

“QGIS PLUGIN DEVELOPMENT TO CLIP TWO LAYERS WITH GEOMETRY”



HAMZA SADAQAT [18-ARID-3301]

TAYYAB HANIF [18-ARID-3320]

SHAHZAD SOHAIL [18-ARID-3329]

**FACULTY OF AGRICULTURAL ENGINEERING AND TECHNOLOGY
INSTITUTE OF GEO-INFORMATION & EARTH OBSERVATION**

**PIR MEHR ALI SHAH
ARID AGRICULTURE UNIVERSITY RAWALPINDI
PAKISTAN
2021**

DEDICATION

This work is dedicated to our respected teachers and my beloved friends.

CONTENTS

	Page
Acknowledgements	iv
Chapter 1 Introduction	1
1.1 background:-	1
1.2 Purpose:	1
1.3 PROBLEM STATEMENT/Limitation:	1
1.4 Pre-requisites:	1
1.5 QGIS Version:	2
1.6 Additional Software required:	2
Chapter 2 Tutorial	2
2.1 PROCEDURE:	2
3.1 RESULTS AND DICUSSION:	20
DEVELOPED BY:	21
DATA SOURCE:	21

Acknowledgements

First thing first, I am thankful to Allah who granted us courage and will to fulfill this task. Then, we want to mention our Teacher Sir Shahinsha, because of his motivation and skillful lectures we completed our project.

HAMZA SADAQAT ABBASI

SHAHZAD SOHAIL

TAYYAB HANIF

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND:-

QGIS has made provision to extend its functionality by writing Plugins in Python and C++. Plugins are categorized as core, stable and experimental. Several Plugins are quite extensively used such as “OpenLayers” and “Semi-Automatic Classification Plugin”. This tutorial will outline the process involved in setting up your development environment, designing the user interface for a plugin and writing code to interact with QGIS.

1.2 PURPOSE:

To build a clip tool which updates the geometry columns.

1.3 PROBLEM STATEMENT/LIMITATION:

Measurement of geometry in GIS is very important. In different spatial analysis for example snow cover assessment, flood damage assessment, cadastral mapping, overlay analysis or different other spatial/location based analysis, accurate geometry measurement is very important. So, the existing clip tool in QGIS does not update the geometry columns. So, we have to build a tool which updates geometry column of the clipped file as well. We shall try to clip Snow cover area of Swat with the district boundary of Swat, KPK, and notice the change in the geometry columns and then we will compare with the clipped file that will be created using our plugin.

1.4 PRE-REQUISITES:

Knowledge of GIS data is a must. Working knowledge of Python programming will be added advantage.

1.5 QGIS VERSION:

3.6.0-Noosa

1.6 ADDITIONAL SOFTWARE REQUIRED:

- (1) Qt Designer (comes bundled with the standard QGIS installation)
- (2) QGIS Plugin - Plugin Builder,
- (3) QGIS Plugin - Reloader plugin and
- (4) a beautiful editor - Sublime Text 2 or Notepad ++ or Python shell

CHAPTER 2

TUTORIAL

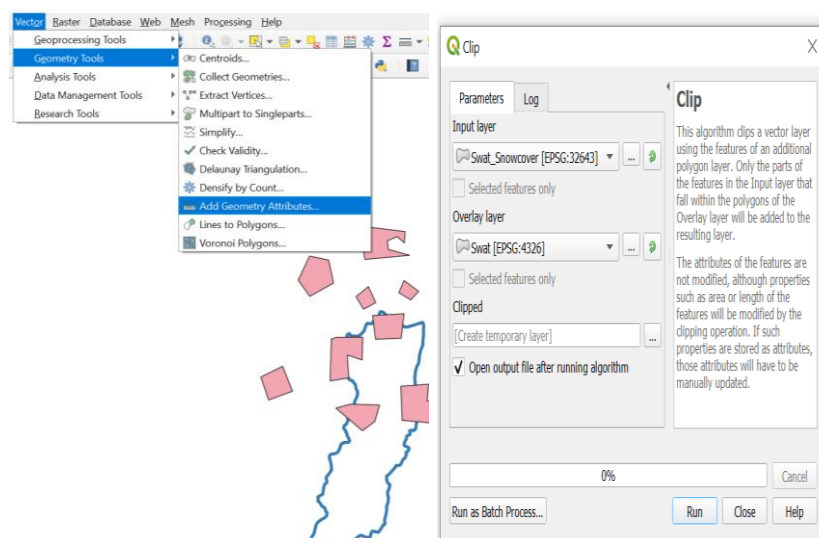
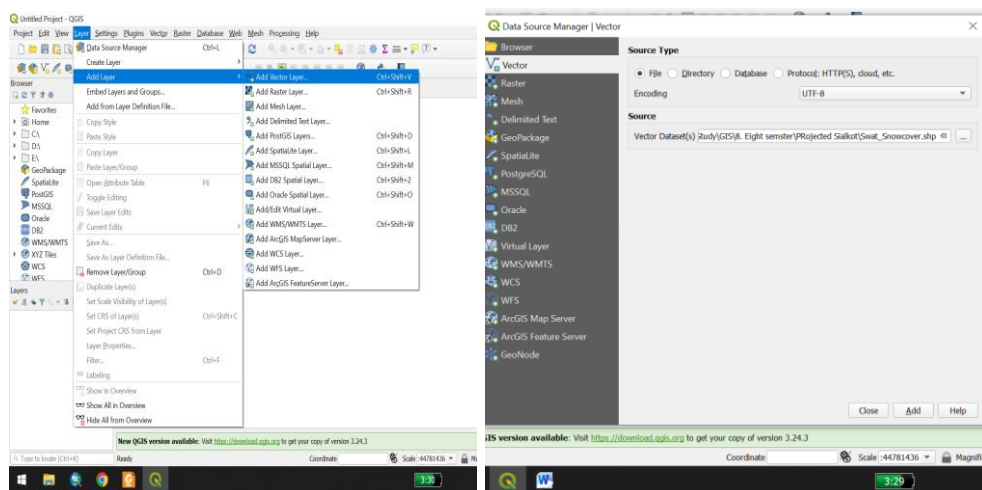
2.1 PROCEDURE:

