In [2]:
```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
#import geoplot as gplt
import contextily
```

In [3]:
```
# Reading in the data from the path
locs_pdf = pd.read_csv('OSM_DollarGeneralLocs.csv')

# Converting the pandas dataframe into a geopandas geodataframe
locs_gdf = gpd.GeoDataFrame(
    locs_pdf, geometry=gpd.points_from_xy(locs_pdf.X, locs_pdf.Y),
    crs="EPSG:4326"
)

# Resetting the index and creating a synthetic ID field
locs_gdf.reset_index(inplace=True)
locs_gdf.rename(columns={'index':'ID'}, inplace=True)
```

In [4]:
```
# To create a buffer, we first need to convert from a g-crs to a p-crs
locs_gdf = locs_gdf.to_crs(3005)

# Next, create aggregation area around each store
buffer_size_mi = 5
buffer_size_m = buffer_size_mi * 1609.344 # meters in a mile

# Creating a copy of the original dataframe to operate on
locs_gdf_buffer = locs_gdf.copy()

# Performing the buffer operation
locs_gdf_buffer["buffer_5mi"] = locs_gdf.buffer(buffer_size_m)

locs_gdf_buffer[['ID','geometry','X','Y','buffer_5mi']].head()
```

Out[4]:

	ID	geometry	X	Y	buffer_5mi
0	0	POINT (4662241.144 445480.319)	-82.458599	38.428581	POLYGON ((4670287.864 445480.319, 4670249.117 ...
1	1	POINT (4667682.555 450628.186)	-82.375886	38.438389	POLYGON ((4675729.275 450628.186, 4675690.528 ...
2	2	POINT (4610865.116 491258.202)	-82.633850	39.047795	POLYGON ((4618911.836 491258.202, 4618873.089 ...
3	3	POINT (4562888.418 588383.044)	-82.443563	40.034121	POLYGON ((4570935.138 588383.044, 4570896.391 ...
4	4	POINT (4519080.712 666961.268)	-82.332082	40.856050	POLYGON ((4527127.432 666961.268, 4527088.684 ...

In [5]:
```
# Joining the buffer to the store locations table
joined = gpd.sjoin(
```

```
    # Right table is the raw store locations data
    locs_gdf,
    # Left table is that of the buffers around the stores
    locs_gdf_buffer.set_geometry("buffer_5mi")[["ID", "buffer_5mi"]],
    # The operation, or spatial predicate, you'll use is `within`
    predicate="within"
)
```

In [6]:
```
# store count
store_count = (
    joined.groupby(
        "ID_left"
    )
    .count()
)

# Converting to a dataframe and cleaning up
store_count_df = store_count.reset_index()
store_count_df = store_count_df[['ID_left','ID_right']]
store_count_df.columns=['ID','Store_Count']

store_count_df.head()
```

Out[6]:

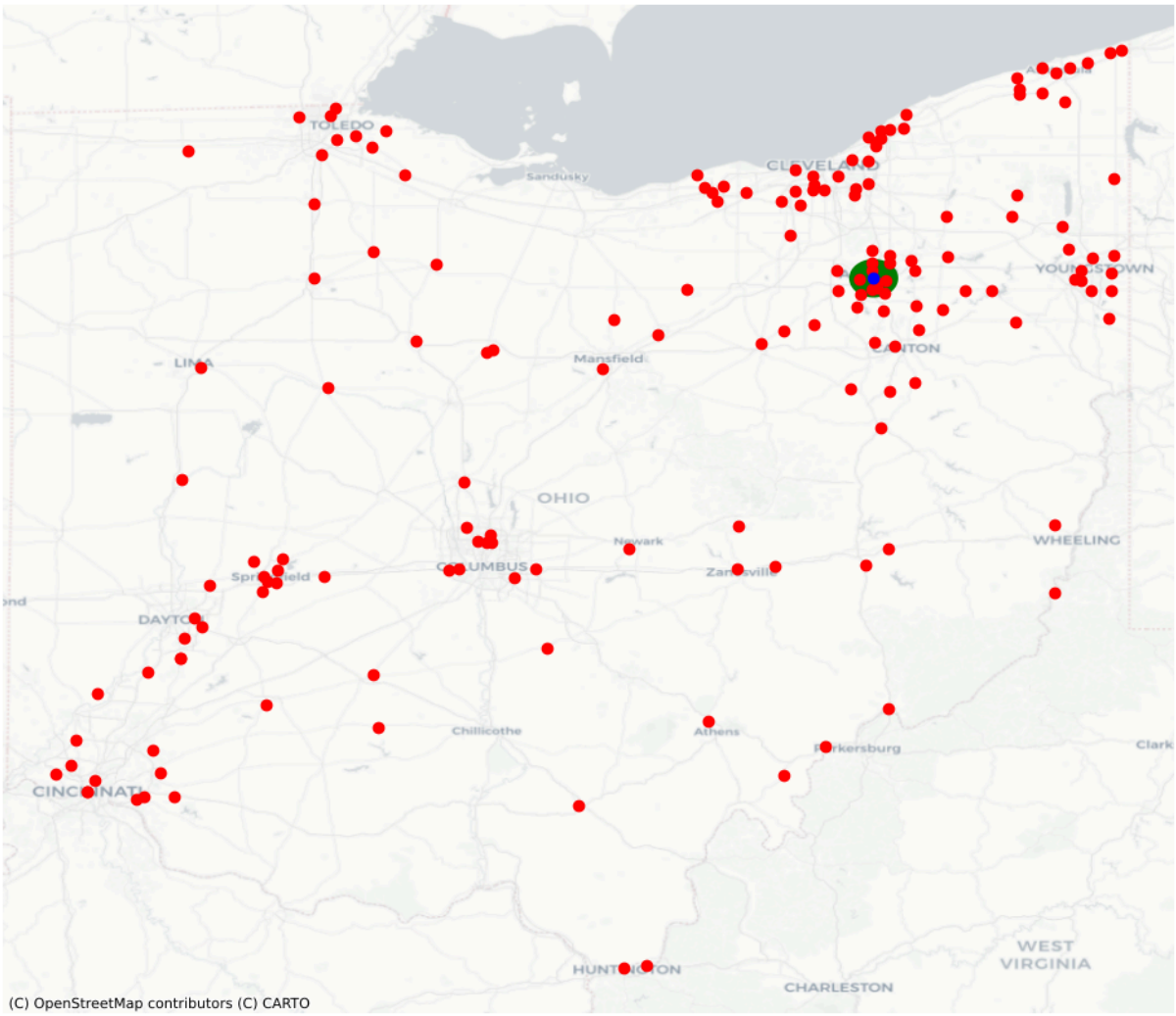
	ID	Store_Count
0	0	2
1	1	2
2	2	1
3	3	1
4	4	1

In [17]:
```
store_count_df.groupby("ID")["Store_Count"].sum().nlargest()
```

Out[17]:
```
ID
55    10
51     9
52     9
53     9
57     9
Name: Store_Count, dtype: int64
```

In [19]:
```
# Changing CRS to make mapping cleaner
locs_gdf_buffer = locs_gdf_buffer.set_geometry("buffer_5mi")[["ID", "buffer_5mi"]]
locs_gdf_buffer = locs_gdf_buffer.to_crs(4326)
locs_gdf = locs_gdf.to_crs(4326)

# Set up figure and axis
f, ax = plt.subplots(1, figsize=(12, 12))

# Plot Buffer around Store ID 45 in green
locs_gdf_buffer[locs_gdf_buffer['ID']==55].plot(ax=ax,color="g")
```

```
# Plot all stores in red
locs_gdf.plot(ax=ax, color="r")

# Plot store ID 2 in blue
locs_gdf[locs_gdf['ID']==55].plot(ax=ax,color="b")

# Add Stamen's Toner basemap
contextily.add_basemap(
    ax,
    crs=locs_gdf.crs.to_string(),
    source=contextily.providers.CartoDB.Positron
)
# Remove axes
ax.set_axis_off()
# Display
plt.show()
```



In [20]:
```
number_55 = locs_gdf_buffer[locs_gdf_buffer['ID']==55]
```

In [21]:
```
number_55.head()
```

Out[21]:

	ID	buffer_5mi
55	55	POLYGON ((-81.42660 41.02741, -81.43238 41.021...

In [ ]: