postgresql_postgis

June 17, 2024

1 Python with PostgreSQL & PostGIS

Note that a PostgreSQL/PostGIS installation and an import of OpenStreetMap data is required for this exercise!

1.1 Libraries and Settings

[ ]:
```
# Libraries
import os
import folium
import psycopg2
import geopandas as gpd
from sqlalchemy import create_engine

# Ignore warnings
import warnings
warnings.filterwarnings("ignore")

print(os.getcwd())
```

```
c:\Users\dimit\Documents\applied_data_science\week_04\spatial_data_analysis\07_P
ython_PostgreSQL_PostGIS
```

1.2 Show tables in the database

[ ]:
```
# Create a connection
conn = psycopg2.connect(user="postgres",
                        password="geheim",
                        host="localhost",
                        port="5432",
                        database="osm_switzerland")

# Create a cursor object
cursor = conn.cursor()

# Execute the SQL query
cursor.execute("SELECT table_name FROM information_schema.tables WHERE␣
 ↪table_schema = 'public'")
```

```
# Fetch all the rows
tables = cursor.fetchall()

try:
    # Print rows
    for table in tables:
        print(table)
except (Exception, psycopg2.Error) as error:
    print("Error while connecting to PostgreSQL", error)
finally:
    # Close connection
    if conn:
        cursor.close()
        conn.close()
```

```
('planet_osm_point',)
('geography_columns',)
('geometry_columns',)
('spatial_ref_sys',)
('planet_osm_polygon',)
('raster_columns',)
('raster_overviews',)
('municipalities_ch',)
('planet_osm_line',)
('planet_osm_roads',)
```

1.3 Query spatial data from PostgreSQL database (1st example)

[ ]:
```
# Create a connection
db_connection_url = "postgresql://postgres:geheim@localhost:5432/
 ↪osm_switzerland"
conn = create_engine(db_connection_url)

# Query the database
sql = """SELECT
        p.osm_id,
        p."addr:street",
        p."addr:housenumber",
        p."addr:city",
        p."addr:postcode",
        p.building,
        st_transform(p.way, 4326) AS geom
        FROM
        public.planet_osm_polygon AS p
        WHERE
        p."addr:street" IS NOT NULL
```

```
        AND p."addr:city" = 'Zürich'
        AND p."addr:postcode" IN ('8001')"""

# Create a GeoDataFrame
gdf_01 = gpd.GeoDataFrame.from_postgis(sql, conn)
gdf_01

# Close the connection
conn.dispose()
```

1.4 Plotting the map

[ ]:
```
# Extract the x (longitude) and y (latitude) coordinates from each polygon
lon = gdf_01.geometry.apply(lambda polygon: polygon.centroid.x)
lat = gdf_01.geometry.apply(lambda polygon: polygon.centroid.y)

# Calculate the median lat/lon coordinates
lon_mean = lon.mean()
lat_mean = lat.mean()

# Initialize the map (use grayscale tiles for better contrast)
m = folium.Map(location=[lat_mean, lon_mean],
               zoom_start=15,
               tiles='CartoDB positron')

# Map settings
folium.Choropleth(
    geo_data=gdf_01,
    name='map',
    fill_color='greenyellow'
).add_to(m)

folium.LayerControl().add_to(m)

# Plot map
m
```

[ ]:
```
<folium.folium.Map at 0x2bbe45b41f0>
```

1.5 Query spatial data from PostgreSQL database (2nd example)

[ ]:
```
# Create a connection
db_connection_url = "postgresql://postgres:geheim@localhost:5432/
 ↪osm_switzerland"
conn = create_engine(db_connection_url)

# Query the database
```

```
sql = """-- Create buffers around major roads and combine these buffers to one␣
 ↪single buffer
       SELECT
       1 as group_id,
       ST_TRANSFORM(ST_UNION(ST_Buffer(p.way::geometry, 5000)), 4326) AS␣
 ↪combined_buffer_geom
       FROM public.planet_osm_roads AS p
       WHERE
       --highway IN ('motorway', 'trunk', 'primary')
       highway IN ('motorway')"""

# Create a GeoDataFrame
gdf_02 = gpd.GeoDataFrame.from_postgis(sql, conn,␣
 ↪geom_col='combined_buffer_geom')
gdf_02

# Close the connection
conn.dispose()
```

1.6 Plotting the map

[ ]:
```
# Extract the x (longitude) and y (latitude) coordinates from each polygon
lon = gdf_02.geometry.apply(lambda polygon: polygon.centroid.x)
lat = gdf_02.geometry.apply(lambda polygon: polygon.centroid.y)

# Calculate the median lat/lon coordinates
lon_mean = lon.mean()
lat_mean = lat.mean()

# Initialize the map (use grayscale tiles for better contrast)
m = folium.Map(location=[lat_mean, lon_mean],
               zoom_start=8,
               tiles='CartoDB positron')

# Map settings
folium.Choropleth(
    geo_data=gdf_02,
    name='map',
    fill_color='greenyellow'
).add_to(m)

folium.LayerControl().add_to(m)

# Plot map
m
```

[ ]:
```
<folium.folium.Map at 0x2bbe1652290>
```

1.7 Query spatial data from PostgreSQL database (3rd example)

[ ]:
```
# Create a connection
db_connection_url = "postgresql://postgres:geheim@localhost:5432/
 ↪osm_switzerland"
conn = create_engine(db_connection_url)

# Query the database
sql = """SELECT
         bfs_nummer,
         name,
         kantonsnum,
         einwohnerz,
         geom
         FROM public.municipalities_ch
         WHERE einwohnerz <= 1000"""

# Create a GeoDataFrame
gdf_03 = gpd.GeoDataFrame.from_postgis(sql, conn, geom_col='geom')
gdf_03

# Close the connection
conn.dispose()
```

1.8 Plotting the map

[ ]:
```
# Extract the x (longitude) and y (latitude) coordinates from each polygon
lon = gdf_03.geometry.apply(lambda polygon: polygon.centroid.x)
lat = gdf_03.geometry.apply(lambda polygon: polygon.centroid.y)

# Calculate the median lat/lon coordinates
lon_mean = lon.mean()
lat_mean = lat.mean()

# Initialize the map (use grayscale tiles for better contrast)
m = folium.Map(location=[lat_mean, lon_mean],
               zoom_start=8,
               tiles='CartoDB positron')

# Map settings
folium.Choropleth(
    geo_data=gdf_03,
    name='map',
    fill_color='greenyellow'
).add_to(m)

folium.LayerControl().add_to(m)
```

```
# Plot map
m
```

[ ]: <folium.folium.Map at 0x2bbe148b9d0>