



## **Internship Report**

Data Pipeline for Geographic Information of Four Crop Types and  
Data Pipeline for Geographic Information of Perennial Plants (Palm and Rubber)

**Presented to**

Sirikul Hutasavi

**By**

Pongphat Mitrathanun

Jatenipat Ausanarassamee

**This report is part of the internship at the**

**Geo-Informatics Application and Management Department of Agriculture**

**Economics and Geo-Social**

**Geo-Informatics and Space Technology Development Agency (Public Organization)**

**June - July 2024**

## Technologies Used

1. Visual Studio Code



Visual Studio Code is a popular text editor developed by Microsoft, widely used by programmers and coders. VS Code is highly versatile and flexible, making it suitable for developing a wide range of applications and programs.

2. Python Programming



Python is a programming language known for its easy-to-read and easy-to-write syntax. It supports multiple platforms, has a comprehensive standard library, and a large user community. Python supports object-oriented programming and can be integrated with other languages effectively. It is commonly used for web development, data analysis, artificial intelligence development, and many other applications.

3. QGIS



QGIS (Quantum GIS) is a free and open-source geographic information system software. It is continuously developed and includes various plugins to support specific functionalities.

## **Libraries Used**

1. GeoPandas



GeoPandas is a Python library designed specifically for working with geospatial data. It extends the capabilities of pandas, which is used for managing tabular data, to handle geospatial data as well. GeoPandas allows reading, writing, and analyzing geospatial data, including managing geometric shapes such as points, lines, and polygons.

### **Installation of GeoPandas**

Install GeoPandas using the command “`pip install geopandas`” then using the command “`import geopandas as gpd`”

2. Pandas



Pandas is a Python library developed for managing and analyzing tabular data. It has two main data structures, Series and DataFrame, which are efficient for handling large and complex data.

### **Installation of Pandas:**

Install Pandas using the command “`pip install pandas`” then using the command “`import pandas as pd`”

### 3. TQDM



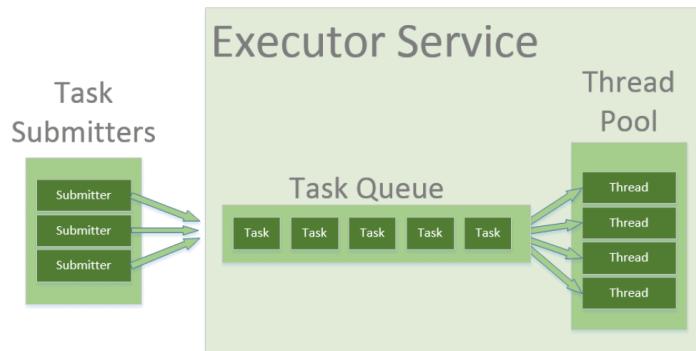
TQDM, derived from the Arabic word "taqadum" meaning progress, is used to display progress bars in the terminal of the editor being used.

```
> /usr/bin/python3 "/home/rachata/pre-project resource/code/test.py"
100%|██████████| 1000/1000 [00:00<00:00, 1186171.95it/s]
```

#### Installation of TQDM:

Install tqdm using the command “pip install tqdm” then using the command “from tqdm import tqdm”

### 4. ThreadPoolExecutor and as\_completed from concurrent.futures



#### ThreadPoolExecutor and as\_completed from concurrent.futures

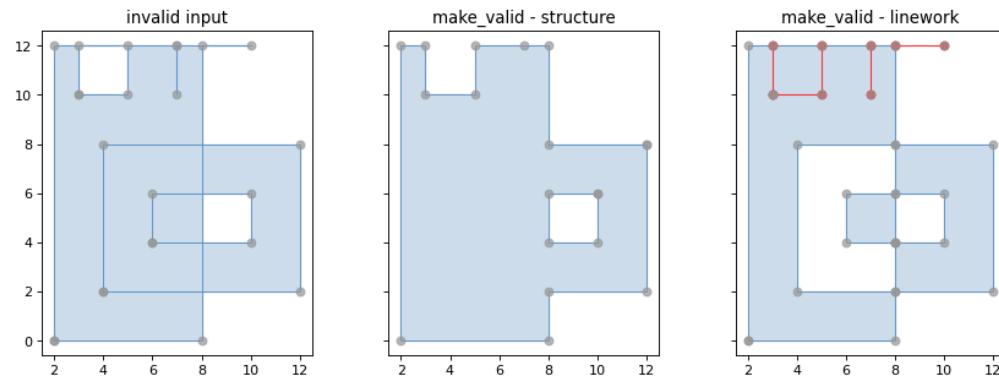
ThreadPoolExecutor is a class in the concurrent.futures module in Python used for managing parallel execution of tasks. It controls the creation and usage of multiple threads to process multiple tasks simultaneously, enhancing data processing efficiency and reducing processing time. It is suitable for data processing, I/O operations, or computations requiring significant time.

`as_completed` is a built-in function in the `concurrent.futures` module for managing parallel execution tasks. It returns an iterator that yields completed tasks as they finish, not in the order they were submitted. This function allows handling results as soon as they are ready without waiting for all tasks to complete.

### **Installation of ThreadPoolExecutor and `as_completed` from `concurrent.futures`**

Install `ThreadPoolExecutor` and `as_completed` from `concurrent.futures` using the command “`from concurrent.futures import ThreadPoolExecutor, as_completed`  
`from shapely.validation import make_valid`”

#### 5. `make_valid` from `shapely.validation`

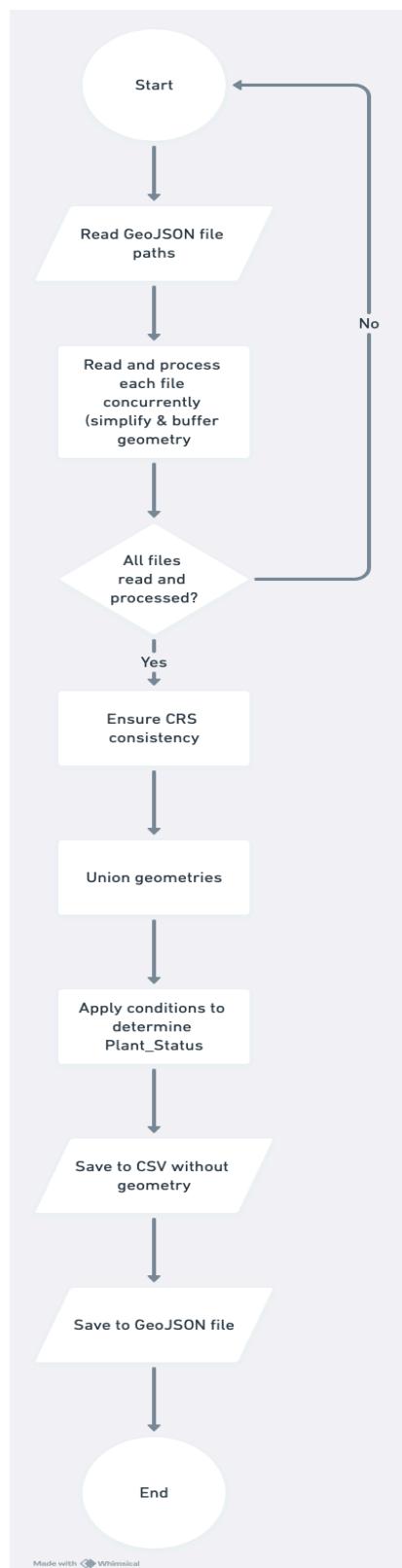


`make_valid` from the `shapely.validation` library is used to correct potentially invalid geometric objects, making them valid according to geometric processing rules. This can resolve issues arising from geometries that do not conform to geometric rules, such as overlaying geographic data.

### **Installation of shapely**

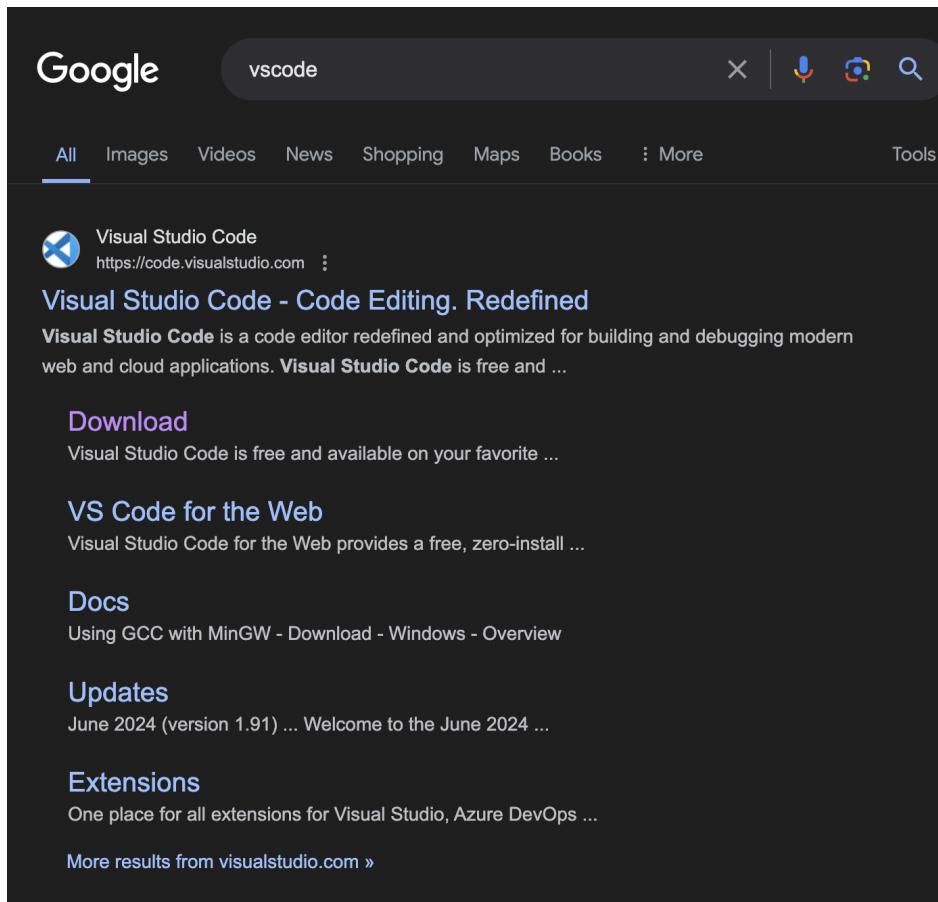
Install `make_valid` from the `shapely.validation` using the command “`pip install shapely`” then using the command “`from shapely.validation import make_valid`”

# Project Workflow



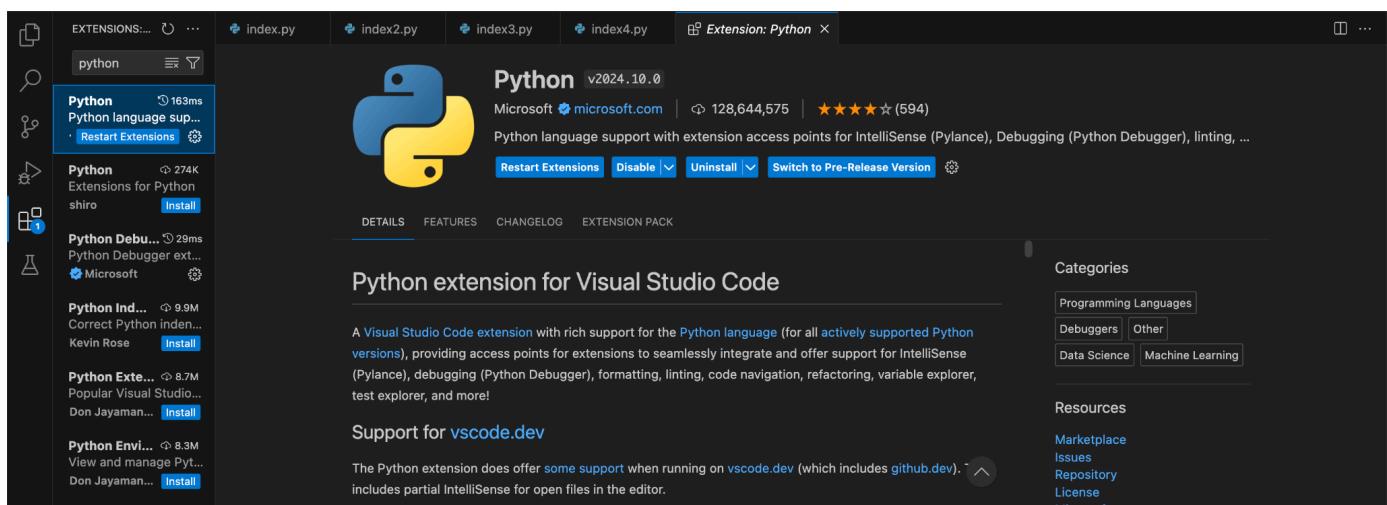
# How to use the program

## Step 1: Install Vscode

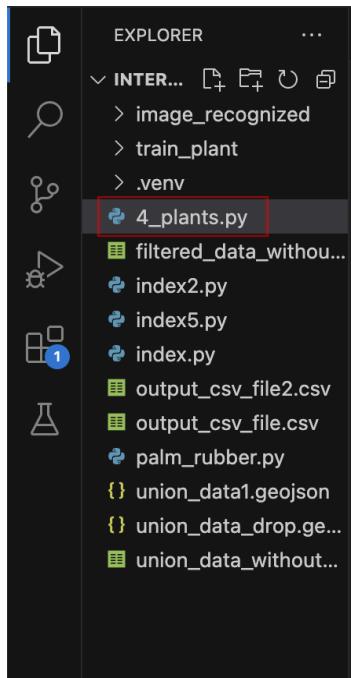


<https://code.visualstudio.com/download>

## Step 2: Open Vscode and Install extension



Step 3: For the Four crop type , Find the file name “4\_plants.py”



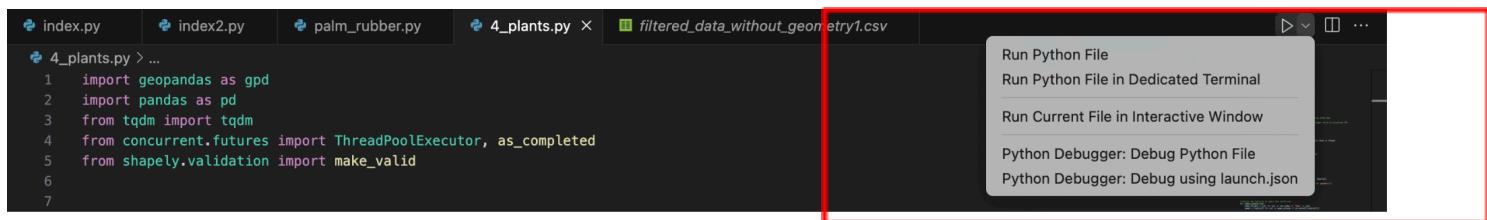
Step 4: Replace file path with the file that you want.

```
8 # List of file paths to the GeoJSON files
9 √ files = [
10     "/Users/vnlight/Desktop/GeoJson/Maize_65.geojson",
11     "/Users/vnlight/Desktop/GeoJson/Palm_65.geojson",
12     "/Users/vnlight/Desktop/GeoJson/Sugarcane_65.geojson"
13     "/Users/vnlight/Desktop/GeoJson/Rubber_65.geojson"
14 ]
15
```

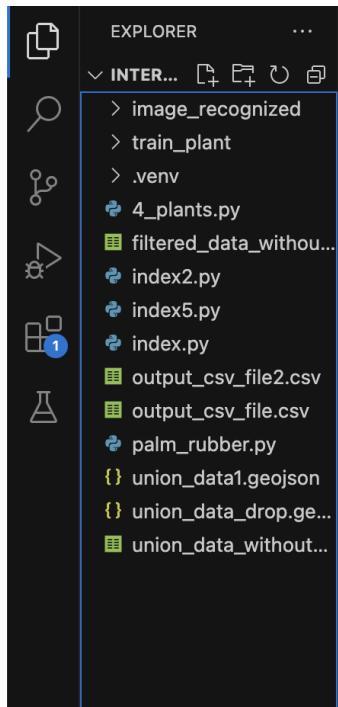
Step 5: You can change the name of the csv file and geojson file by changing the output\_csv\_file and output\_geojson\_file to the file name that you want.

```
76 # Save the GeoDataFrame to a CSV file without the geometry column
77 output_csv_file = 'union_data_without_geometry_4_plant.csv'
78 gdf_union.drop(columns=['geometry']).to_csv(output_csv_file, index=False)
79 print(f"Union data without geometry saved to {output_csv_file}")
80
81
82 # Save to geojson
83 geojson_data = gdf_union.to_json()
84 output_geojson_file = 'union_data_4_plant.geojson'
85 with open(output_geojson_file, 'w') as f:
86     f.write(geojson_data)
87 print(f"Union data saved to {output_geojson_file}")
88
```

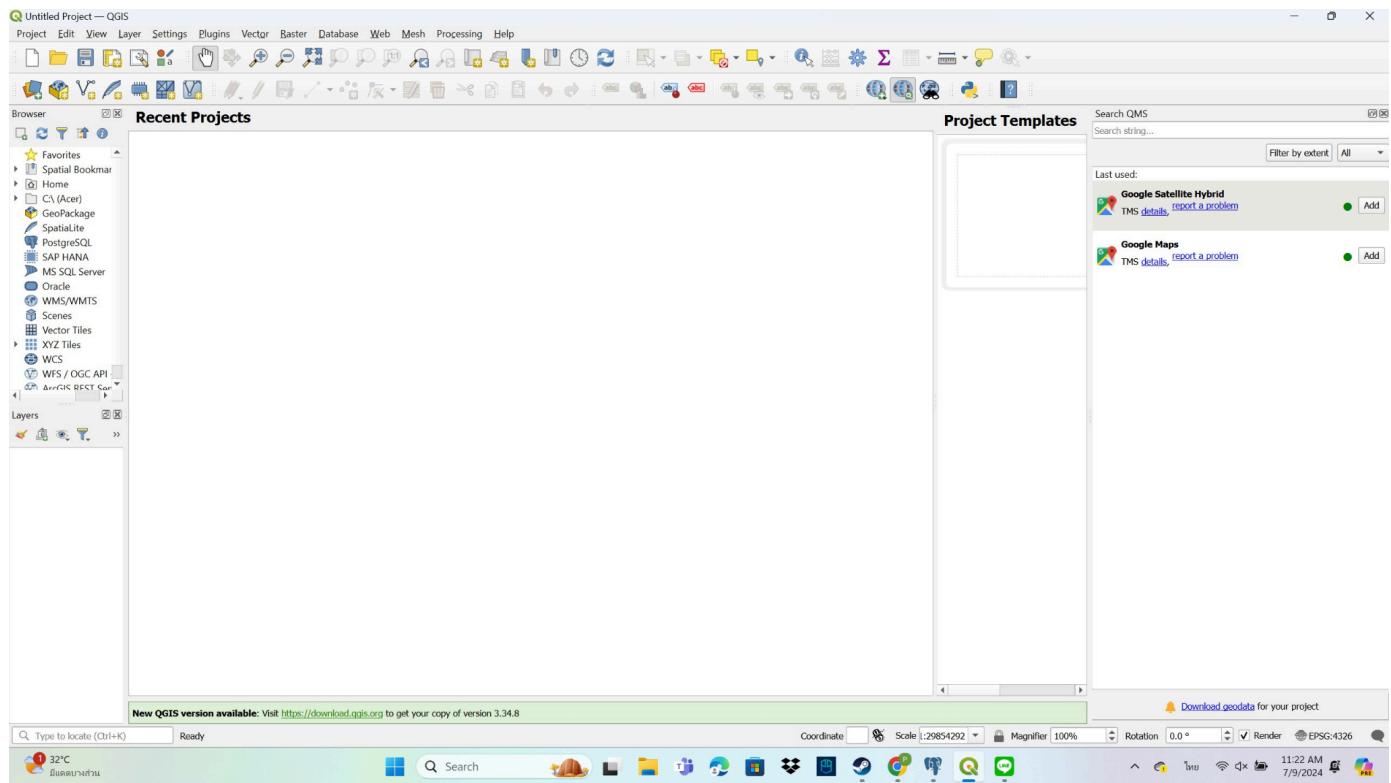
Step 6: After you change all things that are important to run. You can press on run button to run this code (Please ensure that you already have installed all necessary libraries)



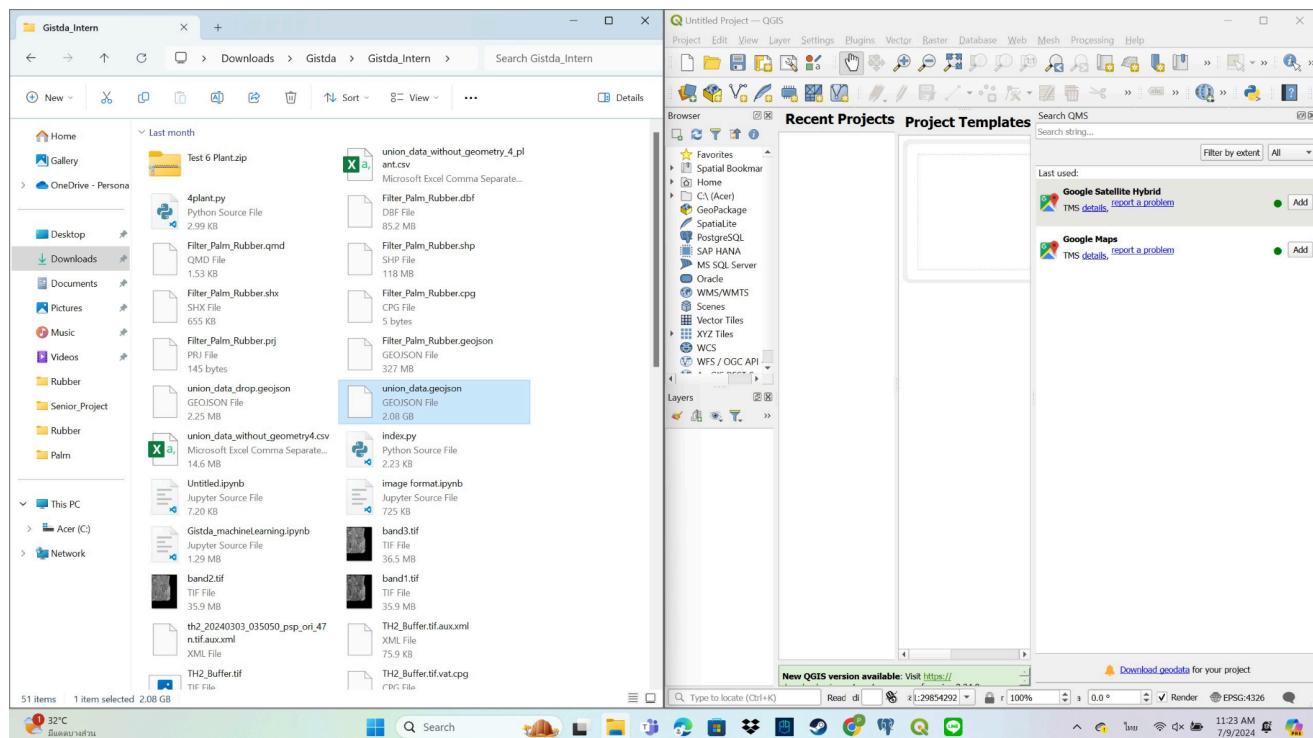
Step 7: You can see the output on the left side of the screen



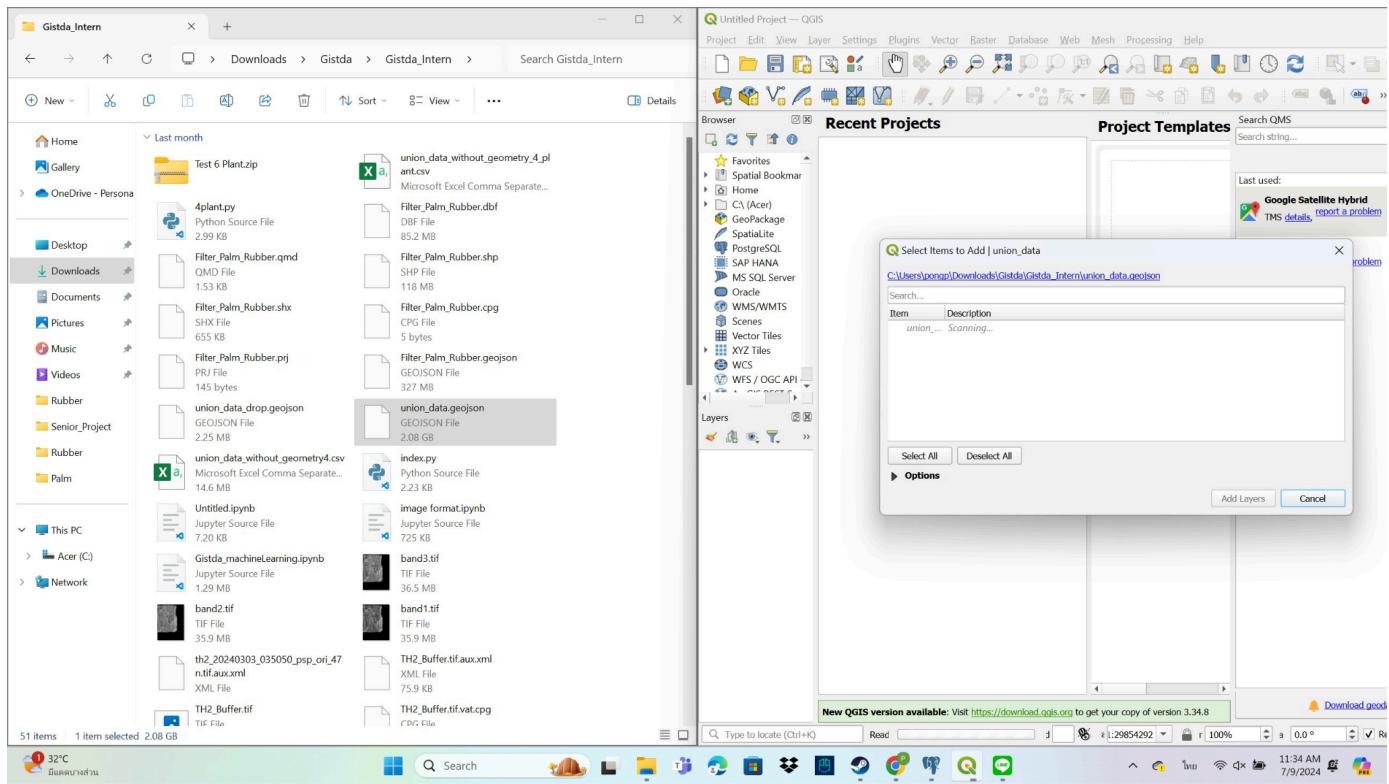
## Step 8: Go to QGIS



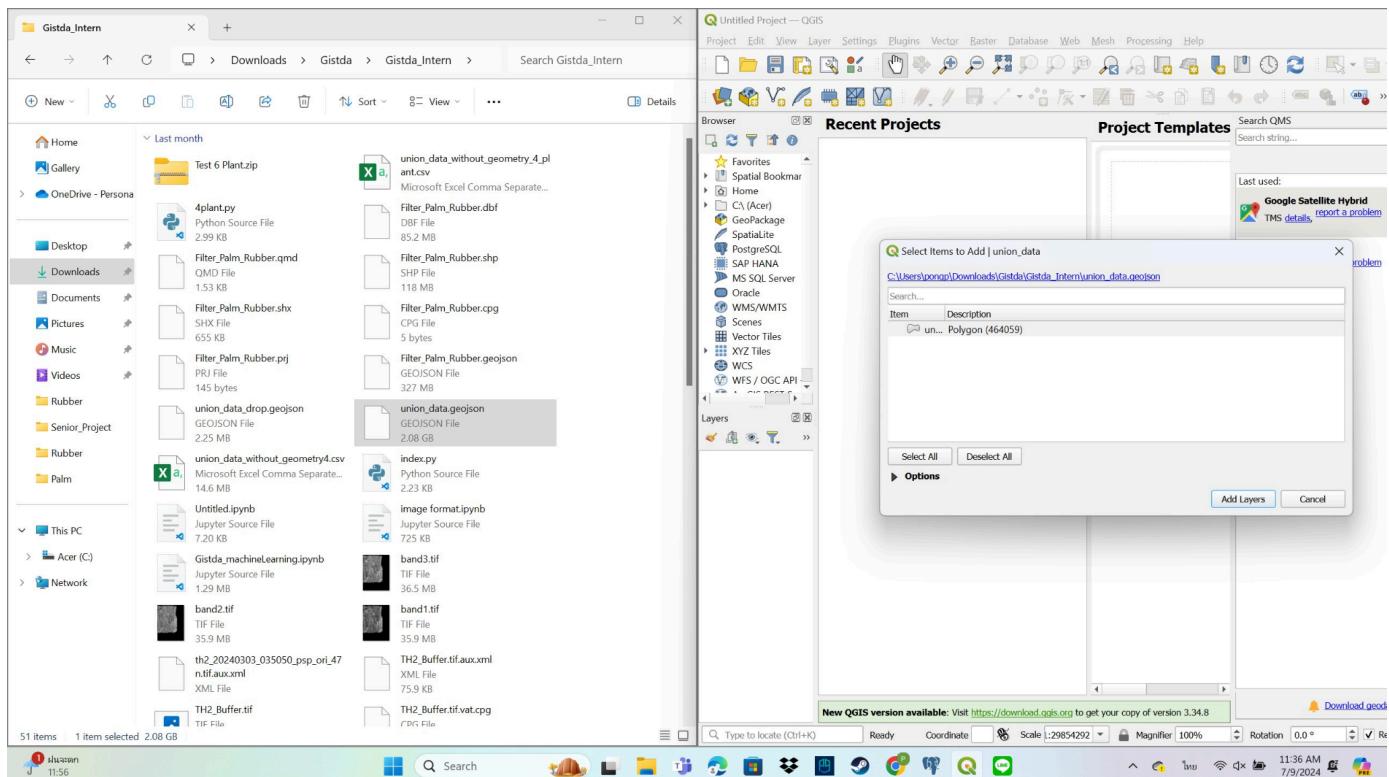
Step 9: Drag and drop the geojson file that you want to use to QGIS. (In this case, I assume to use union\_data.geojson file)



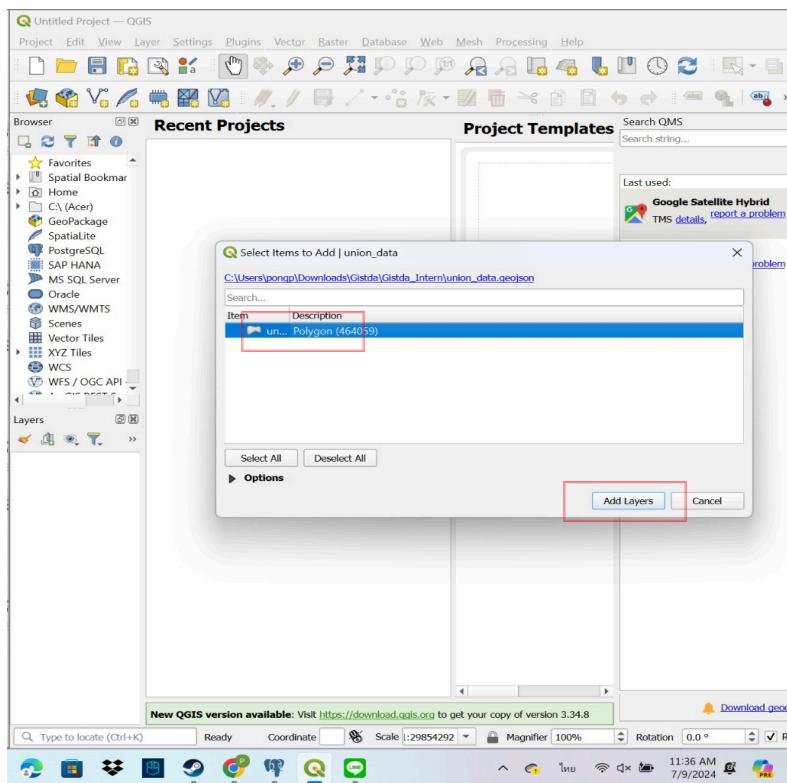
## Step 10: Wait for the file to download



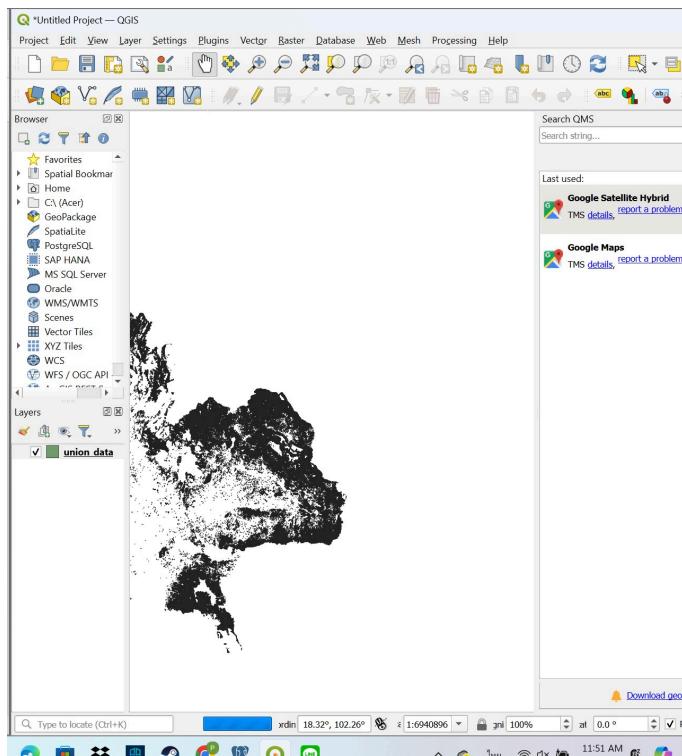
## Step 11: After download, File will appear.



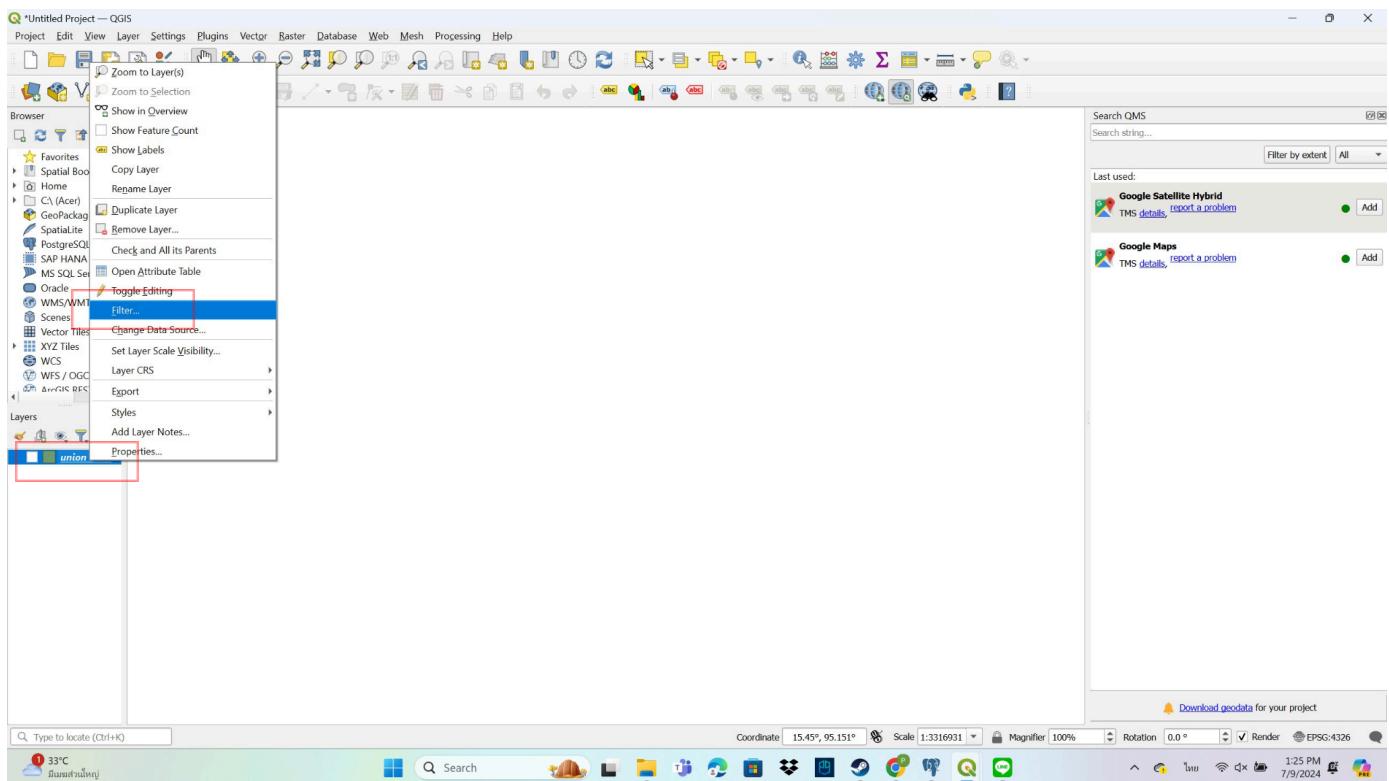
Step 12: Click to the file and Click on the Add layers button.



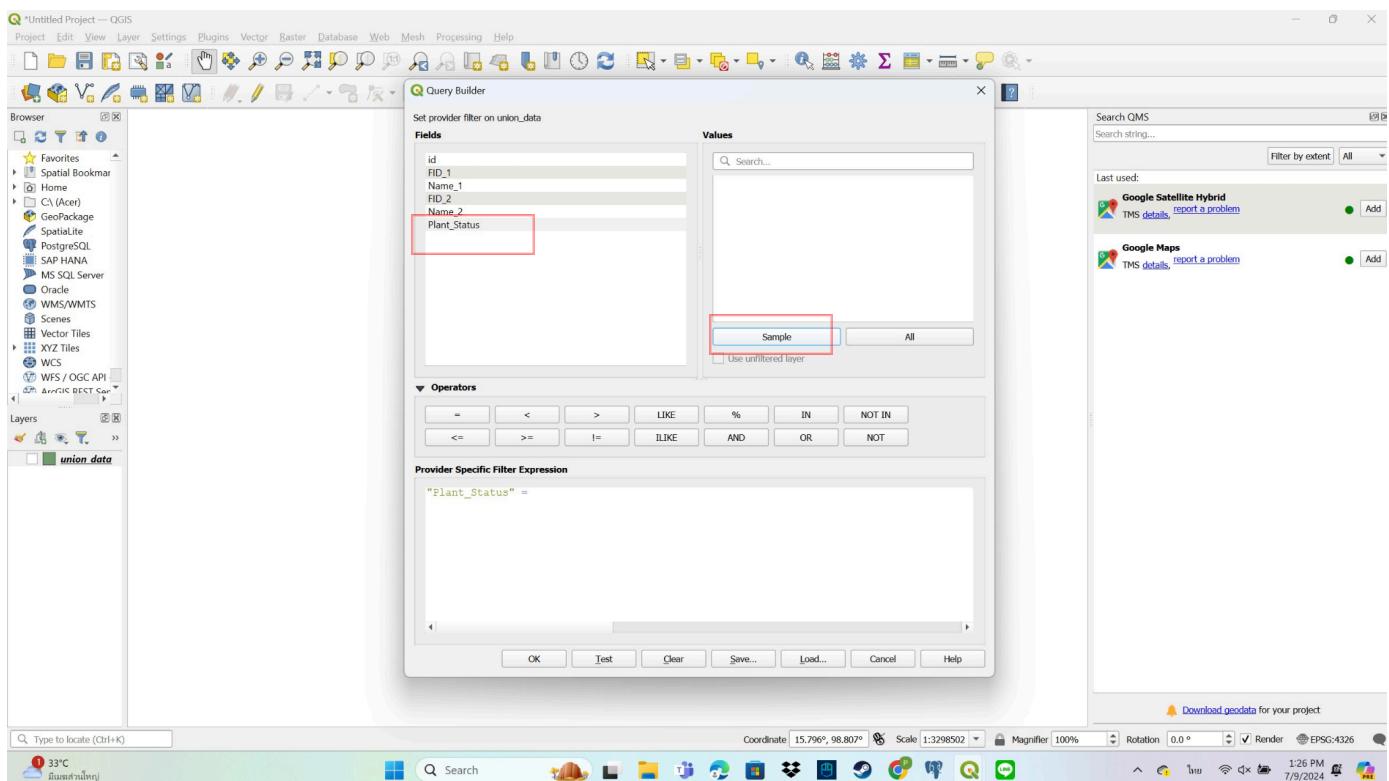
Step 13: It will display an image as shown below



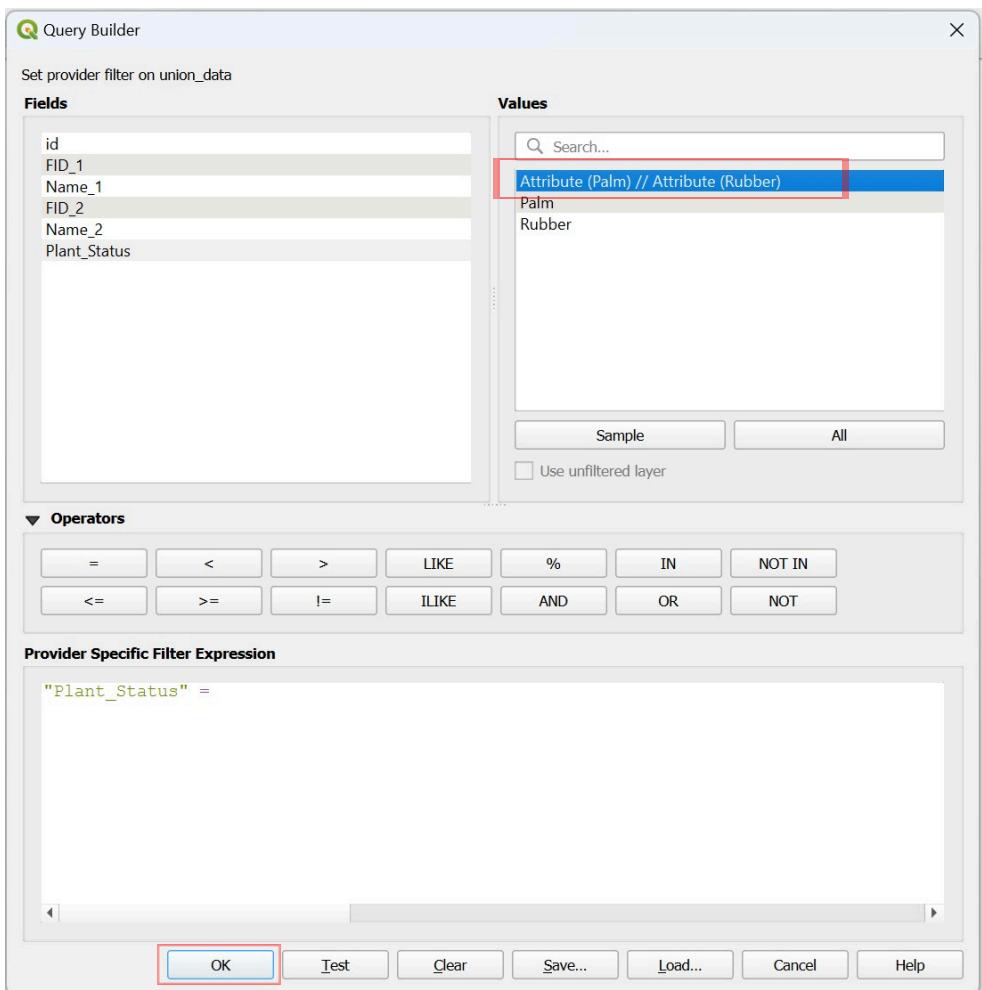
## Step 14: Click to the union\_data and click to the filter



## Step 15: Click to the Plant\_Status then Click to the Sample button



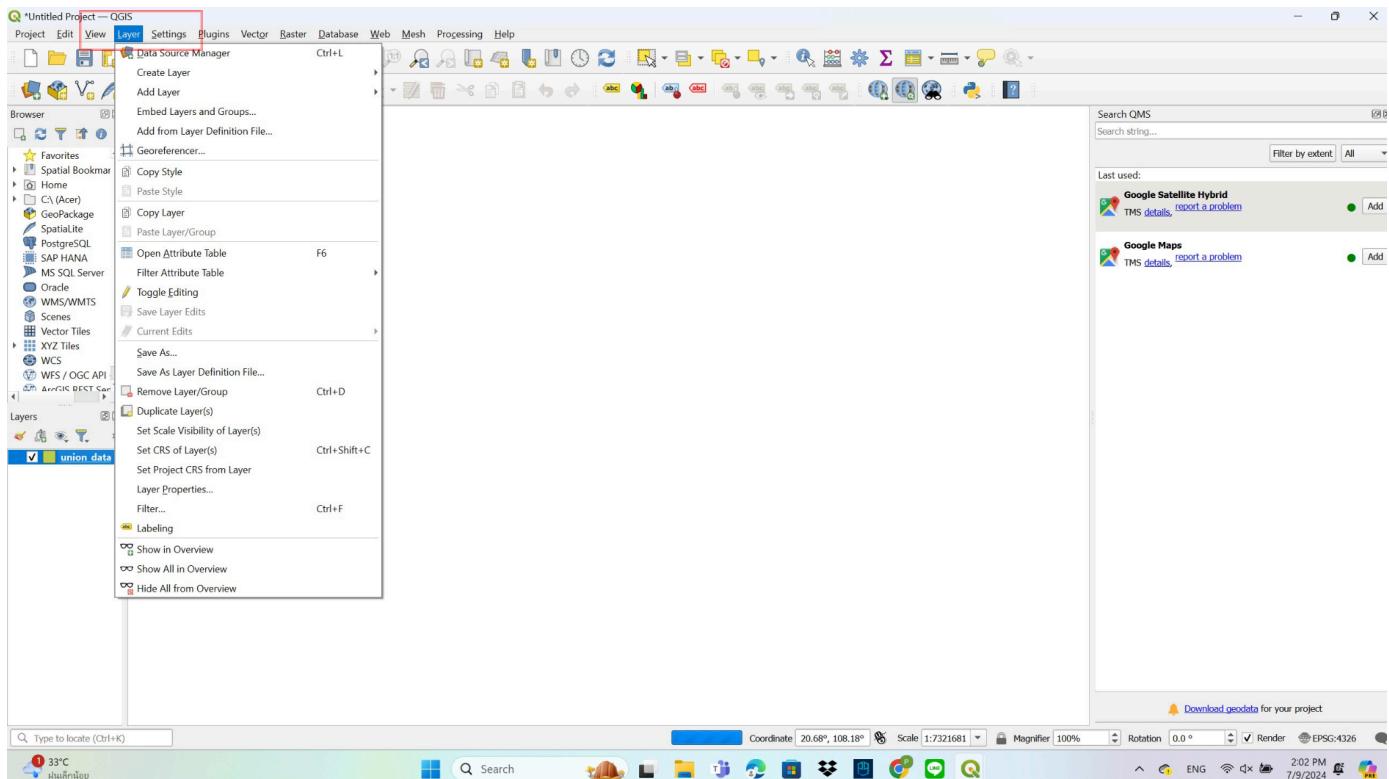
Step 16: Then, it will display the attribute as shown below. Click to the Attribute and Click OK.



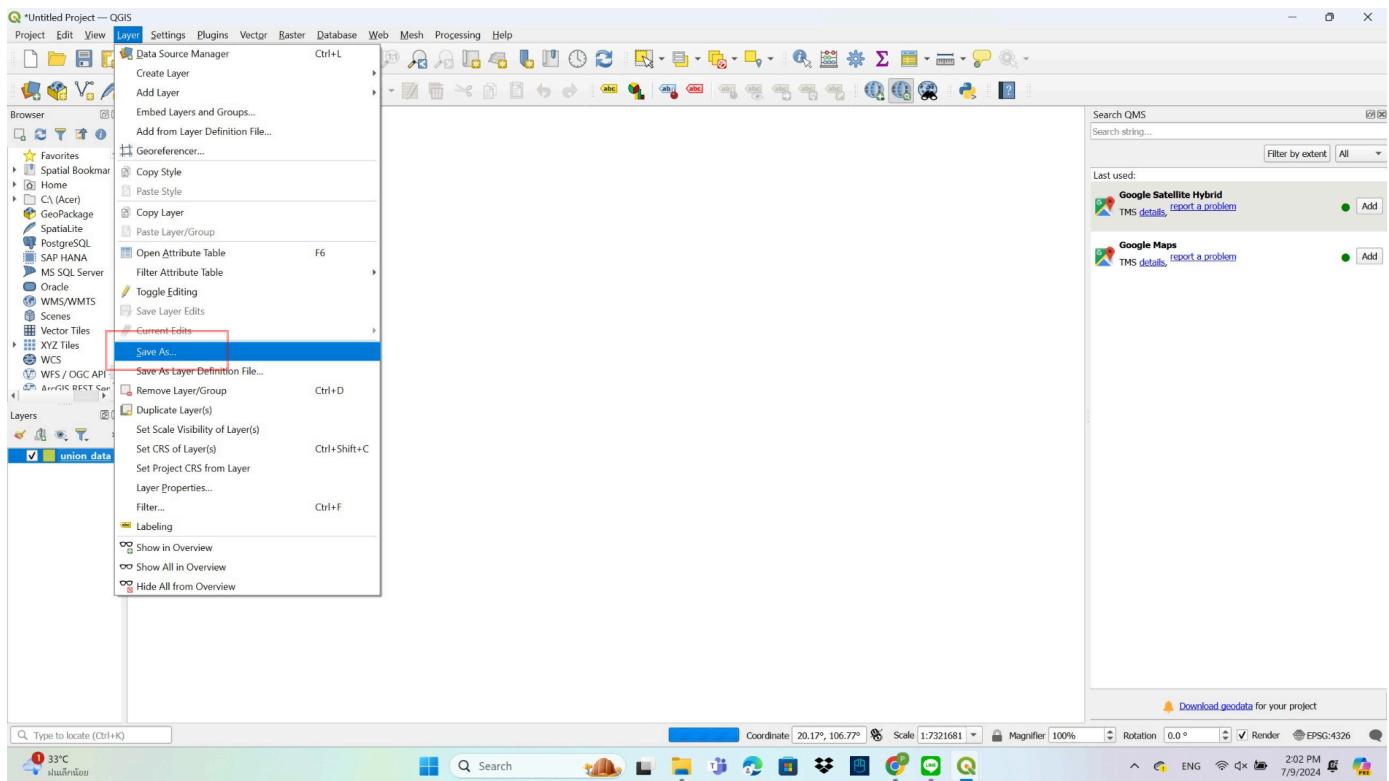
Step 17: After clicking the OK button, It will display the table that shows the result after the filter.

union_data — Features Total: 83923, Filtered: 83923, Selected: 0						X
	id	FID_1	Name_1	FID_2	Name_2	Plant_Status
7	6		3 Palm	61	Rubber	Attribute (P)
8	7		3 Palm	63	Rubber	Attribute (P)
9	8		3 Palm	64	Rubber	Attribute (P)
10	9		3 Palm	65	Rubber	Attribute (P)
11	10		3 Palm	66	Rubber	Attribute (P)
12	11		3 Palm	68	Rubber	Attribute (P)
13	12		3 Palm	69	Rubber	Attribute (P)
14	13		3 Palm	70	Rubber	Attribute (P)

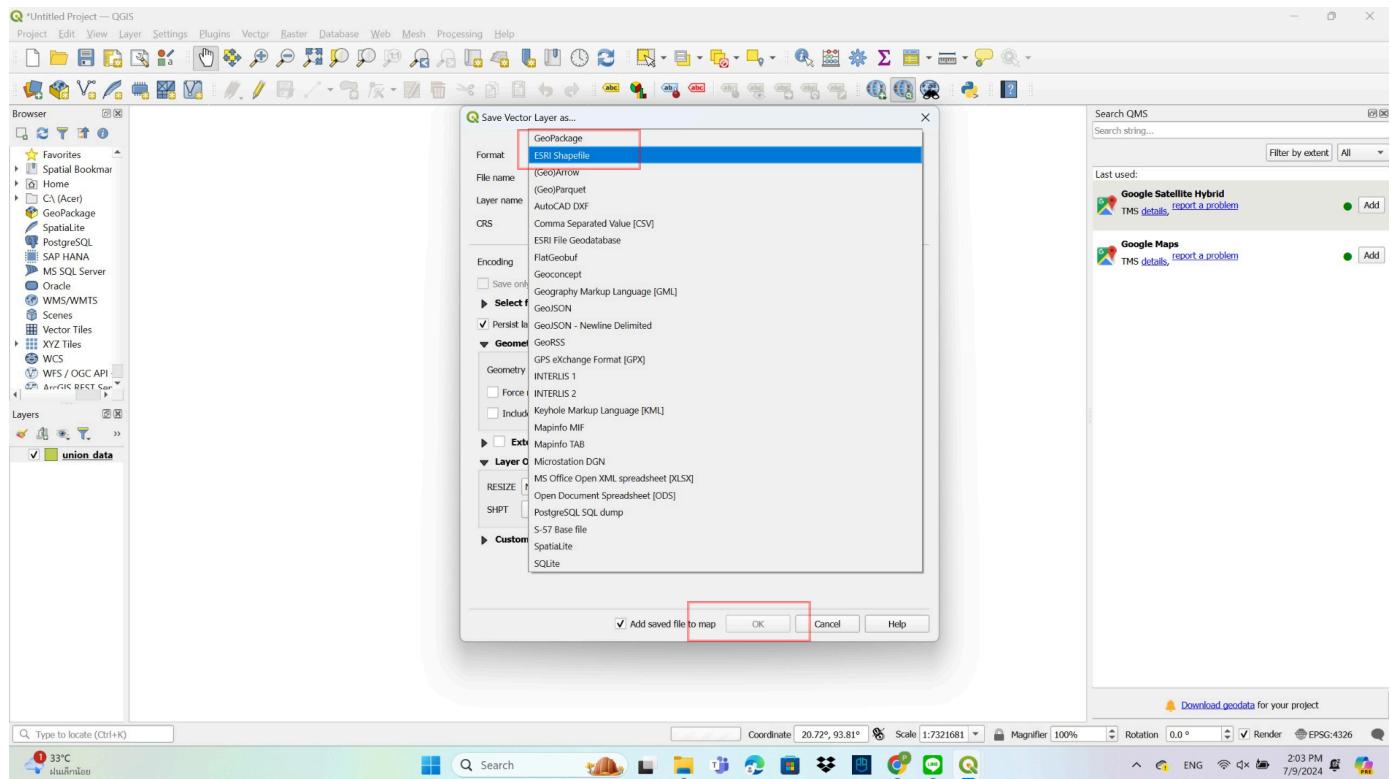
## Step 18: Click on the Layer button



## Step 19: Save as



Step 20: Choose the file type that you want to export and then click the OK button.



**Note: All of these steps are for four crop types. If you want to make Perennial Plants (Palm and Rubber) , you can do the following step as shown below.**

## Step 1: For the Four crop type , Find the file name “palm\_rubber.py”

The screenshot shows a VS Code interface with the title bar "intern\_gistda". The Explorer sidebar on the left lists files like index.py, index2.py, palm\_rubber.py (which is currently selected), index5.py, and 4\_plants.py. The main editor area contains the following Python code:

```
1 import geopandas as gpd
2 import folium
3 import pandas as pd
4 from tqdm import tqdm
5 from concurrent.futures import ThreadPoolExecutor, as_completed
6
7 # List of file paths to the GeoJSON files
8 files = [
9     "/Users/vnlight/Desktop/GeoJson/Maize_65.geojson",
10    "/Users/vnlight/Desktop/GeoJson/Palm_65.geojson",
11    "/Users/vnlight/Desktop/GeoJson/Sugarcane_65.geojson",
12    "/Users/vnlight/Desktop/GeoJson/Rice_in_65.geojson",
13    "/Users/vnlight/Desktop/GeoJson/Rice_out_65.geojson",
14    "/Users/vnlight/Desktop/GeoJson/Rubber_65.geojson"
15]
16
17 def read_and_process_file(file):
18     print("Reading file {file}...")
19     gdf = gpd.read_file(file)
20     print(f"File {file} read successfully!")
21     gdf['geometry'] = gdf['geometry'].simplify(tolerance=0.001)
22     gdf['geometry'] = gdf['geometry'].buffer(0.0001)
23     return gdf
24
25
26 # Read and process GeoJSON files concurrently.
```

The status bar at the bottom indicates "Ln 20, Col 30 Spaces: 4 UTF-8 LF Python 3.12.4 ('venv': venv)".

## Step 2: Replace file path with the file that you want.

```
8 # List of file paths to the GeoJSON files
9 files = [
10    "/Users/vnlight/Desktop/GeoJson/Palm_65.geojson",
11    "/Users/vnlight/Desktop/GeoJson/Rubber_65.geojson"
12]
```

In addition to this, you can follow all the steps above because for four types of crop and perennial plants (palm , rubber) , everything is the same. Things that need to be done is to change the file paths.