FORMATIVE ASSSESSMENT TASK 3

R2311D17355747

BENTA WANJIRU IRUNGU

UEL-DS-7002

Introduction

This paper focuses on demonstrating practical skills in spatial operations and transformation. We will use python software to perform the spatial operations. In this exercise, we will be using three data sets namely municipalities.zip, indigenous group data, and housing data to perform spatial operations required for the five questions provided.

QUESTION 1

PRODUCE A MAP SHOWING THE CENTROIDS OF EACH MUNICIPALITY IN JUST THE STATE OF SÄO PAULO, AND ADD THE OUTER BOUNDARY OF SÄO PAULO STATE

To perfom this task, the first step will be importing the required python libraries which are `geopandas` and `matplotlib`

In [2]:
```
#IMPORTING REQUIRED LIBRARIES
import geopandas as gpd
import matplotlib.pyplot as plt
```

The next step will be opening and reading the contents of the `municipalities.zip` folder. To open the zip folder in python we will import `zipfile` and `os` libraries. our zipped files has four datafiles named `municipalities.dbf` , `municipalities.prj` , `municipalities.shp` and `municipalities.shx` . we will use the `zip_ref.extract()` function to extract the 4specifiled data files within our zip folder.

OPENING THE ZIP FOLDER FOR MUNICIPALITIES DATA. see cell below

In [3]:
```
import zipfile
import os

zip_file_path = "municipalities.zip"
extract_directory = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA"

# Ensure it's a zip file
if zipfile.is_zipfile(zip_file_path):
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        # Extract all files in the zip archive
        for file_name in zip_ref.namelist():
            if file_name.endswith('.dbf') or file_name.endswith('.shp') or fil
                zip_ref.extract(file_name, extract_directory)
                print(f"Extracted {file_name} to {extract_directory}")

else:
    print("Not a valid zip file.")
```

```
Extracted municipalities.dbf to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted municipalities.prj to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted municipalities.shp to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted municipalities.shx to C:\Users\ADMIN\Downloads\SPATIAL_DATA
```

After extracting the files, we will use geopandas to read the `municipalities.shp` file which we will be using in this question. we will use the `gdp.read_file()` funtion to open the `municipalities.shp` data file which is a shapefile file. see cells below

In [4]:
```python
#reading the `municipalities.shp` file using geopandas

shp_file_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\municipalities.shp"

# Read Shapefile using geopandas
gdf = gpd.read_file(shp_file_path)

# Print the GeoDataFrame (summary)
print(gdf)

# You can access the geometry and attributes of the GeoDataFrame
for index, row in gdf.iterrows():
    print(row.geometry)  # Access geometry
    print(row)  # Access attributes
```

```
3     POLYGON ((-48.53014 -3.1953, -48.49342 -3.0943...
4     POLYGON ((-49.9747 -1.30028, -50.01744 -1.3696...
...                                                  ...
5835  POLYGON ((-42.1502 -14.49001, -42.17334 -14.40...
5836  POLYGON ((-44.26816 -20.25036, -44.28688 -20.1...
5837  POLYGON ((-51.50539 -29.08148, -51.47652 -29.0...
5838  POLYGON ((-54.24864 -3.26644, -54.28551 -3.402...
5839  POLYGON ((-48.80551 -28.35525, -48.8373 -28.42...

[5840 rows x 7 columns]
POLYGON ((-48.827804 -2.601164, -48.832337 -2.635788, -48.939222 -2.729819,
-49.031783 -2.857346, -49.036809 -2.90952, -48.991506 -2.977196, -48.990086
-3.062913, -48.918868 -3.10772, -48.917616 -3.161065, -48.966274 -3.212913,
-48.925472 -3.406675, -48.925176 -3.41257, -49.472112 -3.414883, -49.563392
-3.120295, -49.615809 -3.088359, -49.623028 -3.000503, -49.562094 -2.93910
5, -49.453223 -2.89107, -49.467609 -2.745922, -49.39909 -2.714644, -49.3860
75 -2.663609, -49.323696 -2.672726, -49.202063 -2.482448, -49.23145 -2.4129
85, -49.159255 -2.368618, -49.090249 -2.240619, -49.02581 -2.216399, -48.95
1211 -2.141039, -48.963928 -2.064459, -48.874316 -1.993831, -48.905335 -1.9
```

The next step is Opening municipalities and filtering the sao paulo `(SP)` state. from the above file, we note a column named `UF` which is the column used to show the states in Brazil. The states are abbreviated in two letters and include `SP` for `São Paulo`, `RJ` for `Rio de Janeiro`, `MG` for `Minas Gerais`, `RS` for `Rio Grande do Sul`, and `BA` for `Bahia`. To achieve this, we will filter the municipalities data to only include municipalities in Sao Paulo(SP) state using the filter `['UF=='SP']`. See cell below

In [5]:

```python
# Define the path to the shapefile
shp_file_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\municipalities.shp"

# Read the shapefile using geopandas
municipalities = gpd.read_file(shp_file_path)

# Filter the GeoDataFrame for São Paulo state where 'UF' is 'SP'
municipalities_SP = municipalities[municipalities['UF'] == 'SP']

# Print the filtered GeoDataFrame
print(municipalities_SP)

# Optional: Save the filtered data to a new shapefile or other formats
output_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\municipalities_SP.shp"
municipalities_SP.to_file(output_path)
```

```
      COD_MUN                NOME  UF   POP_201  IDHM_10  PIB_PER  \
539   3509908            CANANEIA  SP   12216.0    0.720   9201.0
540   3509908            CANANEIA  SP   12216.0    0.720   9201.0
541   3509908            CANANEIA  SP   12216.0    0.720   9201.0
558   3505351     BARRA DO CHAPEU  SP    5305.0    0.660   7305.0
559   3522653  ITAPIRAPUA PAULISTA  SP    3926.0    0.661   6822.0
...       ...                 ...  ..       ...      ...      ...
5027  3550704       SAO SEBASTIAO  SP   76344.0    0.772  42410.0
5028  3550704       SAO SEBASTIAO  SP   76344.0    0.772  42410.0
5395  3550407           SAO PEDRO  SP   32231.0    0.755  12870.0
5445  3538709          PIRACICABA  SP  369919.0    0.785  29959.0
5452  3501905              AMPARO  SP   66649.0    0.785  30731.0

                                               geometry
539   POLYGON ((-48.09754 -25.31024, -48.24004 -24.9...
540   POLYGON ((-47.91404 -25.16738, -47.9124 -25.16...
541   POLYGON ((-47.86343 -25.12852, -47.85206 -25.1...
558   POLYGON ((-49.25004 -24.44493, -49.24976 -24.4...
559   POLYGON ((-49.20744 -24.70042, -49.31154 -24.6...
...                                                 ...
5027  POLYGON ((-45.78719 -23.86294, -45.78386 -23.8...
5028  POLYGON ((-45.72404 -23.80481, -45.71929 -23.7...
5395  POLYGON ((-47.80306 -22.58174, -47.81845 -22.6...
5445  POLYGON ((-48.08808 -22.6511, -48.07079 -22.64...
5452  POLYGON ((-46.76112 -22.84724, -46.83663 -22.8...

[659 rows x 7 columns]
```

From the abve cell, we can see that there are **659 municipalities** in `SP` state

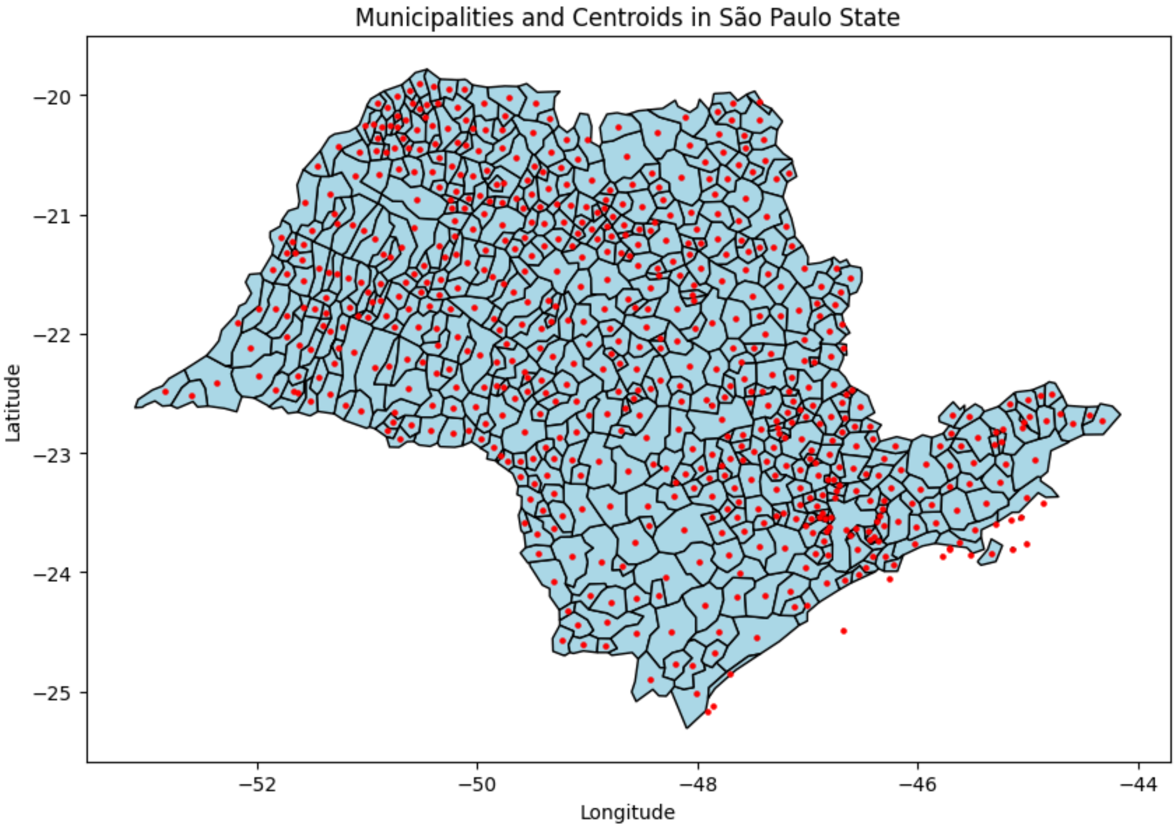
The next step will be **Generating Cetroids for SP municipalities and plting them.**

To achieve this, we will use the `municipalities.geometry.centroid` command to calculates the centroid for each geometry in the generated data, `municipalities_SP` ,which include municicipalities in Sao Paulo State only, and create a column named `centroid` which will hold the calculated centroids. we will then usse the `.plot` command to plot the calculated centroids.

See cell below

In [6]:
```
# Generate the centroids
municipalities_SP['centroid'] = municipalities_SP.geometry.centroid

# Plot the original geometries and their centroids
fig, ax = plt.subplots(figsize=(10, 10))
municipalities_SP.plot(ax=ax, color='lightblue', edgecolor='black')
municipalities_SP.set_geometry('centroid').plot(ax=ax, color='red', markersize

# Set plot title and labels
ax.set_title('Municipalities and Centroids in São Paulo State')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

# Show plot
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\3975828175.py:2: UserWarnin
g: Geometry is in a geographic CRS. Results from 'centroid' are likely incorr
ect. Use 'GeoSeries.to_crs()' to re-project geometries to a projected CRS bef
ore this operation.

  municipalities_SP['centroid'] = municipalities_SP.geometry.centroid
C:\Users\ADMIN\anaconda3\envs\gdal_env\lib\site-packages\geopandas\geodatafra
me.py:1819: 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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  super().__setitem__(key, value)
```

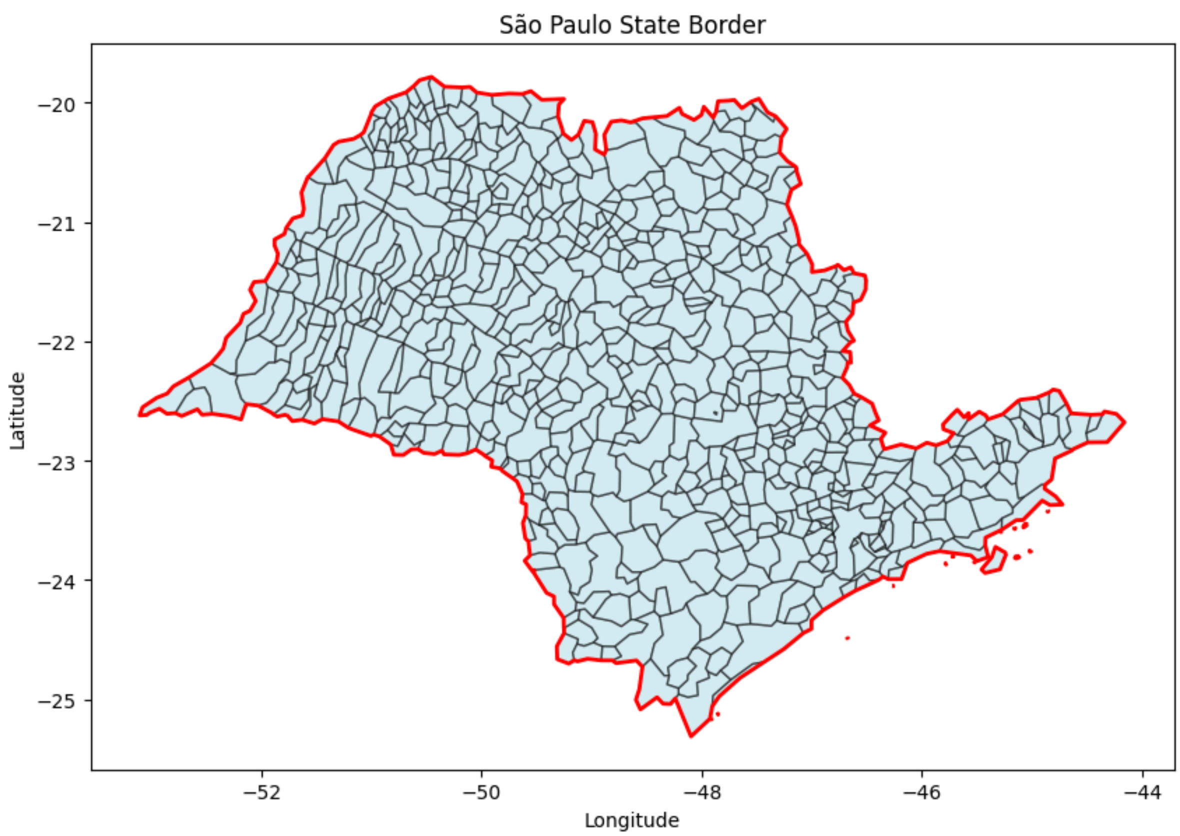


After calculating the centroids, the next step will be `**Dissolving SP state polygons to create SP state borders**`

To achieve this, we will use the `.dissolve()` function on the `municipalities_SP` data and filter by column `UF`. The function mergea all municipalities' polygons into a single polygon representing the state border. we will then proceed to plot the same using the `.plot` function to visualize the state borders.

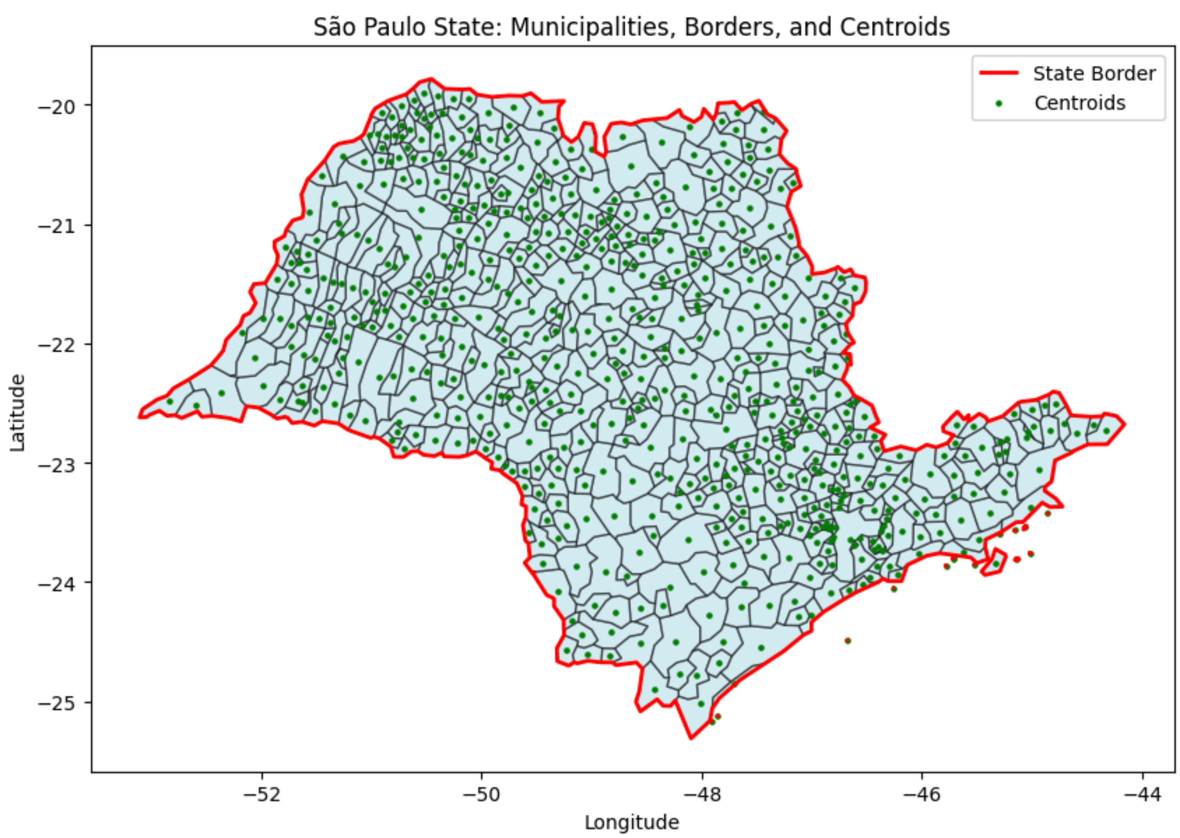
see cell below

In [7]:
```
# Dissolve the municipalities to create a single polygon for São Paulo state
sp_state_border = municipalities_SP.dissolve(by='UF')

# Plot the original municipalities and the dissolved state border
fig, ax = plt.subplots(figsize=(10, 10))

# Plot municipalities
municipalities_SP.plot(ax=ax, color='lightblue', edgecolor='black', alpha=0.5)

# Plot the state border
sp_state_border.boundary.plot(ax=ax, color='red', linewidth=2)

# Set plot title and labels
ax.set_title('São Paulo State Border')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

# Show plot
plt.show()
```



The next step will be **Ploting SP cetroids and borders together**. See the cell below

In [8]:
```
# Plot the original municipalities, the state border, and the centroids
fig, ax = plt.subplots(figsize=(10, 10))

# Plot municipalities
municipalities_SP.plot(ax=ax, color='lightblue', edgecolor='black', alpha=0.5)

# Plot the state border
sp_state_border.boundary.plot(ax=ax, color='red', linewidth=2, label='State Bo

# Plot the centroids
municipalities_SP.set_geometry('centroid').plot(ax=ax, color='green', markersi

# Set plot title and labels
ax.set_title('São Paulo State: Municipalities, Borders, and Centroids')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.legend()

# Show plot
plt.show()
```



QUESTION 2

WHAT IS THE MEAN HUMAN DEVELOPMENT INDEX OF MUNICIPALITIES IN EACH STATE OF BRAZIL?

To answer this question, we will still be using the `municipalities.shp` data from the Brazilian municipalities data. we are required to calculate the mean Human Development Index of municipalities of each state in Brazil. To achieve this, we will use the `IDHM_10` column in the dataset as the variable for Human development index, and then use the `UF` Column as the variable identifying the various municipalities in Brazil.

Step 1: Displaying data using Pandas

First, lets display our municipalities data which we named `gdp` in a tabular format using `pandas`

In [9]:
```
import pandas as pd # imports pandas library

print("Gdf Data:") #displays in tabular form
print(gdf.head()) #displays the 1st five rows
```

```
Gdf Data:
   COD_MUN       NOME  UF  POP_201  IDHM_10  PIB_PER  \
0  1504703       MOJU  PA  72597.0    0.547   3894.0
1  1507953  TAILANDIA  PA  85468.0    0.588   5405.0
2  1500206      ACARA  PA  53787.0    0.506   4389.0
3  1508001   TOME ACU  PA  57914.0    0.586   4765.0
4  1501808     BREVES  PA  94779.0    0.503   3608.0

                                            geometry
0  POLYGON ((-48.8278 -2.60116, -48.83234 -2.6357...
1  POLYGON ((-48.92547 -3.40668, -48.96627 -3.212...
2  POLYGON ((-48.49495 -2.55859, -48.567 -2.48282...
3  POLYGON ((-48.53014 -3.1953, -48.49342 -3.0943...
4  POLYGON ((-49.9747 -1.30028, -50.01744 -1.3696...
```

Step 2: Calculating mean HDI

Now, lets calculate the Mean HDI by states. we will use the `.groupby()` function to group the municipalities into the various states in Brazil, filtering with the column `UF` . After grouping, we will then use the `.mean()` function to calculate the mean Human development Index, filtering by column `IDHM_10` . To display the results in a dataframe, we will use the `.reset_index()` function on the calculated mean. See the cell below

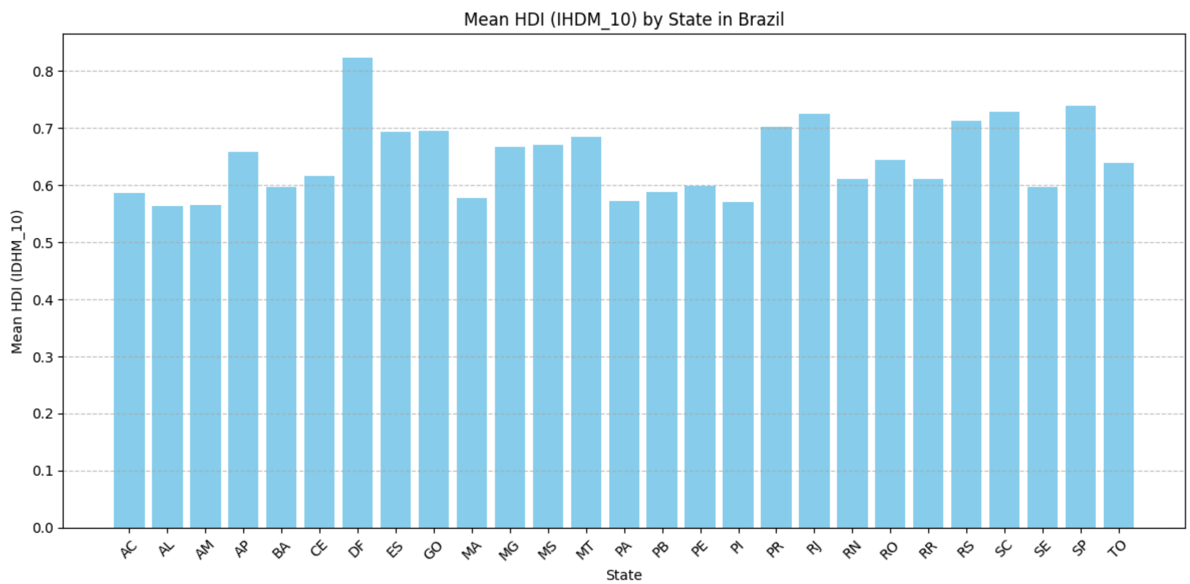
In [10]:
```python
# Step 1: Group by State (UF)
grouped = gdf.groupby('UF')

# Step 2: Calculate Mean HDI (IHDM_10)
mean_hdi = grouped['IDHM_10'].mean()

# Step 3: Reset Index to Display Results as DataFrame
mean_hdi_df = mean_hdi.reset_index()

# Print or display the resulting DataFrame
print("Mean HDI (IDHM_10) by State:")
print(mean_hdi_df)
```

```
Mean HDI (IDHM_10) by State:
    UF   IDHM_10
0   AC  0.586091
1   AL  0.563500
2   AM  0.565113
3   AP  0.657696
4   BA  0.596492
5   CE  0.616630
6   DF  0.824000
7   ES  0.694127
8   GO  0.694668
9   MA  0.577129
10  MG  0.667878
11  MS  0.671101
12  MT  0.684352
13  PA  0.572699
14  PB  0.588371
15  PE  0.598299
16  PI  0.571049
17  PR  0.701748
18  RJ  0.725589
19  RN  0.610850
20  RO  0.644038
21  RR  0.610200
22  RS  0.712383
23  SC  0.729101
24  SE  0.596933
25  SP  0.739829
26  TO  0.639928
```

Step 3: Plotting the results

After, calculating the mean HDI, lets visualize the results using a bar plot.

In [11]:
```
#Plotting the mean HDI values for each state
plt.figure(figsize=(12, 6))
plt.bar(mean_hdi_df['UF'], mean_hdi_df['IDHM_10'], color='skyblue')
plt.title('Mean HDI (IHDM_10) by State in Brazil')
plt.xlabel('State')
plt.ylabel('Mean HDI (IDHM_10)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show plot
plt.show()
```



QUESTION 3

PRODUCE A POLYGON/SHAPEFILE MAPPING THE AREA OF THE MUNICIPALITY ‘GAUCHA DO NORTE’ THAT IS IN THE INDIGENOUS TERRITORY “PARQUE DO XINGU”.

In this task, we will be using the Indigenous territory data zip file. we will upload and extract all the files in the zip, then use the `.shp` data to produce the shapefile mapping of the `Gau do Norte` i the `Parque do Xingu` indigenous territory.

Step 1: uploading, extracting and reading abd visualizing shapefiles

In [12]:

```python
#uploading and extracting
zip_file_path1 = "indigenous.zip"
extract_directory = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA"

# Ensure it's a zip file
if zipfile.is_zipfile(zip_file_path1):
    with zipfile.ZipFile(zip_file_path1, 'r') as zip_ref:
        # Extract all files in the zip archive
        for file_name in zip_ref.namelist():
            if file_name.endswith('.dbf') or file_name.endswith('.shp') or fil
                zip_ref.extract(file_name, extract_directory)
                print(f"Extracted {file_name} to {extract_directory}")

else:
    print("Not a valid zip file.")
```

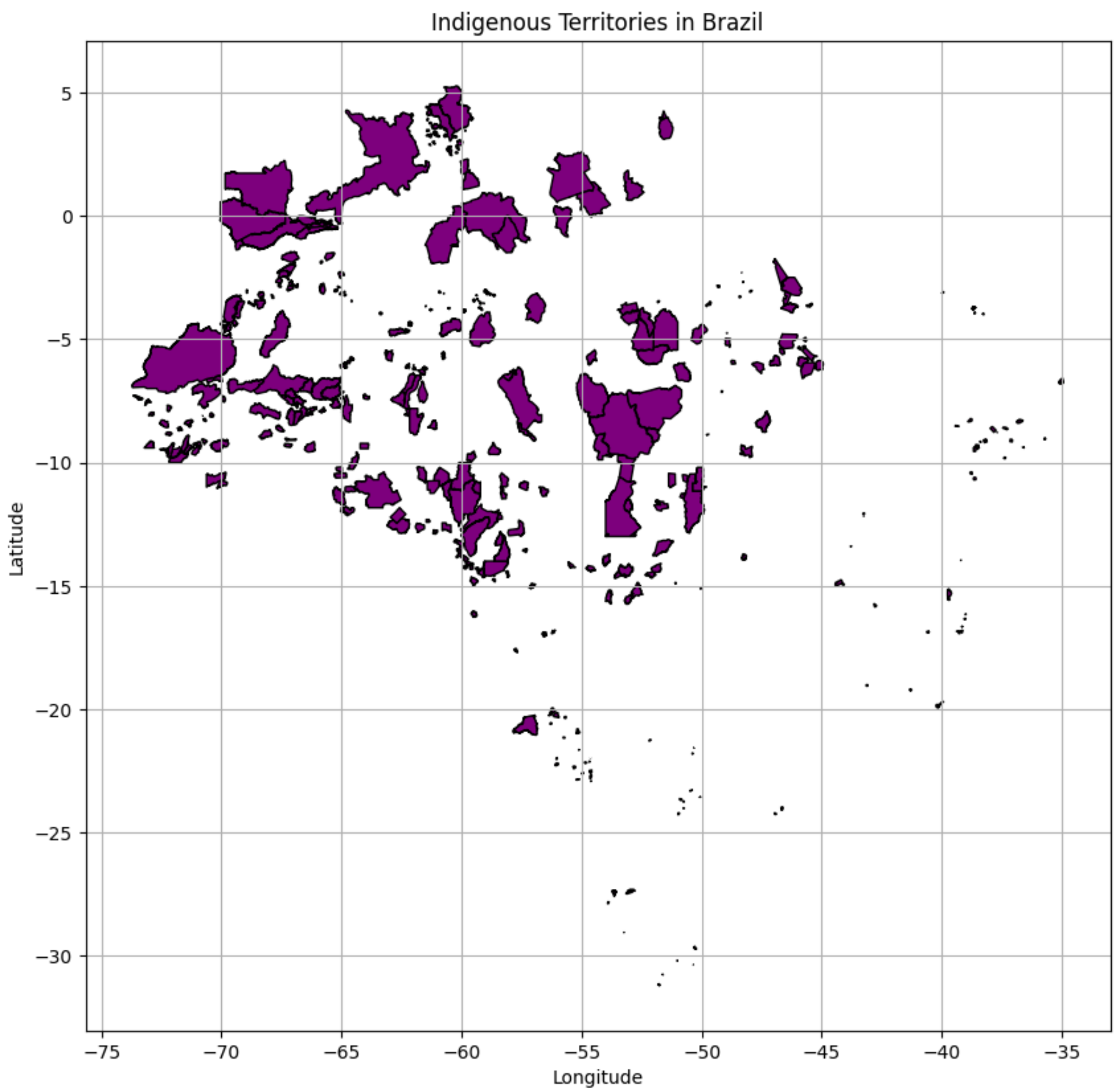
```
Extracted BC250_Terra_Indigena_A.dbf to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted BC250_Terra_Indigena_A.prj to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted BC250_Terra_Indigena_A.shp to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted BC250_Terra_Indigena_A.shx to C:\Users\ADMIN\Downloads\SPATIAL_DATA
```

In [16]:
```
#reading the .shp file
shp_file_path1 = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\BC250_Terra_Indigena_

# Read Shapefile using geopandas
indigenous = gpd.read_file(shp_file_path1)

# Print the GeoDataFrame (summary)
print(indigenous)

# You can access the geometry and attributes of the GeoDataFrame
for index, row in indigenous.iterrows():
    print(row.geometry)  # Access geometry
    print(row)  # Access attributes
```

```
1...
1          3.0         NaN  POLYGON ((-51.03401 -30.19091, -51.03126 -30.
1...
2          4.0         NaN  POLYGON ((-51.60478 -30.76695, -51.60868 -30.
7...
3          7.0         NaN  POLYGON ((-50.35966 -30.35949, -50.35949 -30.
3...
4          1.0         NaN  POLYGON ((-53.2281 -29.07109, -53.22922 -29.0
7...
..         ...         ...
...
427        NaN         NaN  POLYGON ((-53.62068 -27.58497, -53.62376 -27.
5...
428        NaN         NaN  POLYGON ((-53.13552 -27.48002, -53.13552 -27.
4...
429        NaN         NaN  POLYGON ((-53.07222 -27.37842, -53.07268 -27.
3...
430        NaN         NaN  POLYGON ((-59.61879 -10.00834, -59.62129 -10.
0...
```

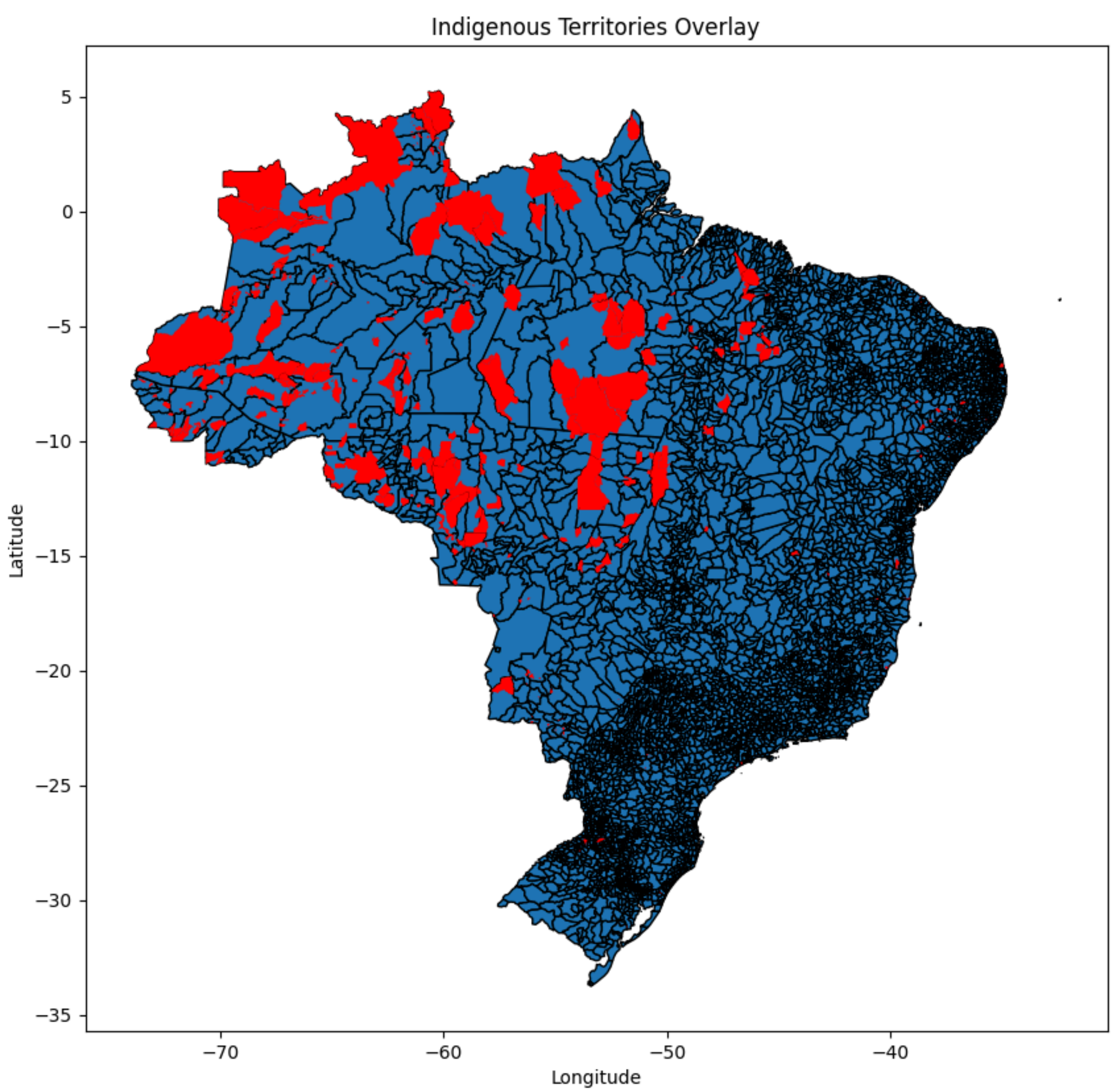
Lets use geopandas to visualize the map of the indigenous territory in Brazil. See cell below

In [17]:
```
# Plot the geometries
plt.figure(figsize=(10, 10))
indigenous.plot(ax=plt.gca(), color='purple', edgecolor='black')

# Set plot title and labels
plt.title('Indigenous Territories in Brazil')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)

# Show plot
plt.show()
```



In [18]:
```
# ploting the indigenous and grouped states in Brazil
indigenous = indigenous.to_crs(epsg=4326)

# Plotting
fig, ax = plt.subplots(figsize=(10, 10))

# Plot Brazilian states
grouped.plot(ax=ax, edgecolor='black')

# Overlay indigenous territories with fill color red
indigenous.plot(ax=ax, color='red')

# Set plot title and labels
plt.title('Indigenous Territories Overlay')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

Out[18]:
```
Text(80.47222222222221, 0.5, 'Latitude')
```



Step 2: Creating and plotting a shapefile for Gaucha do Norte e Xingu

This step involves generating the specific locations of `parque do Xingu` and `Gaucha do Norte` from filtering through the `indigenous` and `gdf` data respectively. we will then use the `to_crs` method to transform the coordinate reference system of Gaucho do Norte and Xingu to EPSG:4326 and save the filtered data into shapefiles. see the cells below

In [19]:
```python
# Filter for "Parque do Xingu" in the indigenous data
xingu = indigenous[indigenous['nome'] == "Parque do Xingu"]

# Filter for "Gaucha do Norte" in the municipalities data
gaucha_do_norte = gdf[gdf['NOME'] == "GAUCHA DO NORTE"]

# Transform the coordinate reference system to EPSG:4326
xingu = xingu.to_crs(epsg=4326)
gaucha_do_norte = gaucha_do_norte.to_crs(epsg=4326)

# Print the filtered GeoDataFrames
print("Filtered Xingu GeoDataFrame:")
print(xingu)
print("\nFiltered Gaucha do Norte GeoDataFrame:")
print(gaucha_do_norte)

# Define output paths for shapefiles
xingu_output_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\xingu.shp"
gaucha_do_norte_output_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\gaucha_d

# Save the filtered GeoDataFrames to shapefiles
xingu.to_file(xingu_output_path)
gaucha_do_norte.to_file(gaucha_do_norte_output_path)

print(f"\nShapefiles created at {xingu_output_path} and {gaucha_do_norte_outpu
```

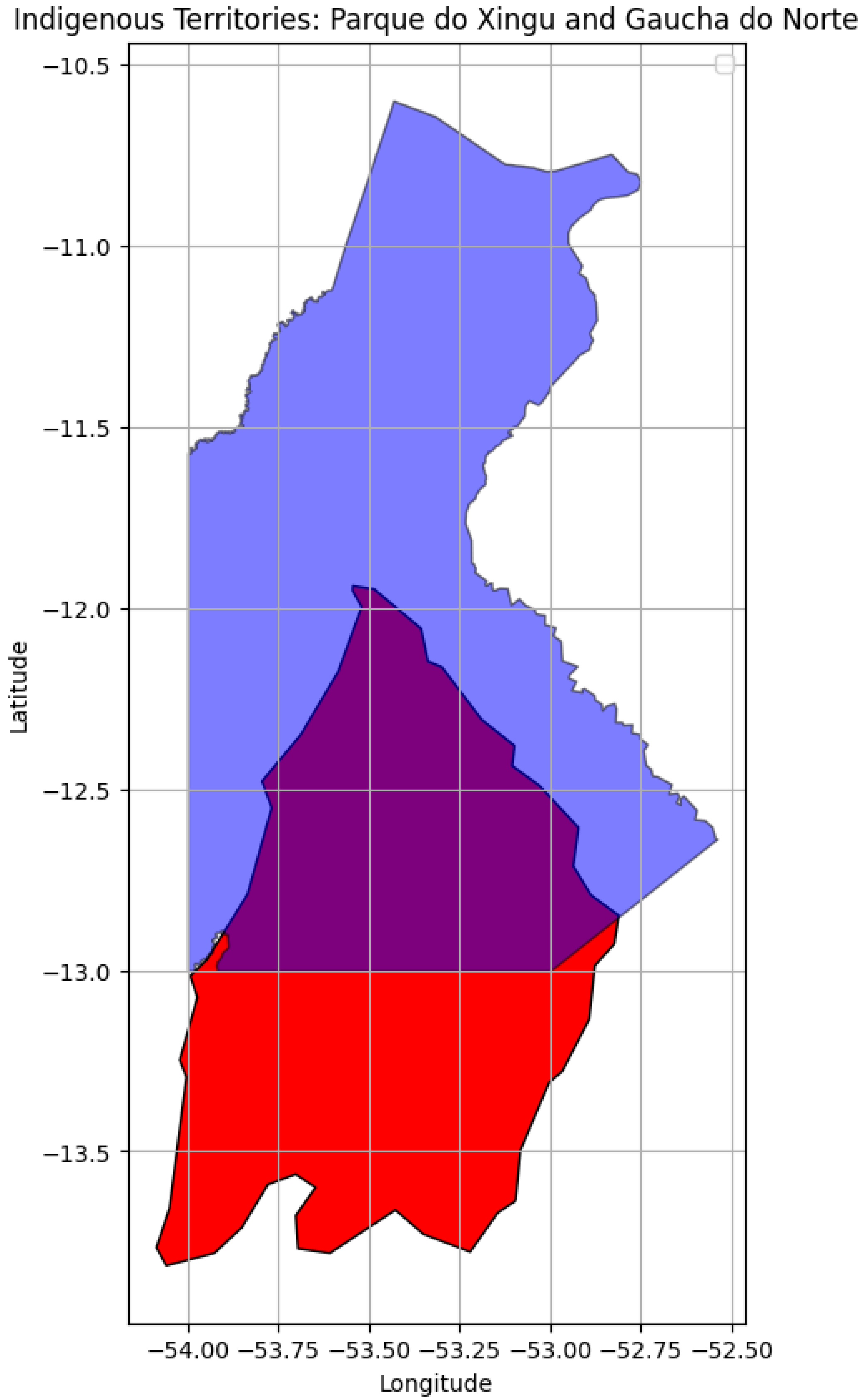
```
Filtered Xingu GeoDataFrame:
     id_objeto             nome        nomeabrev geometriaa  perimetroo  \
405        411  Parque do Xingu  Parque do Xingu        Não     33801.0

     areaoficia        grupoetnic datasituac situacaoju  \
405   2642000.0  Mentuktire, Suyá  1987/5/18  Declarada

                                 nometi  id_produto  id_element  codigofuna
\
405  Terra tradicional - Proc.concluído      250002        29.0         NaN

                                              geometry
405  POLYGON ((-52.91281 -11.05436, -52.92222 -11.0...

Filtered Gaucha do Norte GeoDataFrame:
      COD_MUN             NOME  UF  POP_201  IDHM_10  PIB_PER  \
1200  5103858  GAUCHA DO NORTE  MT   6548.0    0.615  15926.0

                                               geometry
1200  POLYGON ((-53.09758 -13.63437, -53.14672 -13.6...

Shapefiles created at C:\Users\ADMIN\Downloads\SPATIAL_DATA\xingu.shp and
C:\Users\ADMIN\Downloads\SPATIAL_DATA\gaucha_do_norte.shp
```

Now, lets plot the shapefiles to see the overlap

In [20]:

```python
# Plotting the shapefiles (if Gaucha do Norte has valid geometries)
if not gaucha_do_norte.empty and gaucha_do_norte.is_valid.all():
    fig, ax = plt.subplots(1, 1, figsize=(10, 10))

    # Plot "Gaucha do Norte" with fill color red
    gaucha_do_norte.plot(ax=ax, color='red', edgecolor='black', label='Gaucha 

    # Plot "Parque do Xingu" with fill color blue and alpha transparency
    xingu.plot(ax=ax, color='blue', edgecolor='black', alpha=0.5, label='Parqu

    # Set plot title and labels
    plt.title('Indigenous Territories: Parque do Xingu and Gaucha do Norte')
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')

    # Set aspect ratio to equal
    ax.set_aspect('equal')

    plt.legend()
    plt.grid(True)

    # Show plot
    plt.show()
else:
    print("Gaucha do Norte shapefile has invalid or missing geometries.")
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\1409849253.py:19: UserWarni
ng: Legend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#i
mplementing-a-custom-legend-handler (https://matplotlib.org/stable/tutorials/
intermediate/legend_guide.html#implementing-a-custom-legend-handler)
  plt.legend()
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\1409849253.py:19: UserWarni
ng: No artists with labels found to put in legend.  Note that artists whose l
abel start with an underscore are ignored when legend() is called with no arg
ument.
  plt.legend()
```



Step 3: creating a shapefile for intersection and plotting it

In [21]:
```python
# Perform intersection
intersection = gpd.overlay(gaucha_do_norte, xingu, how='intersection')

# Print the resulting GeoDataFrame
print(intersection)
```

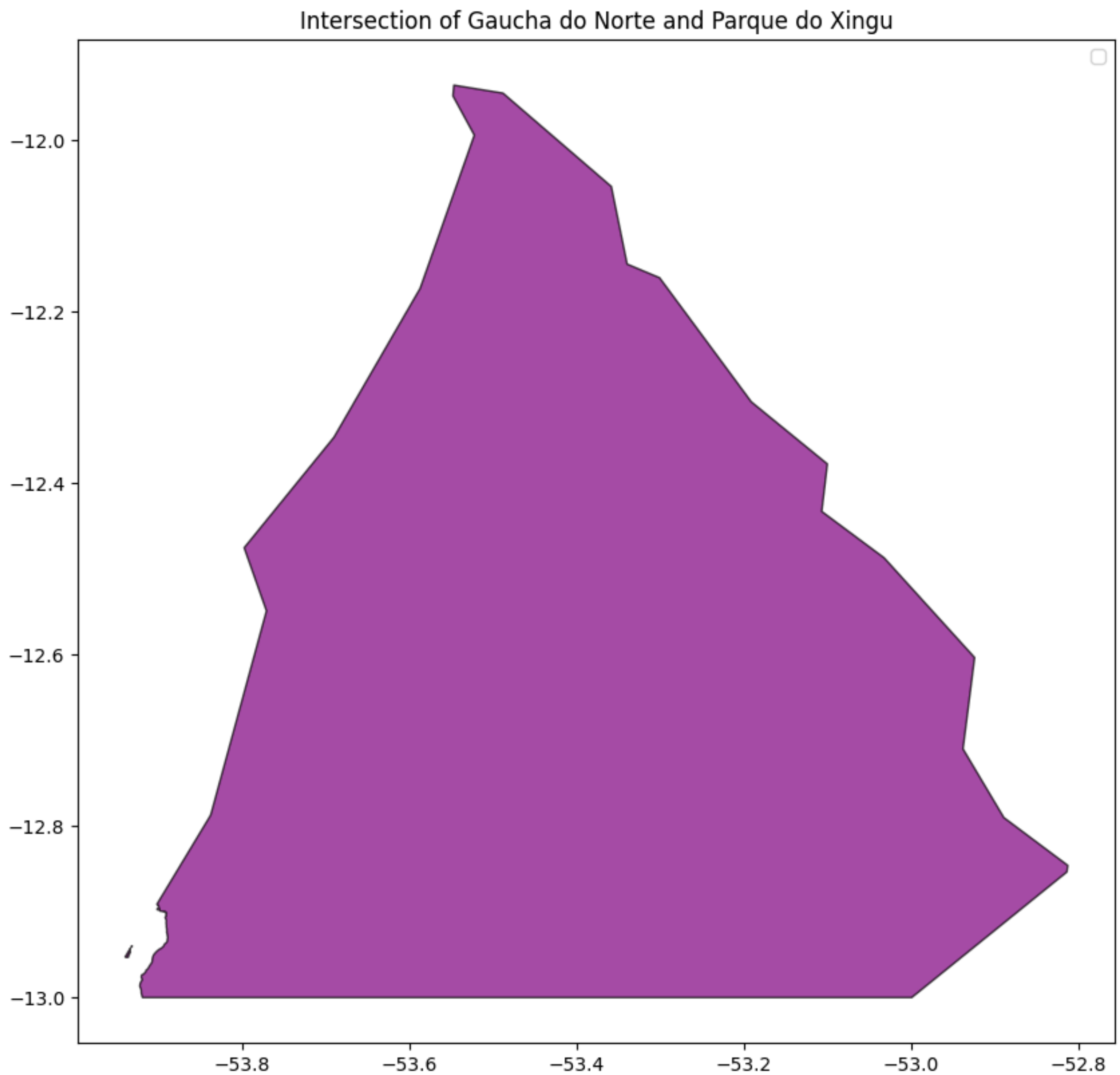
```
   COD_MUN             NOME  UF  POP_201  IDHM_10  PIB_PER  id_objeto  \
0  5103858  GAUCHA DO NORTE  MT   6548.0    0.615  15926.0        411   

              nome        nomeabrev geometriaa  perimetroo  areaoficia  \
0  Parque do Xingu  Parque do Xingu        Não     33801.0   2642000.0   

         grupoetnic datasituac situacaoju                              nometi  \
0  Mentuktire, Suyá  1987/5/18  Declarada  Terra tradicional - Proc.concluído   

   id_produto  id_element  codigofuna  \
0      250002        29.0         NaN   

                                            geometry  
0  MULTIPOLYGON (((-53.93366 -12.94289, -53.93362...  
```

Now lets generate a plot that visualizes only the intersection area of `Gaucho` and `Xingu` . see cell below

In [22]:

```python
# Plot the intersection area
fig, ax = plt.subplots(figsize=(10, 10))

# Plot the intersection with fill color purple
intersection.plot(ax=ax, color='purple', edgecolor='black', alpha=0.7, label='

# Set plot title and legend
plt.title('Intersection of Gaucha do Norte and Parque do Xingu')
plt.legend()

# Display the plot
plt.show()
```

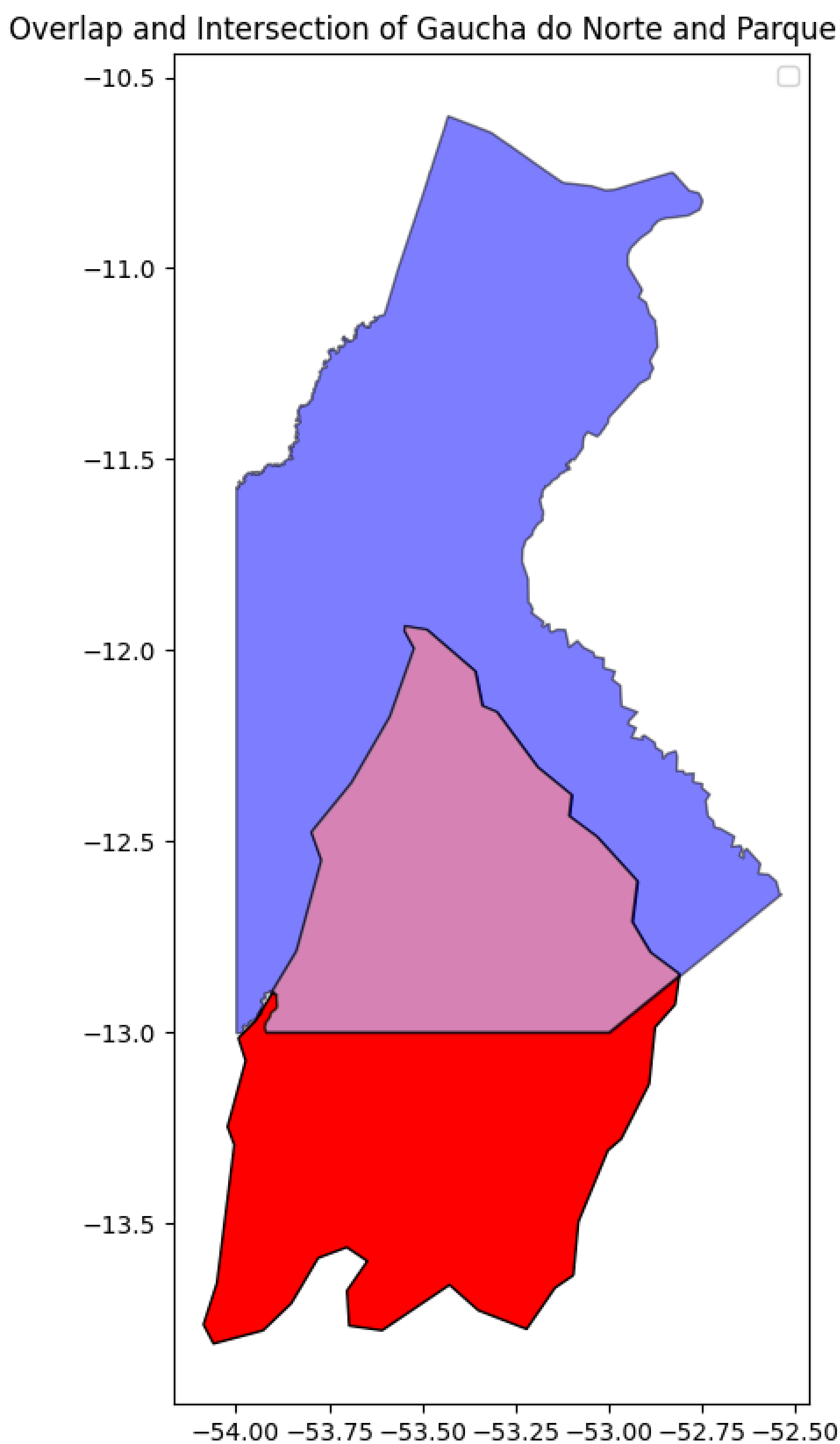
```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\466720264.py:9: UserWarnin
g: Legend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#i
mplementing-a-custom-legend-handler (https://matplotlib.org/stable/tutorials/
intermediate/legend_guide.html#implementing-a-custom-legend-handler)
  plt.legend()
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\466720264.py:9: UserWarnin
g: No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no argu
ment.
  plt.legend()
```



**Step 4: visualizing the overlaping shapefile and intersection of Gaucha and Xingu together. see the cell below

In [23]:
```
# Create a figure and axis
fig, ax = plt.subplots(figsize=(10, 10))

# Plot "Gaucha do Norte" with fill color red
gaucha_do_norte.plot(ax=ax, color='red', edgecolor='black', label='Gaucha do No

# Plot "Parque do Xingu" with fill color blue and alpha transparency
xingu.plot(ax=ax, color='blue', edgecolor='black', alpha=0.5, label='Parque do

# Plot the intersection with fill color purple
intersection.plot(ax=ax, color='pink', edgecolor='black', alpha=0.7, label='Int

# Set plot title and legend
plt.title('Overlap and Intersection of Gaucha do Norte and Parque do Xingu')
plt.legend()

# Display the plot
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\3228510615.py:15: UserWarni
ng: Legend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#i
mplementing-a-custom-legend-handler (https://matplotlib.org/stable/tutorials/
intermediate/legend_guide.html#implementing-a-custom-legend-handler)
  plt.legend()
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\3228510615.py:15: UserWarni
ng: No artists with labels found to put in legend.  Note that artists whose l
abel start with an underscore are ignored when legend() is called with no arg
ument.
  plt.legend()
```



Step 4: calculating the intersection

We will use geopandas function `gdp.overlay()` to calculate the intersection of `Gaucho` and `Xingu` . See the cell below

In [24]:
```python
# Calculate the intersection
intersection = gpd.overlay(gaucha_do_norte, xingu, how='intersection')

# Calculate the area of the intersection in square meters
intersection_area = intersection.area

# Summing up the area of all the intersecting geometries
total_intersection_area = intersection_area.sum()

# Print the total area of the intersection
print(f"Total area of intersection: {total_intersection_area:.2f} square meter
```

```
Total area of intersection: 0.67 square meters

C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\3681295758.py:5: UserWarnin
g: Geometry is in a geographic CRS. Results from 'area' are likely incorrect.
Use 'GeoSeries.to_crs()' to re-project geometries to a projected CRS before t
his operation.

  intersection_area = intersection.area
```

lets recalculate the intersection area taking into consideration the CRS warning above and see the difference

In [25]:
```python

# Define the target CRS
target_crs = 'EPSG:32723'  # UTM zone 23S

# Reproject the GeoDataFrames to the target CRS
gaucha_do_norte_proj = gaucha_do_norte.to_crs(target_crs)
xingu_proj = xingu.to_crs(target_crs)

# Calculate the intersection
intersection = gpd.overlay(gaucha_do_norte_proj, xingu_proj, how='intersection

# Calculate the area of the intersection in square meters
intersection_area = intersection.area

# Summing up the area of all the intersecting geometries
total_intersection_area = intersection_area.sum()

# Print the total area of the intersection
print(f"Total area of intersection: {total_intersection_area:.2f} square meter
```

```
Total area of intersection: 8240145386.92 square meters
```

The total intersection area in square meters before reprojecting the data to the new target crs was **0.67** but it is now **8240145386.92** which We will adopt

QUESTION 4

In the state of Acre (AC), which two social housing (MCMV) projects are closest to each other? Create a 10km buffer around each housing project.

To answer this question, we will follow the following steps;

- Importing the `housing` shapefile from the `"MCMV_new.shp"` data file

**STEP 1: Importing the `housing` shapefile from `MCMV_new.shp` file which is found in the `MCMV_new.zip`

In [26]:
```
import zipfile
import os

zip_file_path = "MCMV_new.zip"
extract_directory = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA"

# Ensure it's a zip file
if zipfile.is_zipfile(zip_file_path):
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        # Extract all files in the zip archive
        for file_name in zip_ref.namelist():
            if file_name.endswith('.dbf') or file_name.endswith('.shp') or fil
                zip_ref.extract(file_name, extract_directory)
                print(f"Extracted {file_name} to {extract_directory}")

else:
    print("Not a valid zip file.")
```

```
Extracted MCMV_new.dbf to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted MCMV_new.prj to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted MCMV_new.shp to C:\Users\ADMIN\Downloads\SPATIAL_DATA
Extracted MCMV_new.shx to C:\Users\ADMIN\Downloads\SPATIAL_DATA
```

In [27]:
```
#reading the `MCMV_new.shp` file using geopandas

shp_file_path = r"C:\Users\ADMIN\Downloads\SPATIAL_DATA\MCMV_new.shp"

# Read Shapefile using geopandas
housing = gpd.read_file(shp_file_path)

# Print the GeoDataFrame (summary)
print(housing)

# You can access the geometry and attributes of the GeoDataFrame
for index, row in housing.iterrows():
    print(row.geometry)  # Access geometry
    print(row)  # Access attributes
```

```
XCOORD                                    -67.955
YCOORD                                   -9.81528
UF                                             AC
UH                                            2.0
Project_ID                                     18
geometry       POINT (-67.9550289999998 -9.81528)
Name: 17, dtype: object
POINT (-67.94628 -3.09544)
XCOORD                         -67.9463
YCOORD                         -3.09544
UF                                   AM
UH                                 56.0
Project_ID                           19
geometry       POINT (-67.94628 -3.09544)
Name: 18, dtype: object
POINT (-67.8099979999999 -9.97472)
XCOORD                                      -67.81
YCOORD                                    -9.97472
UF                                              AC
UH                                          3782.0
```

In [28]:
```
housing.head() # displaying the 1st five entries
```

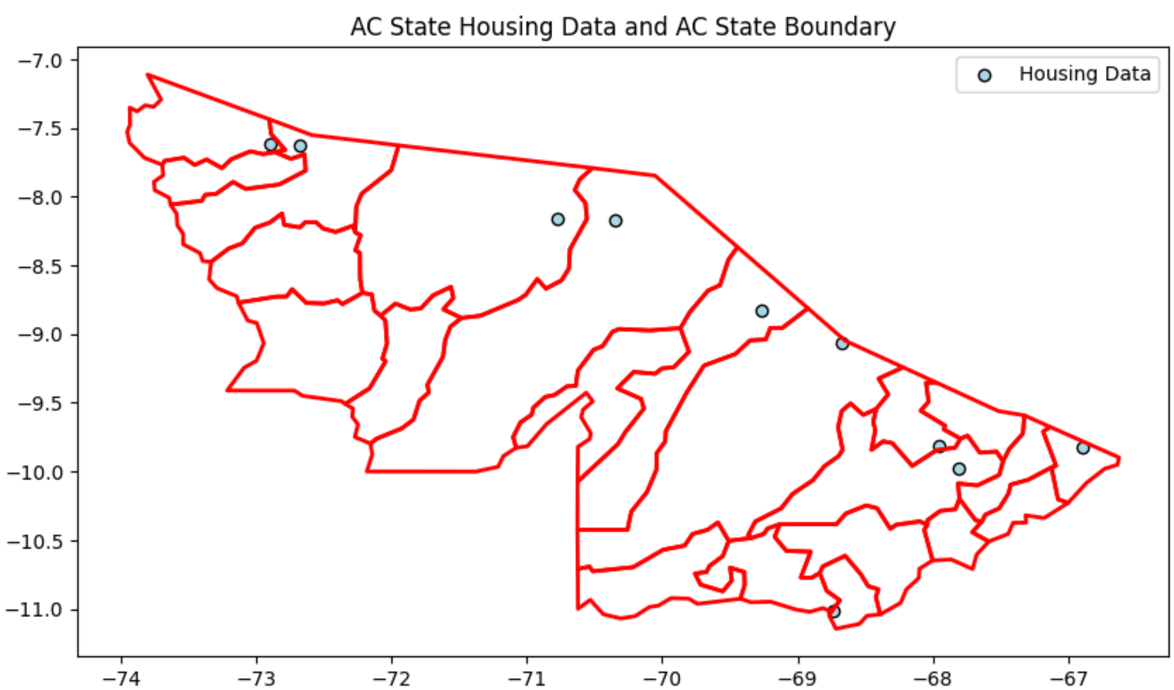
Out[28]:

	XCOORD	YCOORD	UF	UH	Project_ID	geometry
0	-72.8997	-7.61658	AC	1.0	1	POINT (-72.89971 -7.61658)
1	-72.6756	-7.62763	AC	18.0	2	POINT (-72.67558 -7.62762)
2	-72.5907	-7.53797	AM	31.0	3	POINT (-72.59071 -7.53797)
3	-71.6934	-7.04791	AM	30.0	4	POINT (-71.69343 -7.04791)
4	-70.7722	-8.15697	AC	16.0	5	POINT (-70.77215 -8.15697)

Step 2: selecting housing in AC state and plot the results

In [37]:
```python
# Filter housing data for AC state
AC_housing = housing[housing['UF'] == 'AC']
# Filter for Acre state (UF == 'AC') in 'gdf' shapefiles which consists of all
AC_boundaries = gdf[gdf['UF'] == 'AC']

# Plotting
fig, ax = plt.subplots(figsize=(10, 10))

# Plot AC_housing as a line plot
AC_housing.plot(ax=ax, color='lightblue', edgecolor='black', linewidth=1, labe

# Plot state boundaries for Acre
AC_boundaries.plot(ax=ax, color='none', edgecolor='red', linewidth=2, label='A
# Set plot title and legend
plt.title('AC State Housing Data and AC State Boundary')
plt.legend()

# Show plot
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\3585261957.py:17: UserWarni
ng: Legend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#i
mplementing-a-custom-legend-handler (https://matplotlib.org/stable/tutorials/
intermediate/legend_guide.html#implementing-a-custom-legend-handler)
  plt.legend()
```



Step 3: calculating distance between housing points in AC state

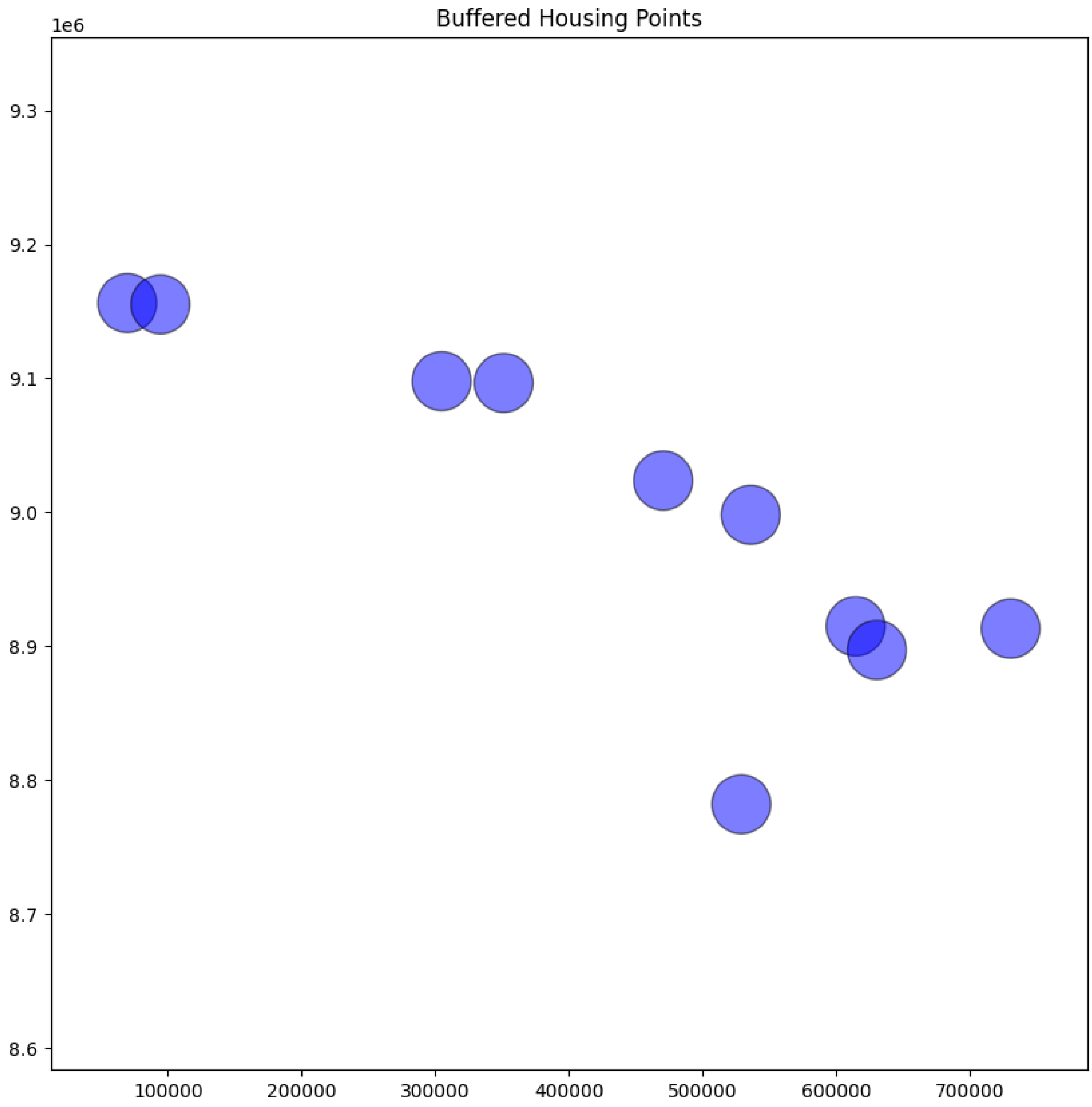
In [30]:
```python
# Select housing points for Acre state (assuming 'UF' column exists)
housing_ac = housing[housing['UF'] == 'AC']

# Transform to a suitable projected coordinate system, e.g., UTM zone 19S (EPSG
housing_ac = housing_ac.to_crs(epsg=29189)

# Calculate distances between all pairs of points
distance_matrix = housing_ac.geometry.apply(lambda geom: housing_ac.geometry.d

# Convert distances to DataFrame
distance_df = distance_matrix.apply(lambda row: row.apply(lambda x: x))

# Display distances (optional)
print("Distance Matrix:")
print(distance_df)
```

```
Distance Matrix:
                0              1              4              5             11
\
0        0.000000   24804.692814  242315.191867  287859.096682  422339.247729
1    24804.692814       0.000000  218072.028757  263435.239451  398549.471619
4   242315.191867  218072.028757       0.000000   46436.005464  181680.919373
5   287859.096682  263435.239451   46436.005464       0.000000  139918.064523
11  422339.247729  398549.471619  181680.919373  139918.064523       0.000000
13  592605.187265  572944.855565  387497.333745  361461.063066  248657.479335
14  492622.070381  468877.052866  251979.334173  209540.526918   70363.187574
17  596084.109394  573086.541968  359890.280003  320046.947665  180507.214632
19  617807.689579  594959.925946  382594.256474  343176.135174  203859.092190
25  704344.747930  680777.648732  464241.059215  421487.871698  282581.530754

               13             14             17             19             25
0   592605.187265  492622.070381  596084.109394  617807.689579  704344.747930
1   572944.855565  468877.052866  573086.541968  594959.925946  680777.648732
4   387497.333745  251979.334173  359890.280003  382594.256474  464241.059215
5   361461.063066  209540.526918  320046.947665  343176.135174  421487.871698
11  248657.479335   70363.187574  180507.214632  203859.092190  282581.530754
13       0.000000  216294.808538  158054.273892  153476.843983  240517.033810
14  216294.808538       0.000000  114423.546017  138161.084384  212262.981591
17  158054.273892  114423.546017       0.000000   23743.747455  116049.871634
19  153476.843983  138161.084384   23743.747455       0.000000  101459.466850
25  240517.033810  212262.981591  116049.871634  101459.466850       0.000000
```

Step 5: creating a buffer around the housing points We will use the `buffer` method in geopaandas to create a buffer with a distance range of 2000 meters around housing points, then visualize the same. see cell below

In [40]:

```
# Create a buffer of 20000 meters around housing points
# Transform to EPSG 29189 (SIRGAS 2000 / UTM zone 19S)
AC_housing = AC_housing.to_crs(epsg=29189)

buffer_distance = 20000  # Adjust the buffer distance as needed
AC_housing['geometry'] = AC_housing.geometry.buffer(buffer_distance)

# Plot the buffered housing points
fig, ax = plt.subplots(figsize=(10, 10))
AC_housing.plot(ax=ax, color='blue', alpha=0.5, edgecolor='k')
ax.set_title('Buffered Housing Points')
plt.axis('equal')  # Ensure equal scaling of x and y axes
plt.show()
```

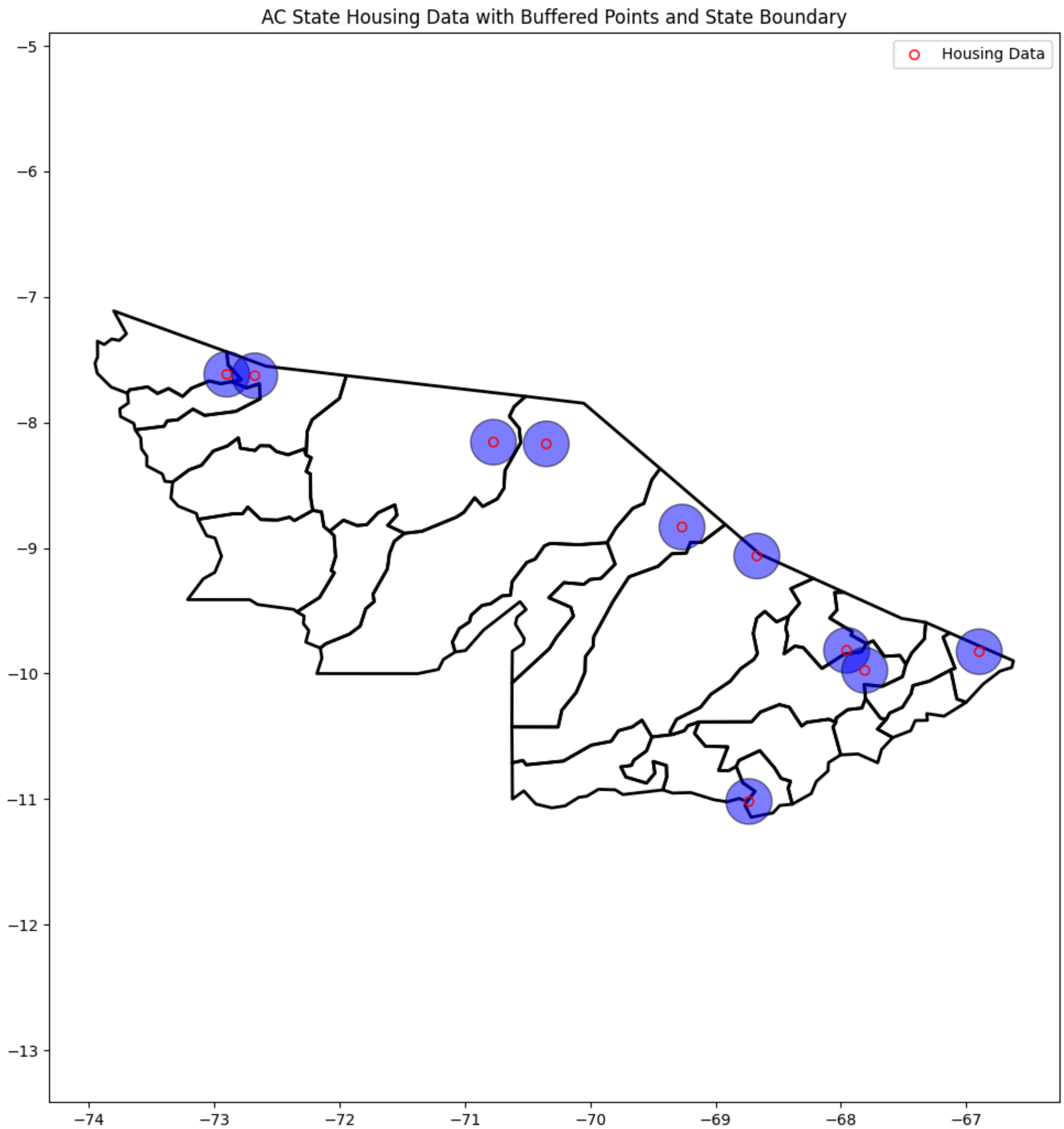


Step 6: Plotting the `AC state housing and boundaries data` together with the `AC State Buffered housing points` data.

In [53]:

```
# Filter housing data for AC state
AC_housing = housing[housing['UF'] == 'AC']
# Filter for Acre state (UF == 'AC') in 'gdf' shapefiles which consists of all
AC_boundaries = gdf[gdf['UF'] == 'AC']

# Create a buffer of 20000 meters around housing points after transforming to
AC_housing_buffered = AC_housing.to_crs(epsg=29189)
buffer_distance = 20000  # Adjust the buffer distance as needed
AC_housing_buffered['geometry'] = AC_housing_buffered.geometry.buffer(buffer_d

# Transform buffered housing points back to the original CRS for consistent pl
AC_housing_buffered = AC_housing_buffered.to_crs(AC_housing.crs)

# Plotting
fig, ax = plt.subplots(figsize=(10, 10))

# Plot state boundaries for Acre
AC_boundaries.plot(ax=ax, color='none', edgecolor='black', linewidth=2, label=

# Plot buffered AC housing data
AC_housing_buffered.plot(ax=ax, color='blue', alpha=0.5, edgecolor='k', label=

# Plot original AC housing data
AC_housing.plot(ax=ax, color='none', edgecolor='red', linewidth=1, label='Hous

# Set plot title and legend
plt.title('AC State Housing Data with Buffered Points and State Boundary')
plt.legend()

# Adjust layout to ensure everything fits within the plot area
plt.tight_layout()

# Ensure equal scaling of x and y axes
plt.axis('equal')

# Show plot
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_11312\2706102809.py:28: UserWarni
ng: Legend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#i
mplementing-a-custom-legend-handler (https://matplotlib.org/stable/tutorials/
intermediate/legend_guide.html#implementing-a-custom-legend-handler)
  plt.legend()
```



QUESTION 5

Across Brazil, which municipalities have the lowest and highest number of MCMV housing units (UH) in its territory? Create a map of the distribution of total housing units by municipality.

This question seeks to identify the lowest and highest housing units in `MCMV` (Minha Casa, Minha Vida) municipality in Brazil. the question also seeks to show the distribution of the housing units across MCMV visually. to achieve this we will:

- Perform a spatial join between the `municipalities` (in our case `gdf`) and `housing` data
- Calculate the total housing units in each municipality
- Pick the municipality with highest housing units
- Visualize the distribution of total housing units per municipality in Brazil

Spatial Join between `gdf` which is municipalities data and `housing` data

we will use the `gpd.sjoin method` to perform an `inner join` which ensures that the joinretains only the matching geometry. the `within` predicate is used to ensure that housing points are within specific municipalities

In [55]:
```
# Ensure both GeoDataFrames use the same CRS
gdf = gdf.to_crs(epsg=4326)  # Assuming the municipality shapefile uses EPSG:4.
housing = housing.to_crs(epsg=4326)  # Ensure the housing data is also in EPSG

# Perform spatial join
joined_gdf = gpd.sjoin(housing, gdf, how="inner", predicate="within")

# Print the result
print(joined_gdf.head())
```

```
    XCOORD    YCOORD UF_left    UH  Project_ID                      geometry  \
0 -72.8997 -7.61658      AC   1.0           1  POINT (-72.89971 -7.61658)
1 -72.6756 -7.62763      AC  18.0           2  POINT (-72.67558 -7.62762)
2 -72.5907 -7.53797      AM  31.0           3  POINT (-72.59071 -7.53797)
3 -71.6934 -7.04791      AM  30.0           4  POINT (-71.69343 -7.04791)
4 -70.7722 -8.15697      AC  16.0           5  POINT (-70.77215 -8.15697)

   index_right  COD_MUN             NOME UF_right  POP_201  IDHM_10  PIB_PER
0          392  1200336      MANCIO LIMA       AC  15890.0    0.625   8111.0
1          374  1200203  CRUZEIRO DO SUL       AC  79819.0    0.664  10643.0
2          395  1301654          GUAJARA       AM  14396.0    0.532   4388.0
3          398  1301803          IPIXUNA       AM  23460.0    0.481   4362.0
4          397  1200609         TARAUACA       AC  36763.0    0.539   8192.0
```

Calculating total housing units per municipality

We will use the `.groupeby` method in geopandas to group the joined data by `NOME` column which stands for the **municipalities name column** and the `UH` column which represents the **housing units column**

In [87]:

```
# Group by municipality and calculate the total number of housing units
total_housing_units = joined_gdf.groupby('NOME')['UH'].sum().reset_index()

# Print the result
print(total_housing_units)
```

```
                      NOME      UH
0          ABADIA DE GOIAS    20.0
1      ABADIA DOS DOURADOS    35.0
2                ABADIANIA    19.0
3                   ABAETE   128.0
4               ABAETETUBA  1257.0
...                    ...     ...
4289             XINGUARA      2.0
4290          XIQUE XIQUE     60.0
4291            ZACARIAS       1.0
4292              ZE DOCA     76.0
4293               ZORTEA     17.0

[4294 rows x 2 columns]
```

Selecting the municipality with most housing units

In [61]:

```
# Find the municipality with the highest number of housing units using the .loc
max_housing_municipality = total_housing_units.loc[total_housing_units['UH'].i

# Print the result
print(f"Municipality with most housing units:")
print(max_housing_municipality)
```

```
Municipality with most housing units:
NOME    RIO DE JANEIRO
UH             46966.0
Name: 3267, dtype: object
```

In [94]:

```
# Find the municipality with the lowest number of housing units
min_housing_municipality = total_housing_units.loc[total_housing_units['UH'].i

# Print the result
print(f"Municipality with the lowest number of housing units:")
print(min_housing_municipality)
```

```
Municipality with the lowest number of housing units:
NOME    ABDON BATISTA
UH                1.0
Name: 8, dtype: object
```

From the above cell, the municipality with the highest number of housing units in Brazil is **RIO DE JANEIRO** with **46,966 units** and the municipality with the lowest number of houing units i Brazil is **ABDON BAISTA** with only **1 unit**

Visualize the distribution of total housing units per municipality in Brazil

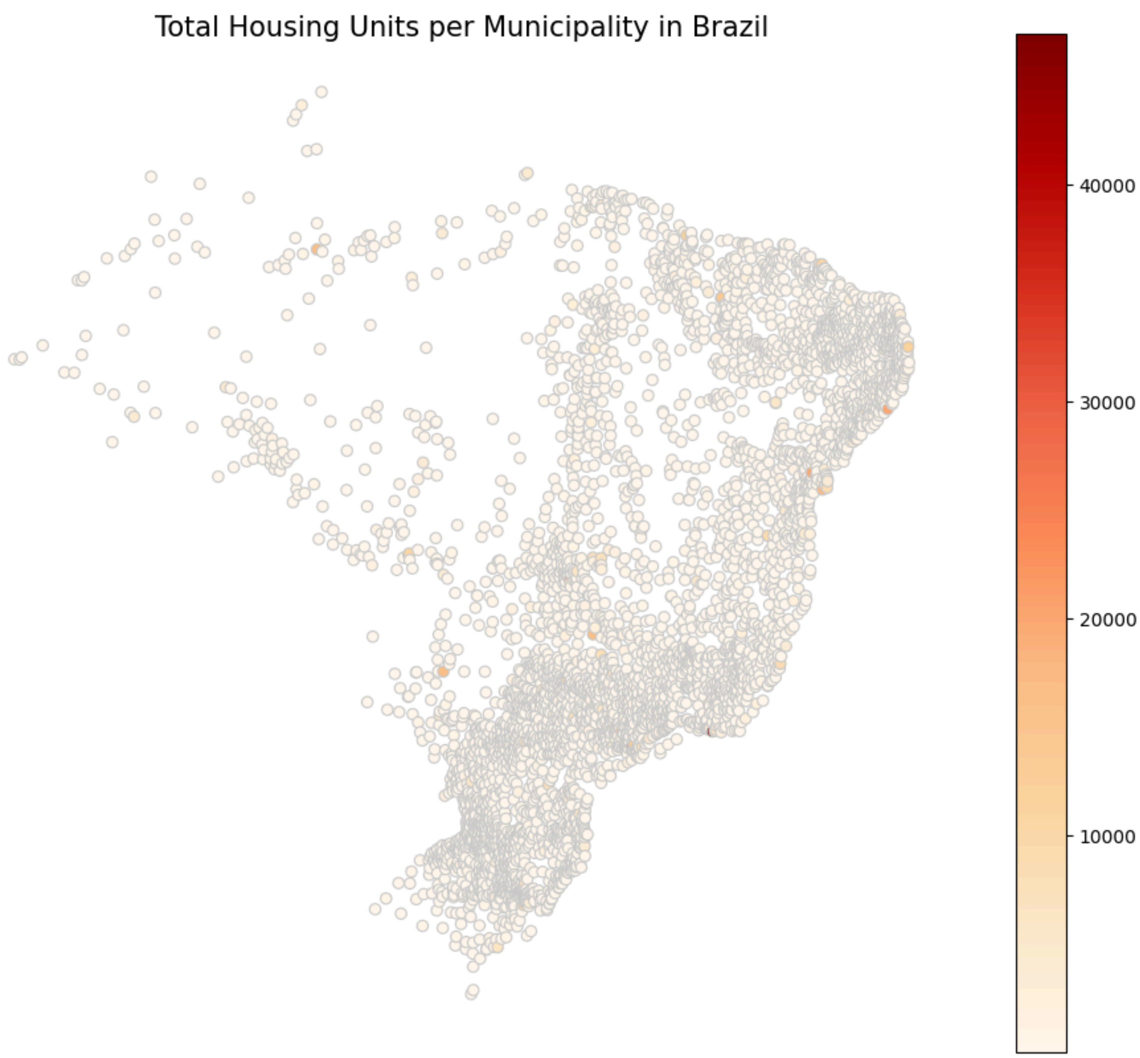
In [91]:
```python
# Merge total_housing_units back into joined_gdf based on 'NOME'
joined_gdf = pd.merge(joined_gdf, total_housing_units, on='NOME', how='left')

# Print the updated joined_gdf to verify
print(joined_gdf.head())
```

```
    XCOORD    YCOORD UF_left  UH_x  Project_ID                    geometry  \
0 -72.8997 -7.61658      AC   1.0           1  POINT (-72.89971 -7.61658)
1 -72.6756 -7.62763      AC  18.0           2  POINT (-72.67558 -7.62762)
2 -72.5907 -7.53797      AM  31.0           3  POINT (-72.59071 -7.53797)
3 -71.6934 -7.04791      AM  30.0           4  POINT (-71.69343 -7.04791)
4 -70.7722 -8.15697      AC  16.0           5  POINT (-70.77215 -8.15697)

   index_right  COD_MUN             NOME UF_right  POP_201  IDHM_10  PIB_PER
\
0          392  1200336      MANCIO LIMA       AC  15890.0    0.625   8111.0
1          374  1200203  CRUZEIRO DO SUL       AC  79819.0    0.664  10643.0
2          395  1301654          GUAJARA       AM  14396.0    0.532   4388.0
3          398  1301803          IPIXUNA       AM  23460.0    0.481   4362.0
4          397  1200609         TARAUACA       AC  36763.0    0.539   8192.0

    UH_y     UH
0    1.0    1.0
1  156.0  156.0
2   31.0   31.0
3   30.0   30.0
4   16.0   16.0
```

The total_housing units column in the joined data is `UH_x` . now, since our data contains both municipalities column and total housing units column, lets visualize the same.

In [92]:

```
# Plotting
fig, ax = plt.subplots(figsize=(12, 10))
joined_gdf.plot(column='UH_x', cmap='OrRd', linewidth=0.8, edgecolor='0.8', le
ax.set_title('Total Housing Units per Municipality in Brazil', fontsize=15)
ax.set_axis_off()

# Show plot
plt.show()
```



SUMMARY In this lab, we have used python to perform various spatial operation tasks. From this lab, we managed to;

- Create a map displaying the central points of each municipality exclusively within the state of São Paulo.
- Calculate and visualize the mean Human Development Index municipalities in each state of Brazil
- Create and plot a shapefile for Gaucha do Norte e Xingu
- Creating and plotting an intersection
- Creating and visualizing data points in GIS, Calculating the distance between data points and creating buffers

- Determine that the municipality with the highest number of housing units in Brazil is RIO DE JANEIRO with 46,966 units and the * municipality with the lowest number of houing units i Brazil is ABDON BAISTA with only 1 unit

GITHUB LINK [https://github.com/b-irungu/SPATIAL-OPERATIONS-IN-GIS.git (https://github.com/b-irungu/SPATIAL-OPERATIONS-IN-GIS.git)](https://github.com/b-irungu/SPATIAL-OPERATIONS-IN-GIS.git)

In [ ]:

```
```