GDP_2018',
 'Job_2014',
 'Job_2018',
 'Income_2014',
 'Income_2018',
 'OwnedUnits_2014',
 'OwnedUnits_2018',
 'HouseValue_2014',
 'HouseValue_2018',
 'Rent_2014',
 'Rent_2018',
 'RentalUnits_2014',
 'RentalUnits_2018',
 'OwnedAffordability_2014',
 'OwnedAffordability_2018',
 'RentalAffordability_2014',
 'RentalAffordability_2018']
```

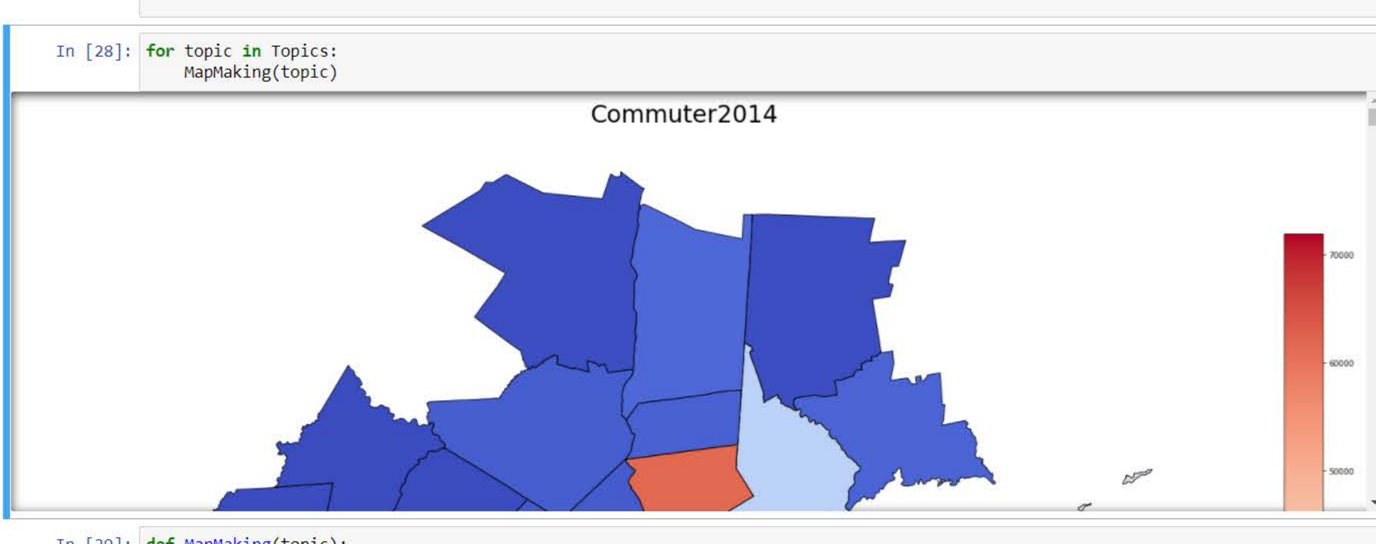
In [ ]:

In [ ]:

In [26]:
```python
def MapMaking(topic):

    fig, axs = plt.subplots(1, 1, figsize=(30, 60))
    ax1= axs

    Factory_Analysis1.plot(ax=ax1,
                           cmap='coolwarm',
                           column = topic+"_2014",
                           legend=True,
                           legend_kwds={'shrink': 0.3}
                                 )

    county_geodata_ready[county_geodata_ready.Region == 'NonNYC_Metro'].plot(ax=ax1,
                                                                  facecolor='none',
                                                                  edgecolor='black',
                                                                   lw=1,
                                                                   alpha=1)

    county_geodata_ready[county_geodata_ready.Region == 'NYC'].plot(ax=ax1,
                                                                  facecolor='none',
                                                                  edgecolor='black',
                                                                   lw=1,
                                                                   alpha=1)


