GeoPandas

In [1]:
```
import pandas as pd
import numpy as np
import geopandas as gpd
import rtree
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading and Writing Files ¶

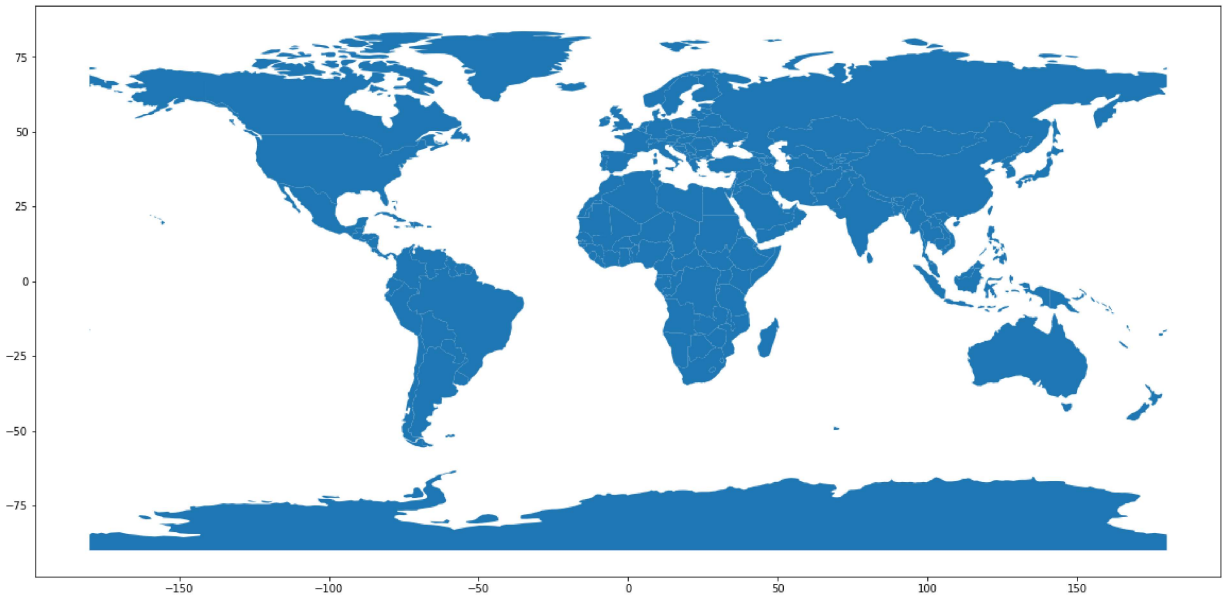
In [2]:
```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))  #read the fi
world.head()
```

Out[2]:

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
0	28400000.0	Asia	Afghanistan	AFG	22270.0	POLYGON ((61.21081709172574 35.65007233330923,...
1	12799293.0	Africa	Angola	AGO	110300.0	(POLYGON ((16.32652835456705 -5.87747039146621...
2	3639453.0	Europe	Albania	ALB	21810.0	POLYGON ((20.59024743010491 41.85540416113361,...
3	4798491.0	Asia	United Arab Emirates	ARE	184300.0	POLYGON ((51.57951867046327 24.24549713795111,...
4	40913584.0	South America	Argentina	ARG	573900.0	(POLYGON ((-65.50000000000003 -55.199999999999...

In [3]:
```
world.plot(figsize=(20,20))
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a5aca358>
```



In [4]:
```
world.geometry.name  #get active geometry column name
```

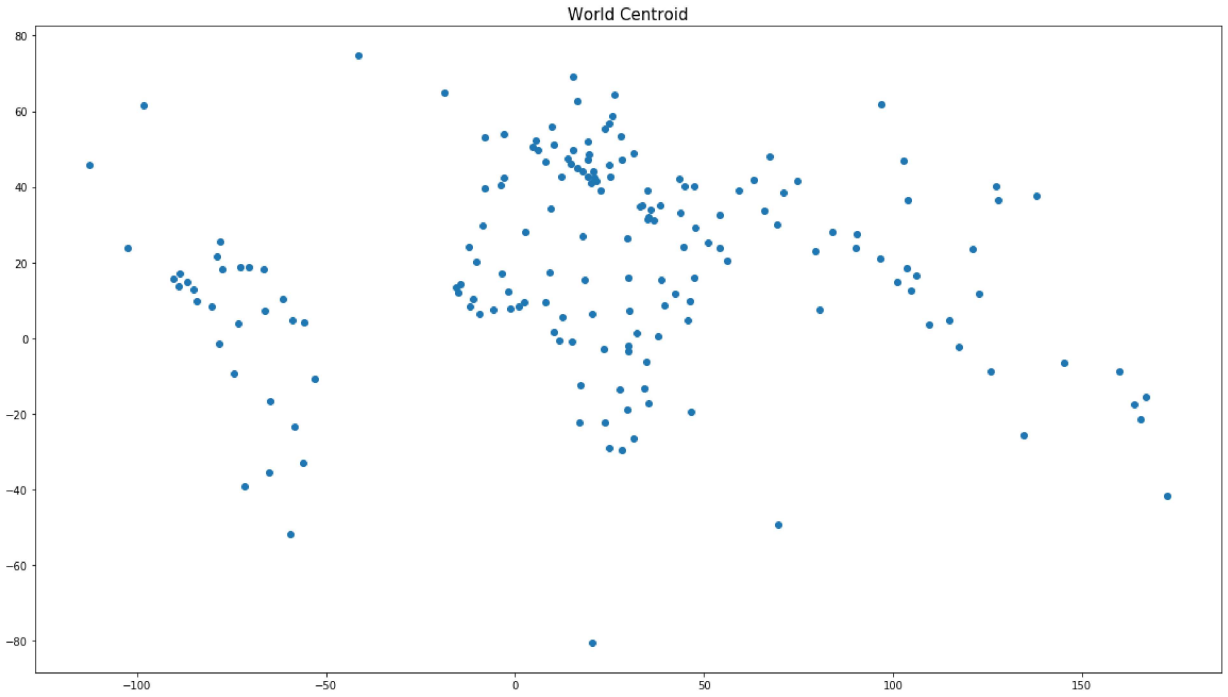
Out[4]:
```
'geometry'
```

In [5]:
```
world['centroid_column'] = world.centroid
world = world.set_geometry('centroid_column')  # set active geometry column by yo
world.geometry.name
```

Out[5]:
```
'centroid_column'
```

In [6]:
```
world.plot(figsize=(20,20))
plt.title('World Centroid', fontsize=15)
```

Out[6]:
```
Text(0.5,1,'World Centroid')
```



In [7]:
```
world.rename(columns={'centroid_column':'centroid'}).set_geometry('centroid', inp
```

In [8]:
```
world.geometry.name
```

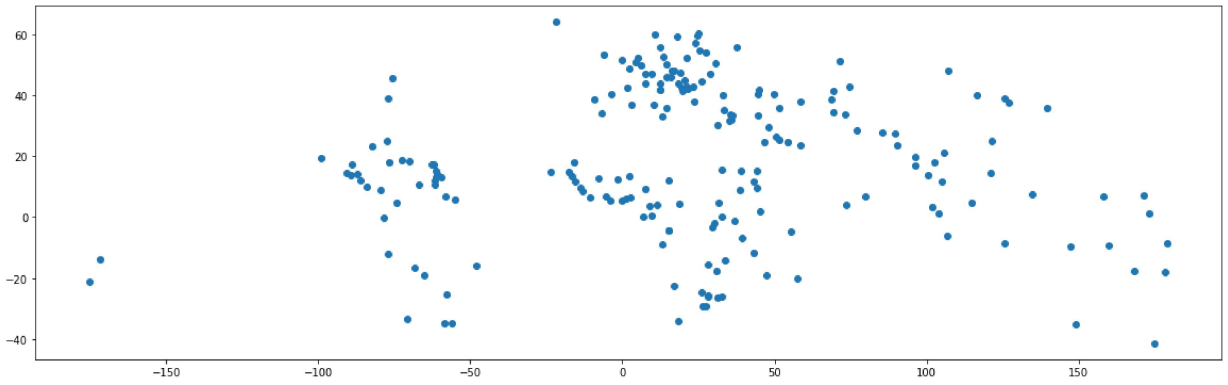
Out[8]:
```
'centroid_column'
```

In [9]:
```
import fiona
from fiona import drivers
help(fiona.open)
```

```
Help on function open in module fiona:

open(path, mode='r', driver=None, schema=None, crs=None, encoding=None, layer
=None, vfs=None, enabled_drivers=None, crs_wkt=None)
    Open file at ``path`` in ``mode`` "r" (read), "a" (append), or
    "w" (write) and return a ``Collection`` object.

    In write mode, a driver name such as "ESRI Shapefile" or "GPX" (see
    OGR docs or ``ogr2ogr --help`` on the command line) and a schema
    mapping such as:

      {'geometry': 'Point',
       'properties': [('class', 'int'), ('label', 'str'),
                      ('value', 'float')]}

    must be provided. If a particular ordering of properties ("fields"
    in GIS parlance) in the written file is desired, a list of (key,
    value) pairs as above or an ordered dict is required. If no ordering
    is needed, a standard dict will suffice.
```

In [10]:
```
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
cities.plot(figsize=(20,20))
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x204a5f33358>



In [11]:
```
#world.to_file("World_SHP.shp")
#world.to_file("World_JSON.geojson", driver='GeoJSON')
#world.to_file("World_Package.gpkg", layer='countries', driver="GPKG")
```

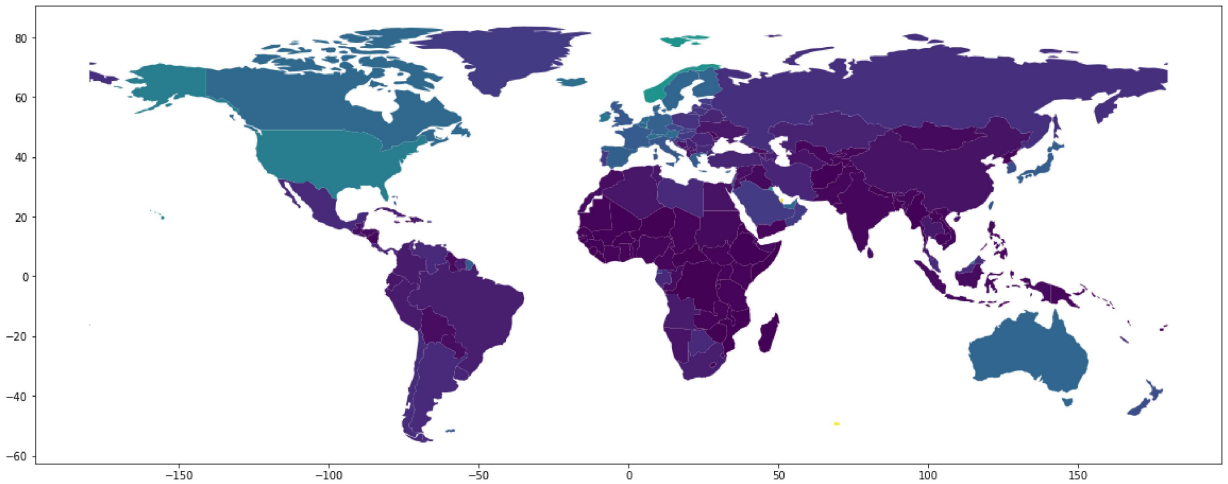
Chloropleth Map

In [12]:
```
world = world[(world.pop_est>0) & (world.name!="Antarctica")]
```

In [13]:
```
world['gdp_per_cap'] = world.gdp_md_est / world.pop_est
```

In [14]:
```
world = world.set_geometry('geometry')
world.plot(column='gdp_per_cap', figsize=(20,20))
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x204a649efd0>

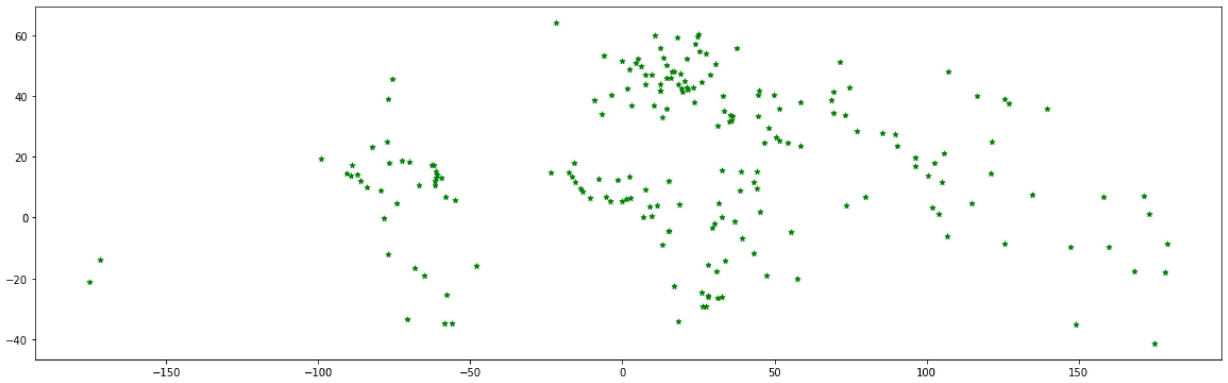


In [15]:
```
world.crs
```

Out[15]:
```
{'init': 'epsg:4326'}
```

In [16]:
```
cities.plot(marker='*', color='green', markersize=25, figsize=(20,20))
```

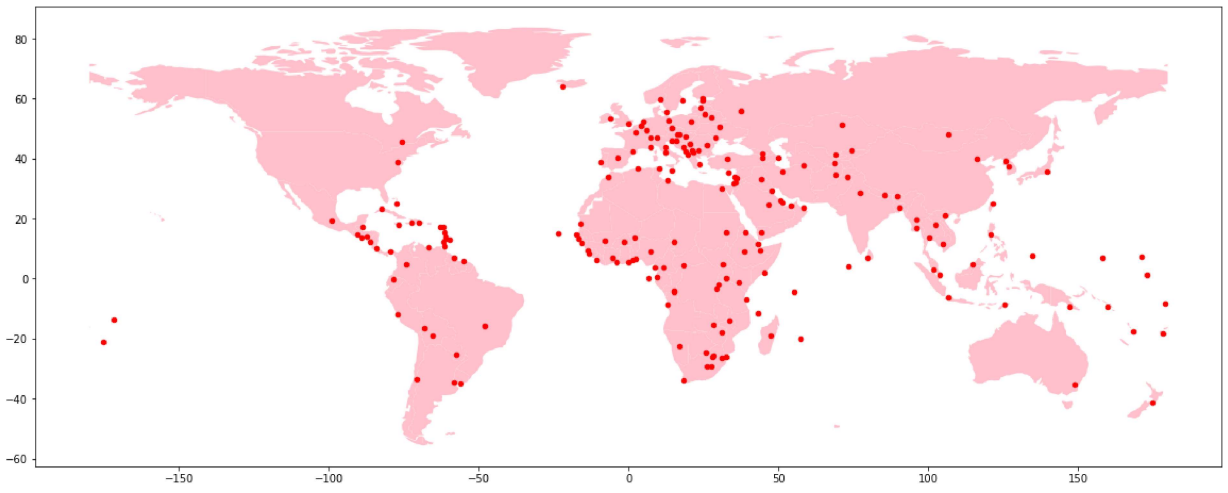
Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a6500198>
```



One graph on another graph

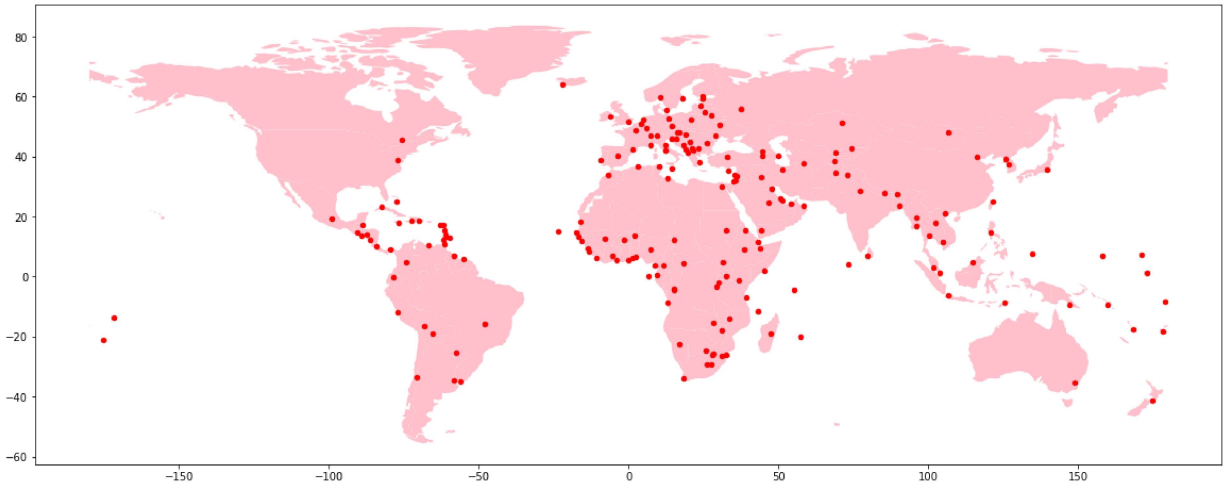
In [17]:
```
base = world.plot(color='pink', figsize=(20,20))
cities.plot(ax=base, marker='o', markersize=20, color='red')
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a6299518>
```

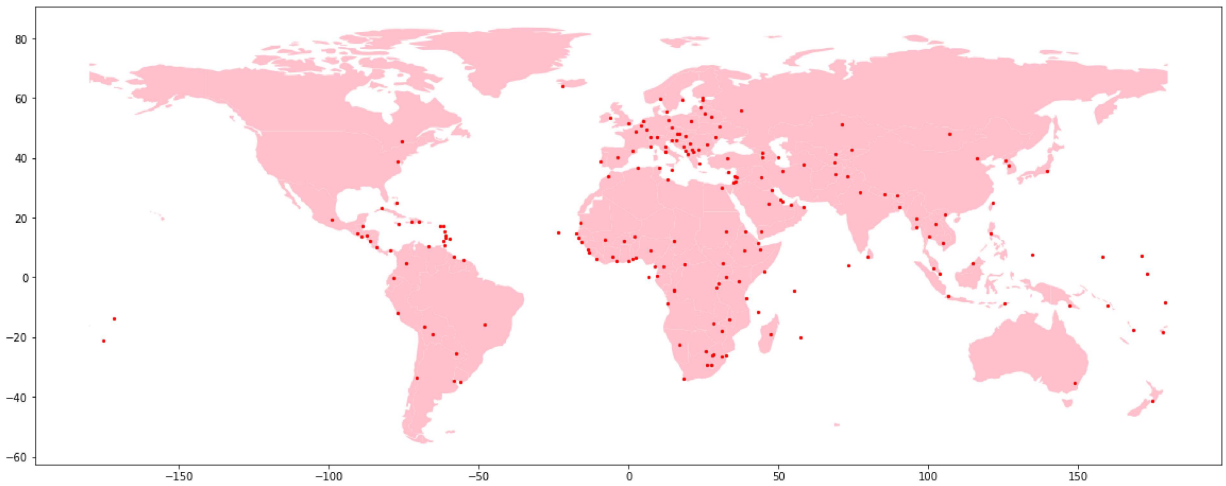


In [18]:
```
cities = cities.to_crs(world.crs)
base = world.plot(color='pink', figsize=(20,20))
cities.plot(ax=base, marker='o', markersize=20, color='red')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x204a6252240>

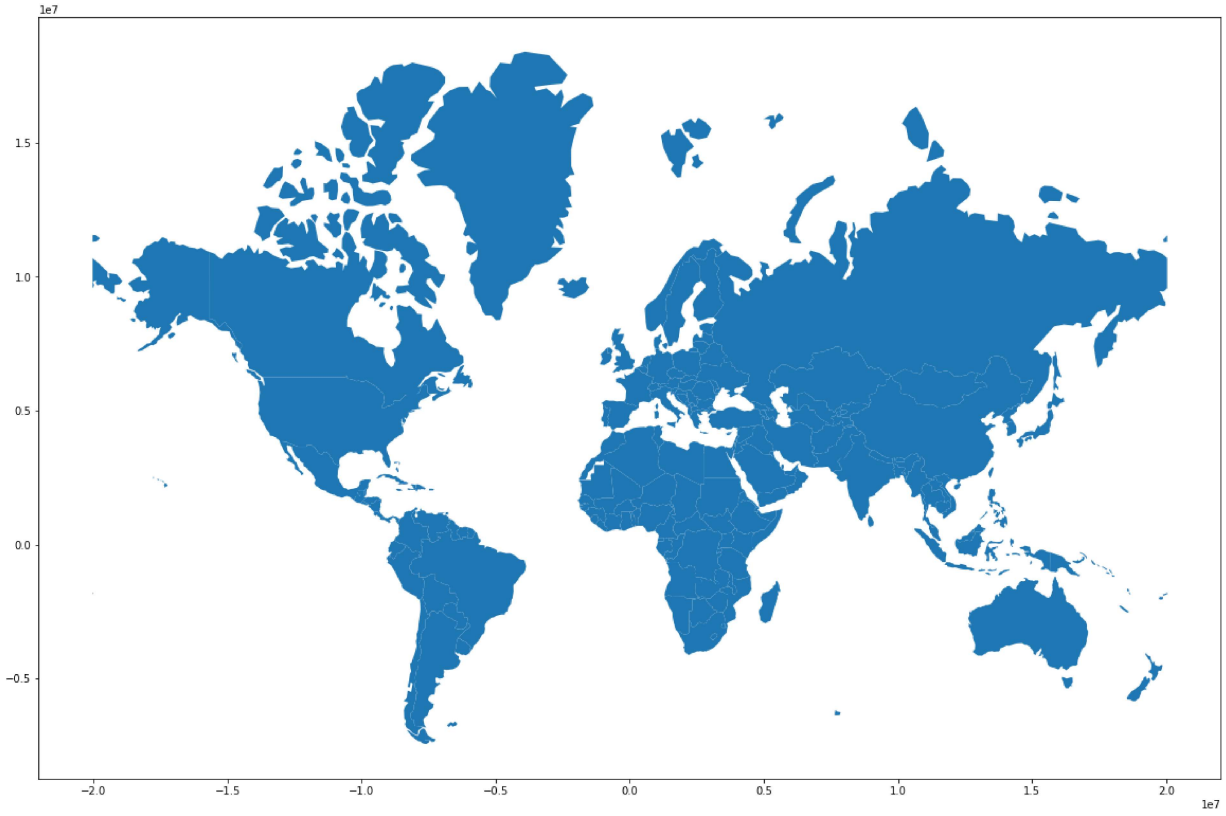


In [19]:
```
fig, ax = plt.subplots(figsize=(20,20))
ax.set_aspect('equal')
world.plot(ax=ax, color='pink')
cities.plot(ax=ax, marker='o', color='red', markersize=5)
plt.show()
```



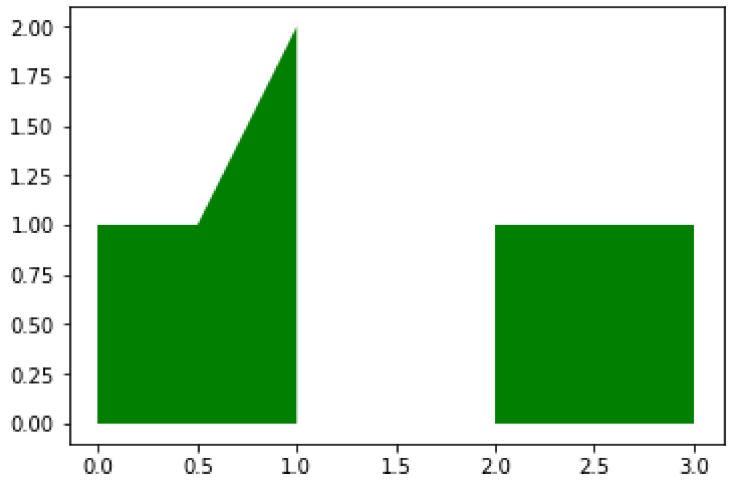
In [20]:
```
world = world.to_crs({'init':'epsg:3395'})
world.plot(figsize=(20,20))
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x204a6bfb908>



In [21]:
```
from shapely.geometry import Polygon
p1 = Polygon([(0, 0), (1, 0), (1, 2)])
p2 = Polygon([(0, 0), (1, 0), (1, 1), (0, 1)])
p3 = Polygon([(2, 0), (3, 0), (3, 1), (2, 1)])
g = gpd.GeoSeries([p1, p2, p3])
g.plot(color='green')
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a6c75a20>
```

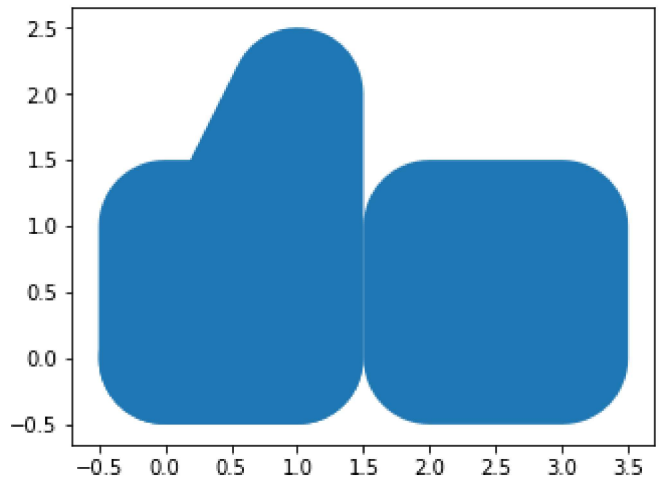


In [22]:
```
g.area
```

Out[22]:
```
0    1.0
1    1.0
2    1.0
dtype: float64
```

In [23]:
```
g.buffer(0.5).plot()
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a6ed1f28>
```



In [24]:
```
import os
boros = gpd.GeoDataFrame.from_file('C:\\Users\\Deepika\\Desktop\\Digital Vidya\\D
```

In [25]:
```
boros.head()
```

Out[25]:

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometr
0	1	0.116357	0.000782	Newark Airport	1	EWR	POLYGO ((933100.918352710 192536.0856972019,.
1	2	0.433470	0.004866	Jamaica Bay	2	Queens	(POLYGO ((1033269.24359129 172126.0078125, 1.
2	3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POLYGO ((1026308.76950666 256767.6975403726,.
3	4	0.043567	0.000112	Alphabet City	4	Manhattan	POLYGO ((992073.466796860 203714.0759887695,.
4	5	0.092146	0.000498	Arden Heights	5	Staten Island	POLYGO ((935843.310493260 144283.335850656, .

In [26]:
```
boros.set_index('OBJECTID', inplace=True)
boros.sort_values(by='OBJECTID')
boros
```

Out[26]:

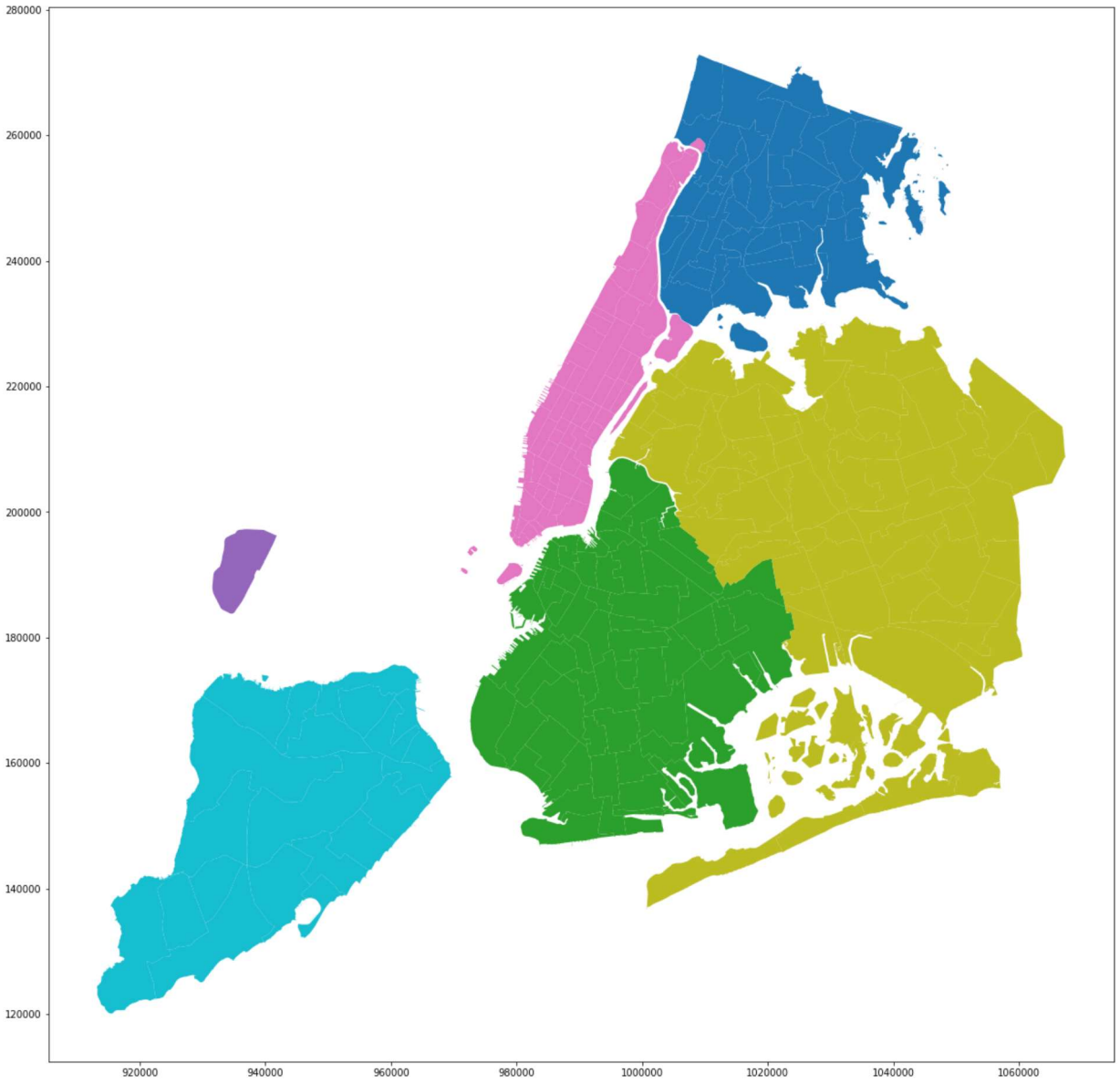
OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	ge
1	0.116357	0.000782	Newark Airport	1	EWR	POL ((933100.9183 192536.085697
2	0.433470	0.004866	Jamaica Bay	2	Queens	(POL ((1033269.243 172126.00781
3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POL ((1026308.769 256767.697540
4	0.043567	0.000112	Alphabet City	4	Manhattan	POL ((992073.4667 203714.075988
5	0.092146	0.000498	Arden Heights	5	Staten	POL ((935843.3104

In [27]:
```
boros.geometry.name
```

Out[27]:
```
'geometry'
```

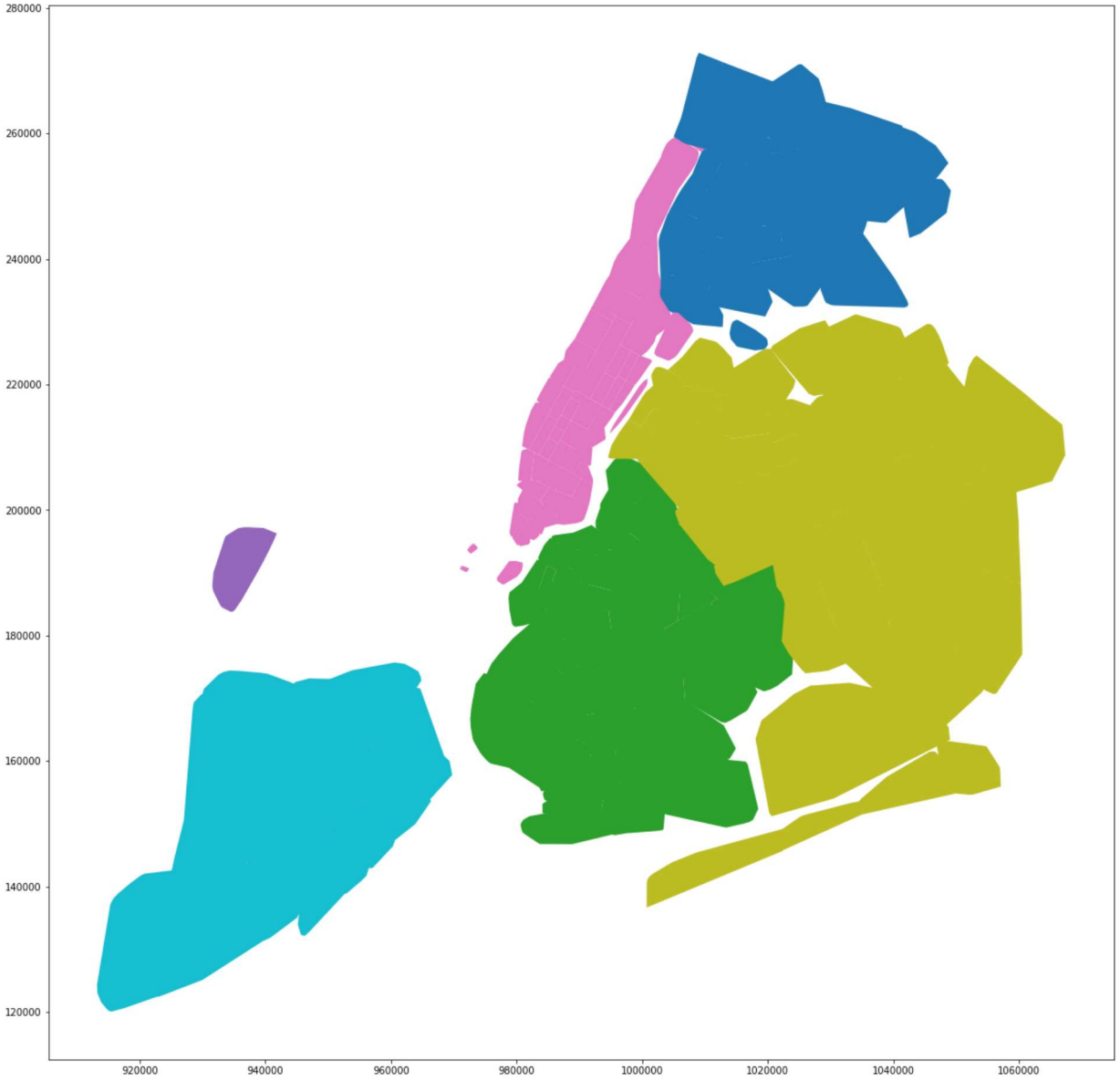
In [28]:
```
boros.plot(column='borough', figsize=(20,20))
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x204a6fa7748>



In [29]:
```
boros['geometry'] = boros['geometry'].convex_hull
boros.plot(column='borough', figsize=(20,20))
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x204a6fe72b0>

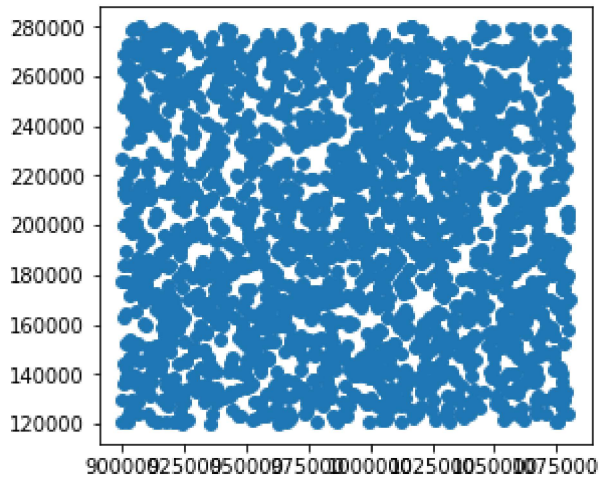


In [30]:
```
from shapely.geometry import Point
xmin, xmax, ymin, ymax = 900000, 1080000, 120000, 280000
xc = (xmax - xmin) * np.random.random(2000) + xmin
yc = (ymax - ymin) * np.random.random(2000) + ymin
pts = gpd.GeoSeries([Point(x, y) for x, y in zip(xc, yc)])
pts
```

Out[30]:
```
0       POINT (1065399.526110841 186200.1540983864)
1       POINT (935378.6862898492 140666.7468276862)
2       POINT (905306.4340212269 240314.5144745465)
3       POINT (935156.3444238239 134248.7381758784)
4       POINT (950302.4786055146 143563.4513212103)
5       POINT (937930.9965538853 187870.4518776465)
6        POINT (976648.814188187 148690.8235938286)
7       POINT (1065759.713884719 265191.3975954338)
8       POINT (1063265.727375751 210867.0314022173)
9       POINT (1004944.052965592 240619.2429294929)
10      POINT (1025807.223192075 193762.5489607491)
11      POINT (1046708.168291946 167832.9236287808)
12      POINT (909358.5341185079 177024.5054820528)
13      POINT (998814.7799965673 265108.6961206695)
14       POINT (967977.221958117 196826.7038034738)
15      POINT (1031671.987471194 273856.9275936169)
16      POINT (1070520.963876346 123968.2468721319)
17      POINT (993056.6900511218 132360.5836605627)
18      POINT (992099.7925911132 203066.7115534214)
```

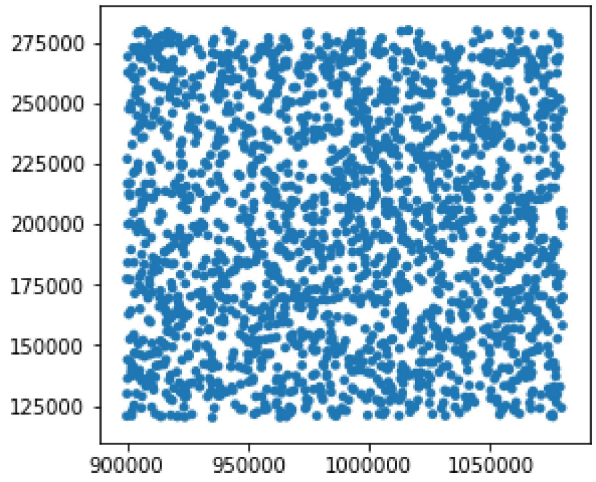
In [31]:
```
pts.plot()
```

Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a700cba8>
```



In [32]:
```
circles = pts.buffer(2000)
circles.plot()
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a8d600b8>
```



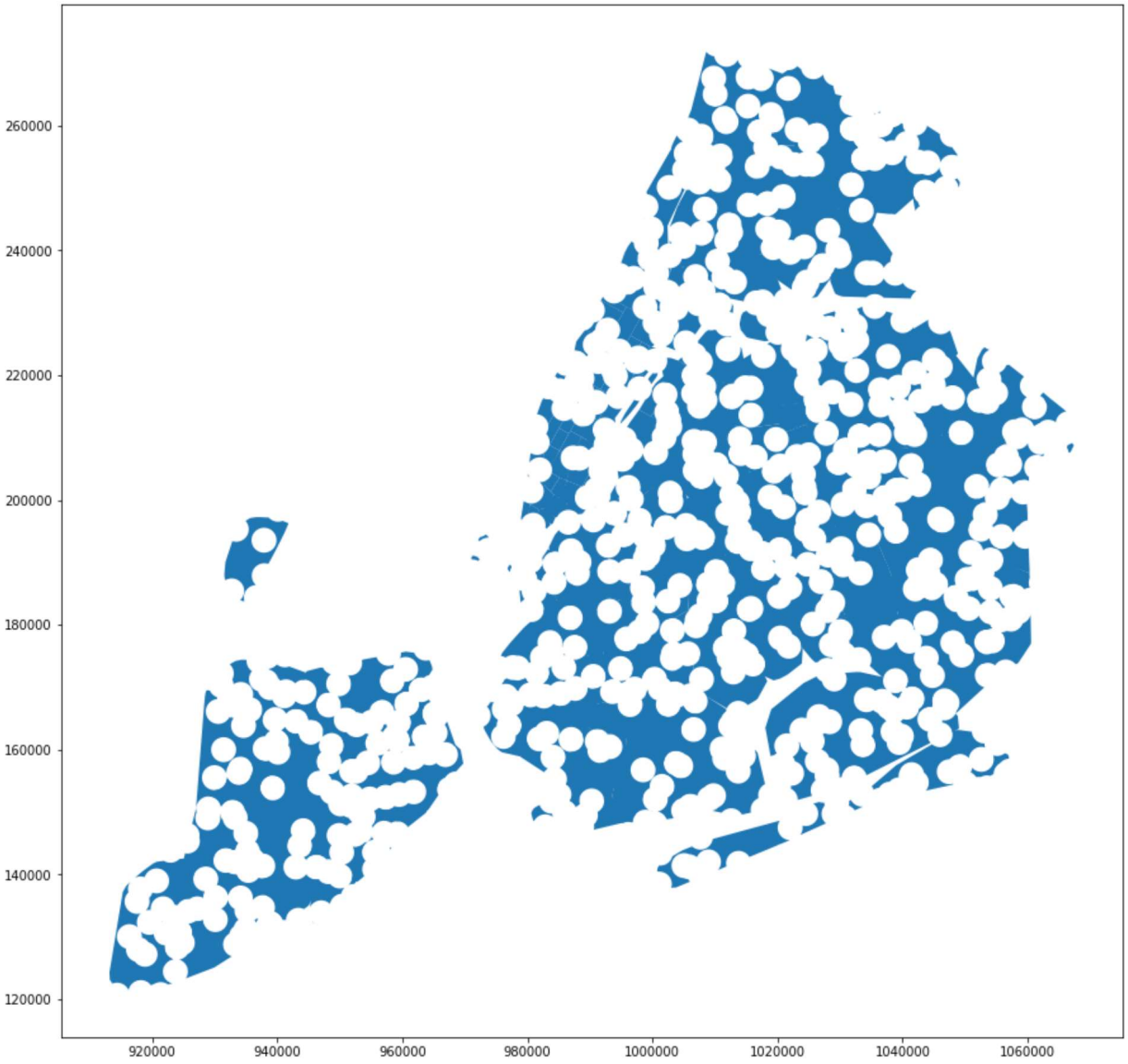
In [33]:
```
mp = circles.unary_union
```

In [34]:
```
holes = boros['geometry'].intersection(mp)
holes.plot(figsize=(15,15))
```

Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a8dc9390>
```

In [35]:
```
boros_with_holes = boros['geometry'].difference(mp)
boros_with_holes.plot(figsize=(15,15))
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x204a8e52a58>



Set Operations with Overlay

In [36]:
```
from geopandas.tools import overlay
from shapely.geometry import Polygon
```

In [37]:
```
polys1 = gpd.GeoSeries([Polygon([(0,0), (2,0), (2,2), (0,2)]),Polygon([(2,2), (4,
polys2 = gpd.GeoSeries([Polygon([(1,1), (3,1), (3,3), (1,3)]), Polygon([(3,3), (5

df1 = gpd.GeoDataFrame({'geometry': polys1, 'df1':[1,2]})
df2 = gpd.GeoDataFrame({'geometry': polys2, 'df2':[1,2]})
```

In [38]:
```
df1,df2
```

Out[38]:
```
(                              geometry  df1
 0  POLYGON ((0 0, 2 0, 2 2, 0 2, 0 0))    1
 1  POLYGON ((2 2, 4 2, 4 4, 2 4, 2 2))    2,
                              geometry  df2
 0  POLYGON ((1 1, 3 1, 3 3, 1 3, 1 1))    1
 1  POLYGON ((3 3, 5 3, 5 5, 3 5, 3 3))    2)
```

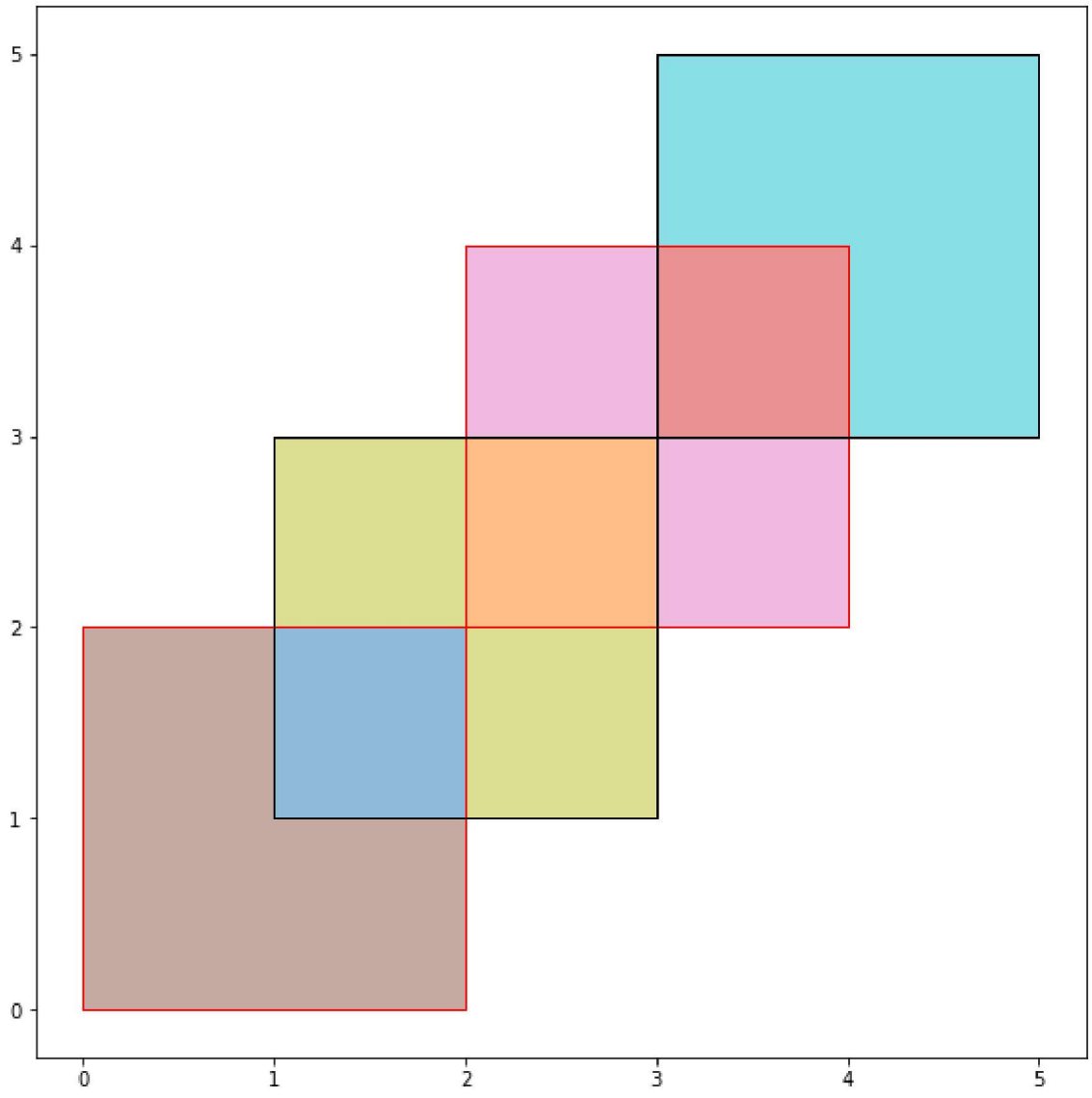
In [39]:
```
res_union = gpd.overlay(df1, df2, how='union')
res_union
```

Out[39]:

	df1	df2	geometry
0	1.0	1.0	POLYGON ((1 2, 2 2, 2 1, 1 1, 1 2))
1	2.0	1.0	POLYGON ((2 2, 2 3, 3 3, 3 2, 2 2))
2	2.0	2.0	POLYGON ((3 4, 4 4, 4 3, 3 3, 3 4))
3	1.0	NaN	POLYGON ((0 0, 0 2, 1 2, 1 1, 2 1, 2 0, 0 0))
4	2.0	NaN	(POLYGON ((2 3, 2 4, 3 4, 3 3, 2 3)), POLYGON ...
5	NaN	1.0	(POLYGON ((1 2, 1 3, 2 3, 2 2, 1 2)), POLYGON ...
6	NaN	2.0	POLYGON ((3 4, 3 5, 5 5, 5 3, 4 3, 4 4, 3 4))

In [40]:
```
ax = res_union.plot(alpha=0.5, cmap='tab10', figsize=(10,10))
df1.plot(ax=ax, facecolor='none', edgecolor='r')
df2.plot(ax=ax, facecolor='none', edgecolor='k')
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x204a98a04a8>



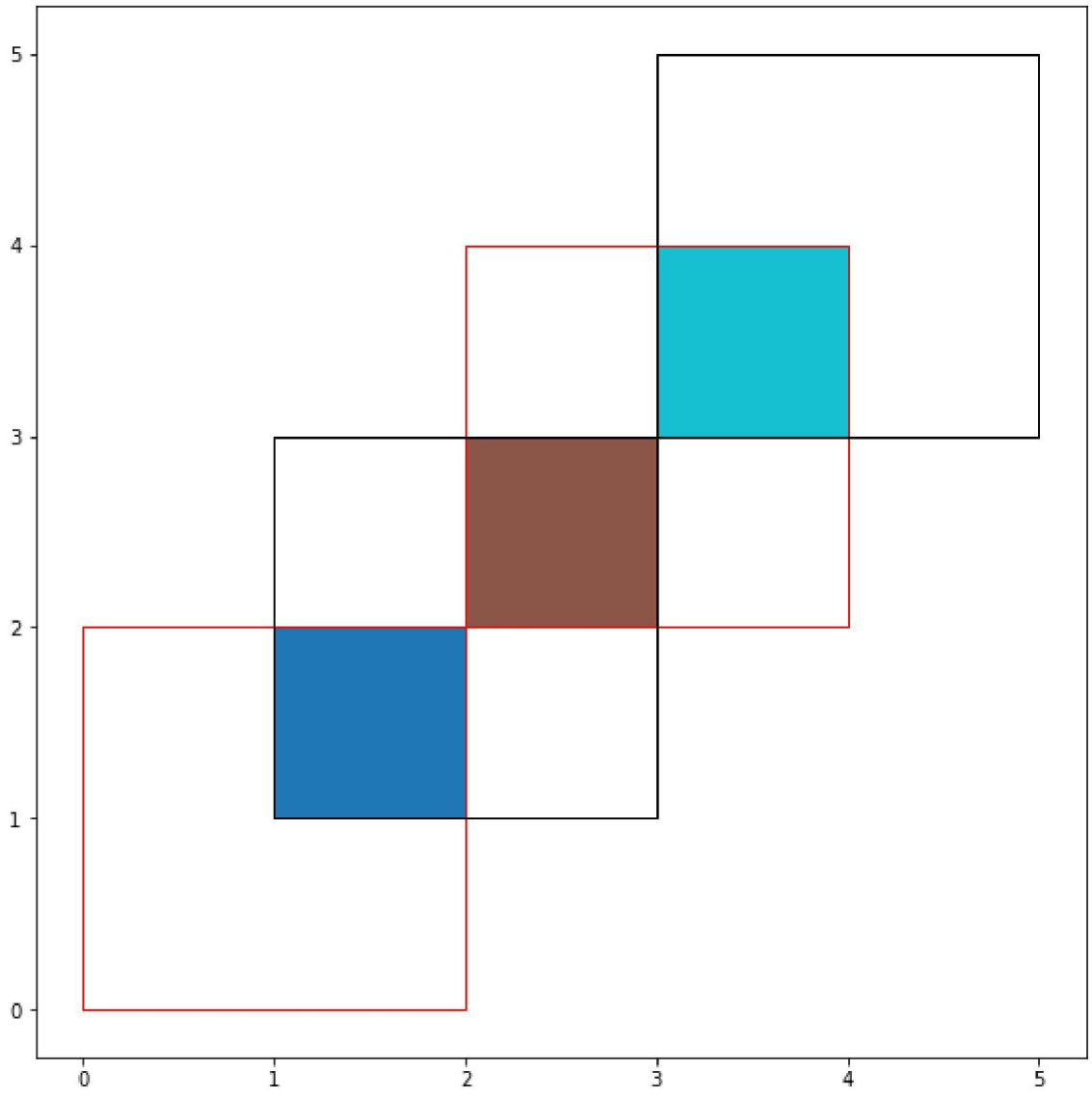
In [41]:
```
res_intersection = gpd.overlay(df1, df2, how='intersection')
res_intersection
```

Out[41]:

	df1	df2	geometry
0	1	1	POLYGON ((1 2, 2 2, 2 1, 1 1, 1 2))
1	2	1	POLYGON ((2 2, 2 3, 3 3, 3 2, 2 2))
2	2	2	POLYGON ((3 4, 4 4, 4 3, 3 3, 3 4))

In [42]:
```
ax = res_intersection.plot(cmap='tab10', figsize=(10,10))
df1.plot(ax=ax, facecolor='none', edgecolor='r')
df2.plot(ax=ax, facecolor='none', edgecolor='k')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x204a98b2278>



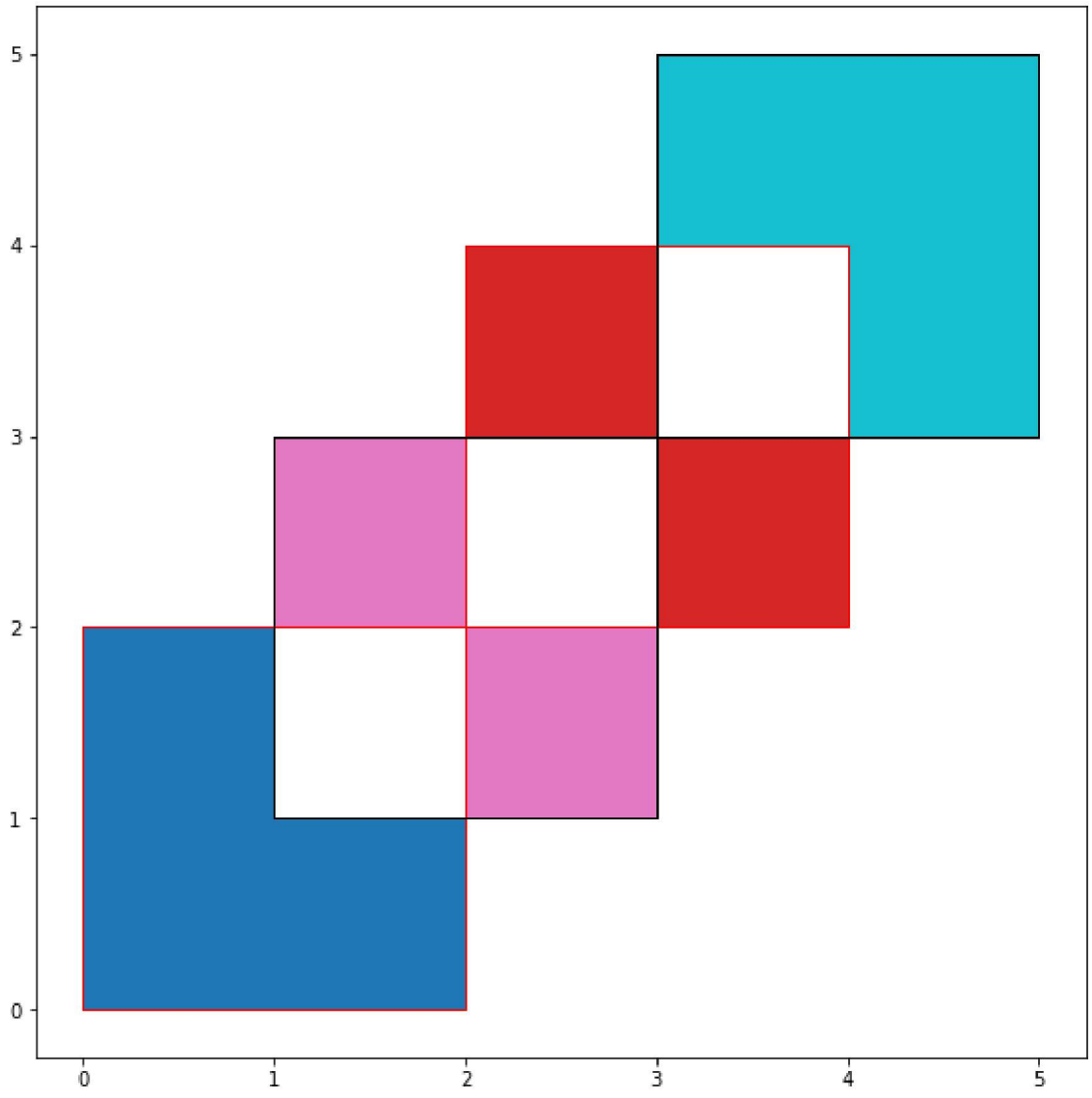
In [43]:
```
res_symdiff = gpd.overlay(df1, df2, how='symmetric_difference')
res_symdiff
```

Out[43]:

	df1	df2	geometry
0	1.0	NaN	POLYGON ((0 0, 0 2, 1 2, 1 1, 2 1, 2 0, 0 0))
1	2.0	NaN	(POLYGON ((2 3, 2 4, 3 4, 3 3, 2 3)), POLYGON ...
2	NaN	1.0	(POLYGON ((1 2, 1 3, 2 3, 2 2, 1 2)), POLYGON ...
3	NaN	2.0	POLYGON ((3 4, 3 5, 5 5, 5 3, 4 3, 4 4, 3 4))

In [44]:
```
ax = res_symdiff.plot(cmap='tab10', figsize=(10,10))
df1.plot(ax=ax, facecolor='none', edgecolor='r')
df2.plot(ax=ax, facecolor='none', edgecolor='k')
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x204a97fbcf8>



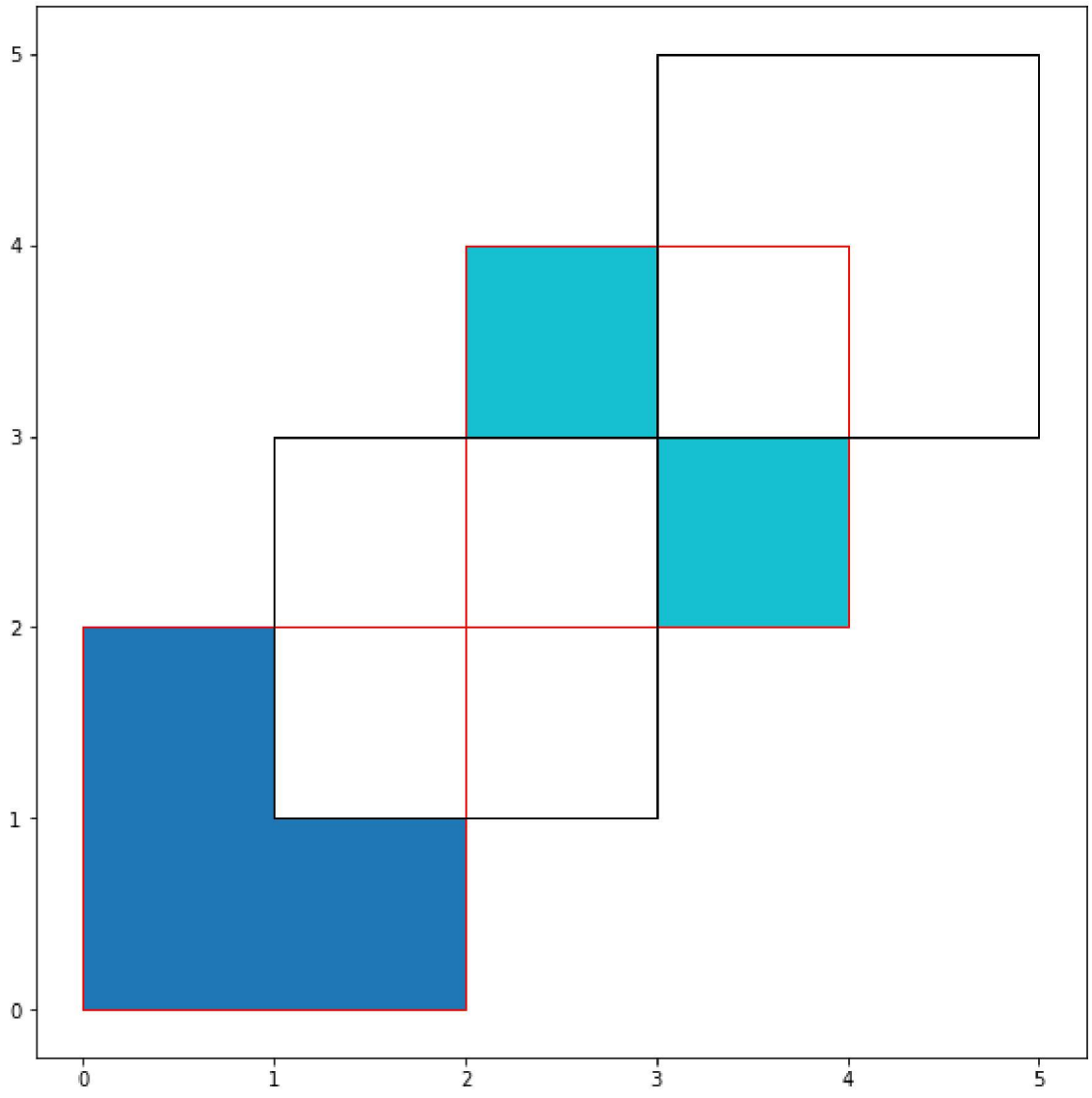
In [45]:
```
res_difference = gpd.overlay(df1, df2, how='difference')
res_difference
```

Out[45]:

	geometry	df1
0	POLYGON ((0 0, 0 2, 1 2, 1 1, 2 1, 2 0, 0 0))	1
1	(POLYGON ((2 3, 2 4, 3 4, 3 3, 2 3)), POLYGON ...	2

In [46]:
```
ax = res_difference.plot(cmap='tab10', figsize=(10,10))
df1.plot(ax=ax, facecolor='none', edgecolor='r')
df2.plot(ax=ax, facecolor='none', edgecolor='k')
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x204a98bfc50>



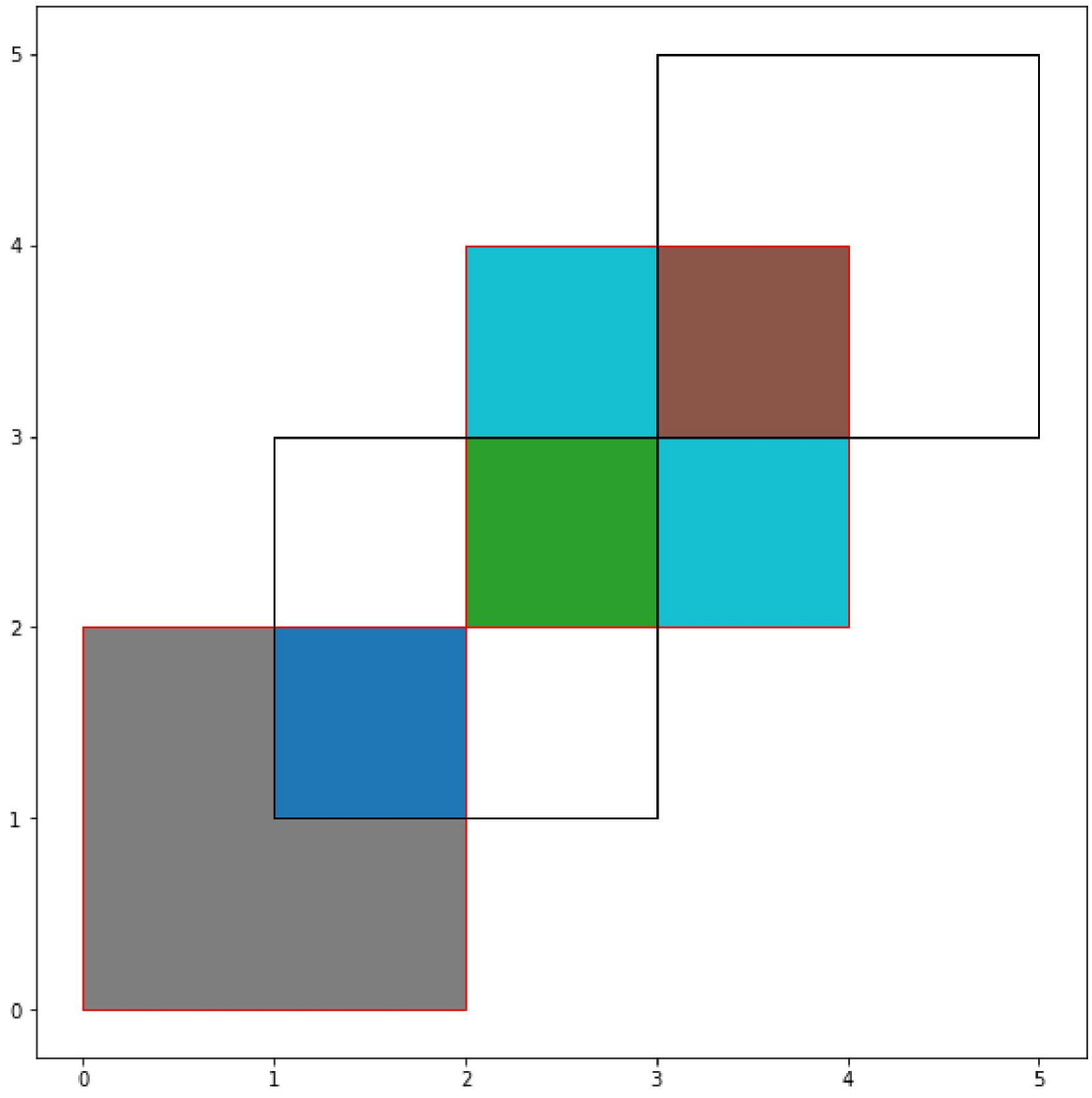
In [47]:
```
res_identity = gpd.overlay(df1, df2, how='identity')
res_identity
```

Out[47]:

	df1	df2	geometry
0	1.0	1.0	POLYGON ((1 2, 2 2, 2 1, 1 1, 1 2))
1	2.0	1.0	POLYGON ((2 2, 2 3, 3 3, 3 2, 2 2))
2	2.0	2.0	POLYGON ((3 4, 4 4, 4 3, 3 3, 3 4))
3	1.0	NaN	POLYGON ((0 0, 0 2, 1 2, 1 1, 2 1, 2 0, 0 0))
4	2.0	NaN	(POLYGON ((2 3, 2 4, 3 4, 3 3, 2 3)), POLYGON ...

In [48]:
```
ax = res_identity.plot(cmap='tab10', figsize=(10,10))
df1.plot(ax=ax, facecolor='none', edgecolor='r')
df2.plot(ax=ax, facecolor='none', edgecolor='k')
```

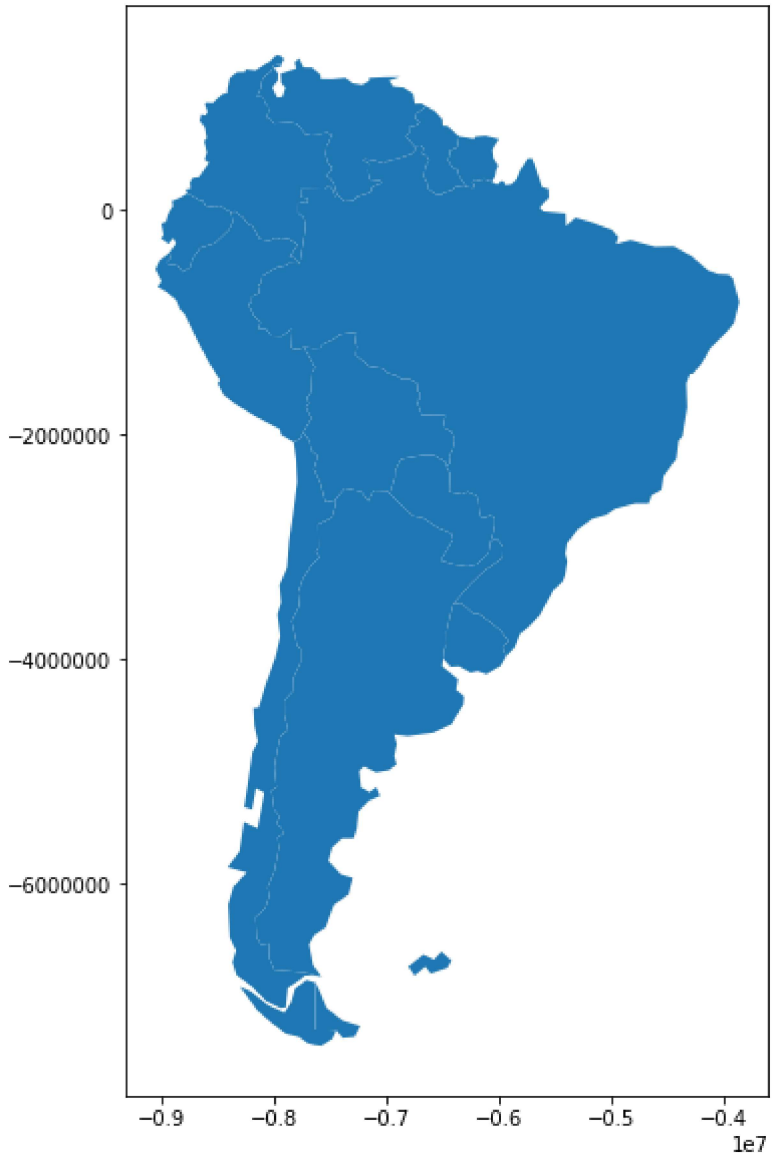
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x204aa911438>



In [49]:
```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
capitals = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))

# Select South Amarica and some columns
countries = world[world['continent'] == "South America"]
countries = countries[['geometry', 'name']]

# Project to crs that uses meters as distance measure
countries = countries.to_crs({'init':'epsg:3395'})
capitals = capitals.to_crs({'init':'epsg:3395'})
```

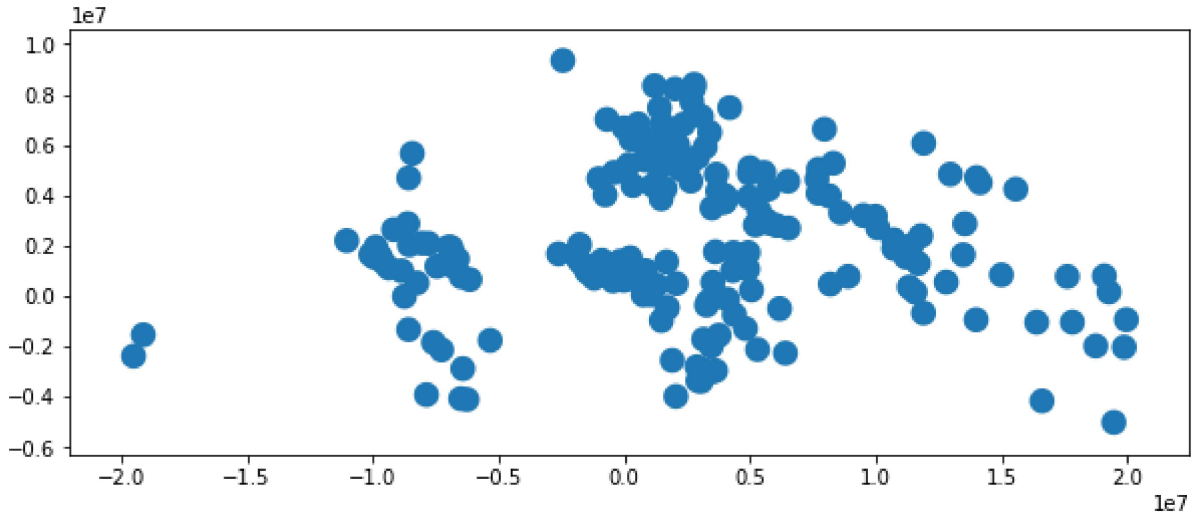
In [50]:
```
countries.plot(figsize=(10,10))
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x204aa981cc0>



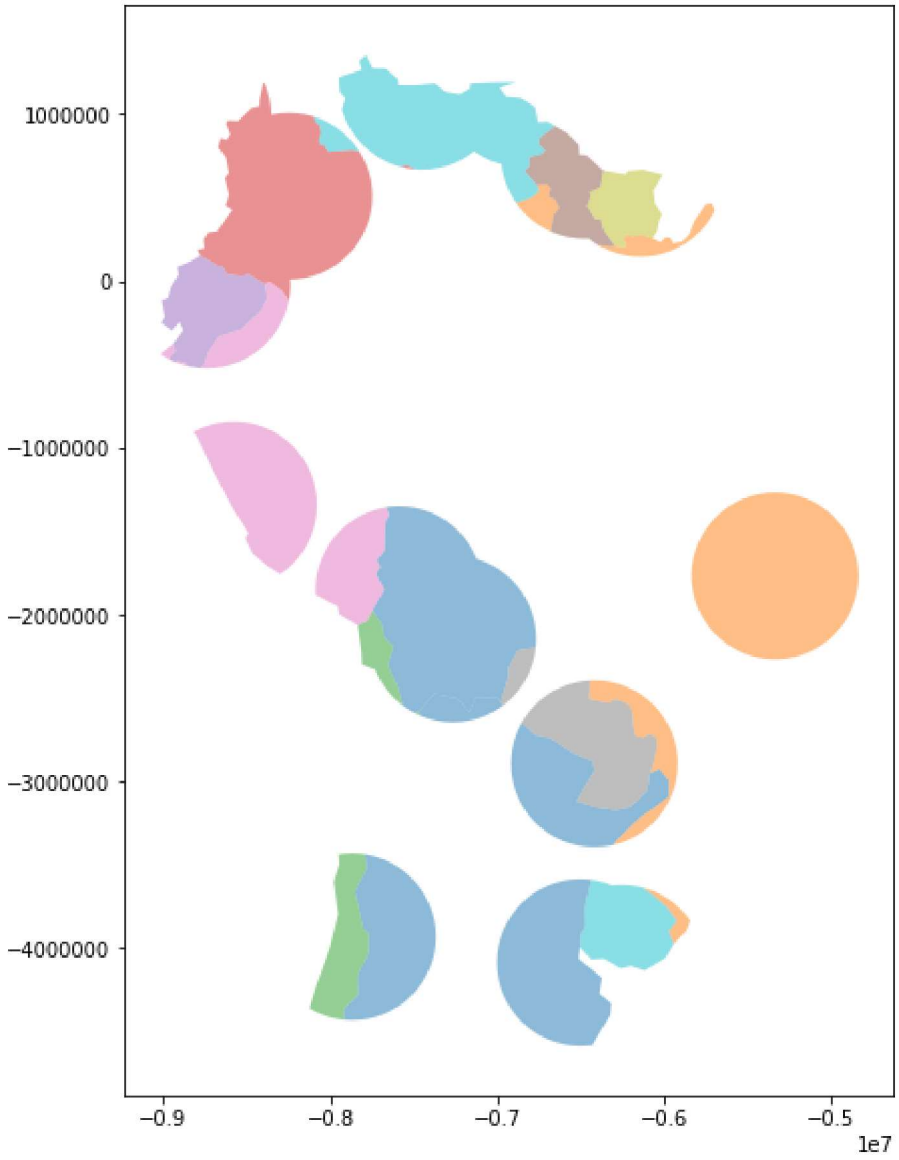
In [51]:
```
capitals['geometry']= capitals.buffer(500000)
capitals.plot(figsize=(10,10))
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x204aa97bfd0>



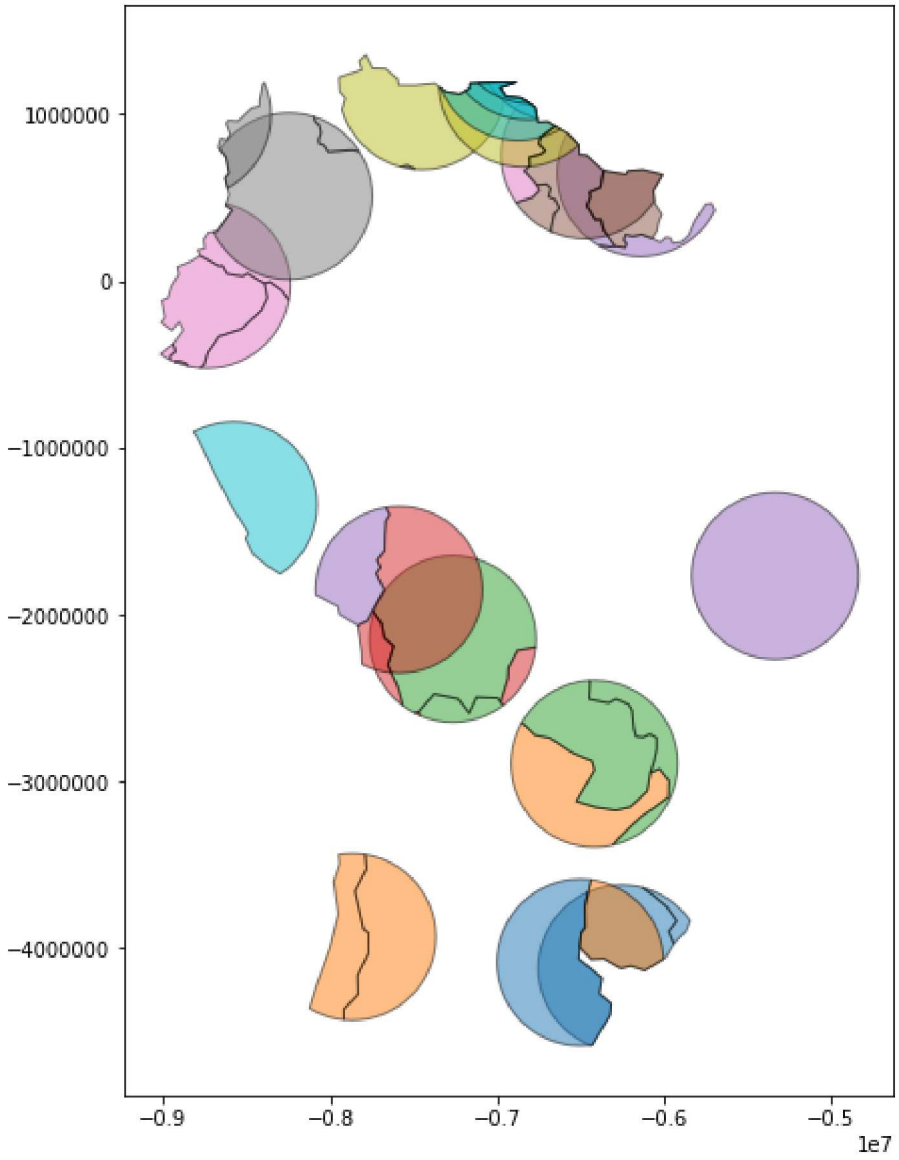
In [52]:
```
intersect_exm = countries['geometry'].intersection(capitals.unary_union)
intersect_exm.plot(alpha=0.5, cmap='tab10', figsize=(10,10))
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x204abf36ba8>



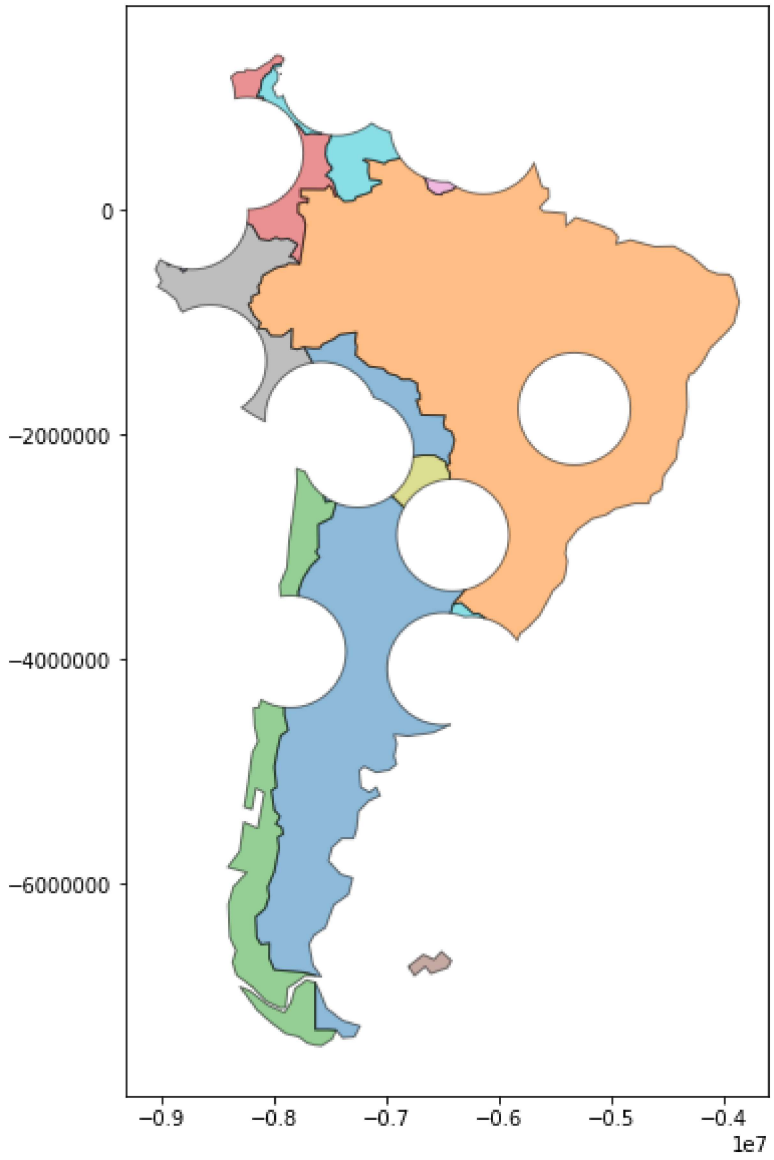
In [53]:
```
country_cores = gpd.overlay(countries, capitals, how='intersection')
country_cores.plot(alpha=0.5, edgecolor='k', cmap='tab10', figsize=(10,10))
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x204abfb61d0>



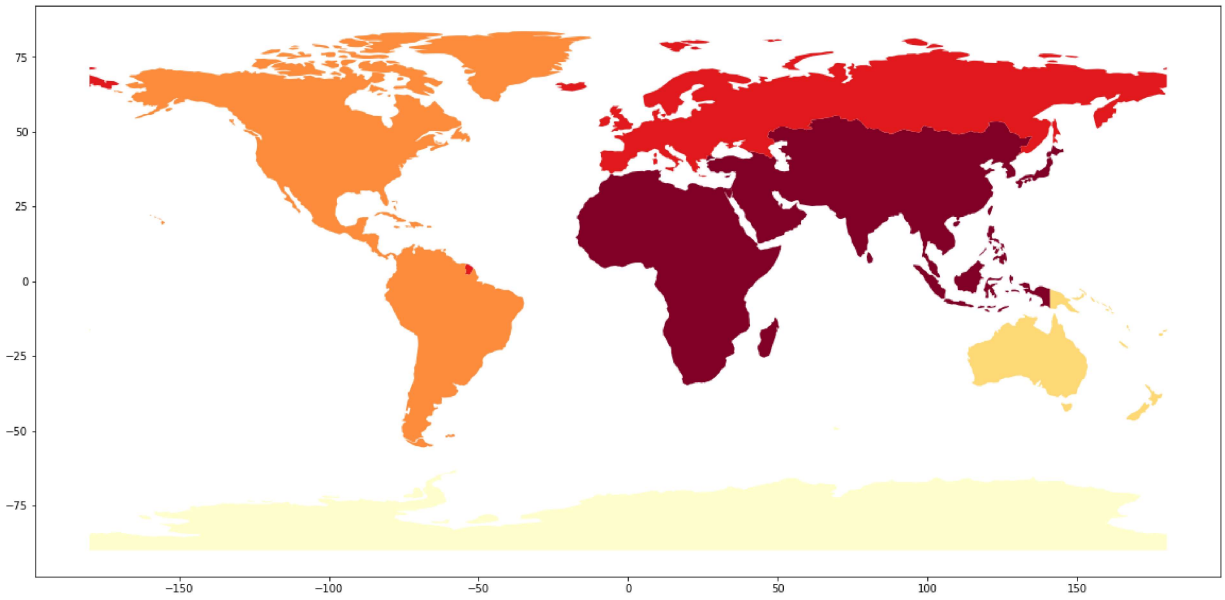
In [54]:
```
country_peripheries = gpd.overlay(countries, capitals, how='difference')
country_peripheries.plot(alpha=0.5, edgecolor='k', cmap='tab10', figsize=(10,10))
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x204abfadcc0>



In [55]:
```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
world = world[['continent', 'geometry', 'pop_est']]
continents = world.dissolve(by='continent', aggfunc='sum')
continents.plot(column = 'pop_est', scheme='quantiles', cmap='YlOrRd', figsize=(2
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204aae62080>
```



Merging Data

1. Attribute Join
2. Spatial Join

In [56]:
```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))

# For attribute join
country_shapes = world[['geometry', 'iso_a3']]
country_names = world[['name', 'iso_a3']]

# For spatial join
countries = world[['geometry', 'name']]
countries = countries.rename(columns={'name':'country'})
```

In [57]:
```
country_shapes.head()
```

Out[57]:

	geometry	iso_a3
0	POLYGON ((61.21081709172574 35.65007233330923,...	AFG
1	(POLYGON ((16.32652835456705 -5.87747039146621...	AGO
2	POLYGON ((20.59024743010491 41.85540416113361,...	ALB
3	POLYGON ((51.57951867046327 24.24549713795111,...	ARE
4	(POLYGON ((-65.50000000000003 -55.199999999999...	ARG

In [58]:
```
country_names.head()
```

Out[58]:

	name	iso_a3
0	Afghanistan	AFG
1	Angola	AGO
2	Albania	ALB
3	United Arab Emirates	ARE
4	Argentina	ARG

In [59]:
```
country_shapes = country_shapes.merge(country_names, on='iso_a3')
country_shapes
```

Out[59]:

	geometry	iso_a3	name
0	POLYGON ((61.21081709172574 35.65007233330923,...	AFG	Afghanistan
1	(POLYGON ((16.32652835456705 -5.87747039146621...	AGO	Angola
2	POLYGON ((20.59024743010491 41.85540416113361,...	ALB	Albania
3	POLYGON ((51.57951867046327 24.24549713795111,...	ARE	United Arab Emirates
4	(POLYGON ((-65.50000000000003 -55.199999999999...	ARG	Argentina
5	POLYGON ((43.58274580259273 41.09214325618257,...	ARM	Armenia
6	(POLYGON ((-59.57209469261153 -80.040178725096...	ATA	Antarctica
7	POLYGON ((68.935 -48.62500000000001, 69.58 -48...	ATF	Fr. S. Antarctic Lands
8	(POLYGON ((145.3979781434948 -40.7925485166058...	AUS	Australia
9	POLYGON ((16.97966678230404 48.12349701597631,...	AUT	Austria
10	(POLYGON ((45.0019873390568 39.7400035670496, ...	AZE	Azerbaijan

In [60]:
```
cities.head()
```

Out[60]:

	name	geometry
0	Vatican City	POINT (12.45338654497177 41.90328217996012)
1	San Marino	POINT (12.44177015780014 43.936095834768)
2	Vaduz	POINT (9.516669472907267 47.13372377429357)
3	Luxembourg	POINT (6.130002806227083 49.61166037912108)
4	Palikir	POINT (158.1499743237623 6.916643696007725)

In [61]:
```
countries.head()
```

Out[61]:

	geometry	country
0	POLYGON ((61.21081709172574 35.65007233330923,...	Afghanistan
1	(POLYGON ((16.32652835456705 -5.87747039146621...	Angola
2	POLYGON ((20.59024743010491 41.85540416113361,...	Albania
3	POLYGON ((51.57951867046327 24.24549713795111,...	United Arab Emirates
4	(POLYGON ((-65.50000000000003 -55.199999999999...	Argentina

In [62]:
```
cities_with_country = gpd.sjoin(cities, countries, how="inner", op='intersects')
cities_with_country.head()
```

Out[62]:

	name	geometry	index_right	country
0	Vatican City	POINT (12.45338654497177 41.90328217996012)	79	Italy
1	San Marino	POINT (12.44177015780014 43.936095834768)	79	Italy
192	Rome	POINT (12.481312562874 41.89790148509894)	79	Italy
2	Vaduz	POINT (9.516669472907267 47.13372377429357)	9	Austria
184	Vienna	POINT (16.36469309674374 48.20196113681686)	9	Austria

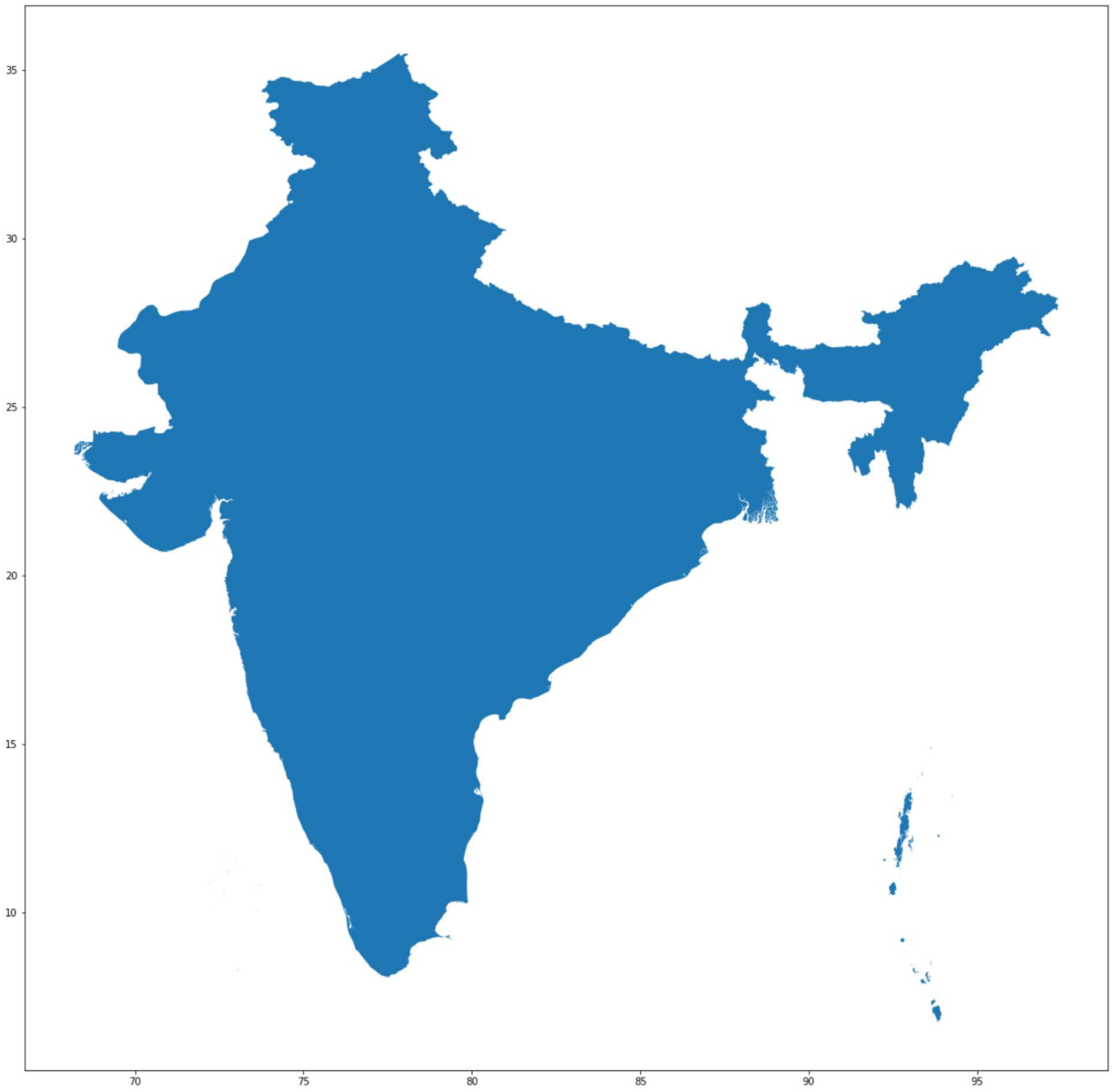
In [63]:
```
india_shape = gpd.read_file("C:\\Users\\Deepika\\Desktop\\Digital Vidya\\Data Sci
india_shape
```

Out[63]:

	GID_0	NAME_0	geometry
0	IND	India	(POLYGON ((93.78772736000001 6.85264015, 93.78...

In [64]:
```
india_shape.plot(figsize=(20,20))
```

Out[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204a6257198>
```



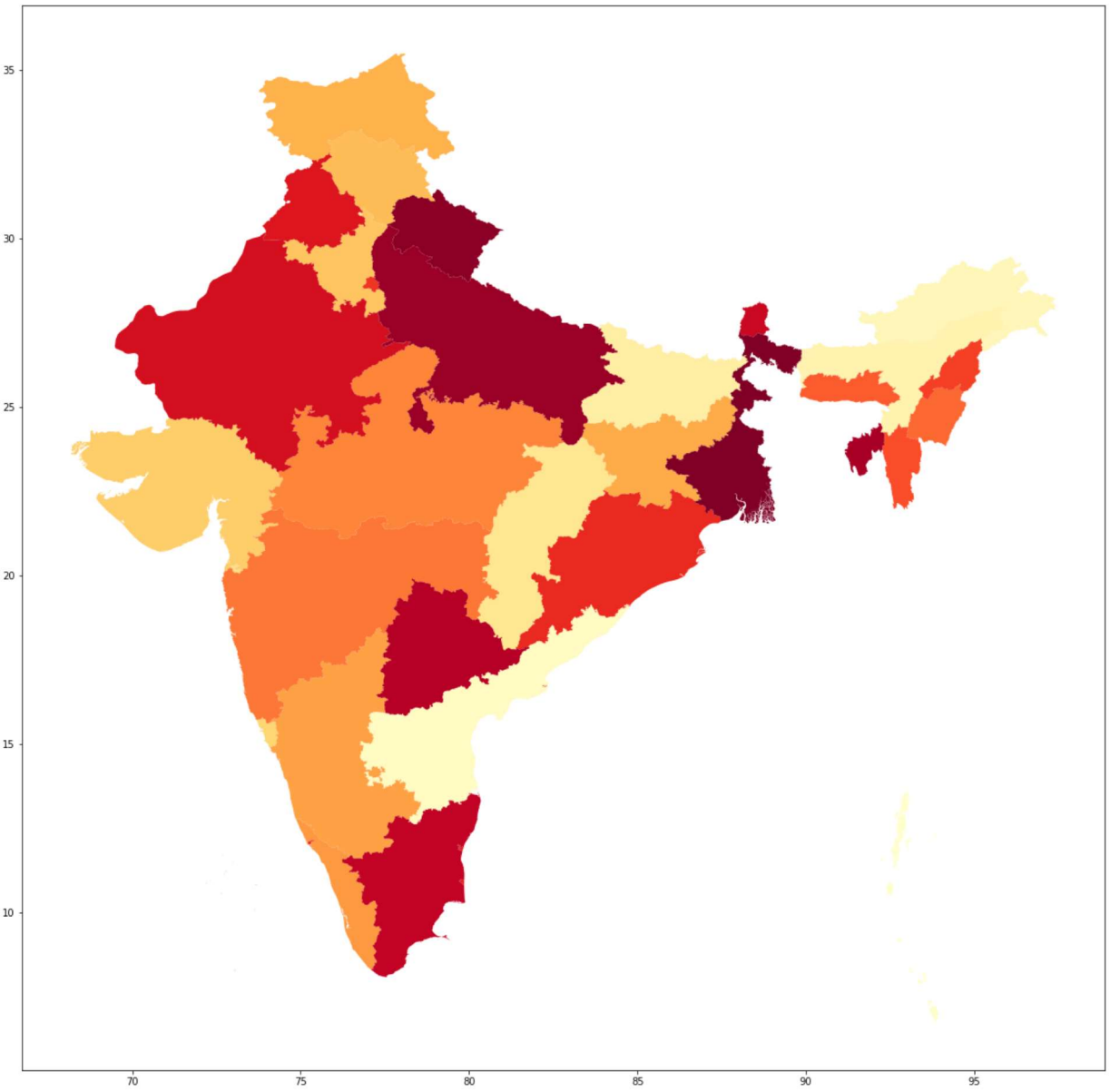
In [65]:
```
india_shape1 = gpd.read_file("C:\\Users\\Deepika\\Desktop\\Digital Vidya\\Data Sc
india_shape1
```

Out[65]:

	GID_0	NAME_0	GID_1	NAME_1	VARNAME_1	NL_NAME_1	TYPI
0	IND	India	IND.1_1	Andaman and Nicobar	Andaman & Nicobar Islands|Andaman et Nicobar|I...	None	Ur Ter
1	IND	India	IND.2_1	Andhra Pradesh	None	None	S
2	IND	India	IND.3_1	Arunachal Pradesh	Agence de la Frontière du Nord-Est(French-obso...	None	S
3	IND	India	IND.4_1	Assam	None	None	S

In [66]:
```
india_shape1.plot(cmap='YlOrRd', figsize=(20,20))
```

Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204b3d65630>
```



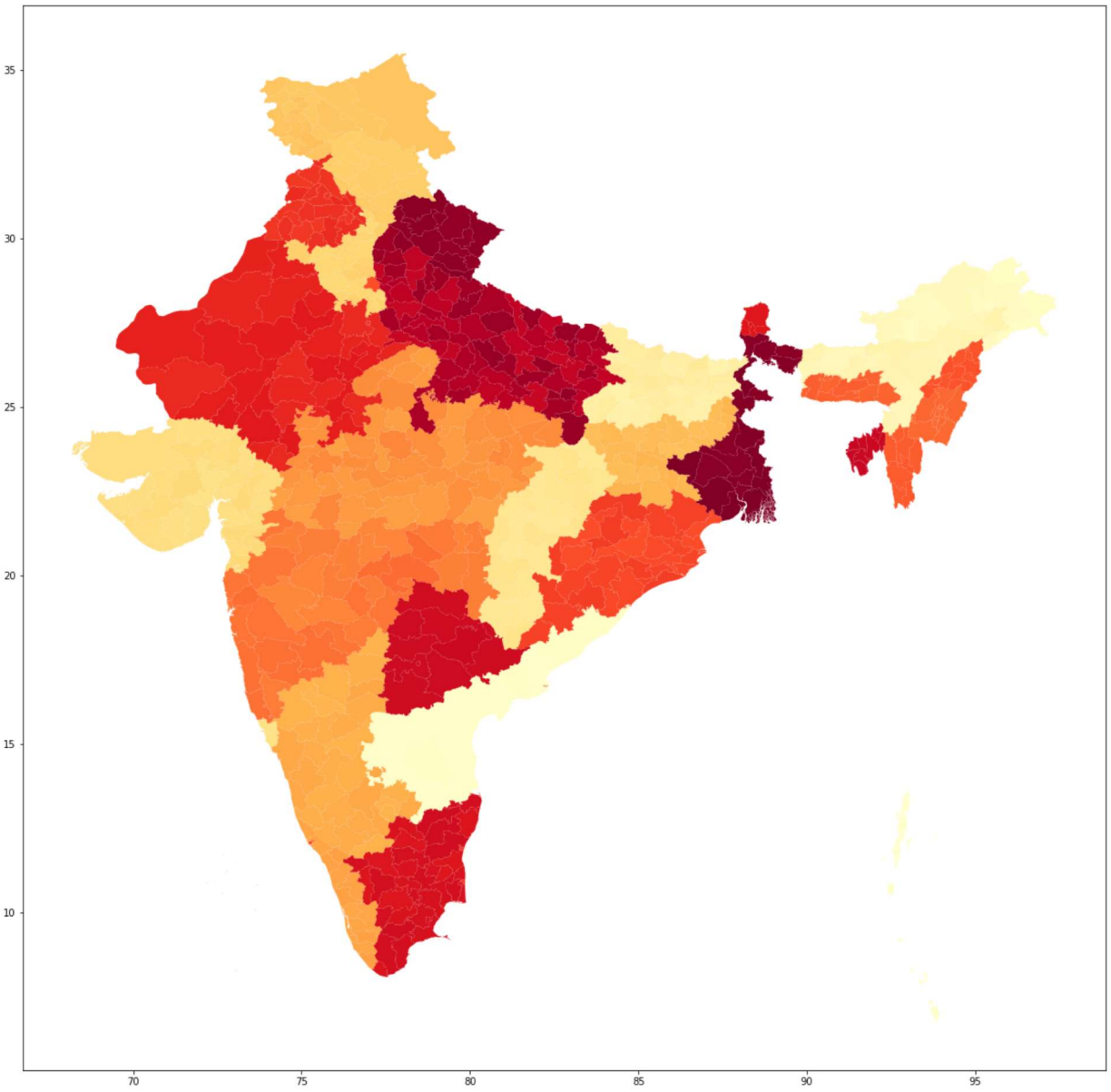
In [67]:
```
india_shape2 = gpd.read_file("C:\\Users\\Deepika\\Desktop\\Digital Vidya\\Data Sc
india_shape2
```

Out[67]:

	GID_0	NAME_0	GID_1	NAME_1	NL_NAME_1	GID_2	NAME_2
0	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.1_1	Nicobar Islands
1	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.2_1	North and Middle Andaman
2	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.3_1	South Andaman
3	IND	India	IND.2_1	Andhra Pradesh	None	IND.2.1_1	Anantapur
4	IND	India	IND.2_1	Andhra	None	IND.2.2_1	Chitt

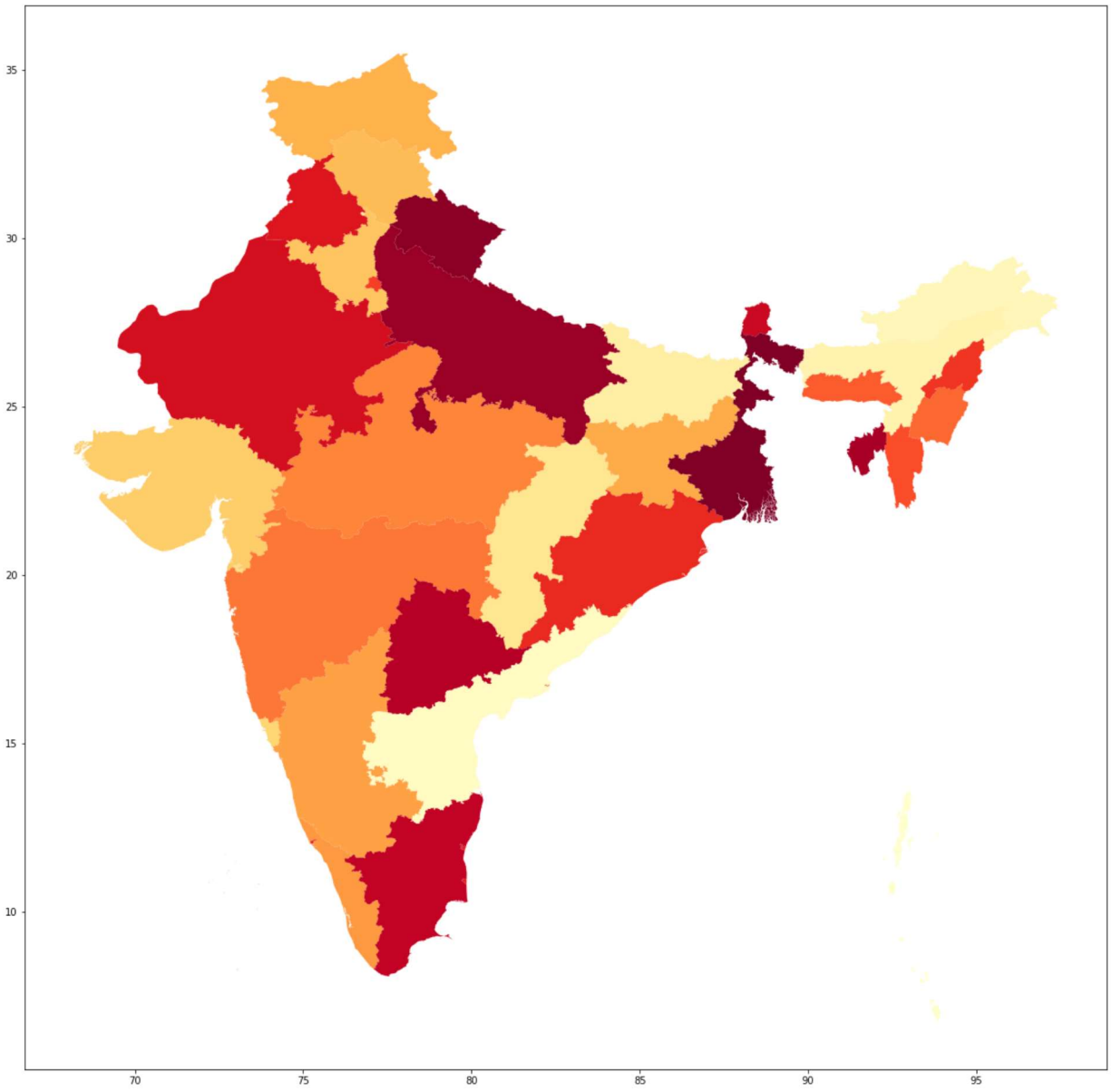
In [68]:
```
india_shape2.plot(cmap='YlOrRd', figsize=(20,20))
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204b3c7abe0>
```



In [69]:
```
india_shape_2_dissolve = india_shape2[['NAME_1', 'geometry']]
india_shape_2_dissolve = india_shape_2_dissolve.dissolve(by='NAME_1', aggfunc='su
india_shape_2_dissolve.plot(cmap='YlOrRd', figsize=(20,20))
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x204b3c081d0>



In [70]:
```
india_shape = gpd.read_file("C:\\Users\\Deepika\\Desktop\\Digital Vidya\\Data Sci
india_shape
```

Out[70]:

	GID_0	NAME_0	GID_1	NAME_1	NL_NAME_1	GID_2	NAME_2	NL_NAME_2
0	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.1_1	Nicobar Islands	None
1	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.2_1	North and Middle Andaman	None
2	IND	India	IND.1_1	Andaman and Nicobar	None	IND.1.3_1	South Andaman	None
3	IND	India	IND.2_1	Andhra Pradesh	None	IND.2.1_1	Anantapur	None
4	IND	India	IND.2_1	Andhra Pradesh	None	IND.2.1_1	Anantapur	None

In [71]:
```
india_shape.plot(cmap='YlOrRd', figsize=(20,20))
```

Out[71]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x204b3bb98d0>
```