In [1]:
```
import pandas as pd
import numpy as np
import altair as alt
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols


import os
os.environ['USE_PYGEOS'] = '0'
import geopandas as gpd
from shapely.geometry import Point, box
import pysal
from pysal.lib import weights
from pysal.model import spreg
import libpysal
from geopandas.tools import sjoin
from geopy.distance import geodesic

#RUNTIME AROUND 38 minutes on full data
```

In [2]:
```
df_costco = pd.read_excel('data/Costco-july-2023-filled.xlsx')

#single family homes
df_zip = pd.read_csv('data/Zip_zhvi_uc_sfr_tier_0.33_0.67_sm_sa_month.csv')

df_match = pd.read_csv('data/uszips.csv')
```

In [3]:
```
df_costco_clean = df_costco
df_costco_clean['Opening Date'] = pd.to_datetime(df_costco_clean['Opening Date'], format='%m/%d/%Y')

# Extract Year, Month, and Day into separate columns
df_costco_clean['Year'] = df_costco_clean['Opening Date'].dt.year
df_costco_clean['Month'] = df_costco_clean['Opening Date'].dt.month
df_costco_clean['Day'] = df_costco_clean['Opening Date'].dt.day

df_costco_clean['Date'] = pd.to_datetime(df_costco_clean[['Year', 'Month', 'Day']])

# Sort the DataFrame based on the 'Date' column
df_costco_sorted = df_costco_clean.sort_values(by='Date')

df_costco_2000_2023 = df_costco_sorted[(df_costco_sorted['Date'] >= '2000-01-31') & (df_costco_sorted['Date'] <= '2023-09-30')]

df_costco_2000_2023 = df_costco_2000_2023[['State', 'City', 'Store Name', 'Zipcode', 'Latitude', 'Longitude', 'Opening Date']]

new_colnames = {'Store Name': 'store_name',
                    'Zipcode': 'zipcode',
                    'Latitude': 'latitude',
                    'Longitude': 'longitude',
                    'Opening Date': 'opening_date'}

df_costco_2000_2023.rename(columns=new_colnames, inplace=True)

df_costco_2000_2023
```

Out[3]:

	State	City	store_name	zipcode	latitude	longitude	opening_date
357	OH	Cincinnati	Deerfield	45249-8220	39.296	-84.303	2000-02-16
268	IL	Niles	Niles	60714-3905	42.015	-87.780	2000-06-01
119	UT	Sandy	Sandy	84070-4171	40.550	-111.895	2000-06-02
250	AZ	Phoenix	N Phoenix	85027-5036	33.660	-112.116	2000-06-22
409	CA	San Diego	Mission Valley	92108-4743	32.782	-117.129	2000-06-29
...	...	...	...	...	...	...	...
345	CO	Longmont	Longmont	80501	40.152	-105.085	2023-05-04
355	OK	Tulsa	North Tulsa	74116	36.223	-95.848	2023-05-25
14	AZ	Buckeye	Buckeye	85326	33.460	-112.501	2023-07-13
364	TX	Georgetown	Georgetown	78628	30.677	-97.664	2023-07-14
472	CO	Denver	NE Denver	80239	39.781	-104.796	2023-07-19

365 rows × 7 columns

In [4]:
```
df_merged = pd.merge(df_zip, df_match, how='left', left_on='RegionName', right_on='zip')

columns_to_select = ['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName', 'State', 'City', 'Metro', 'CountyName', '2000-01-31',
                     '2000-02-29', '2000-03-31', '2000-04-30', '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31', '2000-09-30',
                     '2000-10-31', '2000-11-30', '2000-12-31', '2001-01-31', '2001-02-28', '2001-03-31', '2001-04-30', '2001-05-31',
                     '2001-06-30', '2001-07-31', '2001-08-31', '2001-09-30', '2001-10-31', '2001-11-30', '2001-12-31', '2002-01-31',
                     '2002-02-28', '2002-03-31', '2002-04-30', '2002-05-31', '2002-06-30', '2002-07-31', '2002-08-31', '2002-09-30',
                     '2002-10-31', '2002-11-30', '2002-12-31', '2003-01-31', '2003-02-28', '2003-03-31', '2003-04-30', '2003-05-31',
                     '2003-06-30', '2003-07-31', '2003-08-31', '2003-09-30', '2003-10-31', '2003-11-30', '2003-12-31', '2004-01-31',
                     '2004-02-29', '2004-03-31', '2004-04-30', '2004-05-31', '2004-06-30', '2004-07-31', '2004-08-31', '2004-09-30',
                     '2004-10-31', '2004-11-30', '2004-12-31', '2005-01-31', '2005-02-28', '2005-03-31', '2005-04-30', '2005-05-31',
                     '2005-06-30', '2005-07-31', '2005-08-31', '2005-09-30', '2005-10-31', '2005-11-30', '2005-12-31', '2006-01-31',
                     '2006-02-28', '2006-03-31', '2006-04-30', '2006-05-31', '2006-06-30', '2006-07-31', '2006-08-31', '2006-09-30',
                     '2006-10-31', '2006-11-30', '2006-12-31', '2007-01-31', '2007-02-28', '2007-03-31', '2007-04-30', '2007-05-31',
                     '2007-06-30', '2007-07-31', '2007-08-31', '2007-09-30', '2007-10-31', '2007-11-30', '2007-12-31', '2008-01-31',
                     '2008-02-29', '2008-03-31', '2008-04-30', '2008-05-31', '2008-06-30', '2008-07-31', '2008-08-31', '2008-09-30',
                     '2008-10-31', '2008-11-30', '2008-12-31', '2009-01-31', '2009-02-28', '2009-03-31', '2009-04-30', '2009-05-31',
                     '2009-06-30', '2009-07-31', '2009-08-31', '2009-09-30', '2009-10-31', '2009-11-30', '2009-12-31', '2010-01-31',
                     '2010-02-28', '2010-03-31', '2010-04-30', '2010-05-31', '2010-06-30', '2010-07-31', '2010-08-31', '2010-09-30',
                     '2010-10-31', '2010-11-30', '2010-12-31', '2011-01-31', '2011-02-28', '2011-03-31', '2011-04-30', '2011-05-31',
                     '2011-06-30', '2011-07-31', '2011-08-31', '2011-09-30', '2011-10-31', '2011-11-30', '2011-12-31', '2012-01-31',
                     '2012-02-29', '2012-03-31', '2012-04-30', '2012-05-31', '2012-06-30', '2012-07-31', '2012-08-31', '2012-09-30',
                     '2012-10-31', '2012-11-30', '2012-12-31', '2013-01-31', '2013-02-28', '2013-03-31', '2013-04-30', '2013-05-31',
                     '2013-06-30', '2013-07-31', '2013-08-31', '2013-09-30', '2013-10-31', '2013-11-30', '2013-12-31', '2014-01-31',
                     '2014-02-28', '2014-03-31', '2014-04-30', '2014-05-31', '2014-06-30', '2014-07-31', '2014-08-31', '2014-09-30',
                     '2014-10-31', '2014-11-30', '2014-12-31', '2015-01-31', '2015-02-28', '2015-03-31', '2015-04-30', '2015-05-31',
                     '2015-06-30', '2015-07-31', '2015-08-31', '2015-09-30', '2015-10-31', '2015-11-30', '2015-12-31', '2016-01-31',
                     '2016-02-29', '2016-03-31', '2016-04-30', '2016-05-31', '2016-06-30', '2016-07-31', '2016-08-31', '2016-09-30',
                     '2016-10-31', '2016-11-30', '2016-12-31', '2017-01-31', '2017-02-28', '2017-03-31', '2017-04-30', '2017-05-31',
                     '2017-06-30', '2017-07-31', '2017-08-31', '2017-09-30', '2017-10-31', '2017-11-30', '2017-12-31', '2018-01-31',
                     '2018-02-28', '2018-03-31', '2018-04-30', '2018-05-31', '2018-06-30', '2018-07-31', '2018-08-31', '2018-09-30',
                     '2018-10-31', '2018-11-30', '2018-12-31', '2019-01-31', '2019-02-28', '2019-03-31', '2019-04-30', '2019-05-31',
                     '2019-06-30', '2019-07-31', '2019-08-31', '2019-09-30', '2019-10-31', '2019-11-30', '2019-12-31', '2020-01-31',
                     '2020-02-29', '2020-03-31', '2020-04-30', '2020-05-31', '2020-06-30', '2020-07-31', '2020-08-31', '2020-09-30',
                     '2020-10-31', '2020-11-30', '2020-12-31', '2021-01-31', '2021-02-28', '2021-03-31', '2021-04-30', '2021-05-31',
                     '2021-06-30', '2021-07-31', '2021-08-31','2021-09-30','2021-10-31','2021-11-30','2021-12-31','2022-01-31','2022-02-28',
                     '2022-03-31', '2022-04-30','2022-05-31','2022-06-30','2022-07-31','2022-08-31','2022-09-30','2022-10-31','2022-11-30',
                     '2022-12-31','2023-01-31','2023-02-28','2023-03-31','2023-04-30','2023-05-31','2023-06-30','2023-07-31','2023-08-31',
                     '2023-09-30', 'lat', 'lng']
final_df = df_merged[columns_to_select]

id_vars = ['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName', 'State', 'City', 'Metro', 'CountyName', 'lat', 'lng']
value_vars = final_df.columns.difference(id_vars)
# list(value_vars)
```

```
# # Convert to long format
final_df_long = pd.melt(final_df, id_vars=id_vars, value_vars=value_vars, var_name='Date', value_name='ZHVI')
final_df_long = final_df_long.sort_values(by=['RegionName', 'Date']).reset_index(drop=True)
final_df_long = final_df_long[['State', 'City', 'CountyName', 'RegionName', 'lat', 'lng', 'Date', 'ZHVI']]

final_df_long['Date'] = pd.to_datetime(final_df_long['Date'])
new_column_names = {'RegionName': 'zipcode',
                    'lat': 'latitude',
                    'lng': 'longitude',
                    'ZHVI': 'ZHVI'}

final_df_long.rename(columns=new_column_names, inplace=True)
final_df_long
```

Out[4]:

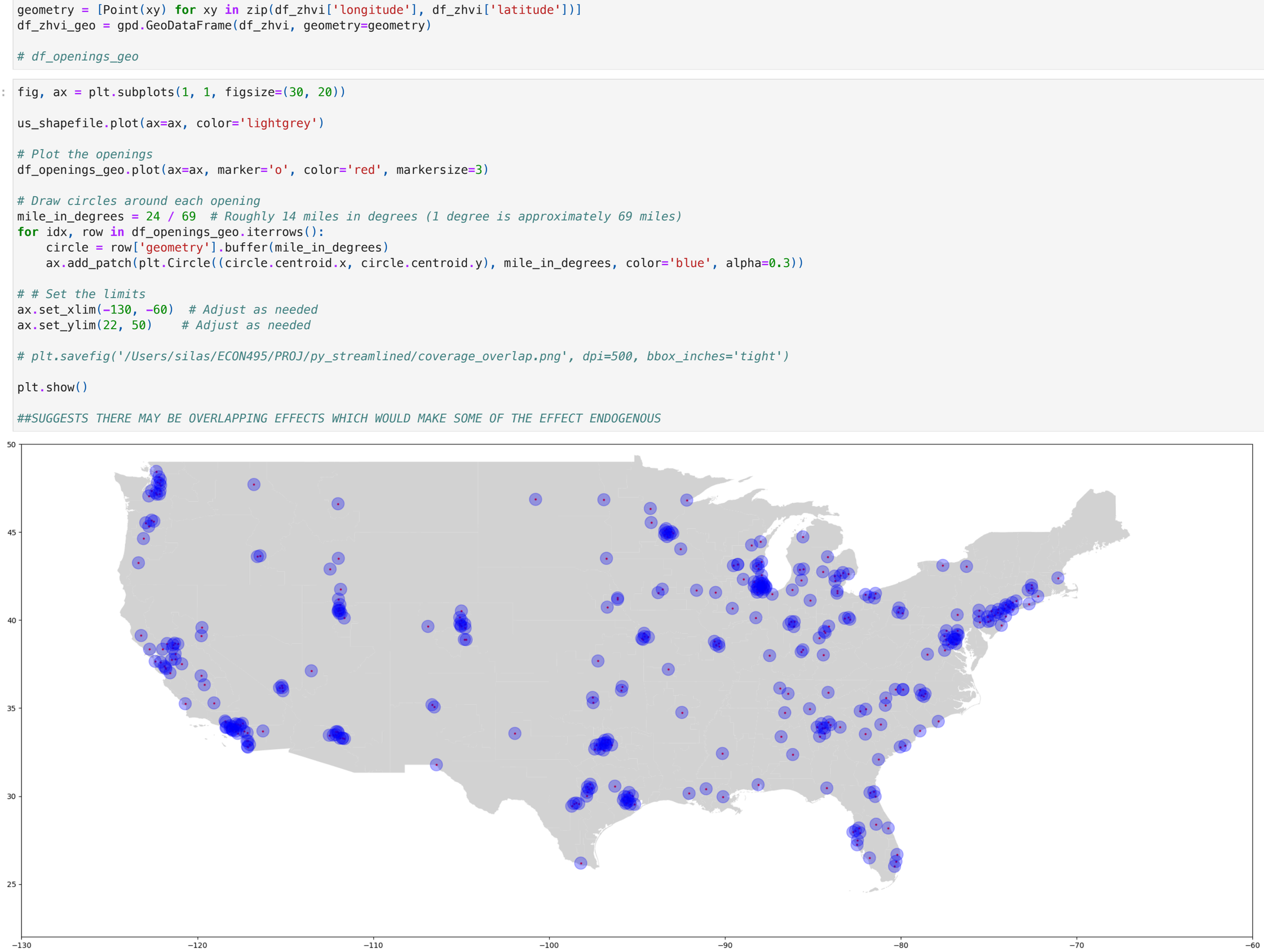
	State	City	CountyName	zipcode	latitude	longitude	Date	ZHVI
0	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746
1	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965
2	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258
3	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449
4	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616
...	...	...	...	...	...	...	...	...
7484095	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-05-31	242285.059571
7484096	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-06-30	243931.317030
7484097	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-07-31	244797.660672
7484098	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-08-31	245424.644616
7484099	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-09-30	245457.161303

7484100 rows × 8 columns

In [5]:
```
df_zhvi = final_df_long
df_zhvi['Log_ZHVI'] = np.log(df_zhvi['ZHVI'])

df_openings = df_costco_2000_2023
```

In [6]:
```
# Set the GDAL environment variable
os.environ['SHAPE_RESTORE_SHX'] = 'YES'

# Now try reading the shapefile
shapefile_path = '/Users/silas/ECON495/PROJ/cb_2022_us_cd118_5m.shp'
# shapefile_path = '/Users/silas/ECON495/PROJ/cb_2022_us_nation_20m.shp'
# shapefile_path = '/Users/silas/ECON495/PROJ/tl_2020_us_zcta520.shp'
map_df = gpd.read_file(shapefile_path)

# Convert DataFrame to GeoDataFrame
gdf = gpd.GeoDataFrame(df_openings, geometry=[Point(xy) for xy in zip(df_openings.longitude, df_openings.latitude)])
store_gdf = gdf
gdf.set_crs(epsg=4326, inplace=True)

us_shapefile = gpd.read_file('/Users/silas/ECON495/PROJ/cb_2022_us_cd118_5m.shp')
# us_shapefile = gpd.read_file('/Users/silas/ECON495/PROJ/tl_2020_us_zcta520.shp')

geometry = [Point(xy) for xy in zip(df_openings['longitude'], df_openings['latitude'])]
df_openings_geo = gpd.GeoDataFrame(df_openings, geometry=geometry)

geometry = [Point(xy) for xy in zip(df_zhvi['longitude'], df_zhvi['latitude'])]
df_zhvi_geo = gpd.GeoDataFrame(df_zhvi, geometry=geometry)

# df_openings_geo
```

In [7]:
```
fig, ax = plt.subplots(1, 1, figsize=(30, 20))

us_shapefile.plot(ax=ax, color='lightgrey')

# Plot the openings
df_openings_geo.plot(ax=ax, marker='o', color='red', markersize=3)

# Draw circles around each opening
mile_in_degrees = 24 / 69  # Roughly 14 miles in degrees (1 degree is approximately 69 miles)
for idx, row in df_openings_geo.iterrows():
    circle = row['geometry'].buffer(mile_in_degrees)
    ax.add_patch(plt.Circle((circle.centroid.x, circle.centroid.y), mile_in_degrees, color='blue', alpha=0.3))

# # Set the limits
ax.set_xlim(-130, -60)  # Adjust as needed
ax.set_ylim(22, 50)    # Adjust as needed

# plt.savefig('/Users/silas/ECON495/PROJ/py_streamlined/coverage_overlap.png', dpi=500, bbox_inches='tight')

plt.show()

##SUGGESTS THERE MAY BE OVERLAPPING EFFECTS WHICH WOULD MAKE SOME OF THE EFFECT ENDOGENOUS
```



In [8]:
```
import numpy as np
import pandas as pd  # Assuming you need pandas for DataFrame manipulation

def haversine(lat1, lon1, lat2, lon2):
    # Radius of the Earth in miles
    R = 3958.8
    # Convert latitude and longitude from degrees to radians
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])

    dlat = lat2 - lat1
    dlon = lon2 - lon1

    a = np.sin(dlat / 2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon / 2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))

    return R * c

def populate_overlap_column(df, max_distance_miles=50, max_years_difference=5):
    df['50_overlapping_within_5_years'] = 0
    df['opening_date'] = pd.to_datetime(df['opening_date'])

    for i in df.index:
        for j in df.index:
            if i != j:
                lat1, lon1 = df.at[i, 'latitude'], df.at[i, 'longitude']
                lat2, lon2 = df.at[j, 'latitude'], df.at[j, 'longitude']
                distance = haversine(lat1, lon1, lat2, lon2)

                date_diff = abs((df.at[i, 'opening_date'] - df.at[j, 'opening_date']).days)
                if distance < max_distance_miles and date_diff <= (max_years_difference * 365):
                    df.at[i, '50_overlapping_within_5_years'] = 1
                    df.at[j, '50_overlapping_within_5_years'] = 1
    return df
```

In [9]:
```
df_openings_geo = populate_overlap_column(df_openings_geo)
df_openings_geo
```

Out[9]:

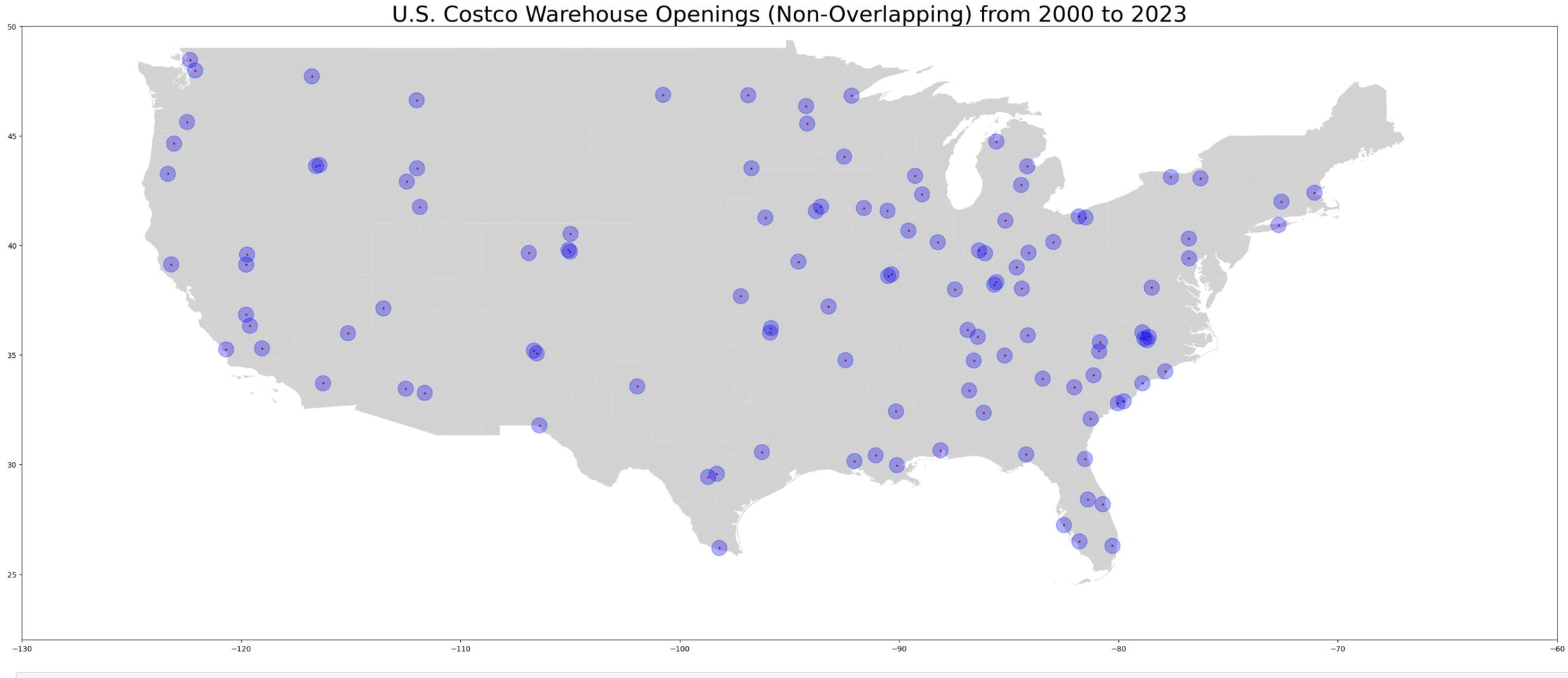
	State	City	store_name	zipcode	latitude	longitude	opening_date	geometry	50_overlapping_within_5_years
357	OH	Cincinnati	Deerfield	45249-8220	39.296	-84.303	2000-02-16	POINT (-84.30300 39.29600)	1
268	IL	Niles	Niles	60714-3905	42.015	-87.780	2000-06-01	POINT (-87.78000 42.01500)	1
119	UT	Sandy	Sandy	84070-4171	40.550	-111.895	2000-06-02	POINT (-111.89500 40.55000)	1
250	AZ	Phoenix	N Phoenix	85027-5036	33.660	-112.116	2000-06-22	POINT (-112.11600 33.66000)	1
409	CA	San Diego	Mission Valley	92108-4743	32.782	-117.129	2000-06-29	POINT (-117.12900 32.78200)	1
...	...	...	...	...	...	...	...	...	...
345	CO	Longmont	Longmont	80501	40.152	-105.085	2023-05-04	POINT (-105.08500 40.15200)	1
355	OK	Tulsa	North Tulsa	74116	36.223	-95.848	2023-05-25	POINT (-95.84800 36.22300)	0
14	AZ	Buckeye	Buckeye	85326	33.460	-112.501	2023-07-13	POINT (-112.50100 33.46000)	0
364	TX	Georgetown	Georgetown	78628	30.677	-97.664	2023-07-14	POINT (-97.66400 30.67700)	1
472	CO	Denver	NE Denver	80239	39.781	-104.796	2023-07-19	POINT (-104.79600 39.78100)	1

365 rows × 9 columns

In [10]:
```
nooverlap_df = df_openings_geo[df_openings_geo['50_overlapping_within_5_years'] == 0]
nooverlap_df.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
Index: 121 entries, 63 to 14
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   State                         121 non-null    object
 1   City                          121 non-null    object
 2   store_name                    121 non-null    object
 3   zipcode                       121 non-null    object
 4   latitude                      121 non-null    float64
 5   longitude                     121 non-null    float64
 6   opening_date                  121 non-null    datetime64[ns]
 7   geometry                      121 non-null    geometry
 8   50_overlapping_within_5_years  121 non-null    int64
dtypes: datetime64[ns](1), float64(2), geometry(1), int64(1), object(4)
memory usage: 9.5+ KB
```

In [11]:
```
# Assuming your existing DataFrame is named 'us_shapefile' and 'nooverlap_df'

fig, ax = plt.subplots(1, 1, figsize=(40, 15))  # Adjust the figsize as needed

us_shapefile.plot(ax=ax, color='lightgrey')

# Plot the openings
nooverlap_df.plot(ax=ax, marker='o', color='red', markersize=3)

# Draw circles around each opening
mile_in_degrees = 24 / 69
for idx, row in nooverlap_df.iterrows():
    circle = row['geometry'].buffer(mile_in_degrees)
    ax.add_patch(plt.Circle((circle.centroid.x, circle.centroid.y), mile_in_degrees, color='blue', alpha=0.3))


# # Set the limits
ax.set_xlim(-130, -60)  # Adjust as needed
ax.set_ylim(22, 50)    # Adjust as needed

# Adjusting aspect ratio
ax.set_aspect(aspect=1.0)

ax.set_title("U.S. Costco Warehouse Openings (Non-Overlapping) from 2000 to 2023", fontsize=30)

# plt.savefig('/Users/silas/ECON495/PROJ/py_streamlined/48overlapcoverage.png', dpi=500, bbox_inches='tight')

plt.show()
```



In [12]:
```python
openings_gdf = nooverlap_df
zhvi_gdf = df_zhvi_geo

mile_in_meters = 1609.34

openings_gdf.set_crs(epsg=4326, inplace=True)  # WGS84 Latitude/Longitude
zhvi_gdf.set_crs(epsg=4326, inplace=True)

# Ensure opening_date is a datetime object
openings_gdf['opening_date'] = pd.to_datetime(openings_gdf['opening_date'])
zhvi_gdf['Date'] = pd.to_datetime(zhvi_gdf['Date'])

# Example: Convert to a UTM zone (e.g., UTM zone 18N for parts of the Northeastern US)
utm_crs = 'EPSG:32618'
openings_gdf = openings_gdf.to_crs(utm_crs)
zhvi_gdf = zhvi_gdf.to_crs(utm_crs)

# openings_gdf.to_csv('all_openings_48overlap_gdf.csv', index=False)
```

```
/Library/anaconda3/envs/ENDGAME/lib/python3.8/site-packages/geopandas/geodataframe.py:1538: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
/Library/anaconda3/envs/ENDGAME/lib/python3.8/site-packages/geopandas/geodataframe.py:1538: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

In [13]:
```python
def determine_store_info_and_post_treatment(row, openings_gdf, mile_in_meters):
    # Calculate distances from the row's geometry to all store openings
    distances = openings_gdf.geometry.distance(row.geometry)
    min_distance = distances.min()

    # Initialize default values
    nearest_store_name = 'None'
    distance_category = 'Outside range'
    post_treatment = False

    # Check if the nearest store is within 25 miles
    if min_distance < mile_in_meters * 25:
        # Find the store that matches the minimum distance
        nearest_store_row = openings_gdf[distances == min_distance].iloc[0]
        nearest_store_name = nearest_store_row['store_name']

        # Determine distance category
        if min_distance < mile_in_meters * 5:
            distance_category = '0-5 miles'
        elif min_distance < mile_in_meters * 10:
            distance_category = '5-10 miles'
        elif min_distance < mile_in_meters * 15:
            distance_category = '10-15 miles'
        elif min_distance < mile_in_meters * 20:
            distance_category = '15-20 miles'
        else:
            distance_category = '20-25 miles'

        # Determine post treatment
        store_opening_date = nearest_store_row['opening_date']
        if row['Date'] >= store_opening_date:
            post_treatment = True

    return distance_category, nearest_store_name, post_treatment, min_distance

def apply_store_info_and_post_treatment(gdf, openings_gdf, mile_in_meters):
    # Lists to store the results
    distance_categories = []
    nearest_store_names = []
    post_treatment_values = []
    min_distances = []

    for _, row in gdf.iterrows():
        distance_category, nearest_store_name, post_treatment, min_distance = determine_store_info_and_post_treatment(row, openings_gdf, mile_in_meters)
        distance_categories.append(distance_category)
        nearest_store_names.append(nearest_store_name)
        post_treatment_values.append(post_treatment)
        min_distances.append(min_distance)