    ax1.axis("off")
    ax1.set_title(topic+""+"2014",fontsize = 30)
```

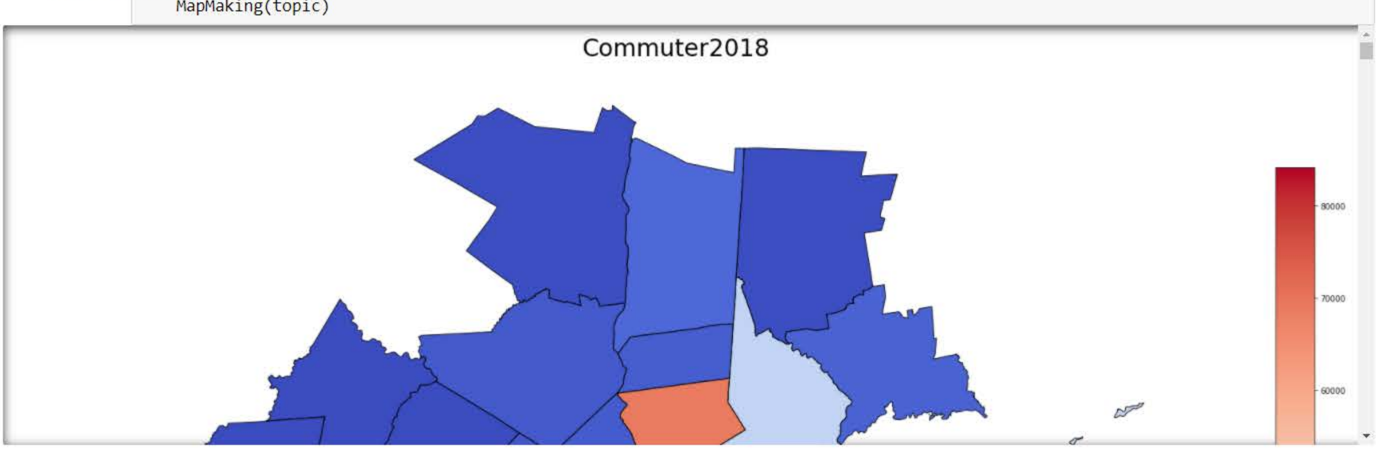
In [27]:
```python
Topics = [
 'Commuter',
 'Population',
 'Age',
 'Education',
 'Homeworker',
 'GDP',
 'Job',
 'Income',
 'OwnedUnits',
 'HouseValue',
 'Rent',
 'RentalUnits',
 'OwnedAffordability',
 'RentalAffordability']
```

In [28]:
```python
for topic in Topics:
    MapMaking(topic)
```



In [29]:
```python
def MapMaking(topic):

    fig, axs = plt.subplots(1, 1, figsize=(30, 60))
    ax1= axs

    Factory_Analysis1.plot(ax=ax1,
                           cmap='coolwarm',
                           column = topic+"_2018",
                           legend=True,
                           legend_kwds={'shrink': 0.3}
                                 )

    county_geodata_ready[county_geodata_ready.Region == 'NonNYC_Metro'].plot(ax=ax1,
                                                                  facecolor='none',
                                                                  edgecolor='black',
                                                                   lw=1,
                                                                   alpha=1)

    county_geodata_ready[county_geodata_ready.Region == 'NYC'].plot(ax=ax1,
                                                                  facecolor='none',
                                                                  edgecolor='black',
                                                                   lw=1,
                                                                   alpha=1)


    ax1.axis("off")
    ax1.set_title(topic+""+"2018",fontsize = 30)
```

In [30]:
```python
for topic in Topics:
    MapMaking(topic)
```



Section 3. Conclusion

We used the function coding to be able to create lots of maps in a short time. This function code is really helping in increase efficiency. In addition, by using the same function for all topics, the graphs look very coherent.

In [ ]: