

```
In [2]: import geopandas as gpd
import pandas as pd
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import seaborn as sns

In [3]: engine = create_engine('postgresql://postgres:MelonSK998@localhost:5432/Poligonos')

In [4]: def cargar_tabla(nombre_tabla):
    query = f"SELECT * FROM {nombre_tabla};"
    gdf = gpd.read_postgis(query, engine, geom_col='geom') # Cambiar 'geometry' por 'geom'
    return gdf

In [5]: gdf_municipio = cargar_tabla("municipio")
gdf_colonia = cargar_tabla("colonia")
gdf_ageb = cargar_tabla("ageb")
gdf_manzana = cargar_tabla("manzana")

In [6]: # Información general
print("Información de Municipio:")
gdf_municipio.info()

print("\nInformación de Colonia:")
gdf_colonia.info()

print("\nInformación de AGEB:")
gdf_ageb.info()

print("\nInformación de Manzana:")
gdf_manzana.info()

# Estadísticas descriptivas
print("Estadísticas de Manzana:")
print(gdf_manzana.describe())

Información de Municipio:
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   gid         1 non-null     int64  
 1   cvegeo     1 non-null     object 
 2   cve_ent     1 non-null     object 
 3   cve_mun     1 non-null     object 
 4   nomgeo     1 non-null     object 
 5   geom        1 non-null     geometry
dtypes: geometry(1), int64(1), object(4)
memory usage: 180.0+ bytes

Información de Colonia:
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   gid         153 non-null    int64  
 1   id_colonia  153 non-null    object 
 2   nombre_col   153 non-null    object 
 3   geom        153 non-null    geometry
dtypes: geometry(1), int64(1), object(2)
memory usage: 4.9+ KB

Información de AGEB:
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   gid         167 non-null    int64  
 1   id_ageb    167 non-null    object 
 2   geom        167 non-null    geometry
dtypes: geometry(1), int64(1), object(1)
memory usage: 4.0+ KB

Información de Manzana:
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 4813 entries, 0 to 4812
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   gid         4813 non-null   int64  
 1   id_manzana  4813 non-null   object 
 2   geom        4813 non-null   geometry
dtypes: geometry(1), int64(1), object(1)
memory usage: 112.9+ KB
Estadísticas de Manzana:
   gid
count  4813.000000
mean   2407.000000
std    1389.537753
min    1.000000
25%   1204.000000
50%   2407.000000
```

```
75%    3610.000000
max    4813.000000
```

```
In [7]: # Valores nulos
print("Valores nulos en Municipio:")
print(gdf_municipio.isnull().sum())

print("\nValores nulos en Colonia:")
print(gdf_colonia.isnull().sum())

print("\nValores nulos en AGEB:")
print(gdf_ageb.isnull().sum())

print("\nValores nulos en Manzana:")
print(gdf_manzana.isnull().sum())
```

```
Valores nulos en Municipio:
```

```
gid      0
cvegeo   0
cve_ent   0
cve_mun   0
nomgeo   0
geom      0
dtype: int64
```

```
Valores nulos en Colonia:
```

```
gid      0
id_colonia  0
nombre_col  0
geom      0
dtype: int64
```

```
Valores nulos en AGEB:
```

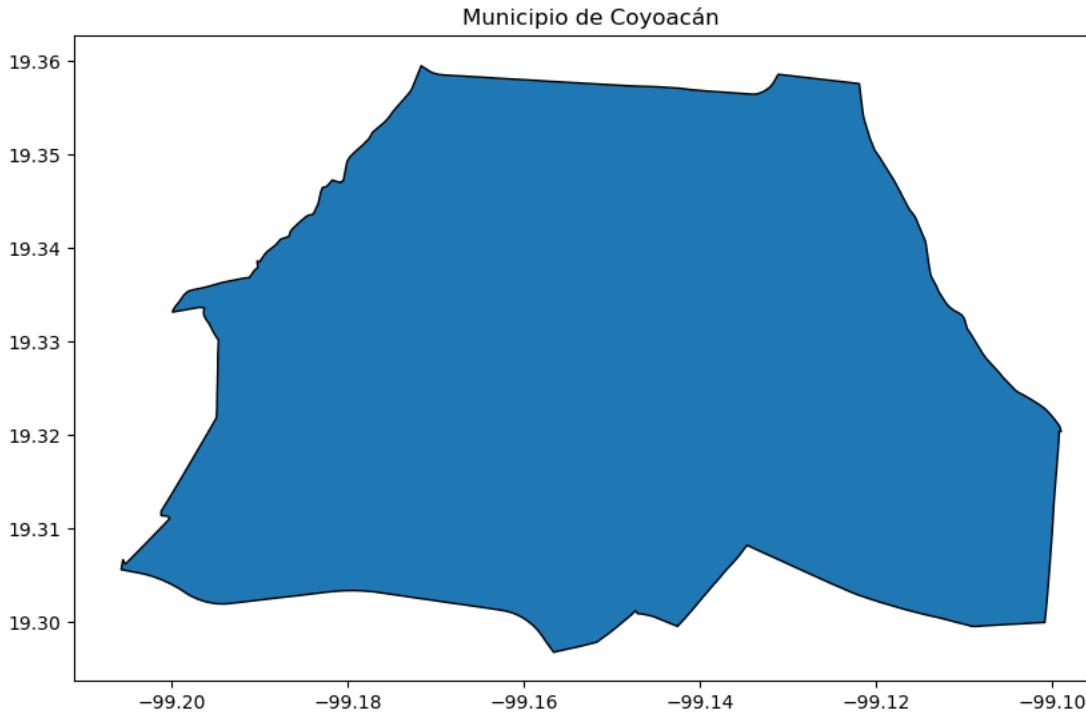
```
gid      0
id_ageb   0
geom      0
dtype: int64
```

```
Valores nulos en Manzana:
```

```
gid      0
id_manzana  0
geom      0
dtype: int64
```