    # Assigning new columns to the DataFrame
    gdf['distance_category'] = distance_categories
    gdf['nearest_store'] = nearest_store_names
    gdf['min_distance_to_store'] = min_distances
    gdf['min_distance_to_store_miles'] = pd.Series(min_distances) / 1609.34
    gdf['post_treatment'] = post_treatment_values

    return gdf
```

In [14]:
```
# Usage
zhvi_gdf = apply_store_info_and_post_treatment(zhvi_gdf, openings_gdf, mile_in_meters)
```

In [15]:
```
zhvi_gdf
```

Out[15]:

	State	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	Log_ZHVI	geometry	distance_category	nearest_store	min_distance_to_store	min_distance_to_
0	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	11.772842	POINT (696487.376 4659457.719)	5-10 miles	Enfield	8.741847e+03	
1	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	11.771722	POINT (696487.376 4659457.719)	5-10 miles	Enfield	8.741847e+03	
2	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	11.771750	POINT (696487.376 4659457.719)	5-10 miles	Enfield	8.741847e+03	
3	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	11.772697	POINT (696487.376 4659457.719)	5-10 miles	Enfield	8.741847e+03	
4	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	11.778614	POINT (696487.376 4659457.719)	5-10 miles	Enfield	8.741847e+03	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7484095	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-05-31	242285.059571	12.397870	POINT (-2715469.671 7777019.818)	Outside range	None	1.246909e+06	
7484096	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-06-30	243931.317030	12.404642	POINT (-2715469.671 7777019.818)	Outside range	None	1.246909e+06	
7484097	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-07-31	244797.660672	12.408187	POINT (-2715469.671 7777019.818)	Outside range	None	1.246909e+06	
7484098	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-08-31	245424.644616	12.410745	POINT (-2715469.671 7777019.818)	Outside range	None	1.246909e+06	
7484099	AK	Wrangell	Wrangell Borough	99929	56.36089	-132.00635	2023-09-30	245457.161303	12.410878	POINT (-2715469.671 7777019.818)	Outside range	None	1.246909e+06	

7484100 rows × 15 columns

In [16]:
```
zhvi_gdf = zhvi_gdf[zhvi_gdf['nearest_store'] != 'None']

openings_gdf['zipcode'] = openings_gdf['zipcode'].str[:5]
```

In [17]:
```
def add_relative_quarters(df, treatment_col='post_treatment', date_col='Date'):
    df = df.copy()
    df[date_col] = pd.to_datetime(df[date_col])

    def get_treatment_start_date(zipcode):
        zipcode_data = df[df['zipcode'] == zipcode]
        treatment_start = zipcode_data[zipcode_data[treatment_col].diff() == 1][date_col]
        return treatment_start.iloc[0] if not treatment_start.empty else pd.NaT

    zipcodes = df['zipcode'].unique()
    treatment_start_dates = {zipcode: get_treatment_start_date(zipcode) for zipcode in zipcodes}

    def calc_relative_quarters(row):
        treatment_date = treatment_start_dates[row['zipcode']]
        if pd.notnull(treatment_date):
            quarters_diff = (row[date_col].year - treatment_date.year) * 4 + (row[date_col].quarter - treatment_date.quarter)
            return quarters_diff
        return np.nan

    # Apply the function and explicitly ensure it's treated as a scalar
    relative_quarters = df.apply(calc_relative_quarters, axis=1)
    if isinstance(relative_quarters, pd.DataFrame):
        raise ValueError("Expected a Series but got a DataFrame.")

    df['quarters_from_opening'] = relative_quarters
    return df

zhvi_gdf = add_relative_quarters(zhvi_gdf)

# zhvi_gdf = zhvi_gdf[(zhvi_gdf['quarters_from_opening'] <= 20) & (zhvi_gdf['quarters_from_opening'] >= -20)]
```

In [18]:
```
def add_relative_months(df, treatment_col='post_treatment', date_col='Date'):
    df = df.copy()
    df[date_col] = pd.to_datetime(df[date_col])

    def get_treatment_start_date(zipcode):
        zipcode_data = df[df['zipcode'] == zipcode]
        treatment_start = zipcode_data[zipcode_data[treatment_col].diff() == 1][date_col]
        return treatment_start.iloc[0] if not treatment_start.empty else pd.NaT

    zipcodes = df['zipcode'].unique()
    treatment_start_dates = {zipcode: get_treatment_start_date(zipcode) for zipcode in zipcodes}

    def calc_relative_months(row):
        treatment_date = treatment_start_dates[row['zipcode']]
        if pd.notnull(treatment_date):
            # Calculate the difference in months considering the years and months of the dates
            months_diff = (row[date_col].year - treatment_date.year) * 12 + (row[date_col].month - treatment_date.month)
            return months_diff
        return np.nan

    # Apply the function and explicitly ensure it's treated as a scalar
    relative_months = df.apply(calc_relative_months, axis=1)
    if isinstance(relative_months, pd.DataFrame):
        raise ValueError("Expected a Series but got a DataFrame.")

    df['months_from_opening'] = relative_months
    return df

# Example usage (assuming zhvi_gdf is your DataFrame):
zhvi_gdf = add_relative_months(zhvi_gdf)
```

In [29]:
```
zhvi_gdf['treatment_0_5'] = (zhvi_gdf['distance_category'] == '0-5 miles').astype(int)
zhvi_gdf['treatment_5_10'] = (zhvi_gdf['distance_category'] == '5-10 miles').astype(int)
zhvi_gdf['treatment_10_15'] = (zhvi_gdf['distance_category'] == '10-15 miles').astype(int)
zhvi_gdf['treatment_15_20'] = (zhvi_gdf['distance_category'] == '15-20 miles').astype(int)
zhvi_gdf['control'] = (zhvi_gdf['distance_category'] == '20-25 miles').astype(int)

zhvi_gdf['post_treatment'] = zhvi_gdf['post_treatment'].astype(int)
```

In [30]:
```
# Create interaction terms
zhvi_gdf['treat5_post'] = zhvi_gdf['treatment_0_5'] * zhvi_gdf['post_treatment']
zhvi_gdf['treat10_post'] = zhvi_gdf['treatment_5_10'] * zhvi_gdf['post_treatment']
zhvi_gdf['treat15_post'] = zhvi_gdf['treatment_10_15'] * zhvi_gdf['post_treatment']
zhvi_gdf['treat20_post'] = zhvi_gdf['treatment_15_20'] * zhvi_gdf['post_treatment']
zhvi_gdf['control_post'] = zhvi_gdf['control'] * zhvi_gdf['post_treatment']

# Convert 'Date' to datetime and extract year and month
zhvi_gdf['Date'] = pd.to_datetime(zhvi_gdf['Date'])
zhvi_gdf['Year'] = zhvi_gdf['Date'].dt.year
zhvi_gdf['Month'] = zhvi_gdf['Date'].dt.month

# Create a unique identifier for each store-year-month combination
zhvi_gdf['Store_Year_Month'] = zhvi_gdf['nearest_store'] + '_' + zhvi_gdf['Year'].astype(str) + '_' + zhvi_gdf['Month'].astype(str)

# zhvi_gdf.to_csv('zhvi_complete.csv', index=False)
```

In [31]:
```
def remove_matching_rows(zhvi_gdf, openings_gdf):
    opening_stores = set(openings_gdf[openings_gdf['50_overlapping_within_5_years'] == 1]['store_name'])

    filtered_df = zhvi_gdf[~(zhvi_gdf['nearest_store'].isin(opening_stores))]

    return filtered_df

# Example usage:
filtered_zhvi_df = remove_matching_rows(zhvi_gdf, openings_gdf)
filtered_zhvi_df
```

Out[31]:

	State	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	Log_ZHVI	geometry	...	treatment_0_5	treatment_5_10	treatment_10_15	treatment_15_20	c
0	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	11.772842	POINT (696487.376 4659457.719)	...	0	1	0	0	
1	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	11.771722	POINT (696487.376 4659457.719)	...	0	1	0	0	
2	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	11.771750	POINT (696487.376 4659457.719)	...	0	1	0	0	
3	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	11.772697	POINT (696487.376 4659457.719)	...	0	1	0	0	
4	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	11.778614	POINT (696487.376 4659457.719)	...	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7480960	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	12.628417	POINT (-2258260.842 9137914.888)	...	0	0	0	1	
7480961	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	12.633130	POINT (-2258260.842 9137914.888)	...	0	0	0	1	
7480962	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	12.638299	POINT (-2258260.842 9137914.888)	...	0	0	0	1	
7480963	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	12.640520	POINT (-2258260.842 9137914.888)	...	0	0	0	1	
7480964	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	12.637306	POINT (-2258260.842 9137914.888)	...	0	0	0	1	

1213245 rows × 30 columns

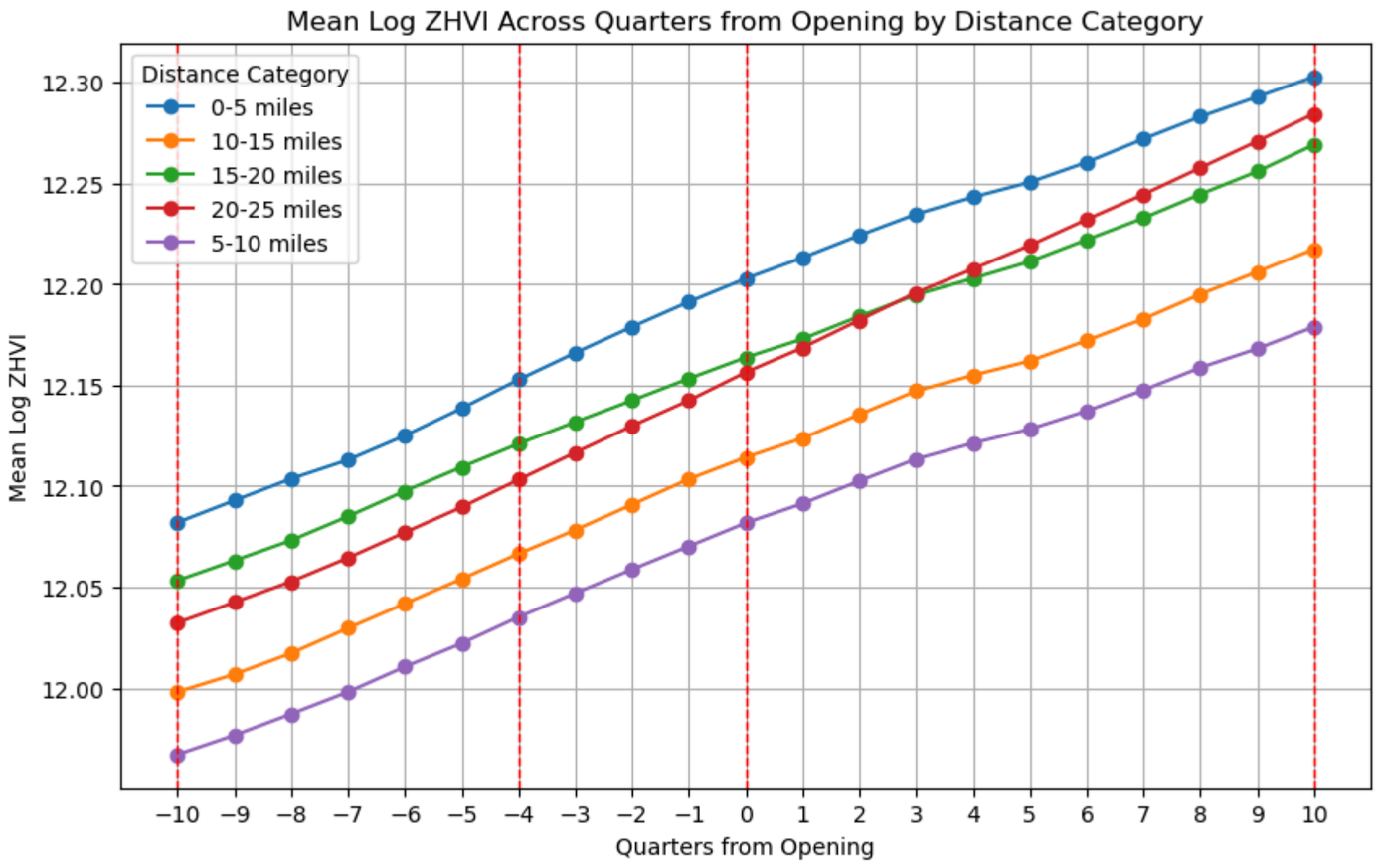
In [32]:
```
filtered_zhvi_df['quarters_from_opening'].unique()
```

Out[32]:
```
array([-18, -17, -16, -15, -14, -13, -12, -11, -10,  -9,  -8,  -7,  -6,
        -5,  -4,  -3,  -2,  -1,   0,   1,   2,   3,   4,   5,   6,   7,
         8,   9,  10,  11,  12,  13,  14,  15,  16,  17,  18,  19,  20,
        21,  22,  23,  24,  25,  26,  27,  28,  29,  30,  31,  32,  33,
        34,  35,  36,  37,  38,  39,  40,  41,  42,  43,  44,  45,  46,
        47,  48,  49,  50,  51,  52,  53,  54,  55,  56,  57,  58,  59,
        60,  61,  62,  63,  64,  65,  66,  67,  68,  69,  70,  71,  72,
        73,  74,  75,  76,  77,  78,  79,  80,  81,  82,  83, -57, -56,
       -55, -54, -53, -52, -51, -50, -49, -48, -47, -46, -45, -44, -43,
       -42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31, -30,
       -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -59, -58,
       -61, -60,  84,  85,  86,  87,  88, -75, -74, -73, -72, -71, -70,
       -69, -68, -67, -66, -65, -64, -63, -62, -77, -76,  89,  90,  91,
       -87, -86, -85, -84, -83, -82, -81, -80, -79, -78, -91, -90, -89,
       -88, -93, -92, -94,  92])
```

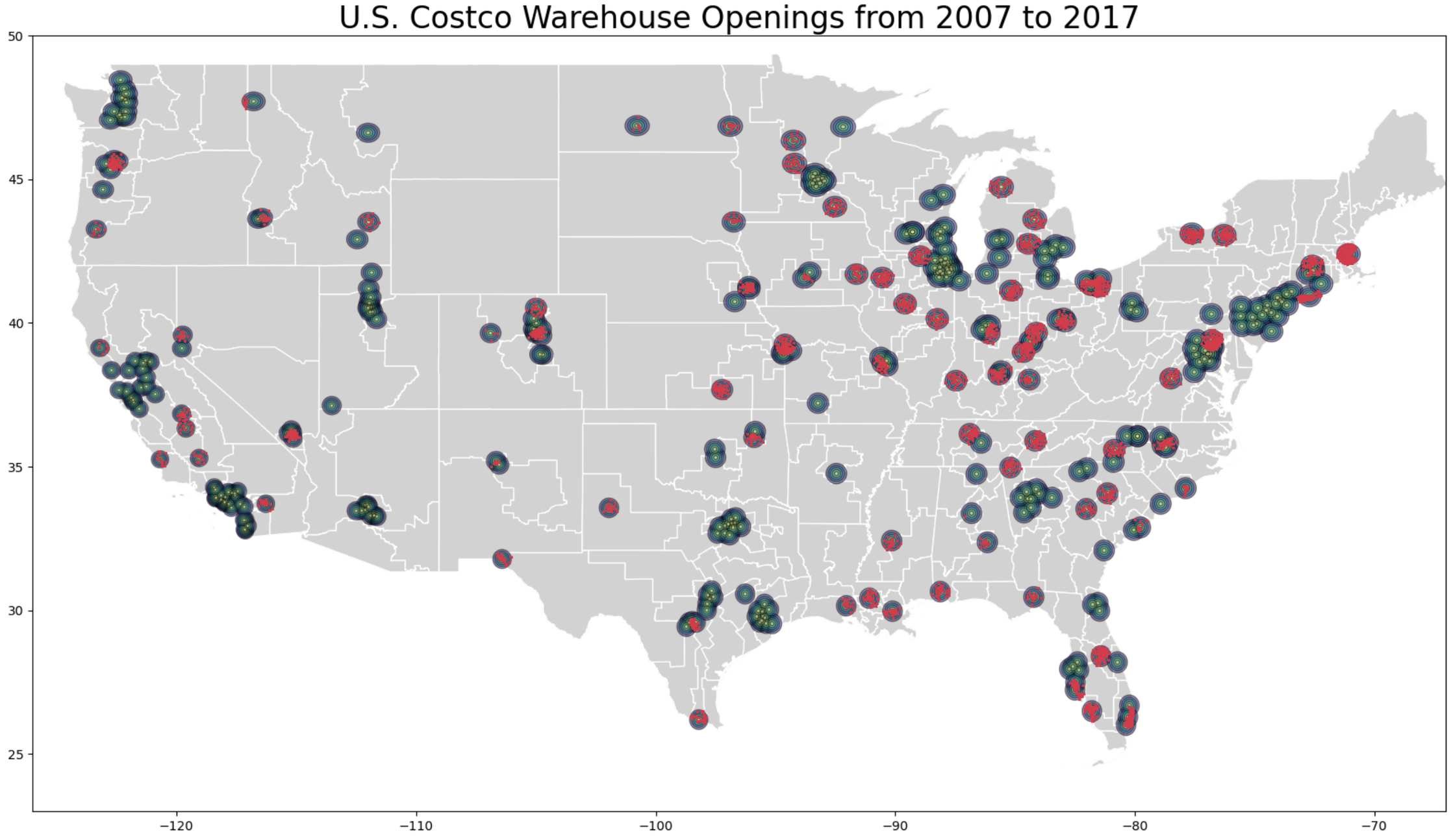
In [33]:
```
# filtered_zhvi_df.to_csv('zhvi_50.csv', index=False)
```

In [24]:
```
merged_with_counties = pd.read_csv("/Users/silas/ECON495/CLOSER/zhvi_50.csv")
```

In [25]:
```
merged_with_counties.dropna(subset=['Log_ZHVI'], inplace=True)

merged_with_counties.drop_duplicates(subset=['Date', 'zipcode'], keep='first', inplace=True)

merged_with_counties = merged_with_counties[(merged_with_counties['quarters_from_opening'] >= -10) & (merged_with_counties['quarters_from_opening'] <= 10)].copy()

merged_with_counties['obs_count'] = merged_with_counties.groupby(['zipcode', 'distance_category'])['quarters_from_opening'].transform('count')

merged_with_counties = merged_with_counties[merged_with_counties['obs_count'] >= 63]
merged_with_counties = merged_with_counties.drop(columns=['obs_count'])
merged_with_counties.shape

# Group by both 'distance_category' and 'quarters_from_opening' and calculate the mean of 'log_zhvi'
grouped_means = merged_with_counties.groupby(['distance_category', 'quarters_from_opening'])['Log_ZHVI'].mean().unstack(level=0)

# Plotting
grouped_means.plot(kind='line', marker='o', figsize=(10, 6))
plt.title('Mean Log ZHVI Across Quarters from Opening by Distance Category')
plt.xlabel('Quarters from Opening')
plt.ylabel('Mean Log ZHVI')
plt.grid(True)
plt.legend(title='Distance Category', loc='best')
plt.xticks(grouped_means.index)

for x in [-10, -4, 0, 10]:
    plt.axvline(x=x, color='red', linestyle='--', linewidth=1)

plt.show()
```



In [26]:
```
shapefile_path = 'data/cb_2022_us_cd118_5m.shp'
map_df = gpd.read_file(shapefile_path)

df_openings_geo = gpd.GeoDataFrame(
    df_openings,
    geometry=gpd.points_from_xy(df_openings.longitude, df_openings.latitude)
)

# Set the CRS for WGS 84 (EPSG:4326) and then convert to UTM Zone 18N
df_openings_geo.set_crs(epsg=4326, inplace=True)
df_openings_geo_utm = df_openings_geo.to_crs(epsg=32618)  # UTM Zone 18N

# Sorting the radii in descending order to ensure the larger buffers are drawn first
radii = sorted([5, 10, 15, 20, 25], reverse=True)

# Plotting the shapefile
fig, ax = plt.subplots(1, 1, figsize=(20, 12))
map_df.plot(ax=ax, edgecolor='white', facecolor='lightgrey')

# Plotting the buffers in descending order of their radii
colors = ['#5e4fa2', '#3288bd', '#66c2a5', '#abdda4', '#e6f598', '#fee08b', '#fdae61', '#f46d43']

for radius, color in zip(radii, colors):
    # Buffer creation and transformation to WGS 84 CRS
    buffer = df_openings_geo_utm.geometry.buffer(radius * 1609.34).to_crs(epsg=4326)
    buffer.plot(ax=ax, alpha=0.5, edgecolor='black', facecolor=color)

# Plotting the original points
# Assuming merged_with_counties is defined and contains 'latitude' and 'longitude' columns
gdf_temp = gpd.GeoDataFrame(
    merged_with_counties,
    geometry=gpd.points_from_xy(merged_with_counties.longitude, merged_with_counties.latitude)
)
gdf_temp.plot(ax=ax, marker='o', color='#d53e4f', markersize=0.1)

ax.set_xlim(-126, -67)  # Adjust as needed
ax.set_ylim(23, 50)    # Adjust as needed

# Adjusting aspect ratio
ax.set_aspect(aspect=1.2)

# Setting the title
ax.set_title('U.S. Costco Warehouse Openings from 2007 to 2017', fontsize = 24)

# Saving the figure
# plt.savefig('/Users/silas/ECON495/PROJ/ACS_data.png', dpi=500, bbox_inches='tight')

plt.show()
```


U.S. Costco Warehouse Openings from 2007 to 2017
50
45
40
35
30
25
−120
−110
−100
−90
−80
−70

In [1]:
```
import pandas as pd
import numpy as np
import altair as alt
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import os

from shapely import wkt
from shapely.geometry import Point, box
import pysal
import libpysal

import geopandas as gpd
from geopandas.tools import sjoin
from geopy.distance import geodesic
from geopandas import sjoin

from autocensus import Query
```

```
/Library/anaconda3/envs/ENDGAME/lib/python3.8/site-packages/geopandas/_compat.py:124: UserWarning: The Shapely GEOS version (3.11.2-CAPI-1.17.2) is incompatible with the GEOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions between both will be slow.
  warnings.warn(
```

In [2]:
```
from autocensus import Query

query = Query(
    estimate=5,
    years=[2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022],
    variables=['B01003_001E', 'B19013_001E', 'B23006_023E', 'B02001_002E'],
    for_geo='county:*',
    in_geo=['state:*'],
    geometry='polygons',
    census_api_key='095fa8aaf42a3003ba7ce37c76a71a2037ab053e',
    resolution='500k'
)

try:
    dataframe = query.run()
except Exception as e:
    print(f"An error occurred: {e}")
    # Optionally, add any cleanup or retry logic here

# #B01003_001E : Total Population
# #B19013_001E : Household Income
# #B23006_023E : Bachelor's degree
# #B02001_002E : White alone

acs_df = query.run()
```

```
Retrieving variables...
Warning: B23006_023E is not a recognized variable for 2009
Retrieving ACS tables...
Retrieving shapefiles...
Merging ACS tables and variables...
Merging annotations...
Merging shapefiles...
Finalizing data...
Retrieving variables...
Retrieving ACS tables...
Retrieving shapefiles...
Merging ACS tables and variables...
Merging annotations...
Merging shapefiles...
Finalizing data...
```

In [3]:
```
#ORIGINAL 2 DATASETS
df_openings = pd.read_csv("/Users/silas/ECON495/CLOSER/50_openings.csv")

df_zhvi = pd.read_csv("/Users/silas/ECON495/CLOSER/zhvi_50.csv")

# Fill missing City values with the most common 'City' for each 'CountyName'
most_common_cities = df_zhvi.groupby('CountyName')['City'].agg(pd.Series.mode).reset_index()
most_common_cities['City'] = most_common_cities['City'].apply(lambda x: x.iloc[0] if isinstance(x, pd.Series) else x)
df_zhvi = pd.merge(df_zhvi, most_common_cities, on='CountyName', how='left', suffixes=('', '_most_common'))
df_zhvi['City'] = df_zhvi.apply(lambda row: row['City_most_common'] if pd.isna(row['City']) else row['City'], axis=1)
df_zhvi.drop('City_most_common', axis=1, inplace=True)
df_zhvi
```

Out[3]:

	State	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	Log_ZHVI	geometry	...	treatment_0_5	treatment_5_10	treatment_10_15	treatment_15_
0	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	11.772842	POINT (696487.3761403256 4659457.718801506)	...	0	1	0	
1	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	11.771722	POINT (696487.3761403256 4659457.718801506)	...	0	1	0	
2	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	11.771750	POINT (696487.3761403256 4659457.718801506)	...	0	1	0	
3	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	11.772697	POINT (696487.3761403256 4659457.718801506)	...	0	1	0	
4	MA	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	11.778614	POINT (696487.3761403256 4659457.718801506)	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	12.628417	POINT (-2258260.842104718 9137914.888358535)	...	0	0	0	
1213241	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	12.633130	POINT (-2258260.842104718 9137914.888358535)	...	0	0	0	
1213242	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	12.638299	POINT (-2258260.842104718 9137914.888358535)	...	0	0	0	
1213243	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	12.640520	POINT (-2258260.842104718 9137914.888358535)	...	0	0	0	
1213244	AK	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	12.637306	POINT (-2258260.842104718 9137914.888358535)	...	0	0	0	

1213245 rows × 30 columns

In [4]:
```
#Create county to county name mapping
geo_id_to_name_mapping = acs_df.drop_duplicates(subset=['geo_id'])[['geo_id', 'name']].set_index('geo_id')['name'].to_dict()
```

```
geo_id_to_name_mapping_upper = {k: v.upper() for k, v in geo_id_to_name_mapping.items()}
geo_id_to_name_mapping_upper = {k: v.replace("'", "") for k, v in geo_id_to_name_mapping_upper.items()}
geo_id_to_name_mapping_upper = {k: v.replace("ST. ", "SAINT ").replace("ST. ", "SAINT") for k, v in geo_id_to_name_mapping_upper.items()}
geo_id_to_name_mapping_upper = {k: v.replace("DE WITT", "DEWITT").replace("DE WITT", "DEWITT") for k, v in geo_id_to_name_mapping_upper.items()}
geo_id_to_name_mapping_upper = {k: v.replace("DO?A", "DONA").replace("DO?A", "DONA") for k, v in geo_id_to_name_mapping_upper.items()}

geo_df = pd.DataFrame(list(geo_id_to_name_mapping_upper.items()), columns=['geo_id', 'name'])
# geo_df.to_csv('full_acs_map.csv', index=False)
```

In [5]:
```
# If acs_df is not yet a GeoDataFrame but the geometries are correct, convert it
if not isinstance(acs_df, gpd.GeoDataFrame):
    gdf = gpd.GeoDataFrame(acs_df, geometry='geometry')

# Ensure CRS is set; assuming geometries are in WGS84
gdf.set_crs("EPSG:4326", inplace=True, allow_override=True)
```

Out[5]:

	name	geo_id	geo_type	year	date	variable_code	variable_label	variable_concept	annotation	value	geometry
0	Abbeville County, South Carolina	0500000US45001	county	2009	2009-12-31	B01003_001E	Estimate!!Total	TOTAL POPULATION	NaN	25347.0	None
1	Abbeville County, South Carolina	0500000US45001	county	2010	2010-12-31	B01003_001E	Estimate!!Total	TOTAL POPULATION	NaN	25643.0	None
2	Abbeville County, South Carolina	0500000US45001	county	2011	2011-12-31	B01003_001E	Estimate!!Total	TOTAL POPULATION	NaN	25515.0	None
3	Abbeville County, South Carolina	0500000US45001	county	2012	2012-12-31	B01003_001E	Estimate!!Total	TOTAL POPULATION	NaN	25387.0	None
4	Abbeville County, South Carolina	0500000US45001	county	2013	2013-12-31	B01003_001E	Estimate!!Total	TOTAL POPULATION	NaN	25233.0	MULTIPOLYGON (((-82.74113 34.20879, -82.68430 ...
...	...	...	...	...	...	...	...	...	...	...	...
177130	Ziebach County, South Dakota	0500000US46137	county	2018	2018-12-31	B23006_023E	Estimate!!Total!!Bachelor's degree or higher	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FO...	NaN	224.0	MULTIPOLYGON (((-102.00044 44.69000, -102.0001...
177131	Ziebach County, South Dakota	0500000US46137	county	2019	2019-12-31	B23006_023E	Estimate!!Total:!!Bachelor's degree or higher:	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FO...	NaN	230.0	MULTIPOLYGON (((-102.00044 44.86364, -102.0001...
177132	Ziebach County, South Dakota	0500000US46137	county	2020	2020-12-31	B23006_023E	Estimate!!Total:!!Bachelor's degree or higher:	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FO...	NaN	218.0	MULTIPOLYGON (((-102.00044 44.86364, -102.0001...
177133	Ziebach County, South Dakota	0500000US46137	county	2021	2021-12-31	B23006_023E	Estimate!!Total:!!Bachelor's degree or higher:	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FO...	NaN	179.0	MULTIPOLYGON (((-102.00044 44.86364, -102.0001...
177134	Ziebach County, South Dakota	0500000US46137	county	2022	2022-12-31	B23006_023E	Estimate!!Total:!!Bachelor's degree or higher:	Educational Attainment by Employment Status fo...	NaN	177.0	MULTIPOLYGON (((-102.00044 44.86364, -102.0001...

177135 rows × 11 columns

In [6]:
```
# State abbreviation to full name mapping
state_full_names = {
    'AL': 'Alabama', 'AK': 'Alaska', 'AZ': 'Arizona', 'AR': 'Arkansas', 'CA': 'California',
    'CO': 'Colorado', 'CT': 'Connecticut', 'DE': 'Delaware', 'FL': 'Florida', 'GA': 'Georgia',
    'HI': 'Hawaii', 'ID': 'Idaho', 'IL': 'Illinois', 'IN': 'Indiana', 'IA': 'Iowa',
    'KS': 'Kansas', 'KY': 'Kentucky', 'LA': 'Louisiana', 'ME': 'Maine', 'MD': 'Maryland',
    'MA': 'Massachusetts', 'MI': 'Michigan', 'MN': 'Minnesota', 'MS': 'Mississippi', 'MO': 'Missouri',
    'MT': 'Montana', 'NE': 'Nebraska', 'NV': 'Nevada', 'NH': 'New Hampshire', 'NJ': 'New Jersey',
    'NM': 'New Mexico', 'NY': 'New York', 'NC': 'North Carolina', 'ND': 'North Dakota', 'OH': 'Ohio',
    'OK': 'Oklahoma', 'OR': 'Oregon', 'PA': 'Pennsylvania', 'RI': 'Rhode Island', 'SC': 'South Carolina',
    'SD': 'South Dakota', 'TN': 'Tennessee', 'TX': 'Texas', 'UT': 'Utah', 'VT': 'Vermont',
    'VA': 'Virginia', 'WA': 'Washington', 'WV': 'West Virginia', 'WI': 'Wisconsin', 'WY': 'Wyoming'
}

# Create new column with full state names
df_zhvi['state_full'] = df_zhvi['State'].map(state_full_names)
columns_except_state_full = [col for col in df_zhvi.columns if col != 'state_full']
new_column_order = ['state_full'] + columns_except_state_full
df_zhvi = df_zhvi[new_column_order]
df_zhvi.drop(columns=['State'], inplace=True)

# ADDITIONAL SPECIFIC CLEANING
df_zhvi.loc[df_zhvi['CountyName'] == "District of Columbia", 'state_full'] = "District of Columbia"

#continue mapping
df_zhvi['name'] = (df_zhvi['CountyName'] + ", " + df_zhvi['state_full']).str.upper()

columns_except_state_full = [col for col in df_zhvi.columns if col != 'name']
new_column_order = ['name'] + columns_except_state_full
df_zhvi = df_zhvi[new_column_order]

# ADDITIONAL SPECIFIC CLEANING
df_zhvi['name'] = df_zhvi['name'].str.replace('DE KALB COUNTY, INDIANA', 'DEKALB COUNTY, INDIANA', regex=False, case=False)

df_zhvi
```

```
/var/folders/b5/lpp9ws755wg9hd3qg6v6np7h0000gn/T/ipykernel_7147/780576954.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_zhvi.drop(columns=['State'], inplace=True)
```

Out[6]:

	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	Log_ZHVI	...	treatment_0_5	treatment_5_10	treatment_10_15	treatment
0	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	11.772842	...	0	1	0	
1	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	11.771722	...	0	1	0	
2	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	11.771750	...	0	1	0	
3	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	11.772697	...	0	1	0	
4	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	11.778614	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	12.628417	...	0	0	0	
1213241	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	12.633130	...	0	0	0	
1213242	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	12.638299	...	0	0	0	
1213243	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	12.640520	...	0	0	0	
1213244	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	12.637306	...	0	0	0	

1213245 rows × 31 columns

In [7]:
```
name_to_geo_id_mapping_upper = {v: k for k, v in geo_id_to_name_mapping_upper.items()}
df_zhvi['geo_id'] = df_zhvi['name'].map(name_to_geo_id_mapping_upper)

columns_except_state_full = [col for col in df_zhvi.columns if col != 'geo_id']
new_column_order = ['geo_id'] + columns_except_state_full
df_zhvi = df_zhvi[new_column_order]

df_zhvi['Year'] = df_zhvi['Year'].astype(int)

df_zhvi
# Write df_zhvi to a CSV file named 'df_zhvi.csv'
# df_zhvi.to_csv('zhvi_48nooverlap_GEOID_FULL_gdf.csv', index=False)
```

Out[7]:

	geo_id	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	...	treatment_0_5	treatment_5_10	treatment_10_15	tr
0	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	...	0	1	0	
1	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	...	0	1	0	
2	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	...	0	1	0	
3	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	...	0	1	0	
4	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	...	0	0	0	
1213241	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	...	0	0	0	
1213242	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	...	0	0	0	
1213243	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	...	0	0	0	
1213244	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	...	0	0	0	

1213245 rows × 32 columns

In [8]:
```
#Create countyname to geo_id mapping
county_geo_id_map = df_zhvi[['geo_id', 'name']].drop_duplicates()
county_geo_id_map.reset_index(drop=True, inplace=True)
county_geo_id_map
```

Out[8]:

	geo_id	name
0	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS
1	0500000US25015	HAMPSHIRE COUNTY, MASSACHUSETTS
2	0500000US25027	WORCESTER COUNTY, MASSACHUSETTS
3	0500000US25017	MIDDLESEX COUNTY, MASSACHUSETTS
4	0500000US25009	ESSEX COUNTY, MASSACHUSETTS
...	...	...
508	0500000US53057	SKAGIT COUNTY, WASHINGTON
509	0500000US53029	ISLAND COUNTY, WASHINGTON
510	0500000US53011	CLARK COUNTY, WASHINGTON
511	0500000US53063	SPOKANE COUNTY, WASHINGTON
512	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA

513 rows × 2 columns

In [9]:
```
# print(name_to_geo_id_mapping_upper)
```

SUCCESSFUL MAPPING ON COUNTY GEO_ID W ZHVI DATA COMPLETE for 2009-2022

In [10]:
```
# Convert years to the same data type if necessary
acs_df['year'] = acs_df['year'].astype(int)

# Pivot acs_df to have 'variable_concept' as columns
acs_pivoted = acs_df.pivot_table(index=['geo_id', 'year'], columns='variable_concept', values='value', aggfunc='first').reset_index()

# Identify columns that match the pattern for MEDIAN HOUSEHOLD INCOME across different years or specifications
income_columns = [col for col in acs_pivoted.columns if "MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS" in col]

# Define a function to return the first non-NaN value found across specified columns for a row
def combine_income_columns(row):
    for col in income_columns:
        if not pd.isna(row[col]):
            return row[col]
    return np.nan  # Return NaN if all values are NaN

# Apply the function across the DataFrame rows
acs_pivoted['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'] = acs_pivoted.apply(combine_income_columns, axis=1)

# Optionally, drop the original income columns to clean up the DataFrame
acs_pivoted.drop(columns=income_columns, inplace=True, errors='ignore')
acs_pivoted
```

Out[10]:

variable_concept	geo_id	year	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS	Educational Attainment by Employment Status for the Population 25 to 64 Years	Median Household Income in the Past 12 Months (in 2022 Inflation-Adjusted Dollars)	RACE	Race	TOTAL POPULATION	Total Population	MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS
0	0500000US01001	2009	NaN	NaN	NaN	39215.0	NaN	49584.0	NaN	51463.0
1	0500000US01001	2010	6582.0	NaN	NaN	42031.0	NaN	53155.0	NaN	53255.0
2	0500000US01001	2011	6691.0	NaN	NaN	42577.0	NaN	53944.0	NaN	53899.0
3	0500000US01001	2012	6883.0	NaN	NaN	43084.0	NaN	54590.0	NaN	53773.0
4	0500000US01001	2013	6640.0	NaN	NaN	42997.0	NaN	54907.0	NaN	53682.0
...	...	...	...	...	...	...	...	...	...	...
45084	0500000US72153	2018	5532.0	NaN	NaN	28189.0	NaN	36439.0	NaN	14954.0
45085	0500000US72153	2019	5381.0	NaN	NaN	26607.0	NaN	35428.0	NaN	14743.0
45086	0500000US72153	2020	5224.0	NaN	NaN	24931.0	NaN	34501.0	NaN	14813.0
45087	0500000US72153	2021	5623.0	NaN	NaN	24533.0	NaN	34704.0	NaN	16444.0
45088	0500000US72153	2022	NaN	5843.0	19827.0	NaN	24982.0	NaN	33988.0	NaN

45089 rows × 10 columns

In [11]:
```
# Assuming acs_pivoted is already your DataFrame
acs_pivoted['EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS'] = acs_pivoted['EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 T
acs_pivoted['RACE'] = acs_pivoted['RACE'].combine_first(acs_pivoted['Race'])
acs_pivoted['TOTAL POPULATION'] = acs_pivoted['TOTAL POPULATION'].combine_first(acs_pivoted['Total Population'])
acs_pivoted['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'] = acs_pivoted['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'].combine_first(acs_pivoted['Median Household Income
acs_pivoted.drop(columns=['Educational Attainment by Employment Status for the Population 25 to 64 Years', 'Race', 'Total Population', 'Median Household Income in the Past 12
acs_pivoted.head()
```

Out[11]:

variable_concept	geo_id	year	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS	RACE	TOTAL POPULATION	MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS
0	0500000US01001	2009	NaN	39215.0	49584.0	51463.0
1	0500000US01001	2010	6582.0	42031.0	53155.0	53255.0
2	0500000US01001	2011	6691.0	42577.0	53944.0	53899.0
3	0500000US01001	2012	6883.0	43084.0	54590.0	53773.0
4	0500000US01001	2013	6640.0	42997.0	54907.0	53682.0

In [12]:
```
nan_counts = acs_pivoted.isna().sum()
print(nan_counts)
```

```
variable_concept
geo_id                                                                0
year                                                                  0
EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS    3222
RACE                                                                  0
TOTAL POPULATION                                                      0
MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS                         5
dtype: int64
```

In [13]:
```
nan_distribution_by_year = acs_pivoted.groupby('year').apply(lambda x: x.isna().sum())
nan_distribution_by_year
```

Out[13]:

variable_concept	geo_id	year	EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS	RACE	TOTAL POPULATION	MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS
year						
2009	0	0	3221	0	0	0
2010	0	0	0	0	0	0
2011	0	0	0	0	0	0
2012	0	0	0	0	0	0
2013	0	0	0	0	0	0
2014	0	0	0	0	0	0
2015	0	0	0	0	0	1
2016	0	0	0	0	0	0
2017	0	0	0	0	0	0
2018	0	0	1	0	0	1
2019	0	0	0	0	0	0
2020	0	0	0	0	0	1
2021	0	0	0	0	0	1
2022	0	0	0	0	0	1

In [14]:
```
acs_pivoted['year'].value_counts()
```

Out[14]:
```
year
2022    3222
2009    3221
2010    3221
2011    3221
2012    3221
2013    3221
2020    3221
2021    3221
2014    3220
2015    3220
2016    3220
2017    3220
2018    3220
2019    3220
Name: count, dtype: int64
```

In [15]:
```
acs_pivoted['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'].describe()
```

Out[15]:
```
count     45084.000000
mean      49056.019209
std       14996.032463
min       10499.000000
25%       39623.250000
50%       47023.500000
75%       56117.250000
max      170463.000000
Name: MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS, dtype: float64
```

In [16]:
```
print(acs_pivoted['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'].isnull().sum())
print(acs_pivoted['TOTAL POPULATION'].isnull().sum())
print(acs_pivoted['RACE'].isnull().sum())
print(acs_pivoted['EDUCATIONAL ATTAINMENT BY EMPLOYMENT STATUS FOR THE POPULATION 25 TO 64 YEARS'].isnull().sum())
```

```
5
0
0
3222
```

In [17]:
```
# acs_pivoted.boxplot(column=['MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS'])
# plt.title('Box Plot of MEDIAN HOUSEHOLD INCOME IN THE PAST 12 MONTHS')
# plt.ylabel('Income')
# plt.show()
```

In [18]:
```
# Merge df_zhvi with the pivoted acs_df on 'geo_id' and 'Year'
df_zhvi_enriched = pd.merge(df_zhvi, acs_pivoted, how='left', left_on=['geo_id', 'Year'], right_on=['geo_id', 'year'])

# Drop the extra 'year' column from the merge if it exists
df_zhvi_enriched = df_zhvi_enriched.drop(columns=['year'], errors='ignore')
df_zhvi_enriched
```

Out[18]:

	geo_id	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	...	control	treat5_post	treat10_post	treat15_post	tre
0	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	...	0	0	0	0	
1	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	...	0	0	0	0	
2	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	...	0	0	0	0	
3	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	...	0	0	0	0	
4	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1213240	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	...	0	0	0	0	
1213241	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	...	0	0	0	0	
1213242	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	...	0	0	0	0	
1213243	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	...	0	0	0	0	
1213244	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	...	0	0	0	0	

1213245 rows × 36 columns

In [19]:
```
# Write df_zhvi to a CSV file named 'df_zhvi.csv'
# df_zhvi_enriched.to_csv('zhvi_48nooverlap_ACS_FULL_gdf.csv', index=False)

#conclusion: with this method, we have very little data coverage - need to find alternatives so we have more info
```

In [20]:
```
# Import modules
import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd
from census import Census
from us import states
import os
```

In [21]:
```
# Set API key
c = Census("095fa8aaf42a3003ba7ce37c76a71a2037ab053e")
```

In [22]:
```
#TRY TO GET HOUSING RELATED DATA FROM ACS
old_acs_df = pd.DataFrame()

# B25001_001E   Estimate!!Total HOUSING UNITS
# B25003_002E   Estimate!!Total!!Owner occupied TENURE
# B25017_001E   Estimate!!Total ROOMS
# B25024_001E   Estimate!!Total UNITS IN STRUCTURE
# B25034_001E   Estimate!!Total YEAR STRUCTURE BUILT
# B25035_001E   Estimate!!Median year structure built   MEDIAN YEAR STRUCTURE BUILT
# B25036_013E   Estimate!!Total!!Renter occupied        TENURE BY YEAR STRUCTURE BUILT
# B25041_001E   Estimate!!Total BEDROOMS
# B25044_001E   Estimate!!Total TENURE BY VEHICLES AVAILABLE
# B25047_001E   Estimate!!Total PLUMBING FACILITIES FOR ALL HOUSING UNITS
# B25051_001E   Estimate!!Total KITCHEN FACILITIES FOR ALL HOUSING UNITS

# B25063_001E   Estimate!!Total GROSS RENT
# B25064_001E   Estimate!!Median gross rent     MEDIAN GROSS RENT (DOLLARS)
# B25075_001E   Estimate!!Total Property VALUE

# SELECTED SHORTLIST
# B25001_001E   Estimate!!Total HOUSING UNITS
# B25017_001E   Estimate!!Total ROOMS
# B25034_001E   Estimate!!Total YEAR STRUCTURE BUILT
# B25047_001E   Estimate!!Total PLUMBING FACILITIES FOR ALL HOUSING UNITS
```

In [23]:
```
old_acs_df = pd.DataFrame()

# #B01003_001E : Total Population : B01003_001E
# #B19013_001E : Household Income :  B19001_001E
# #B23006_023E : Bachelor's Degree : B23006_023E
# #B02001_002E : White alone : B02001_002E

# # : Total Population : B01003_001E
# # : Household Income :  B19001_001E
# # : Bachelor's Degree Plus : B16010_041E
# # : White alone : B02001_002E

#ACS 5 Year Coverage: (2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009)
#ACS 1 Year Coverage: (2008, 2007, 2006, 2005)
#PL Coverage: (2000)

#Part 1: 2021-2009
for year in range(2009, 2021):
    temp_data = c.acs5.state_county(fields=('B01003_001E', 'B19001_001E', 'B15002_001E', 'B02001_002E'),
                                            # 'B25001_001E', 'B25017_001E', 'B25034_001E', 'B25047_001E'),
                                    state_fips="*",
                                    county_fips="*",
                                    year=year)
    temp_df = pd.DataFrame(temp_data)
    temp_df['Year'] = year
    old_acs_df = pd.concat([old_acs_df, temp_df], ignore_index=True)
```

```
#Part 2: 2008-2005
for year in range(2005, 2008):
    temp_data = c.acs1.state_county(fields=('B01003_001E', 'B19001_001E', 'B15002_001E', 'B02001_002E'),
                                            # 'B25001_001E', 'B25017_001E', 'B25034_001E', 'B25047_001E'),
                                    state_fips="*",
                                    county_fips="*",
                                    year=year)
    temp_df = pd.DataFrame(temp_data)
    temp_df['Year'] = year
    old_acs_df = pd.concat([old_acs_df, temp_df], ignore_index=True)


old_acs_df = old_acs_df.rename(columns={
    'B01003_001E': 'pop',
    'B19001_001E': 'house_income',
    'B15002_001E': 'bachelors_deg_plus',
    'B02001_002E': 'white',
    # 'B25001_001E': 'units',
    # 'B25017_001E': 'rooms',
    # 'B25034_001E': 'year_built',
    # 'B25047_001E': 'plumbing',
    'Year': 'year'
})

old_acs_df['fips_code'] = old_acs_df['state'] + old_acs_df['county']
old_acs_df
```

Out[23]:

	pop	house_income	bachelors_deg_plus	white	state	county	year	fips_code
0	19695.0	6936.0	12547.0	13512.0	13	091	2009	13091
1	11641.0	3790.0	7824.0	5354.0	13	093	2009	13093
2	95330.0	36755.0	57940.0	31956.0	13	095	2009	13095
3	122657.0	42423.0	76101.0	75365.0	13	097	2009	13097
4	11812.0	4433.0	7485.0	5651.0	13	099	2009	13099
...	...	...	...	...	...	...	...	...
41023	109737.0	42759.0	74013.0	101531.0	34	041	2007	34041
41024	629292.0	252952.0	412054.0	430556.0	35	001	2007	35001
41025	198791.0	69836.0	118178.0	171250.0	35	013	2007	35013
41026	70059.0	18004.0	40129.0	15490.0	35	031	2007	35031
41027	117866.0	39125.0	76063.0	77162.0	35	043	2007	35043

41028 rows × 8 columns

In [24]:
```
old_acs_df['year'].unique()
```

Out[24]:
```
array([2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
       2020, 2005, 2006, 2007])
```

In [25]:
```
with open('data/state_fips.txt', 'r') as file:
    lines = file.readlines()
    data = [line.split() for line in lines]

state_map = pd.DataFrame(data, columns=['Code', 'State', 'space1', 'space2'])

for index, row in state_map.iterrows():
    state_parts = [part for part in [row['State'], row['space1'], row['space2']] if part is not None]
    state_map.at[index, 'State'] = ' '.join(state_parts)

state_map.drop(columns=['space1', 'space2'], inplace=True)

with open('data/county_fips.txt', 'r') as file:
    lines = file.readlines()
    data = [line.split() for line in lines]

county_map = pd.DataFrame(data, columns=['Code', 'County', 'space1', 'space2', 'space3', 'space4', 'space5'])

for index, row in county_map.iterrows():
    state_parts = [part for part in [row['County'], row['space1'], row['space2']] if part is not None]
    county_map.at[index, 'County'] = ' '.join(state_parts)

county_map.drop(columns=['space1', 'space2', 'space3', 'space4', 'space5'], inplace=True)

county_map['State'] = county_map['Code'].str[:2].map(state_map.set_index('Code')['State'])
county_map['name'] = county_map['County'].str.upper() + ', ' + county_map['State']
county_map.drop(columns=['County', 'State'], inplace=True)

county_map
```

Out[25]:

	Code	name
0	01000	ALABAMA, ALABAMA
1	01001	AUTAUGA COUNTY, ALABAMA
2	01003	BALDWIN COUNTY, ALABAMA
3	01005	BARBOUR COUNTY, ALABAMA
4	01007	BIBB COUNTY, ALABAMA
...	...	...
3191	56037	SWEETWATER COUNTY, WYOMING
3192	56039	TETON COUNTY, WYOMING
3193	56041	UINTA COUNTY, WYOMING
3194	56043	WASHAKIE COUNTY, WYOMING
3195	56045	WESTON COUNTY, WYOMING

3196 rows × 2 columns

In [26]:
```
# Merge old_acs_df with county_map based on 'fips_code' and 'Code' columns
old_acs_county_df = pd.merge(old_acs_df, county_map, left_on='fips_code', right_on='Code', how='left')
old_acs_county_df = old_acs_county_df[old_acs_county_df['state'] != '72']
old_acs_county_df.drop(columns=['Code', 'state', 'county'], inplace=True)

old_acs_county_df['name'] = old_acs_county_df['name'].str.replace('ST.', 'SAINT')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('EAST BATON ROUGE, LOUISIANA', 'EAST BATON ROUGE PARISH, LOUISIANA')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('WEST BATON ROUGE, LOUISIANA', 'WEST BATON ROUGE PARISH, LOUISIANA')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('DE WITT COUNTY, ILLINOIS', 'DEWITT COUNTY, ILLINOIS')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('PRINCE GEORGE, MARYLAND', 'PRINCE GEORGES COUNTY, MARYLAND')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('DE KALB COUNTY, INDIANA', 'DEKALB COUNTY, INDIANA')
old_acs_county_df['name'] = old_acs_county_df['name'].replace('SAN LUIS OBISPO, CALIFORNIA', 'SAN LUIS OBISPO COUNTY, CALIFORNIA')

conditions = [
    {'year': 2009, 'name': "WHITFIELD COUNTY, GEORGIA"},
    {'year': 2009, 'name': "HAMILTON COUNTY, TENNESSEE"},
```

```
    {'year': 2009, 'name': "BRADLEY COUNTY, TENNESSEE"},
    {'year': 2009, 'name': "KERN COUNTY, CALIFORNIA"},
    {'year': 2009, 'name': "WASHINGTON COUNTY, OREGON"},
    {'year': 2009, 'name': "CLACKAMAS COUNTY, OREGON"},
    {'year': 2009, 'name': "MULTNOMAH COUNTY, OREGON"},
    {'year': 2009, 'name': "DOUGLAS COUNTY, OREGON"},
    {'year': 2009, 'name': "CLARK COUNTY, WASHINGTON"},
]

for condition in conditions:
    row_to_duplicate = old_acs_county_df[
        (old_acs_county_df['year'] == condition['year']) &
        (old_acs_county_df['name'] == condition['name'])
    ].copy()

    if not row_to_duplicate.empty:
        row_to_duplicate['year'] = 2008
        old_acs_county_df = pd.concat([old_acs_county_df, row_to_duplicate], ignore_index=True)

rows_2005 = old_acs_county_df[old_acs_county_df['year'] == 2005].copy()
duplicated_rows = pd.DataFrame()

# Iterate over the range of years 2000 to 2004
for year in range(2000, 2005):
    temp_rows = rows_2005.copy()  # Copy the 2005 rows
    temp_rows['year'] = year  # Set the year to the current iteration year
    duplicated_rows = pd.concat([duplicated_rows, temp_rows], ignore_index=True)

# Append the duplicated rows back to the original DataFrame
old_acs_county_df = pd.concat([old_acs_county_df, duplicated_rows], ignore_index=True)

rows_2009 = old_acs_county_df[old_acs_county_df['year'] == 2009].copy()

fips_codes_2009 = rows_2009['fips_code'].unique()

earliest_fips = []
for fips_code in fips_codes_2009:
    years = old_acs_county_df[old_acs_county_df['fips_code'] == fips_code]['year'].unique()
    if min(years) == 2009:
        earliest_fips.append(fips_code)

# Step 2: Duplicate these rows for each year from 2000 to 2008
duplicated_rows = pd.DataFrame()
for fips_code in earliest_fips:
    # Filter rows for each fips_code from 2009
    temp_rows = old_acs_county_df[(old_acs_county_df['fips_code'] == fips_code) & (old_acs_county_df['year'] == 2009)].copy()
    for year in range(2000, 2009):  # From 2000 to 2008
        temp_copy = temp_rows.copy()
        temp_copy['year'] = year
        duplicated_rows = pd.concat([duplicated_rows, temp_copy], ignore_index=True)

# Step 3: Append the duplicated rows back to the original DataFrame
old_acs_county_df = pd.concat([old_acs_county_df, duplicated_rows], ignore_index=True)

# Filter rows for the year 2020
rows_2020 = old_acs_county_df[old_acs_county_df['year'] == 2020].copy()

# Empty DataFrame to hold duplicated rows
duplicated_rows = pd.DataFrame()

# Iterate over the range of years 2021 to 2023
for year in range(2021, 2024):
    temp_rows = rows_2020.copy()  # Copy the 2020 rows
    temp_rows['year'] = year  # Set the year to the current iteration year
    duplicated_rows = pd.concat([duplicated_rows, temp_rows], ignore_index=True)

# Append the duplicated rows back to the original DataFrame
old_acs_county_df = pd.concat([old_acs_county_df, duplicated_rows], ignore_index=True)

old_acs_county_df
```

Out[26]:

	pop	house_income	bachelors_deg_plus	white	year	fips_code	name
0	19695.0	6936.0	12547.0	13512.0	2009	13091	DODGE COUNTY, GEORGIA
1	11641.0	3790.0	7824.0	5354.0	2009	13093	DOOLY COUNTY, GEORGIA
2	95330.0	36755.0	57940.0	31956.0	2009	13095	DOUGHERTY COUNTY, GEORGIA
3	122657.0	42423.0	76101.0	75365.0	2009	13097	DOUGLAS COUNTY, GEORGIA
4	11812.0	4433.0	7485.0	5651.0	2009	13099	EARLY COUNTY, GEORGIA
...	...	...	...	...	...	...	...
74550	14572.0	6032.0	10192.0	12983.0	2023	27129	RENVILLE COUNTY, MINNESOTA
74551	15259.0	5940.0	10375.0	13960.0	2023	27135	ROSEAU COUNTY, MINNESOTA
74552	96015.0	32791.0	62908.0	87454.0	2023	27141	SHERBURNE COUNTY, MINNESOTA
74553	36710.0	14836.0	24704.0	33146.0	2023	27147	STEELE COUNTY, MINNESOTA
74554	24603.0	9888.0	17027.0	22681.0	2023	27153	TODD COUNTY, MINNESOTA

74555 rows × 7 columns

In [27]:
```
unique_years = old_acs_county_df['year'].unique()

# If you want to convert this array into a Python list
unique_years_list = list(unique_years)

# Printing the list of unique years
print(unique_years_list)
```

```
[2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2005, 2006, 2007, 2008, 2000, 2001, 2002, 2003, 2004, 2021, 2022, 2023]
```

In [28]:
```
df_zhvi = df_zhvi.rename(columns={'Year': 'year'})
df_zhvi
```

Out[28]:

	geo_id	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	ZHVI	...	treatment_0_5	treatment_5_10	treatment_10_15	tr
0	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	129682.239746	...	0	1	0	
1	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	129536.988965	...	0	1	0	
2	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	129540.688258	...	0	1	0	
3	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	129663.344449	...	0	1	0	
4	0500000US25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	130432.902616	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	305106.660169	...	0	0	0	
1213241	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	306547.972216	...	0	0	0	
1213242	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	308136.681574	...	0	0	0	
1213243	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	308821.909108	...	0	0	0	
1213244	0500000US02090	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	307830.967624	...	0	0	0	

1213245 rows × 32 columns

In [29]:
```
old_acs_county_df['name'] = old_acs_county_df['name'].str.upper().str.strip()
df_zhvi['name'] = df_zhvi['name'].str.upper().str.strip()

# Use pd.merge with 'how=left' to keep all rows from df_zhvi and add information from old_acs_county_df
merged_df = pd.merge(df_zhvi, old_acs_county_df, on=['name', 'year'], how='left')

# Assuming you want to re-order columns to move 'geo_id', 'fips_code' to the front
new_column_order = ['geo_id', 'fips_code'] + [col for col in merged_df.columns if col not in ['geo_id', 'fips_code']]
merged_df = merged_df[new_column_order]
merged_df
```

Out[29]:

	geo_id	fips_code	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	...	control	treat5_post	treat10_post	treat15_post	treat20_
0	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	...	0	0	0	0	
1	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	...	0	0	0	0	
2	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	...	0	0	0	0	
3	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	...	0	0	0	0	
4	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	...	0	0	0	0	
1213241	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	...	0	0	0	0	
1213242	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	...	0	0	0	0	
1213243	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	...	0	0	0	0	
1213244	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	...	0	0	0	0	

1213245 rows × 37 columns

In [30]:
```
merged_df['pop'].isna().mean()
```

Out[30]: 0.03784973356576784

In [31]:
```
merged_df['house_income'].isna().mean()
```

Out[31]: 0.03784973356576784

In [32]:
```
print(merged_df['nearest_store'].nunique())
print(merged_df['quarters_from_opening'].unique())
```

```
120
[-18 -17 -16 -15 -14 -13 -12 -11 -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83 -57 -56 -55 -54 -53 -52
 -51 -50 -49 -48 -47 -46 -45 -44 -43 -42 -41 -40 -39 -38 -37 -36 -35 -34
 -33 -32 -31 -30 -29 -28 -27 -26 -25 -24 -23 -22 -21 -20 -19 -59 -58 -61
 -60  84  85  86  87  88 -75 -74 -73 -72 -71 -70 -69 -68 -67 -66 -65 -64
 -63 -62 -77 -76  89  90  91 -87 -86 -85 -84 -83 -82 -81 -80 -79 -78 -91
 -90 -89 -88 -93 -92 -94  92]
```

In [33]:
```
merged_df['year'].unique()
```

Out[33]:
```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021,
       2022, 2023])
```

In [53]:
```
#To Calculate Density, we must find the land area based on the shape file (no var in ACS)
import os
os.environ['SHAPE_RESTORE_SHX'] = 'YES'

counties = gpd.read_file('data/cb_2013_us_county_500k.shp')

# Calculate land area for each county
counties['land_area_sqm'] = counties.geometry.area  # Area is in square meters

geometry_series = merged_df.apply(lambda row: Point(float(row['longitude']), float(row['latitude'])), axis=1)
merged_gdf = gpd.GeoDataFrame(merged_df, geometry=geometry_series)
merged_gdf.crs = counties.crs
merged_with_counties = gpd.sjoin(merged_gdf, counties[['geometry', 'land_area_sqm']], how='left', predicate='intersects')

merged_with_counties.drop(columns=['index_right'], inplace=True)
merged_with_counties['pop_density'] = merged_with_counties['pop'] / (merged_with_counties['land_area_sqm']*100_000_000)

merged_with_counties
```

Out[53]:

	geo_id	fips_code	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	...	treat10_post	treat15_post	treat20_post	control_post	
0	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	...	0	0	0	0	4
1	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	...	0	0	0	0	4
2	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	...	0	0	0	0	4
3	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	...	0	0	0	0	4
4	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	...	0	0	0	0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	...	0	0	1	0	
1213241	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	...	0	0	1	0	
1213242	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	...	0	0	1	0	
1213243	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	...	0	0	1	0	
1213244	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	...	0	0	1	0	

1213245 rows × 39 columns

In [35]:
```
# merged_with_counties.to_csv("ext_zhvi.csv")
```

In [36]:
```
# merged_with_counties = merged_with_counties.drop(['income_per_capita', 'bachelors_plus_per_capita', 'white_per_capita'], axis=1)

merged_with_counties.to_csv('zhvi_50_est.csv', index=False)

merged_with_counties
```

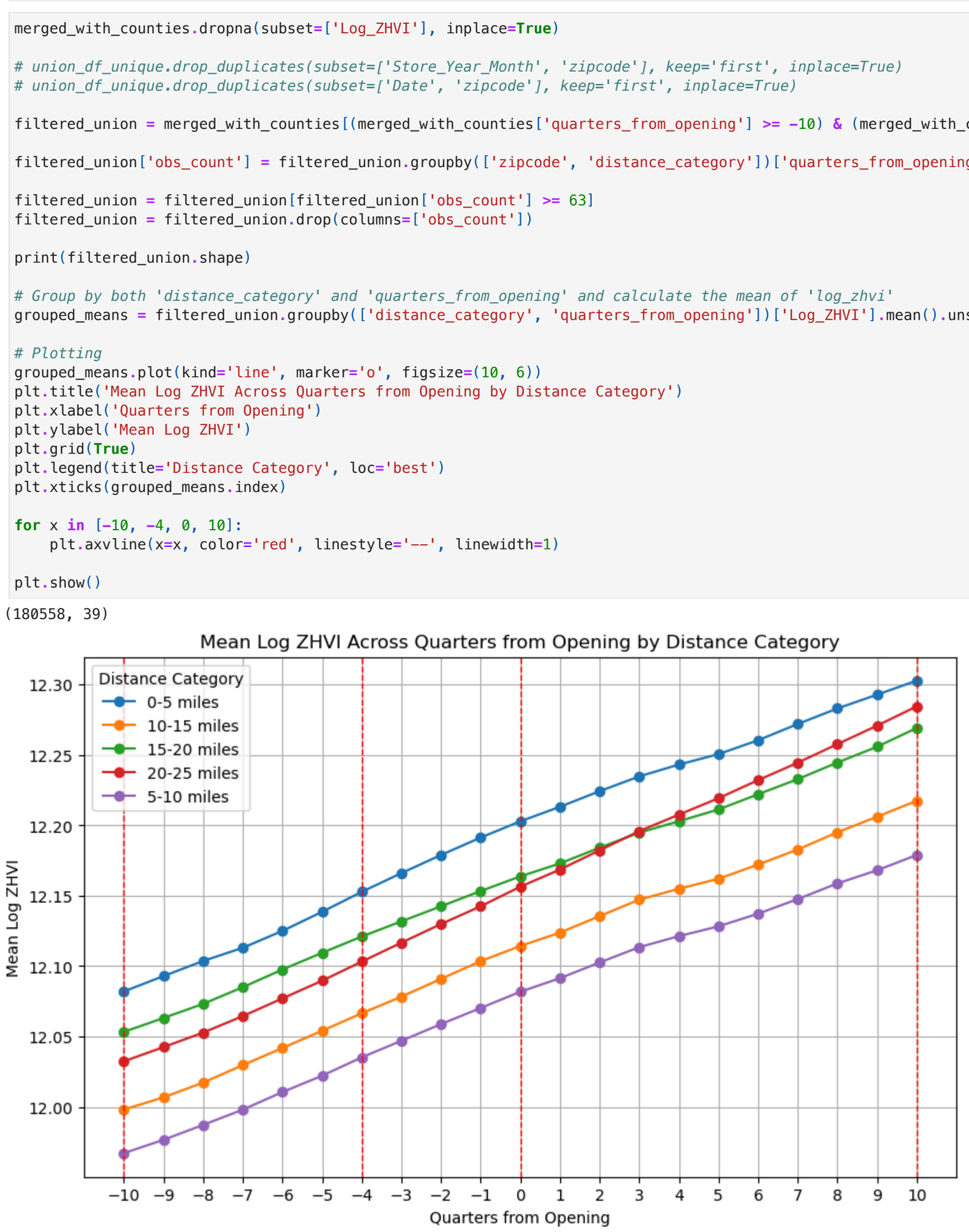
Out[36]:

	geo_id	fips_code	name	state_full	City	CountyName	zipcode	latitude	longitude	Date	...	treat10_post	treat15_post	treat20_post	control_post	
0	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-01-31	...	0	0	0	0	4
1	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-02-29	...	0	0	0	0	4
2	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-03-31	...	0	0	0	0	4
3	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-04-30	...	0	0	0	0	4
4	0500000US25013	25013	HAMPDEN COUNTY, MASSACHUSETTS	Massachusetts	Agawam	Hampden County	1001	42.06262	-72.62521	2000-05-31	...	0	0	0	0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1213240	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-05-31	...	0	0	1	0	
1213241	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-06-30	...	0	0	1	0	
1213242	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-07-31	...	0	0	1	0	
1213243	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-08-31	...	0	0	1	0	
1213244	0500000US02090	NaN	FAIRBANKS NORTH STAR BOROUGH, ALASKA	Alaska	Fairbanks	Fairbanks North Star Borough	99709	64.87402	-148.22458	2023-09-30	...	0	0	1	0	

1213245 rows × 39 columns

In [37]:
```
merged_with_counties.dropna(subset=['Log_ZHVI'], inplace=True)

# union_df_unique.drop_duplicates(subset=['Store_Year_Month', 'zipcode'], keep='first', inplace=True)
# union_df_unique.drop_duplicates(subset=['Date', 'zipcode'], keep='first', inplace=True)

filtered_union = merged_with_counties[(merged_with_counties['quarters_from_opening'] >= -10) & (merged_with_counties['quarters_from_opening'] <= 10)].copy()

filtered_union['obs_count'] = filtered_union.groupby(['zipcode', 'distance_category'])['quarters_from_opening'].transform('count')

filtered_union = filtered_union[filtered_union['obs_count'] >= 63]
filtered_union = filtered_union.drop(columns=['obs_count'])

print(filtered_union.shape)

# Group by both 'distance_category' and 'quarters_from_opening' and calculate the mean of 'log_zhvi'
grouped_means = filtered_union.groupby(['distance_category', 'quarters_from_opening'])['Log_ZHVI'].mean().unstack(level=0)

# Plotting
grouped_means.plot(kind='line', marker='o', figsize=(10, 6))
plt.title('Mean Log ZHVI Across Quarters from Opening by Distance Category')
plt.xlabel('Quarters from Opening')
plt.ylabel('Mean Log ZHVI')
plt.grid(True)
plt.legend(title='Distance Category', loc='best')
plt.xticks(grouped_means.index)

for x in [-10, -4, 0, 10]:
    plt.axvline(x=x, color='red', linestyle='--', linewidth=1)

plt.show()
```

```
(180558, 39)
```

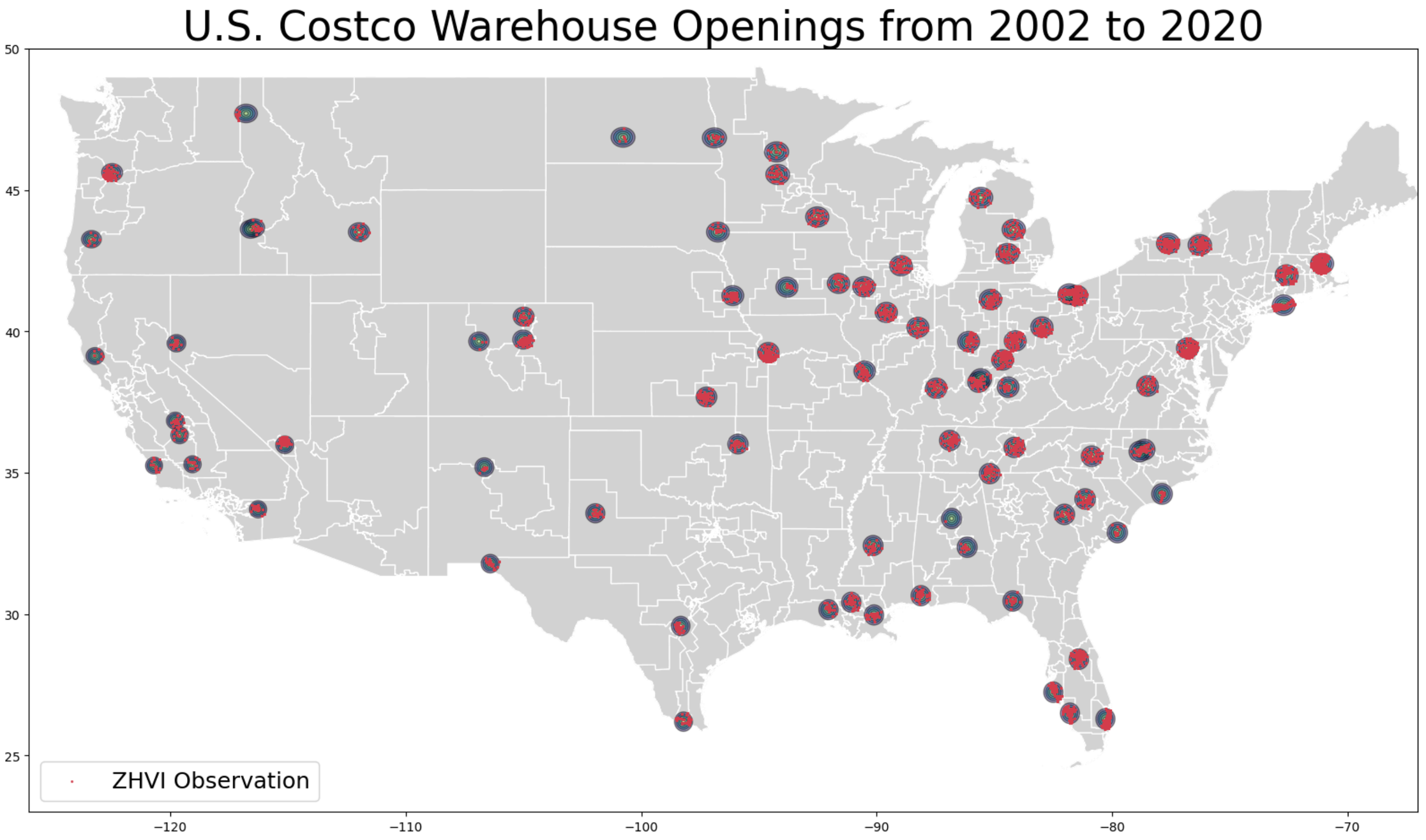


In [38]:
```
filtered_union['distance_category'].value_counts()
```

Out[38]:
```
distance_category
5-10 miles     43029
15-20 miles    40068
10-15 miles    39060
20-25 miles    38241
0-5 miles      20160
Name: count, dtype: int64
```

In [39]:
```
import matplotlib.pyplot as plt
import geopandas as gpd
```

```
import matplotlib.patches as mpatches

unique_nearest_stores = filtered_union['nearest_store'].unique()

filtered_openings = df_openings[df_openings['store_name'].isin(unique_nearest_stores)]

shapefile_path = '/Users/silas/ECON495/PROJ/cb_2022_us_cd118_5m.shp'
map_df = gpd.read_file(shapefile_path)

df_openings_geo = gpd.GeoDataFrame(
    filtered_openings,
    geometry=gpd.points_from_xy(filtered_openings.longitude, filtered_openings.latitude)
)

# Set the CRS for WGS 84 (EPSG:4326) and then convert to UTM Zone 18N
df_openings_geo.set_crs(epsg=4326, inplace=True)
df_openings_geo_utm = df_openings_geo.to_crs(epsg=32618)  # UTM Zone 18N

# Sorting the radii in descending order to ensure the larger buffers are drawn first
radii = sorted([5, 10, 15, 20, 25], reverse=True)

# Plotting the shapefile
fig, ax = plt.subplots(1, 1, figsize=(20, 12))
map_df.plot(ax=ax, edgecolor='white', facecolor='lightgrey')


# Plotting the buffers in descending order of their radii
colors = ['#5e4fa2', '#3288bd', '#66c2a5', '#abdda4', '#e6f598', '#fee08b', '#fdae61', '#f46d43']
radii_legend_patches = []

for radius, color in zip(radii, colors):
    # Buffer creation and transformation to WGS 84 CRS
    buffer = df_openings_geo_utm.geometry.buffer(radius * 1609.34).to_crs(epsg=4326)
    buffer.plot(ax=ax, alpha=0.5, edgecolor='black', facecolor=color)
    # Add patch to the list for the legend
    radii_legend_patches.append(mpatches.Patch(color=color, label=f'{radius} miles buffer'))

# Plotting the original points with label for legend
filtered_union.plot(ax=ax, marker='o', color='#d53e4f', markersize=1, label='ZHVI Observation')

# Adjusting aspect ratio, setting the title, etc.
ax.set_xlim(-126, -67)  # Adjust as needed
ax.set_ylim(23, 50)    # Adjust as needed

# Adjusting aspect ratio
ax.set_aspect(aspect=1.2)

# Setting the title
ax.set_title('U.S. Costco Warehouse Openings from 2002 to 2020', fontsize = 33)
# Adding the legend for ZHVI observations
plt.legend(loc='lower left', bbox_to_anchor=(0, 0), fontsize=18, title_fontsize=20)



# Adding back the first legend which was removed by the second legend call
plt.show()
```



In [40]:
```
filtered_openings
```

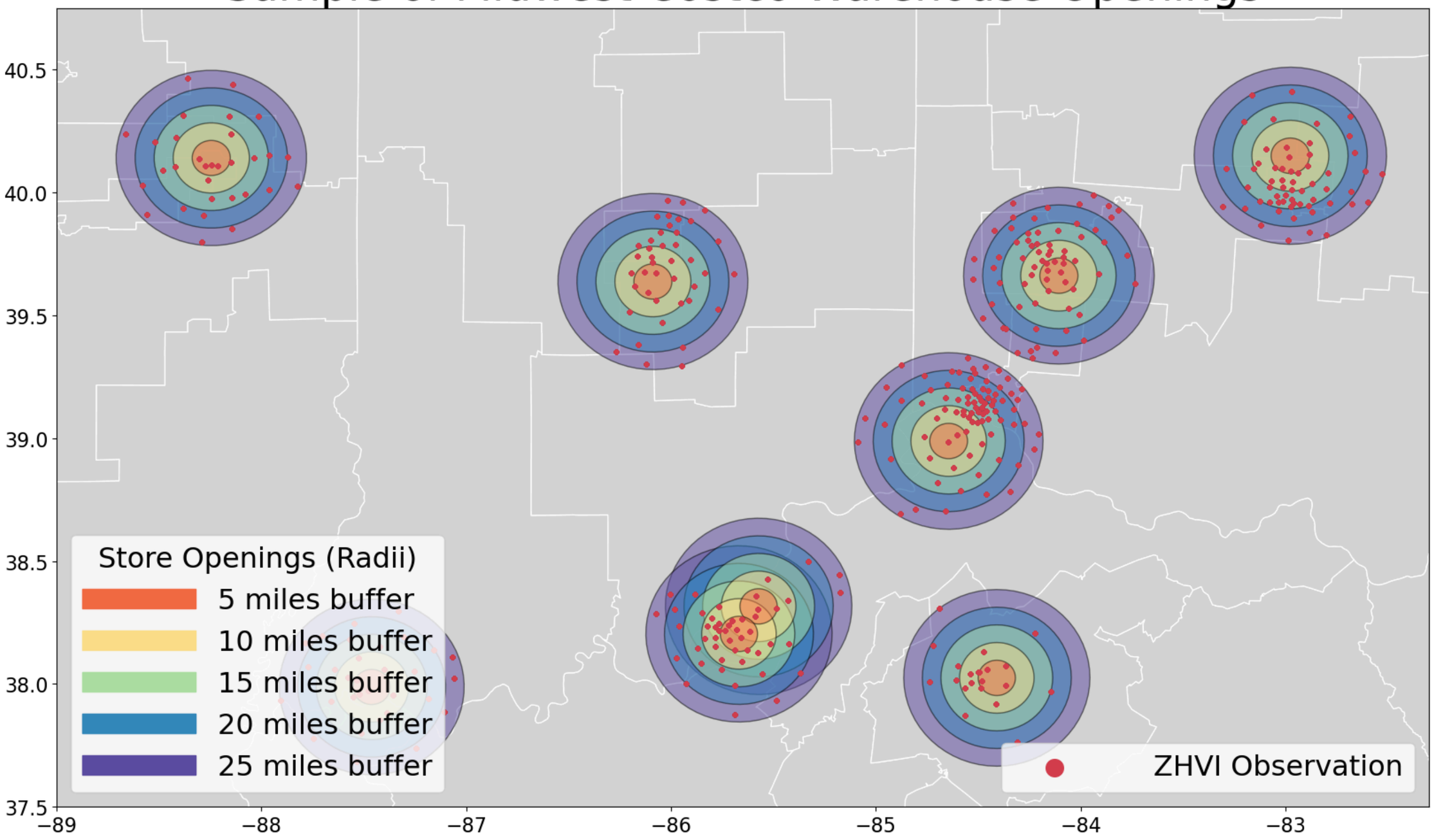
Out[40]:

	State	City	store_name	zipcode	latitude	longitude	opening_date	geometry	50_overlapping_within_5_years
13	MA	Everett	Everett MA	02149-2428	42.397	-71.072	2002-11-21	POINT (823301.0768518804 4701334.701357687)	0
14	CA	Fresno	N Fresno	93720-2920	36.839	-119.788	2002-11-26	POINT (-3575643.563736804 5159992.443480482)	0
15	TX	El Paso	El Paso	79925-3416	31.783	-106.410	2003-11-13	POINT (-2537863.4108963246 3984197.1846671025)	0
16	AL	Hoover	Hoover	35244-2346	33.379	-86.812	2003-11-20	POINT (-601760.4821663199 3756327.0903135277)	0
18	ID	Coeur D' Alene	Coeur D' Alene	83815-3723	47.709	-116.783	2004-04-22	POINT (-2582894.3815328344 6192016.141085387)	0
...	...	...	...	...	...	...	...	...	...
96	ID	Idaho Falls	Idaho Falls	83401	43.513	-111.985	2020-08-14	POINT (-2486460.583367863 5532639.746337421)	0
97	ND	Bismarck	Bismarck	58503-6581	46.865	-100.770	2020-08-25	POINT (-1457710.2009736325 5522315.456149072)	0
98	IL	Champaign	Champaign	61820	40.141	-88.245	2020-10-22	POINT (-629910.5171052357 4528428.830565146)	0
99	ID	Meridian	Meridian	83646	43.661	-116.436	2020-10-30	POINT (-2829570.7596595585 5747176.698602446)	0
100	MI	Midland	Midland	48642	43.603	-84.170	2020-11-12	POINT (-240229.00550479593 4868813.3301279275)	0

85 rows × 9 columns

In [85]:

```
import matplotlib.pyplot as plt
import geopandas as gpd
import matplotlib.patches as mpatches

unique_nearest_stores = filtered_union['nearest_store'].unique()

filtered_openings = df_openings[df_openings['store_name'].isin(unique_nearest_stores)]

shapefile_path = '/Users/silas/ECON495/PROJ/cb_2022_us_cd118_5m.shp'
map_df = gpd.read_file(shapefile_path)

df_openings_geo = gpd.GeoDataFrame(
    filtered_openings,
    geometry=gpd.points_from_xy(filtered_openings.longitude, filtered_openings.latitude)
)

# Set the CRS for WGS 84 (EPSG:4326) and then convert to UTM Zone 18N
df_openings_geo.set_crs(epsg=4326, inplace=True)
df_openings_geo_utm = df_openings_geo.to_crs(epsg=32618)  # UTM Zone 18N

# Sorting the radii in descending order to ensure the larger buffers are drawn first
radii = sorted([5, 10, 15, 20, 25], reverse=True)

plt.rcParams.update({'font.size': 16, 'legend.title_fontsize': 'x-large'})

# Plotting the shapefile
fig, ax = plt.subplots(1, 1, figsize=(20, 12))
map_df.plot(ax=ax, edgecolor='white', facecolor='lightgrey')


# Plotting the buffers in descending order of their radii
colors = ['#5e4fa2', '#3288bd', '#abdda4', '#fee08b', '#f46d43']
radii_legend_patches = []

for radius, color in zip(radii, colors):
    # Buffer creation and transformation to WGS 84 CRS
    buffer = df_openings_geo_utm.geometry.buffer(radius * 1609.34).to_crs(epsg=4326)
    buffer.plot(ax=ax, alpha=0.5, edgecolor='black', facecolor=color)
    # Add patch to the list for the legend
    radii_legend_patches.append(mpatches.Patch(color=color, label=f'{radius} miles buffer'))

radii_legend_patches.reverse()

# Plotting the original points with label for legend
filtered_union.plot(ax=ax, marker='o', color='#d53e4f', markersize=8, label='ZHVI Observation')

ax.set_xlim(-89, -82.3)  # Longitudes from -104°W to -80°W
ax.set_ylim(37.5, 40.75)     # Latitudes from 36°N to 49°N

# Adjusting aspect ratio
ax.set_aspect(aspect=1.2)

# Setting the title
ax.set_title('Sample of Midwest Costco Warehouse Openings', fontsize=35)

# Adding the legend
# Adding the legend for store openings
store_openings_legend = plt.legend(handles=radii_legend_patches,
                                   title="Store Openings (Radii)",
                                   loc='lower left',
                                   bbox_to_anchor=(0, 0),
                                   fontsize='x-large',
                                   title_fontsize='x-large',
                                   handlelength=4,
                                   handletextpad=0.8,
                                   markerscale=5)

# Adding the legend for ZHVI observations
plt.legend(loc='lower right',
           bbox_to_anchor=(1, 0),
           fontsize='x-large',
           title_fontsize='x-large',
           handlelength=3,
           handletextpad=2,
           markerscale=5)

# Adding back the first legend which was removed by the second legend call
ax.add_artist(store_openings_legend)

plt.show()
```



In [42]:
```
# ZHVI data from 2000-01-31 to 2023-09-30
# Costco opening date from 2002-11-21 to 2020-11-12
```

In [43]:
```
merged_with_counties.shape
```

Out[43]: (1073191, 39)

In [54]:
```
# Increase the 'quarters_from_opening' value by 3 years (12 quarters)
merged_with_counties_placebo = merged_with_counties
merged_with_counties_placebo.loc[:, 'quarters_from_opening'] += 12
merged_with_counties_placebo.loc[:, 'months_from_opening'] += 36
merged_with_counties_placebo.loc[:, 'post_treatment'] = np.where(merged_with_counties_placebo['months_from_opening'] > 0, 1, 0)

merged_with_counties.to_csv('plus3yrs_zhvi_50_complete_gdf.csv', index=False)
```

In [55]:
```
# Increase the 'quarters_from_opening' value by 2.5 years (10 quarters)
merged_with_counties_placebo = merged_with_counties
merged_with_counties_placebo.loc[:, 'quarters_from_opening'] -= 2
merged_with_counties_placebo.loc[:, 'months_from_opening'] -= 6
merged_with_counties_placebo.loc[:, 'post_treatment'] = np.where(merged_with_counties_placebo['months_from_opening'] > 0, 1, 0)

merged_with_counties_placebo.to_csv('plus2.5yrs_zhvi_50_complete_gdf.csv', index=False)
```

In [56]:
```
# Increase the 'quarters_from_opening' value by 2 years (8 quarters)
merged_with_counties_placebo = merged_with_counties
merged_with_counties_placebo.loc[:, 'quarters_from_opening'] -= 2
merged_with_counties_placebo.loc[:, 'months_from_opening'] -= 6
merged_with_counties_placebo.loc[:, 'post_treatment'] = np.where(merged_with_counties_placebo['months_from_opening'] > 0, 1, 0)

merged_with_counties_placebo.to_csv('plus2yrs_zhvi_50_complete_gdf.csv', index=False)
```

In [ ]:
```
```