Subject: **Convert New Orleans Parcel JSON to Shapefile**

Name: Parvin Momenian

LSUID: 895384980

Course: 2025 Spring GEOG 4057 for Lei Wang

Problem Statement

Urban planners, tax assessors, and policymakers require spatial datasets that represent land parcel boundaries and their associated values to support urban development and decision-making. The 2018 Market Value Analysis dataset from New Orleans is provided in JSON format with geometries encoded as WKT (Well-Known Text), making it unusable directly in ArcGIS. The goal of this project is to develop an automated GIS solution that converts this JSON data into a usable shapefile format, enabling geospatial analysis and map visualization within ArcGIS Pro.

Summary

This project demonstrates how to convert a JSON file containing land parcel data for New Orleans (in WKT format) into a shapefile using a Python Toolbox integrated with ArcGIS Pro. The final output is a shapefile visualized and exported as a professional layout map.

Data Sources

- **Dataset:** Market Value Analysis 2018
- **Source:** data.nola.gov
- **Format:** JSON with meta (field metadata) and data (attribute and geometry values) sections
- **Geometry Type:** MULTIPOLYGON in WKT format

Tools and Technologies

- Python 3 (within ArcGIS Pro environment)
- ArcGIS Pro + Python Toolbox (.pyt)

Libraries used:

- arcpy for ArcGIS integration
- pandas and json for data handling
- geopandas and shapely for spatial data processing

- **Preprocessing Steps:**
  - Download and inspect the JSON file
  - Parse JSON to extract records and field names
  - Convert WKT geometries into valid geospatial objects
  - Create a shapefile with appropriate attribute fields

Steps

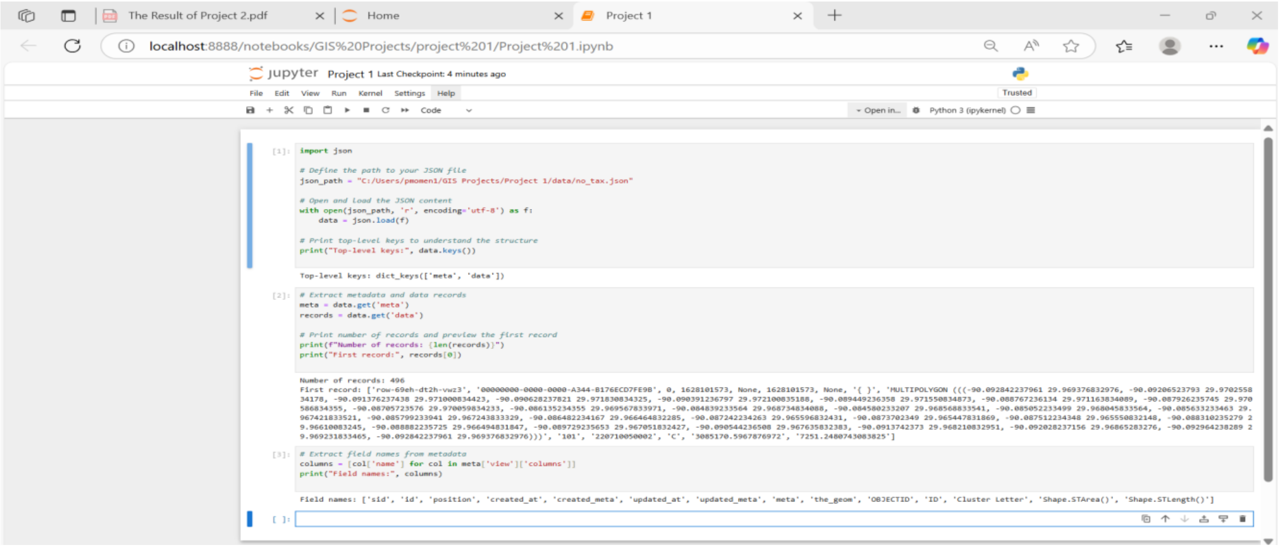
Jupyter Notebook Testing

```
import json
# Define the path to your JSON file
json_path = "C:/Users/pmomen1/GIS Projects/Project 1/data/no_tax.json"
# Open and load the JSON content
with open(json_path, 'r', encoding='utf-8') as f:
    data = json.load(f)
# Print top-level keys to understand the structure
print("Top-level keys:", data.keys())
```

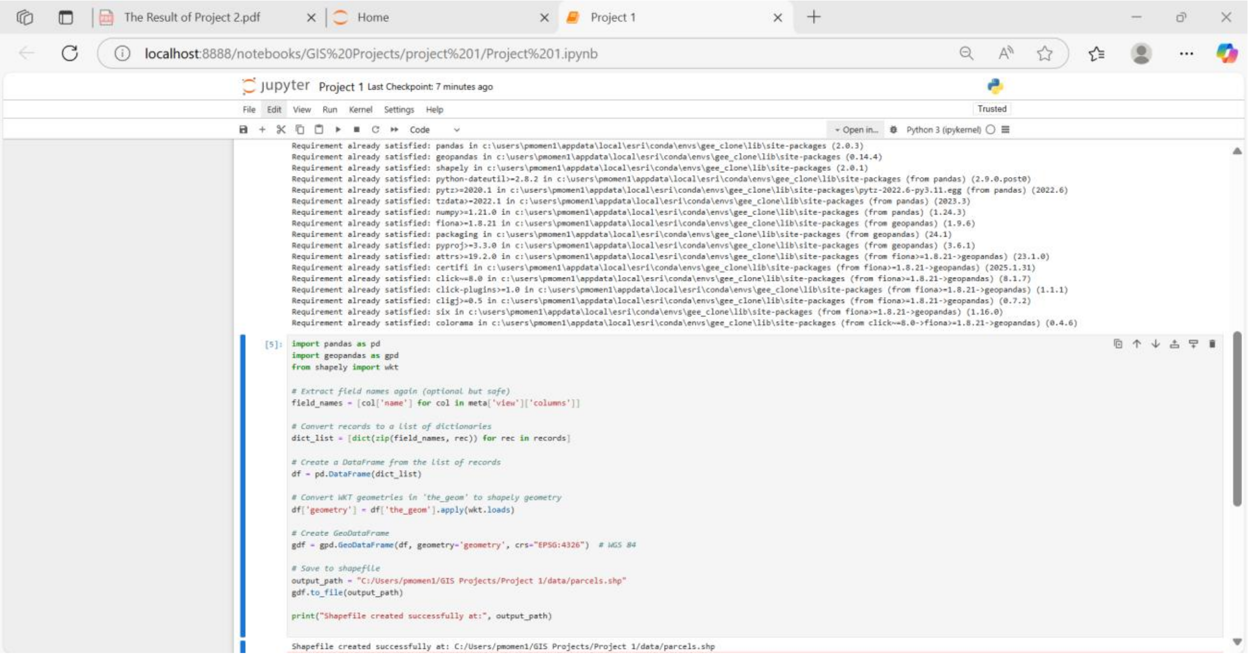
```
# Extract metadata and data records
meta = data.get('meta')
records = data.get('data')
# Print number of records and preview the first record
print(f"Number of records: {len(records)}")
print("First record:", records[0])
```

```
# Extract field names from metadata
columns = [col['name'] for col in meta['view']['columns']]
print("Field names:", columns)
```

```
import pandas as pd

import geopandas as gpd

from shapely import wkt

# Extract field names again (optional but safe)

field_names = [col['name'] for col in meta['view']['columns']]

# Convert records to a list of dictionaries

dict_list = [dict(zip(field_names, rec)) for rec in records]

# Create a DataFrame from the list of records

df = pd.DataFrame(dict_list)

# Convert WKT geometries in 'the_geom' to shapely geometry

df['geometry'] = df['the_geom'].apply(wkt.loads)

# Create GeoDataFrame

gdf = gpd.GeoDataFrame(df, geometry='geometry', crs="EPSG:4326")  # WGS 84

# Save to shapefile

output_path = "C:/Users/pmomen1/GIS Projects/Project 1/data/parcels.shp"

gdf.to_file(output_path)

print("Shapefile created successfully at:", output_path)
```


The Result of Project 2.pdf
Home
Project 1
localhost:8888/notebooks/GIS%20Projects/project%201/Project%201.ipynb
jupyter Project 1 Last Checkpoint: 4 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)

```
[1]: import json

     # Define the path to your JSON file
     json_path = "C:/Users/pmomen1/GIS Projects/Project 1/data/no_tax.json"

     # Open and Load the JSON content
     with open(json_path, 'r', encoding='utf-8') as f:
         data = json.load(f)

     # Print top-level keys to understand the structure
     print("Top-level keys:", data.keys())

     Top-level keys: dict_keys(['meta', 'data'])

[2]: # Extract metadata and data records
     meta = data.get('meta')
     records = data.get('data')

     # Print number of records and preview the first record
     print(f"Number of records: {len(records)}")
     print("First record:", records[0])

     Number of records: 496
     First record: ['row-69eh-dt2h-vwz3', '00000000-0000-0000-A344-B176ECD7FE9B', 0, 1628101573, None, 1628101573, None, '{ }', 'MULTIPOLYGON (((-90.092842237961 29.969376832976, -90.09206523793 29.9702558
     34178, -90.091376237438 29.971000834423, -90.090628237821 29.971830834325, -90.090391236797 29.972100835188, -90.089449236358 29.971550834873, -90.088767236134 29.971163834089, -90.087926235745 29.970
     586834355, -90.08705723576 29.970059834233, -90.086135234355 29.969567833971, -90.084839233564 29.968734834088, -90.084580233207 29.968568833541, -90.085052233499 29.968045833564, -90.085633233463 29.
     967421833521, -90.085799233941 29.967243833329, -90.086482234167 29.966464832285, -90.087242234263 29.965596832431, -90.0873702349 29.965447831869, -90.087512234348 29.965550832148, -90.088310235279 2
     9.96610083245, -90.088882235725 29.966494831847, -90.089729235653 29.967051832427, -90.090544236508 29.967635832383, -90.0913742373 29.968210832951, -90.092028237156 29.96865283276, -90.092964238289 2
     9.969231833465, -90.092842237961 29.969376832976)))', '101', '220710050002', 'C', '3085170.5967876972', '7251.2480743083825']

[3]: # Extract field names from metadata
     columns = [col['name'] for col in meta['view']['columns']]
     print("Field names:", columns)

     Field names: ['sid', 'id', 'position', 'created_at', 'created_meta', 'updated_at', 'updated_meta', 'meta', 'the_geom', 'OBJECTID', 'ID', 'Cluster Letter', 'Shape.STArea()', 'Shape.STLength()']

[ ]:
```



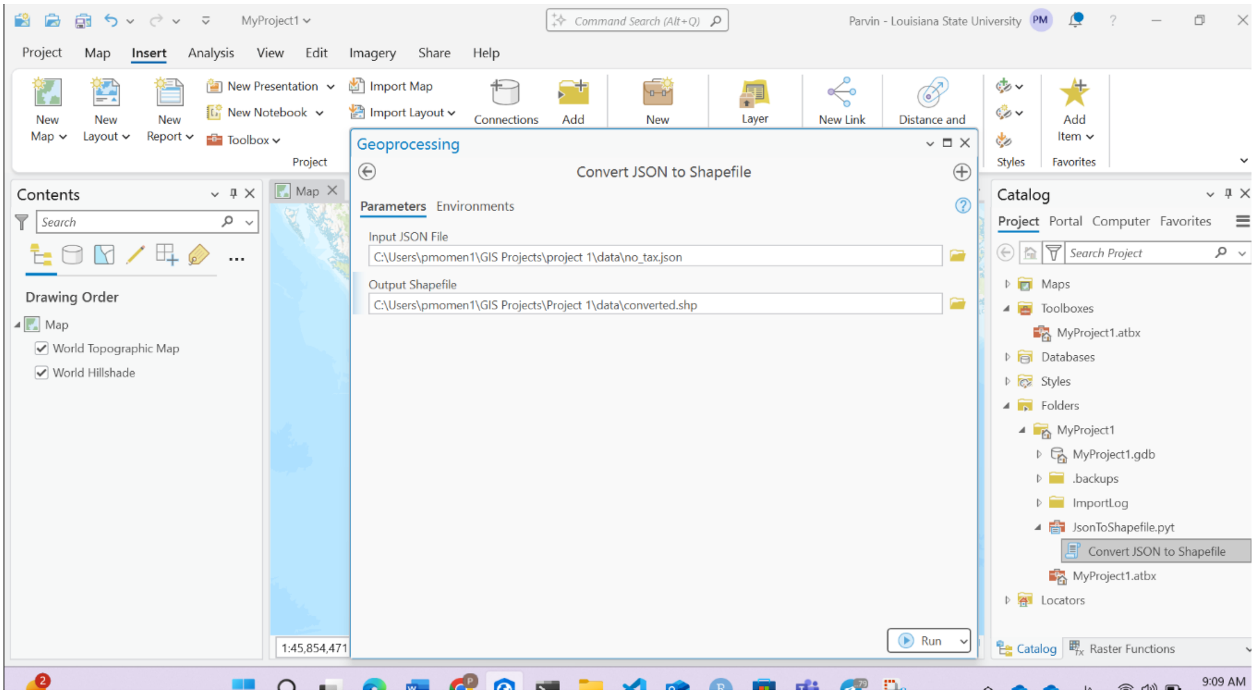
The Result of Project 2.pdf
Home
Project 1
localhost:8888/notebooks/GIS%20Projects/project%201/Project%201.ipynb
jupyter Project 1 Last Checkpoint: 7 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Trusted
Code
Open in...
Python 3 (ipykernel)

```
Requirement already satisfied: pandas in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (2.0.3)
Requirement already satisfied: geopandas in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (0.14.4)
Requirement already satisfied: shapely in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages\pytz-2022.6-py3.11.egg (from pandas) (2022.6)
Requirement already satisfied: tzdata>=2022.1 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: fiona>=1.8.21 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from geopandas) (1.9.6)
Requirement already satisfied: packaging in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from geopandas) (24.1)
Requirement already satisfied: pyproj>=3.3.0 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from geopandas) (3.6.1)
Requirement already satisfied: attrs>=19.2.0 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (23.1.0)
Requirement already satisfied: certifi in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (2025.1.31)
Requirement already satisfied: click~=8.0 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (0.7.2)
Requirement already satisfied: six in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from fiona>=1.8.21->geopandas) (1.16.0)
Requirement already satisfied: colorama in c:\users\pmomen1\appdata\local\esri\conda\envs\gee_clone\lib\site-packages (from click~=8.0->fiona>=1.8.21->geopandas) (0.4.6)
```

```
[5]: import pandas as pd
     import geopandas as gpd
     from shapely import wkt

     # Extract field names again (optional but safe)
     field_names = [col['name'] for col in meta['view']['columns']]

     # Convert records to a list of dictionaries
     dict_list = [dict(zip(field_names, rec)) for rec in records]

     # Create a DataFrame from the list of records
     df = pd.DataFrame(dict_list)

     # Convert WKT geometries in 'the_geom' to shapely geometry
     df['geometry'] = df['the_geom'].apply(wkt.loads)

     # Create GeoDataFrame
     gdf = gpd.GeoDataFrame(df, geometry='geometry', crs="EPSG:4326")  # WGS 84

     # Save to shapefile
     output_path = "C:/Users/pmomen1/GIS Projects/Project 1/data/parcels.shp"
     gdf.to_file(output_path)

     print("Shapefile created successfully at:", output_path)
```

```
Shapefile created successfully at: C:/Users/pmomen1/GIS Projects/Project 1/data/parcels.shp
```

Python Toolbox Development



Convert JSON to Shapefile

Input JSON File
C:\Users\pmomen1\GIS Projects\project 1\data\no_tax.json

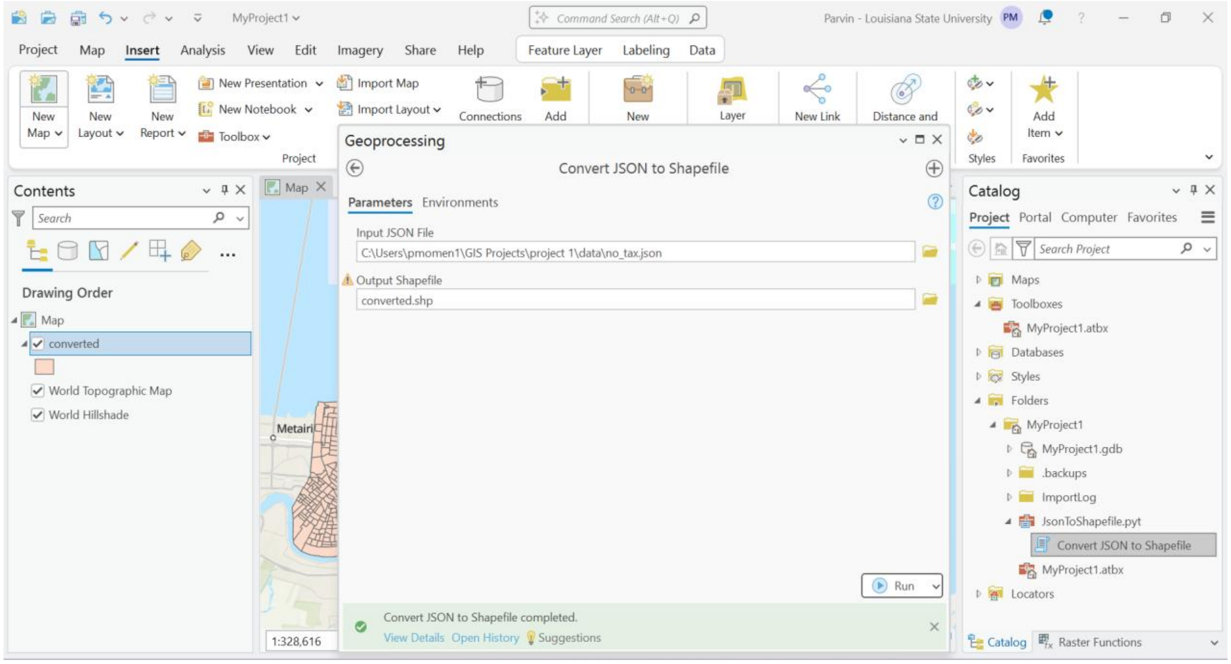
Output Shapefile
C:\Users\pmomen1\GIS Projects\Project 1\data\converted.shp

```
import arcpy
import json
import pandas as pd
import geopandas as gpd
from shapely import wkt
class Toolbox(object):
    def __init__(self):
        self.label = "JSON to Shapefile Toolbox"
        self.alias = "json2shp"
        self.tools = [JsonToShapefile]
class JsonToShapefile(object):
    def __init__(self):
        self.label = "Convert JSON to Shapefile"
        self.description = "Converts a JSON file with WKT geometries to a shapefile"
    def getParameterInfo(self):
        params = [
            arcpy.Parameter(
                displayName="Input JSON File",
                name="input_json",
                datatype="DEFile",
                parameterType="Required",
                direction="Input"
            ),
            arcpy.Parameter(
                displayName="Output Shapefile",
                name="output_shp",
                datatype="DEFeatureClass",
                parameterType="Required",
                direction="Output"
            )
        ]
        return params
```

```
def execute(self, parameters, messages):
  input_json = parameters[0].valueAsText
  output_shp = parameters[1].valueAsText
  try:
    # Load JSON content
    with open(input_json, 'r', encoding='utf-8') as f:
      data = json.load(f)
    # Extract meta and data sections
    meta = data.get('meta')
    records = data.get('data')
    field_names = [col['name'] for col in meta['view']['columns']]
    dict_list = [dict(zip(field_names, rec)) for rec in records]
    # Convert to DataFrame and then GeoDataFrame
    df = pd.DataFrame(dict_list)
    df['geometry'] = df['the_geom'].apply(wkt.loads)
    gdf = gpd.GeoDataFrame(df, geometry='geometry', crs="EPSG:4326")
    # Save to shapefile
    gdf.to_file(output_shp)
    messages.addMessage(" Shapefile successfully created at: " + output_shp)
  except Exception as e:
    messages.addErrorMessage(" Error occurred: " + str(e))
```

Results and Visualization

- Generated shapefile successfully displayed in ArcGIS Pro



Geoprocessing

Convert JSON to Shapefile

Parameters Environments

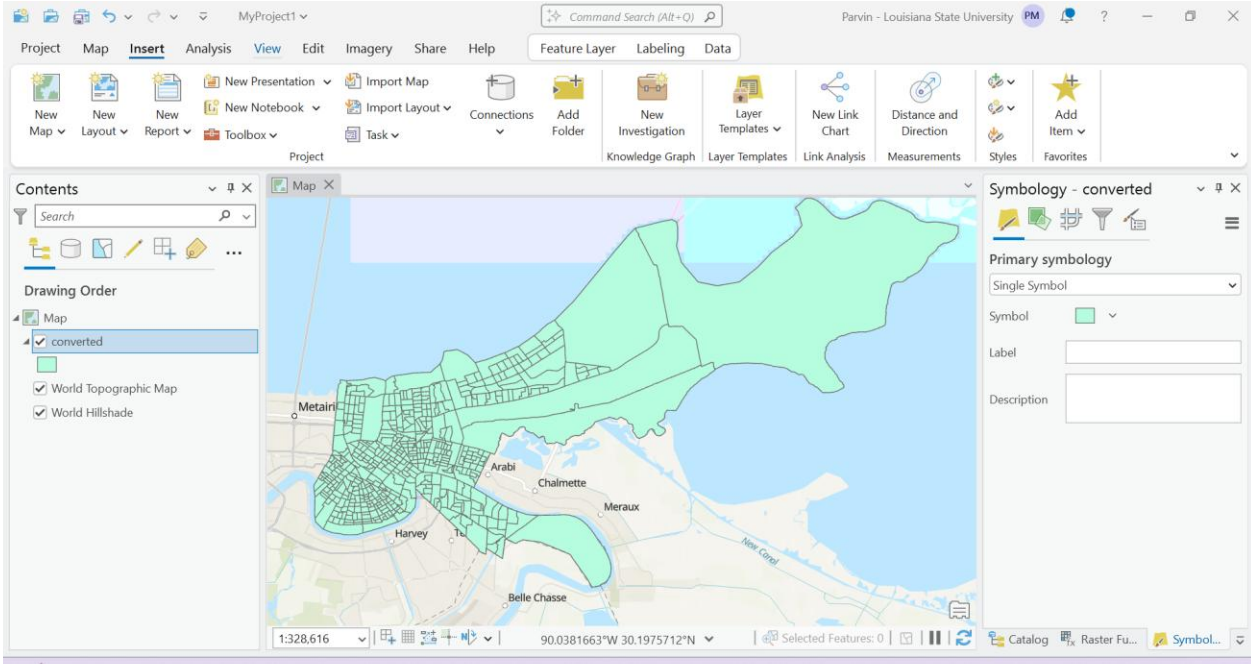
Input JSON File
C:\Users\pmomen1\GIS Projects\project 1\data\no_tax.json

Output Shapefile
converted.shp

Run

Convert JSON to Shapefile completed.
View Details Open History Suggestions

- Applied Unique Values symbology using the "Cluster Le" field.



Symbology - converted

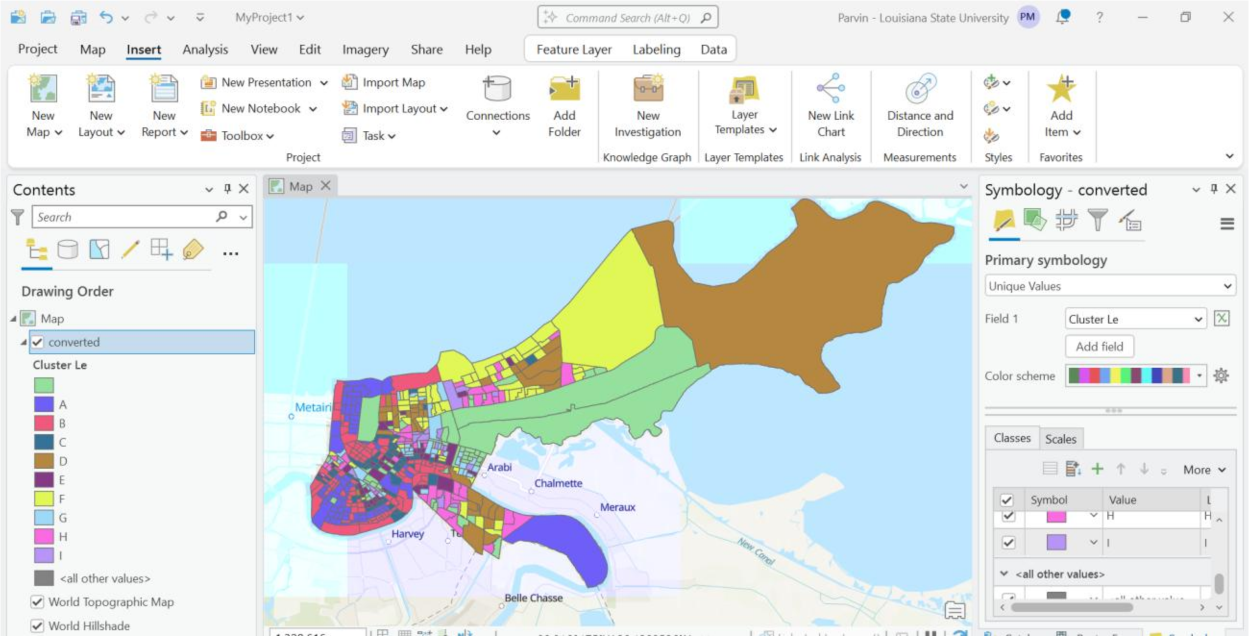
Primary symbology
Single Symbol

Symbol 🟩

Label

Description

- Designed a layout with title, legend, scale bar, and north arrow.



Symbology - converted

Primary symbology: Unique Values

Field 1: Cluster Le

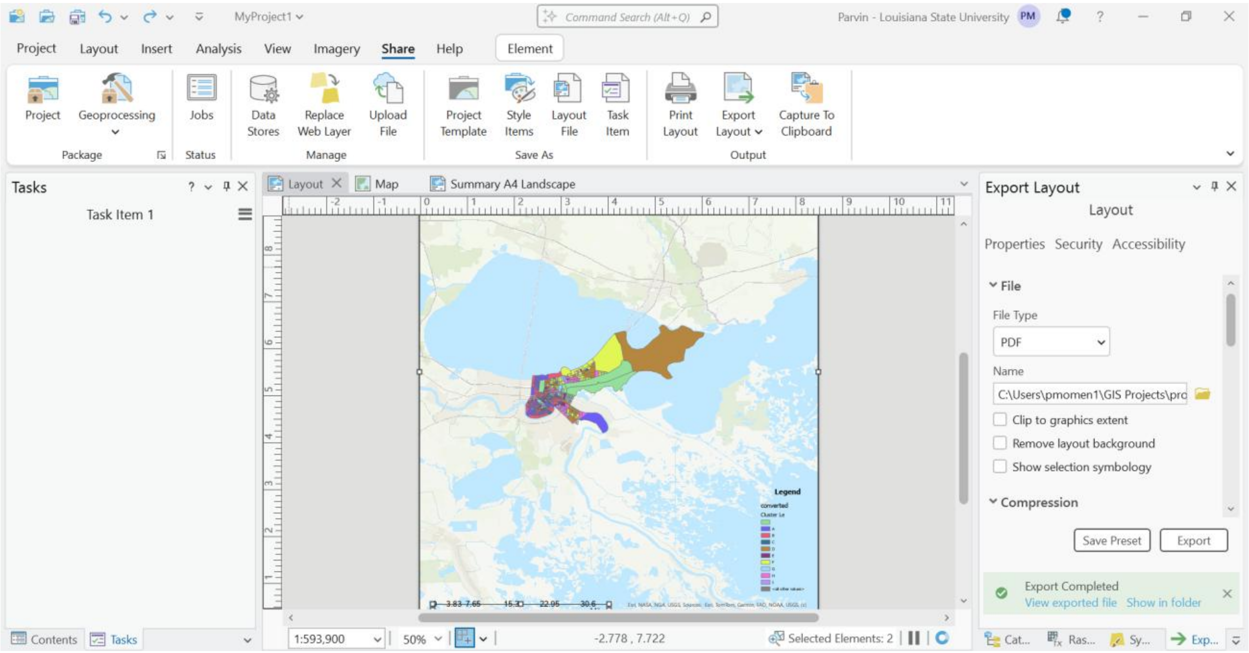
Color scheme

Contents — Drawing Order

- Map
  - ☑ converted
    - Cluster Le
      - 🟩
      - 🟦 A
      - 🟥 B
      - 🟦 C
      - 🟫 D
      - 🟪 E
      - 🟨 F
      - 🟦 G
      - 🟪 H
      - 🟪 I
      - ⬛ <all other values>
  - ☑ World Topographic Map
  - ☑ World Hillshade

Export Output

- Exported the layout to PDF using ArcGIS Pro's Export Layout tool.