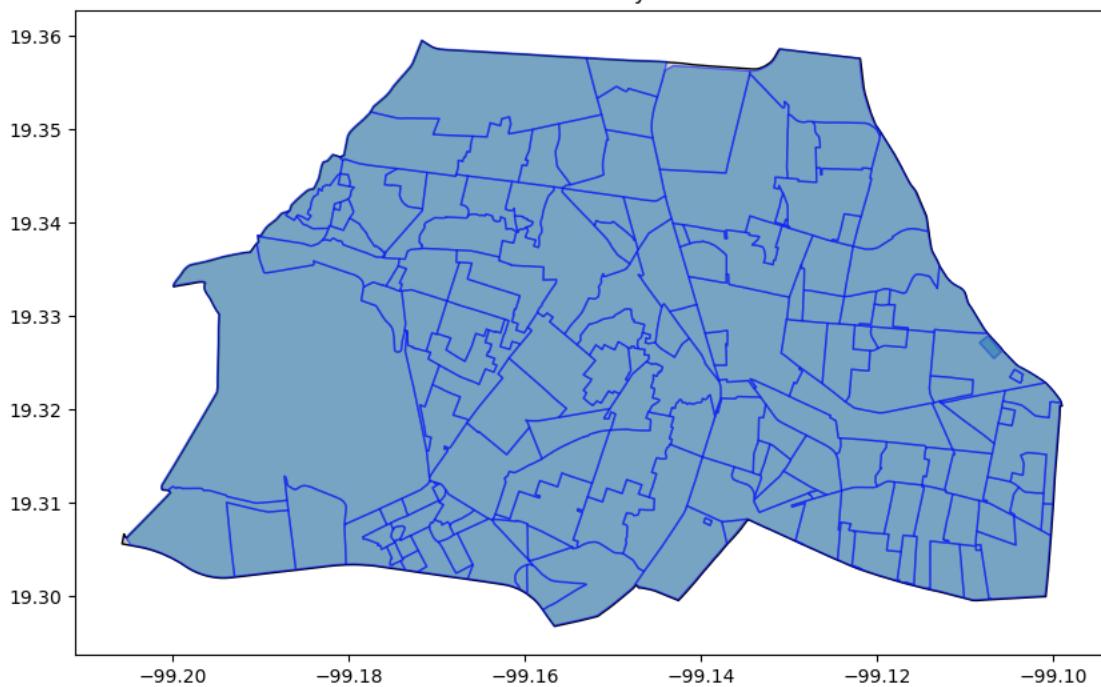
```
In [8]: import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Plotear Municipio
gdf_municipio.plot(edgecolor='black', figsize=(10, 10))
plt.title('Municipio de Coyoacán')
plt.show()
```



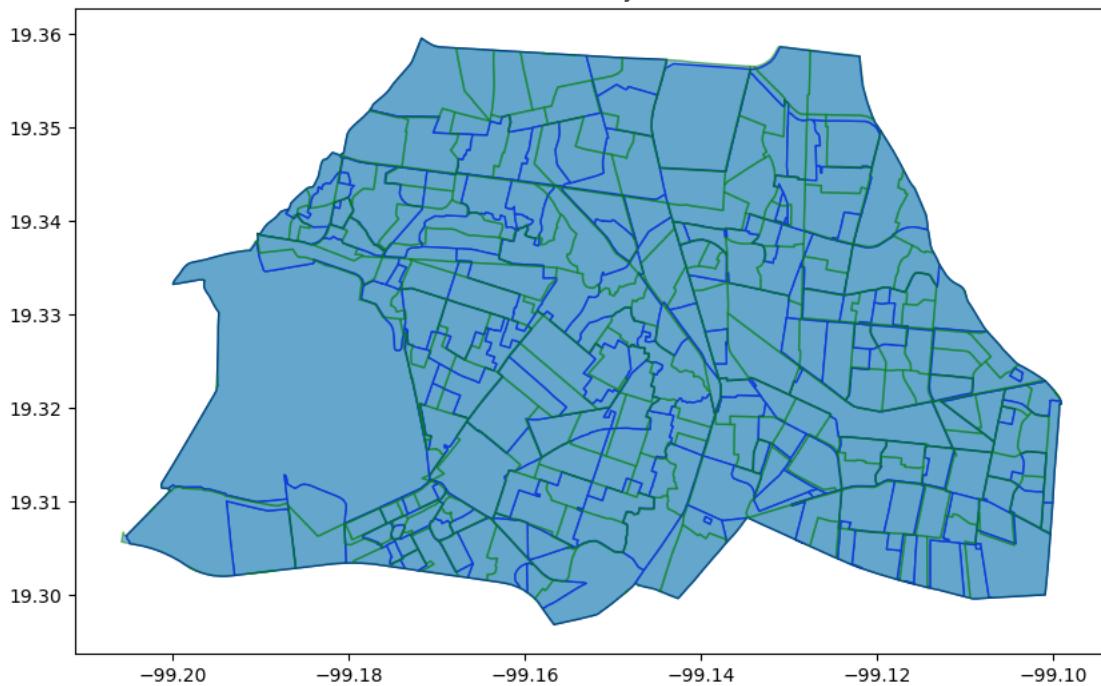
```
In [9]: # Plotear Colonias sobre Municipio
ax = gdf_municipio.plot(edgecolor='black', figsize=(10, 10), color='lightgrey')
gdf_colonia.plot(ax=ax, edgecolor='blue', alpha=0.5)
plt.title('Colonias en Coyoacán')
plt.show()
```

Colonias en Coyoacán



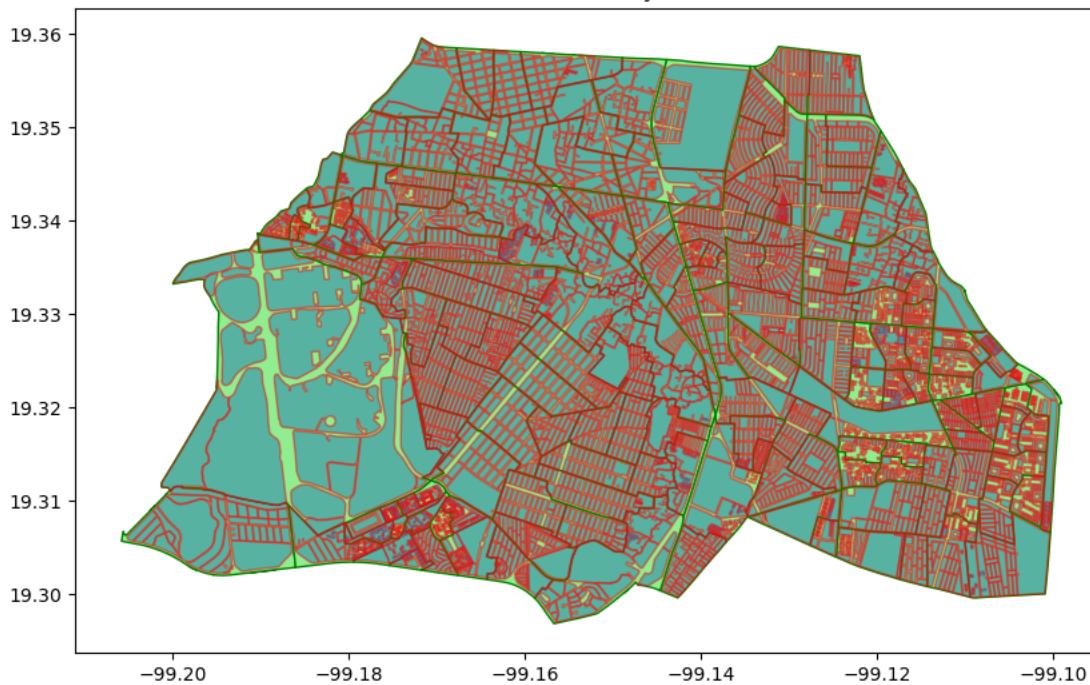
```
In [10]: # Plotear AGEB sobre Colonias
ax = gdf_colonia.plot(edgecolor='blue', figsize=(10, 10), color='lightblue')
gdf_ageb.plot(ax=ax, edgecolor='green', alpha=0.5)
plt.title('AGEBs en Coyoacán')
plt.show()
```

AGEBs en Coyoacán



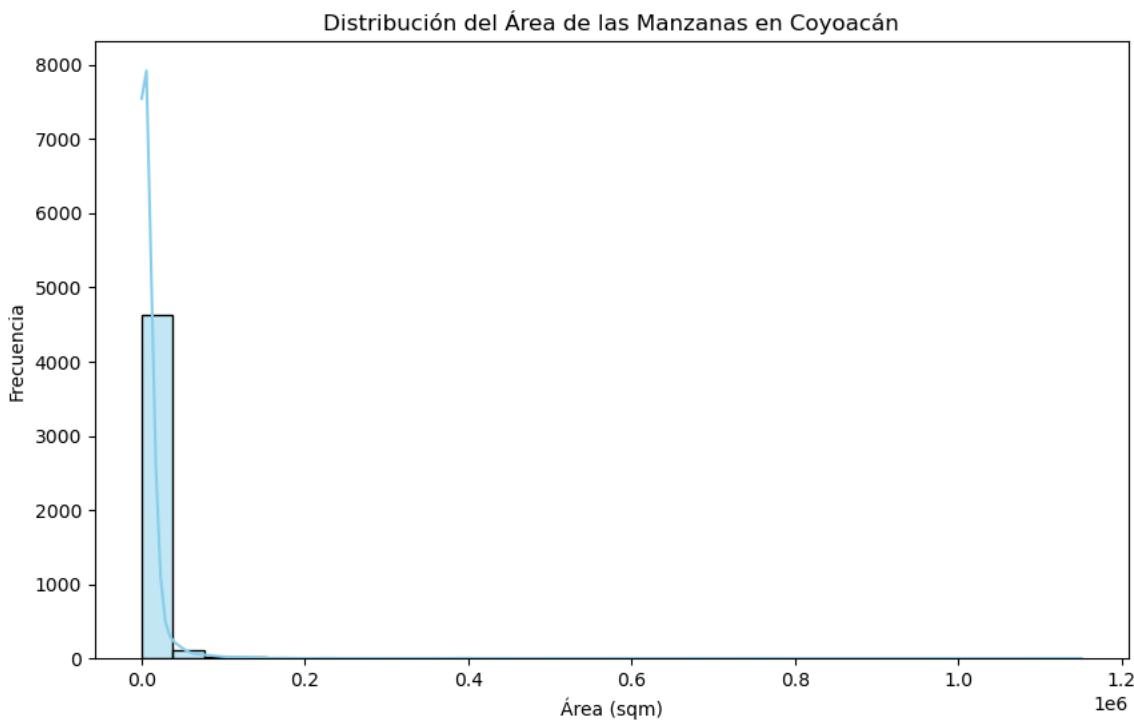
```
In [11]: # Plotear Manzanas sobre AGEBs
ax = gdf_ageb.plot(edgecolor='green', figsize=(10, 10), color='lightgreen')
gdf_manzana.plot(ax=ax, edgecolor='red', alpha=0.5)
plt.title('Manzanas en Coyoacán')
plt.show()
```

Manzanas en Coyoacán



```
In [12]: # Calcular el área en metros cuadrados (cambiar el CRS a proyectado para precisión)
gdf_manzana_proj = gdf_manzana.to_crs(epsg=3857) # EPSG:3857 es un CRS proyectado común
gdf_manzana['area_sqm'] = gdf_manzana_proj.geometry.area

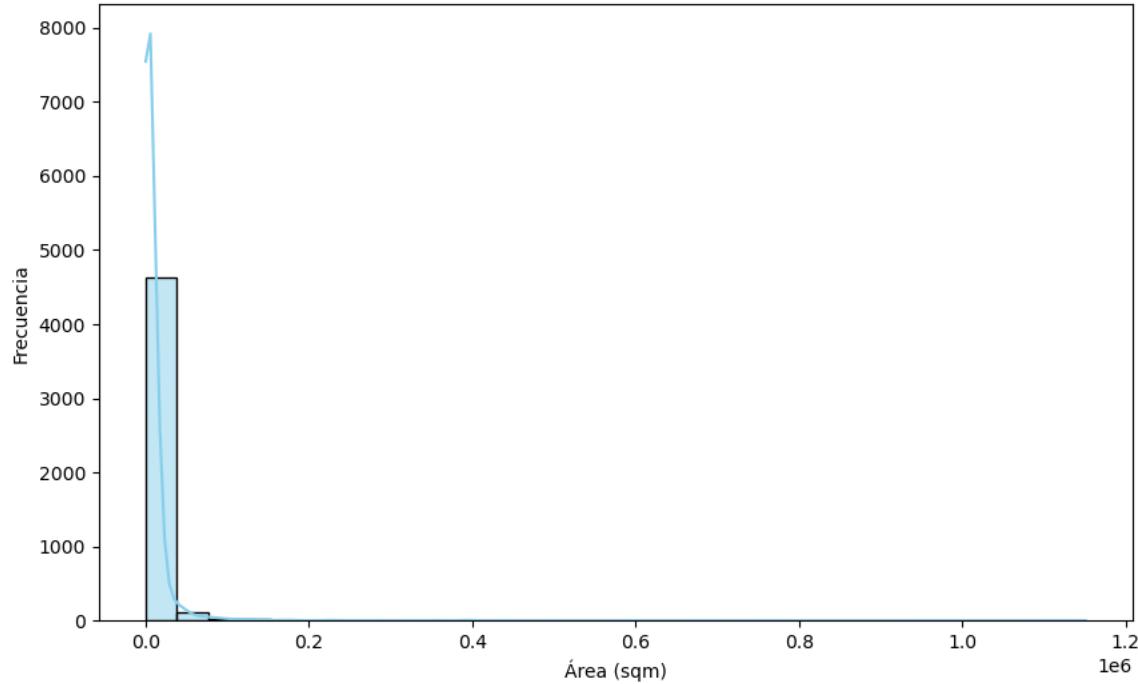
# Histograma del área de las manzanas
plt.figure(figsize=(10,6))
sns.histplot(gdf_manzana['area_sqm'], bins=30, kde=True, color='skyblue')
plt.title('Distribución del Área de las Manzanas en Coyoacán')
plt.xlabel('Área (sqm)')
plt.ylabel('Frecuencia')
plt.show()
```



```
In [13]: # Calcular el área en metros cuadrados (cambiar el CRS a proyectado para precisión)
gdf_manzana_proj = gdf_manzana.to_crs(epsg=3857) # EPSG:3857 es un CRS proyectado común
gdf_manzana['area_sqm'] = gdf_manzana_proj.geometry.area

# Histograma del área de las manzanas
plt.figure(figsize=(10,6))
sns.histplot(gdf_manzana['area_sqm'], bins=30, kde=True, color='skyblue')
plt.title('Distribución del Área de las Manzanas en Coyoacán')
plt.xlabel('Área (sqm)')
plt.ylabel('Frecuencia')
plt.show()
```

Distribución del Área de las Manzanas en Coyoacán



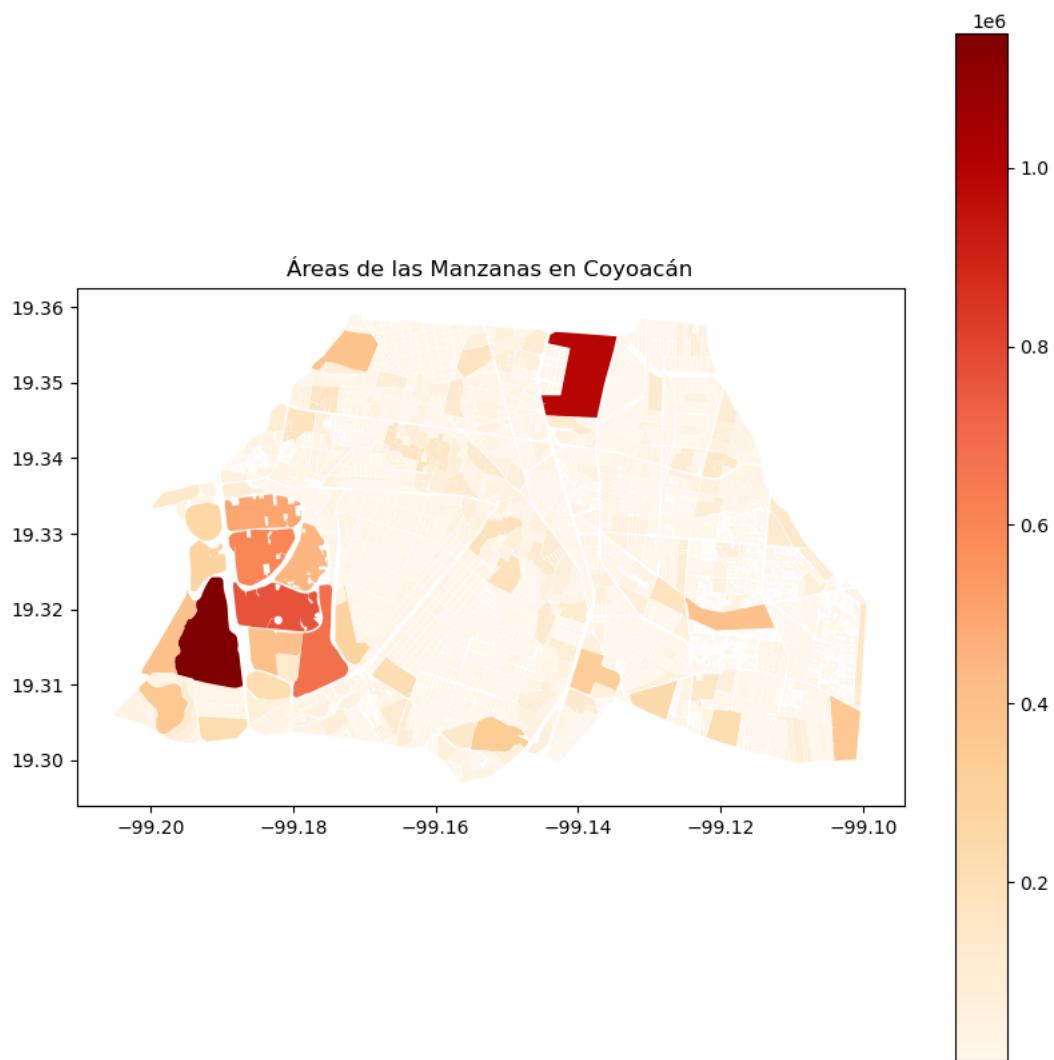
```
In [14]: import matplotlib.pyplot as plt

# Plotear Municipio
ax = gdf_municipio.plot(edgecolor='black', figsize=(10, 10), color='lightgrey')
# Superponer Colonias
gdf_colonia.plot(ax=ax, edgecolor='blue', alpha=0.5)
# Superponer AGEBS
gdf_ageb.plot(ax=ax, edgecolor='green', alpha=0.5)
# Superponer Manzanas
gdf_manzana.plot(ax=ax, edgecolor='red', alpha=0.3)
plt.title('Mapa Completo de Coyoacán')
plt.show()
```

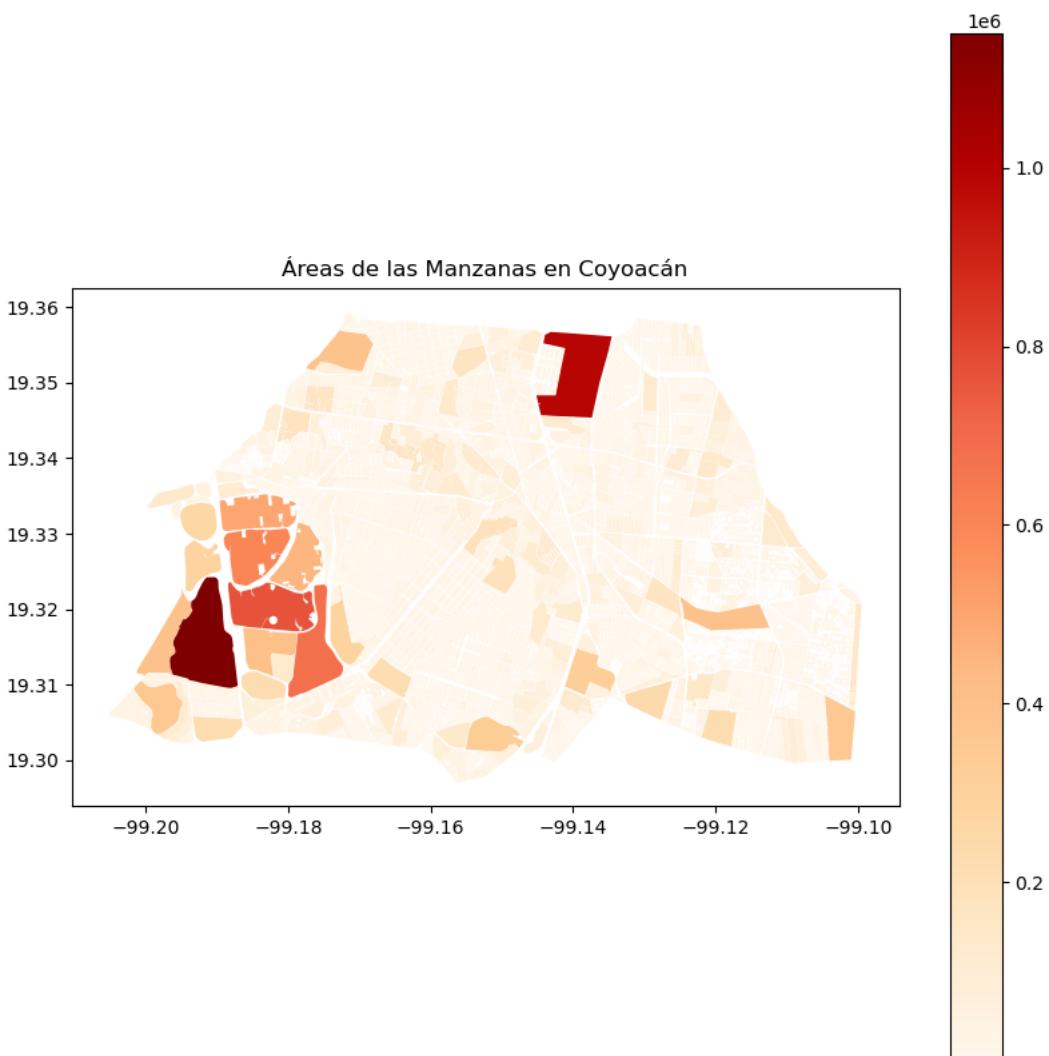
Mapa Completo de Coyoacán



```
In [15]: # Mapa de Manzanas coloreadas por área
gdf_manzana.plot(column='area_sqm', cmap='OrRd', legend=True, figsize=(10,10))
plt.title('Áreas de las Manzanas en Coyoacán')
plt.show()
```



```
In [16]: # Mapa de Manzanas coloreadas por área
gdf_manzana.plot(column='área_sqm', cmap='OrRd', legend=True, figsize=(10,10))
plt.title('Áreas de las Manzanas en Coyoacán')
plt.show()
```



```
In [17]: # Transformar a un CRS común (por ejemplo, EPSG:4326 para WGS84)
crs_target = "EPSG:4326"
gdf_ageb = gdf_ageb.to_crs(crs_target)
gdf_manzana = gdf_manzana.to_crs(crs_target)
gdf_colonia = gdf_colonia.to_crs(crs_target)
```

```
In [19]: # Unión espacial entre AGEF y manzanas
agef_to_manzanas = gpd.sjoin(gdf_ageb,
                             gdf_manzana,
                             how="inner",
                             predicate="intersects")

# Visualizar el resultado
print(agef_to_manzanas.head())
```

gid_left	id_ageb	geom
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...

index_right	gid_right	id_manzana	area_sqm
3	3583	3564	415.618044
3	710	696	208.813564
3	2878	2851	527.219111
3	2490	2463	242.186136
3	2613	2589	533.982116

```
In [20]: agef_to_colonias = gpd.sjoin(gdf_ageb,
                                    gdf_colonia,
                                    how="inner",
                                    predicate="intersects")
print(agef_to_colonias.head())
```

gid_left	id_ageb	geom
0	1	MULTIPOLYGON (((-99.20241 19.30902, -99.20254 ...
1	3	MULTIPOLYGON (((-99.19025 19.33798, -99.19061 ...
2	4	MULTIPOLYGON (((-99.14833 19.30045, -99.14819 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3	44	MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...

index_right	gid_right	id_colonia	nombre_col
0	103	103	PEDREGAL DE SAN ANGEL (AMPL)
1	39	32	COPILCO UNIVERSIDAD
2	20	16	BOSQUES DE TETLAMEYAN

```
In [21]: def load_data(file_path):
    print("[DATA LOADER] Intentando cargar datos desde: {file_path}")
    gdf = gpd.read_file(file_path)
```

```
    if gdf.crs != "EPSG:4326":
        print("[DATA LOADER] Reproyectando a EPSG:4326...")
        gdf = gdf.to_crs("EPSG:4326")
        gdf.to_file(file_path, driver="GeoJSON")
        print("[DATA LOADER] Reproyección completada y archivo guardado.")

    print("[DATA LOADER] Archivo cargado correctamente. Total de registros: {len(gdf)}")
    print("[DATA LOADER] Columnas disponibles en el archivo:", list(gdf.columns))
    print("[DATA LOADER] Ejemplo de geometrías:\n", gdf["geometry"].head())

    print("[DATA LOADER] Calculando métricas adicionales...")
    gdf["relacion_genero"] = gdf["densidad_hombres"] / gdf["densidad_mujeres"]
    gdf["dependencia_infantil"] = gdf["p_3ymas"] / gdf["pob_total"]
    print("[DATA LOADER] Nuevas métricas calculadas exitosamente.")
    return gdf
```

```
In [22]: datos_agef = load_data("../clean_data/coyoacan_poblacion_2020_clean.geojson")
```

```
[DATA LOADER] Intentando cargar datos desde: ../clean_data/coyoacan_poblacion_2020_clean.geojson
[DATA LOADER] Archivo cargado correctamente. Total de registros: 154
[DATA LOADER] Columnas disponibles en el archivo: ['alc', 'loc', 'amb_loc', 'ageb', 'pob_total', 'p_3ymas', 'p_12yms', 'p_nacoe', 'p_vivoe', 'p_hli', 'p_hli_nh', 'p_hli_h', 'p_afrmx', 'p_p12ym_sl', 'p_p12ym_c', 'p_p12ym_sp', 'p_catlc', 'p_criev', 'p_trsrl', 'p_sinrl', 't_nacoe', 't_vivoe', 't_hli', 't_hli_nh', 't_hli_h', 't_afrmx', 't_p12ym_sl', 't_p12ym_c', 't_p12ym_sp', 't_catlc', 't_criev', 't_trsrl', 't_sinrl', 'pob_hombres', 'pob_mujeres', 'area_km2', 'densidad_pob_total', 'densidad_hombres', 'densidad_mujeres', 'anio', 'geometry']
[DATA LOADER] Ejemplo de geometrías:
0  POLYGON ((-99.10148 19.30718, -99.10399 19.308...
1  POLYGON ((-99.10086 19.3139, -99.10095 19.3126...
2  POLYGON ((-99.09929 19.32045, -99.09935 19.319...
3  POLYGON ((-99.10364 19.31808, -99.10359 19.318...
4  POLYGON ((-99.11532 19.30638, -99.11509 19.306...
Name: geometry, dtype: geometry
[DATA LOADER] Calculando métricas adicionales...
[DATA LOADER] Nuevas métricas calculadas exitosamente.
```

```
In [23]: print(agef_to_manzanas.columns)
print(datos_agef.columns)
```

```
Index(['gid_left', 'id_ageb', 'geom', 'index_right', 'gid_right', 'id_manzana',
       'area_sqm'],
      dtype='object')
Index(['alc', 'loc', 'amb_loc', 'ageb', 'pob_total', 'p_3ymas', 'p_12yms',
       'p_nacoe', 'p_vivoe', 'p_hli', 'p_hli_nh', 'p_hli_h', 'p_afrmx',
       'p_p12ym_sl', 'p_p12ym_c', 'p_p12ym_sp', 'p_catlc', 'p_criev',
       'p_trsrl', 'p_sinrl', 't_nacoe', 't_vivoe', 't_hli', 't_hli_nh',
       't_hli_h', 't_afrmx', 't_p12ym_sl', 't_p12ym_c', 't_p12ym_sp',
       't_catlc', 't_criev', 't_trsrl', 't_sinrl', 'pob_hombres',
       'pob_mujeres', 'area_km2', 'densidad_pob_total', 'densidad_hombres',
       'densidad_mujeres', 'anio', 'geometry', 'relacion_genero',
       'dependencia_infantil'],
      dtype='object')
```

```
In [24]: datos_con_manzanas = agef_to_manzanas.merge(
    datos_agef,
    left_on="id_ageb",
    right_on="ageb"
)
print(datos_con_manzanas.head())
```

```
Empty GeoDataFrame
Columns: [gid_left, id_ageb, geom, index_right, gid_right, id_manzana, area_sqm, alc, loc, amb_loc, ageb, pob_total, p_3ymas, p_12yms, p_nacoe, p_vivoe, p_hli, p_hli_nh, p_hli_h, p_afrmx, p_p12ym_sl, p_p12ym_c, p_p12ym_sp, p_catlc, p_criev, p_trsrl, p_sinrl, t_nacoe, t_vivoe, t_hli, t_hli_nh, t_hli_h, t_afrmx, t_p12ym_sl, t_p12ym_c, t_p12ym_sp, t_catlc, t_criev, t_trsrl, t_sinrl, pob_hombres, pob_mujeres, area_km2, densidad_pob_total, densidad_hombres, densidad_mujeres, anio, geometry, relacion_genero, dependencia_infantil]
Index: []
```

```
[0 rows x 50 columns]
```

```
In [25]: print("Valores únicos en id_ageb (agef_to_manzanas):", agef_to_manzanas["id_ageb"].unique())
print("Valores únicos en ageb (datos_agef):", datos_agef["ageb"].unique())
```

```
Valores únicos en id_ageb (agef_to_manzanas): ['1228' '1533' '1459' '0516' '1783' '0357' '0361' '0728' '1444' '054A'
'0111' '1092' '0179' '0906' '1425' '1730' '0018' '1105' '168A' '1618'
'1073' '0037' '1660' '1694' '175A' '1764' '1779' '1798' '0234' '0060'
'1139' '0022' '0041' '008A' '0075' '0713' '0412' '1124' '0094' '0107'
'0130' '1745' '015A' '1707' '1726' '1317' '1478' '1143' '1529' '1675'
'0164' '0183' '0198' '0200' '0215' '022A' '0249' '0268' '0893' '0380'
'0395' '0408' '0535' '0821' '0836' '0840' '0855' '0889' '093A' '0520'
'1020' '129A' '136A' '1374' '1389' '1393' '1406' '0643' '143A' '150A'
'1514' '1567' '1590' '1482' '0145' '0287' '0291' '0304' '0319' '0465'
'0323' '1213' '0342' '0376' '1336' '0427' '0959' '0802' '0431' '1321'
'0446' '0450' '0817' '0499' '0501' '0696' '0785' '1162' '0982' '1232'
'1247' '1251' '1270' '0554' '0573' '0605' '0624' '0639' '0658' '0662'
'079A' '086A' '0997' '1001' '1016' '1069' '1410' '1340' '1355' '1497'
'1463' '1622' '0126' '0588' '0056' '1302' '1586' '1266' '0272' '0677'
'0681' '1285' '1158' '1181' '0978' '1656' '1571' '1552' '1548' '061A'
'0770' '0484' '0709' '0963' '1163' '1641']
```

```
Valores únicos en ageb (datos_agef): ['0900300011514' '0900300011497' '0900300011463' '0900300011482'
'0900300011073' '0900300011213' '0900300011092' '0900300011069'
'0900300011444' '090030001150A' '0900300011181' '090030001093A'
'0900300011406' '0900300011393' '0900300010959' '0900300011389'
'0900300011336' '0900300011302' '0900300011779' '0900300011694'
```

```
'090030001143A' '0900300011355' '0900300011374' '090030001136A'
'0900300011340' '0900300010906' '090030001129A' '090030001175A'
'0900300011707' '0900300011478' '0900300011459' '0900300011321'
'0900300011764' '0900300011285' '0900300011317' '0900300011266'
'0900300010665' '0900300011251' '0900300011270' '0900300011745'
'0900300011247' '0900300010588' '0900300010520' '0900300010535'
'0900300010516' '0900300010342' '090030001022A' '0900300010200'
'0900300010357' '0900300011105' '0900300011798' '0900300011618'
'0900300011641' '0900300011783' '0900300011586' '0900300011622'
'0900300011637' '0900300011529' '0900300011571' '0900300011533'
'0900300010770' '0900300011590' '0900300011548' '0900300011552'
'0900300011567' '0900300010963' '090030001079A' '0900300010817'
'0900300010785' '0900300011425' '0900300011410' '0900300011016'
'0900300011020' '0900300011660' '0900300011124' '0900300011656'
'0900300011001' '0900300010978' '0900300010997' '0900300010982'
'090030001168A' '0900300011675' '0900300010893' '0900300011730'
'0900300011726' '0900300010889' '0900300010855' '0900300010677'
'0900300010696' '0900300010681' '0900300010709' '0900300010484'
'0900300010465' '0900300011143' '0900300011139' '0900300010304'
'0900300010821' '0900300010840' '0900300010836' '0900300010643'
'0900300010662' '0900300010658' '0900300010573' '0900300010446'
'0900300010450' '0900300010291' '0900300010624' '090030001061A'
'0900300011162' '0900300011158' '0900300010431' '0900300010427'
'0900300010272' '0900300010268' '0900300010130' '0900300010126'
'0900300010554' '090030001054A' '0900300010408' '0900300010412'
'0900300010376' '0900300010380' '0900300010361' '0900300010395'
'0900300010249' '0900300010287' '090030001015A' '0900300010145'
'0900300010056' '0900300010060' '0900300010037' '0900300010041'
'0900300010234' '0900300010111' '0900300010022' '0900300010018'
'090030001086A' '0900300010728' '0900300010713' '0900300010501'
'0900300010499' '0900300011228' '0900300011232' '0900300010183'
'0900300010179' '0900300010198' '0900300010323' '0900300010319'
'0900300010164' '0900300010075' '0900300010215' '0900300010107'
'090030001008A' '0900300010094']
```

In [26]: # Extraer los últimos 4 caracteres de la columna 'ageb'
datos_agef["id_ageb"] = datos_agef["ageb"].str[-4:]

In [27]: coincidencias = set(agef_to_manzanas["id_ageb"]) & set(datos_agef["id_ageb"])
print("Valores coincidentes después del ajuste:", coincidencias)

Valores coincidentes después del ajuste: {'143A', '0287', '0450', '0840', '1389', '0094', '1181', '1444', '0268', '0963', '1656', '1317', '0200', '0982', '0681', '1660', '1552', '0183', '1336', '1162', '1355', '0198', '0075', '0018', '0342', '1247', '0465', '086A', '1232', '0272', '093A', '0376', '0395', '1425', '0107', '0431', '1783', '1529', '0319', '022A', '129A', '0836', '1586', '168A', '0624', '1143', '1393', '1798', '0855', '1694', '0821', '0499', '0817', '0677', '136A', '1340', '1707', '0713', '1228', '0696', '1270', '1124', '0022', '1622', '150A', '0446', '1514', '1463', '0484', '1410', '1092', '1745', '0501', '1406', '1730', '0588', '0770', '1321', '0959', '0234', '1001', '0573', '1020', '0709', '0179', '1533', '0785', '1459', '054A', '1726', '0427', '0516', '1478', '1567', '1571', '0304', '1590', '1105', '1139', '1637', '0041', '0535', '0658', '0361', '0126', '0130', '1618', '0889', '079A', '0215', '0893', '0380', '0662', '1016', '0056', '0357', '1497', '1073', '0978', '0643', '0323', '0249', '0164', '0906', '015A', '1482', '1213', '1302', '0520', '0997', '1285', '1158', '0291', '0111', '061A', '1779', '0728', '008A', '1641', '0605', '0554', '1675', '0412', '0145', '0037', '1069', '1548', '0060', '0408', '1266', '1764', '1374', '175A', '1251'}

In [28]: datos_con_manzanas = agef_to_manzanas.merge(
 datos_agef,
 left_on="id_ageb",
 right_on="id_ageb")
)

```
print(datos_con_manzanas.head())
gid_left id_ageb                                     geom \
0        44    1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
1        44    1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
2        44    1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
3        44    1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
4        44    1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...

index_right  gid_right id_manzana      area_sqm      alc      loc amb_loc \
0            3583     3564      3564  415.618044  Coyoacán  Coyoacán Urbana
1              710      696      696  208.813564  Coyoacán  Coyoacán Urbana
2            2878     2851      2851  527.219111  Coyoacán  Coyoacán Urbana
3            2490     2463      2463  242.186136  Coyoacán  Coyoacán Urbana
4            2613     2589      2589  533.982116  Coyoacán  Coyoacán Urbana

...  pob_hombres  pob_mujeres  area_km2  densidad_pob_total \
0 ...          1107         1173  0.066397  34339.068265
1 ...          1107         1173  0.066397  34339.068265
2 ...          1107         1173  0.066397  34339.068265
3 ...          1107         1173  0.066397  34339.068265
4 ...          1107         1173  0.066397  34339.068265

densidad_hombres  densidad_mujeres  anio \
0   16672.521302    17666.546963  2020
1   16672.521302    17666.546963  2020
2   16672.521302    17666.546963  2020
3   16672.521302    17666.546963  2020
4   16672.521302    17666.546963  2020

geometry  relacion_genero \
0  POLYGON ((-99.13051 19.34164, -99.13033 19.341...  0.943734
1  POLYGON ((-99.13051 19.34164, -99.13033 19.341...  0.943734
2  POLYGON ((-99.13051 19.34164, -99.13033 19.341...  0.943734
3  POLYGON ((-99.13051 19.34164, -99.13033 19.341...  0.943734
4  POLYGON ((-99.13051 19.34164, -99.13033 19.341...  0.943734

dependencia_infantil
0           0.969298
```

```
1      0.969298
2      0.969298
3      0.969298
4      0.969298
```

[5 rows x 50 columns]

```
In [29]: print(datos_con_manzanas["id_manzana"].nunique()) # Número de manzanas únicas
print(datos_con_manzanas["id_ageb"].nunique()) # Número de AGEF únicas
```

```
4742
154
```

```
In [30]: datos_con_manzanas.to_csv("../clean_data/poligonos/datos_con_manzanas.csv", index=False)
```

```
In [31]: # Renombrar la columna existente (opción 1)
datos_con_manzanas.rename(columns={"index_right": "index_manzanas"}, inplace=True)
```

```
In [32]: # Realizar la unión espacial
datos_con_colonias = gpd.sjoin(
    datos_con_manzanas,
    gdf_colonia,
    how="inner",
    predicate="intersects"
)
```

```
# Verificar el resultado
print(datos_con_colonias.head())
```

```
gid_left id_ageb                                geom \
0        44     1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...
1        44     1228 MULTIPOLYGON (((-99.13051 19.34164, -99.13033 ...

index_manzanas  gid_right id_manzana      area_sqm      alc      loc \
0            3583       3564     3564  415.618044  Coyoacán  Coyoacán
1             710        696      696  208.813564  Coyoacán  Coyoacán

amb_loc ... densidad_hombres  densidad_muujeres  anio \
0   Urbana ...  16672.521302      17666.546963  2020
1   Urbana ...  16672.521302      17666.546963  2020

geometry  relacion_genero \
0  POLYGON ((-99.13051 19.34164, -99.13033 19.341...
1  POLYGON ((-99.13051 19.34164, -99.13033 19.341...

dependencia_infantil  index_right  gid  id_colonia      nombre_col
0          0.969298         5     60      150  EX EJIDO DE CHURUBUSCO
0          0.969298        17    136      108  SANTA MARTHA DEL SUR
0          0.969298        99    101      134  PASEOS DE TAXQUEÑA II
0          0.969298        24     19      012  CAMPESTRE CHURUBUSCO
1          0.969298         5     60      150  EX EJIDO DE CHURUBUSCO
```

[5 rows x 54 columns]

```
In [33]: datos_con_colonias.to_csv("../clean_data/datos_con_colonias.csv", index=False)
```

```
In [34]: print(datos_con_colonias.columns)
```

```
Index(['gid_left', 'id_ageb', 'geom', 'index_manzanas', 'gid_right',
       'id_manzana', 'area_sqm', 'alc', 'loc', 'amb_loc', 'ageb', 'pob_total',
       'p_3ymas', 'p_12yms', 'p_nacoe', 'p_vivoe', 'p_hli', 'p_hl_nh',
       'p_hli_h', 'p_afrmx', 'p_p12ym_s1', 'p_p12ym_c', 'p_p12ym_sp',
       'p_catlc', 'p_criev', 'p_trsrl', 'p_sinrl', 't_nacoe', 't_vivoe',
       't_hli', 't_hl_nh', 't_hli_h', 't_afrmx', 't_p12ym_s1', 't_p12ym_c',
       't_p12ym_sp', 't_catlc', 't_criev', 't_trsrl', 't_sinrl', 'pob_hombres',
       'pob_muujeres', 'area_km2', 'densidad_pob_total', 'densidad_hombres',
       'densidad_muujeres', 'anio', 'geometry', 'relacion_genero',
       'dependencia_infantil', 'index_right', 'gid', 'id_colonia',
       'nombre_col'],
      dtype='object')
```

```
In [35]: def procesar_datos(df, nivel_granularidad, metricas):
```

```
    """Agrupa y procesa los datos según el nivel de granularidad y las métricas seleccionadas.
```

```
    Args:
```

```
        df (GeoDataFrame): DataFrame con los datos geoespaciales.
        nivel_granularidad (str): Nivel de granularidad ('manzana', 'ageb', 'colonia').
        metricas (list): Lista de métricas a calcular. Ejemplo: ['pob_total', 'densidad_pob_total'].
```

```
    Returns:
```

```
        GeoDataFrame: DataFrame procesado con las métricas agregadas por nivel de granularidad.
    """
```

```
# Mapear columnas según el nivel de granularidad
```

```
niveles = {
```

```

    "manzana": "id_manzana",
    "ageb": "id_ageb",
    "colonia": "id_colonia"
}

if nivel_granularidad not in niveles:
    raise ValueError(f"Nivel de granularidad no válido. Debe ser uno de: {list(niveles.keys())}")

columna_grupo = niveles[nivel_granularidad]

# Agrupar y calcular métricas
datos_procesados = df.groupby(columna_grupo).agg({
    "geometry": "first", # Mantener la geometría
    **{metrica: "sum" for metrica in metricas} # Calcular suma para las métricas
}).reset_index()

# Convertir a GeoDataFrame
return gpd.GeoDataFrame(datos_procesados, geometry="geometry")

```

```

In [36]: import matplotlib.pyplot as plt

def graficar_mapa(df, nivel_granularidad, metrica, cmap="viridis"):
    """
    Grafica un mapa con la métrica seleccionada en el nivel de granularidad especificado.

    Args:
        df (GeoDataFrame): DataFrame procesado con las métricas agregadas.
        nivel_granularidad (str): Nivel de granularidad ('manzana', 'ageb', 'colonia').
        metrica (str): Métrica a visualizar.
        cmap (str): Colormap para la visualización.

    Returns:
        None: Muestra el mapa.
    """
    plt.figure(figsize=(12, 8))
    df.plot(
        column=metrica,
        cmap=cmap,
        legend=True,
        edgecolor="black",
        linewidth=0.5
    )
    plt.title(f"{metrica.capitalize()} por {nivel_granularidad.capitalize()}")
    plt.axis("off")
    plt.show()

```

```

In [37]: datos_ageb = procesar_datos(
    df=datos_con_colonias,
    nivel_granularidad="ageb",
    metricas=["pob_total", "densidad_pob_total"]
)

print(datos_ageb.head())

```

	id_ageb	geometry	pob_total
0	0018	POLYGON ((-99.16776 19.35846, -99.16774 19.358...	92848
1	0022	POLYGON ((-99.16351 19.35801, -99.16354 19.357...	39225
2	0037	POLYGON ((-99.16046 19.35804, -99.16045 19.357...	35700
3	0041	POLYGON ((-99.15888 19.35797, -99.15769 19.357...	252504
4	0056	POLYGON ((-99.14933 19.3512, -99.14958 19.3497...	296020

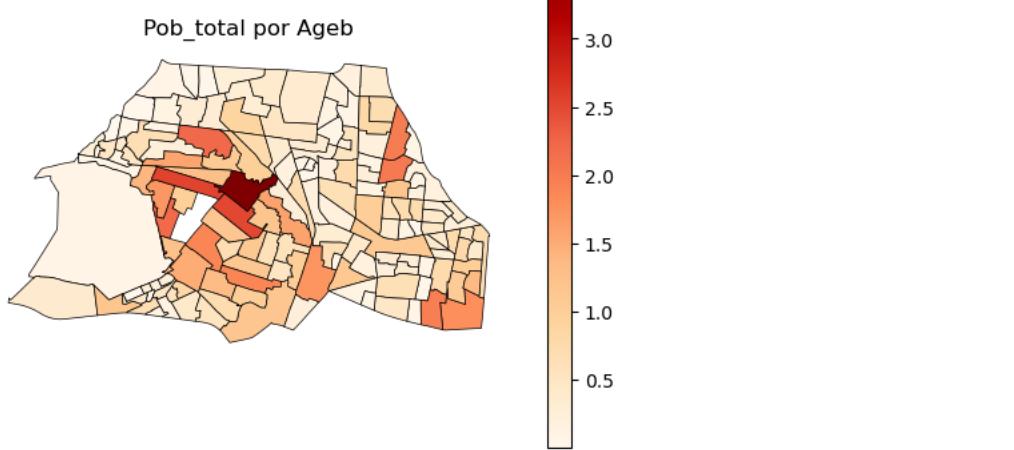
	densidad_pob_total
0	133725.963049
1	150683.282328
2	120761.464934
3	553264.601731
4	627164.963594

```

In [38]: graficar_mapa(
    df=datos_ageb,
    nivel_granularidad="ageb",
    metrica="pob_total",
    cmap="OrRd"
)

```

<Figure size 1200x800 with 0 Axes>



```
In [39]: import geopandas as gpd

def procesar_datos(df, nivel_granularidad, metricas):
    """
    Agrupa y procesa los datos según el nivel de granularidad y las métricas seleccionadas.

    Args:
        df (GeoDataFrame): DataFrame con los datos geoespaciales.
        nivel_granularidad (str): Nivel de granularidad ('manzana', 'ageb', 'colonia').
        metricas (list): Lista de métricas a calcular. Ejemplo: ['pob_total', 'densidad_pob_total'].

    Returns:
        GeoDataFrame: DataFrame procesado con las métricas agregadas por nivel de granularidad.
    """
    niveles = {
        "manzana": "id_manzana",
        "ageb": "id_ageb",
        "colonia": "id_colonia"
    }

    if nivel_granularidad not in niveles:
        raise ValueError(f"Nivel de granularidad no válido. Debe ser uno de: {list(niveles.keys())}")

    columna_grupo = niveles[nivel_granularidad]

    # Agrupar y calcular métricas
    datos_procesados = df.groupby(columna_grupo).agg({
        "geometry": "first", # Mantener la geometría
        **{metrica: "sum" for metrica in metricas} # Calcular suma para las métricas
    }).reset_index()

    return gpd.GeoDataFrame(datos_procesados, geometry="geometry")
```

```
In [40]: import plotly.express as px

def graficar_mapa_interactivo_plotly(df, nivel_granularidad, metrica, nombre_mapa="Mapa Interactivo"):
    """
    Genera un mapa interactivo en Plotly para visualizar una métrica por un nivel de granularidad.

    # Mapear columna según nivel de granularidad
    niveles = {
        "manzana": "id_manzana",
        "ageb": "id_ageb",
        "colonia": "id_colonia" # Aquí ajustamos el mapeo
    }

    if nivel_granularidad not in niveles:
        raise ValueError(f"Nivel de granularidad no válido. Debe ser uno de: {list(niveles.keys())}")

    columna_grupo = niveles[nivel_granularidad]

    # Convertir GeoDataFrame a GeoJSON
    geojson = df.set_index(columna_grupo)[["geometry"]].__geo_interface__

    # Crear el mapa interactivo
    fig = px.choropleth(
        df,
        geojson=geojson,
        locations=columna_grupo,
        color=metrica,
        title=nombre_mapa,
        color_continuous_scale="Viridis",
        projection="mercator"
    )

    # Ajustar la vista del mapa
    fig.update_geos(fitbounds="locations", visible=False)
    fig.update_layout(
        margin={"r": 0, "t": 50, "l": 0, "b": 0},
        coloraxis_colorbar={"title": metrica.capitalize()})
```

```
)
```

```
return fig
```

```
In [41]: # Procesar datos por colonia
```

```
datos_colonia = procesar_datos(  
    df=datos_con_colonias,  
    nivel_granularidad="manzana",  
    metricas=["pob_total", "densidad_pob_total"]  
)
```

```
In [42]: print(datos_colonia.columns)
```

```
Index(['id_manzana', 'geometry', 'pob_total', 'densidad_pob_total'], dtype='object')
```

```
In [43]: # Generar el mapa interactivo con la métrica pob_total
```

```
fig = graficar_mapa_interactivo_plotly(  
    df=datos_colonia,  
    nivel_granularidad="manzana", # Ahora se mapea correctamente a id_colonia  
    metrica="pob_total",  
    nombre_mapa="Población Total por Colonia"  
)
```

```
# Mostrar el mapa  
fig.show()
```

Población Total por Colonia