Export Layout — Layout

Properties | Security | Accessibility

File

File Type: PDF

Name: C:\Users\pmomen1\GIS Projects\pro

☐ Clip to graphics extent

☐ Remove layout background

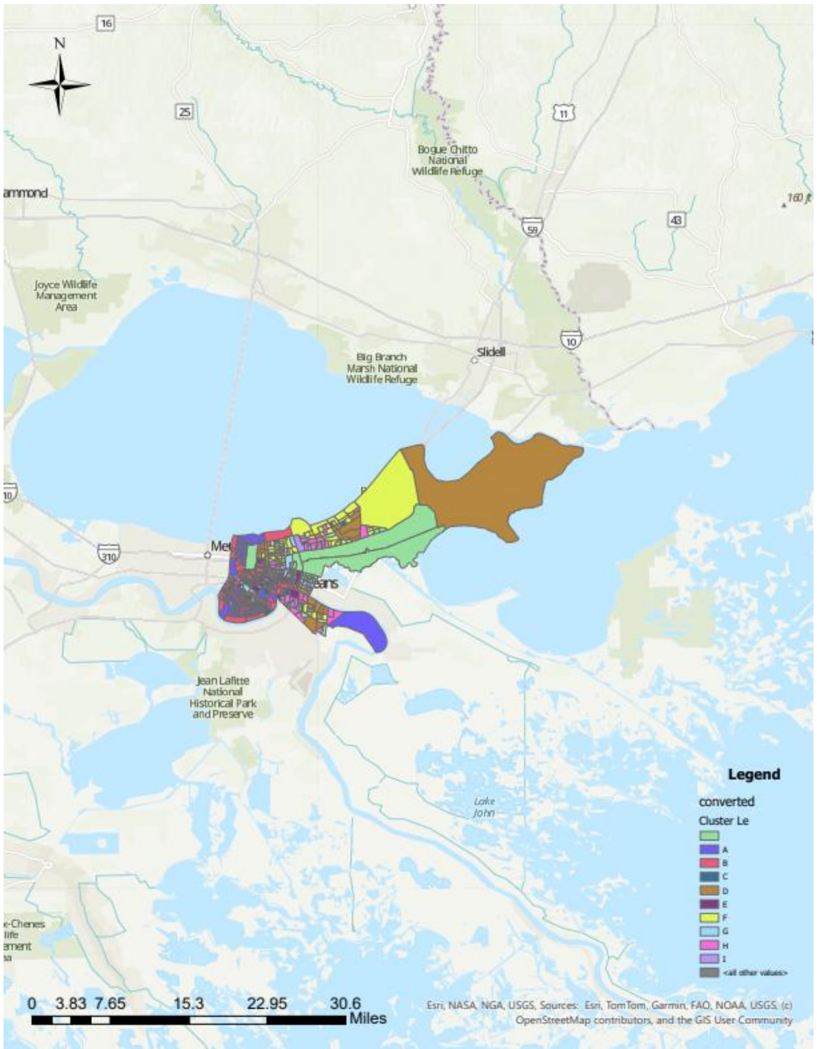
☐ Show selection symbology

Compression

Save Preset | Export

Export Completed — View exported file | Show in folder

1:593,900 | 50% | -2.778 , 7.722 | Selected Elements: 2


N
Bogue Chitto National Wildlife Refuge
Joyce Wildlife Management Area
Big Branch Marsh National Wildlife Refuge
Slidell
Jean Lafitte National Historical Park and Preserve
Lake John
160 ft
Legend
converted
Cluster Le
A
B
C
D
E
F
G
H
I
<all other values>
0 3.83 7.65 15.3 22.95 30.6
Miles
Esri, NASA, NGA, USGS, Sources: Esri, TomTom, Garmin, FAO, NOAA, USGS, (c) OpenStreetMap contributors, and the GIS User Community

References

- Market Value Analysis 2018, data.nola.gov
- Esri ArcGIS Pro Python Documentation
- Python libraries: pandas, geopandas, shapely, arcpy

Academic Integrity Note

This project report and code were developed with the assistance of OpenAI ChatGPT for code review.