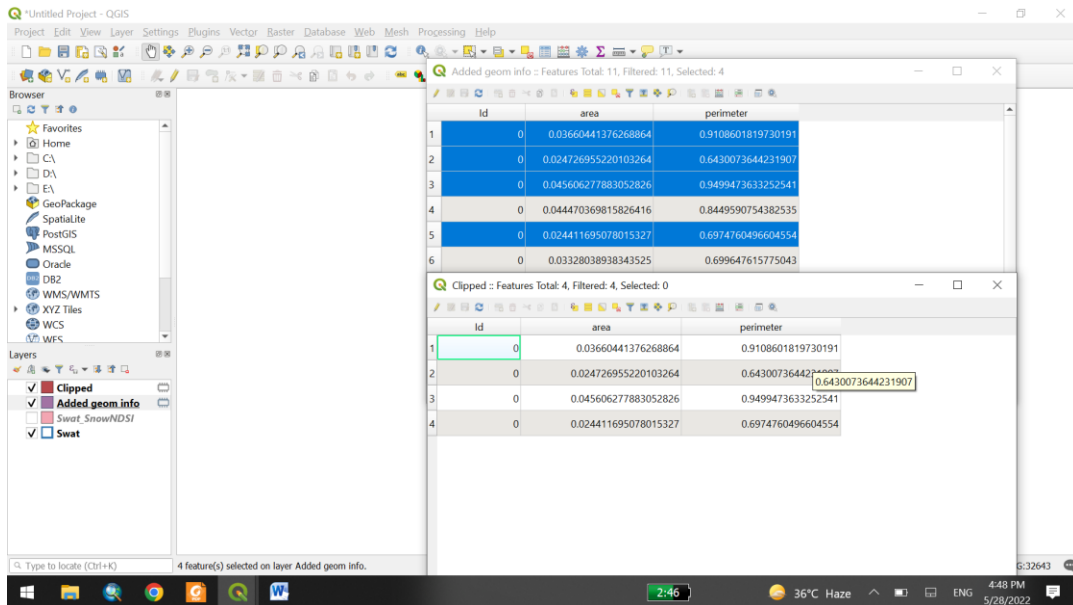
For the ease of understanding, both the layers used are in the UTM 43N / WGS 84 “projected” coordinate system. The map units are in meters

1. Add 2 overlapping vector layers. Click on Layer - Add Layer - Add Vector Layer. Navigate to the folder shared with you, press Ctrl and select the two files “Swat_snowcover.shp” and “Swat.shp”
2. Click on Vector - Geometry tools - Add / Export Geometry columns. Select “Swat_snowcover.shp” in the drop and click Ok. Close the box, after process is over.
3. Now right click on the new layer “Added geom info”, click on Open Attribute Table. Note the values of AREA and PERIMETER columns (esp., for features

with POLY_ID). These are the area and perimeter values before the clip operation.

- Now perform the clip operation and save the output file as “clip before”. Open the Attribute table of the “clip before.shp” file and compare the area and perimeter values noted down in the previous step. You will notice that although the features have changed, the area and perimeter values have not been updated.

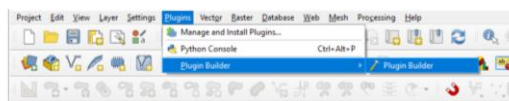




- Now we shall try to create a tool to clip Snow cover area of Swat with the district boundary of Swat, KPK and update the geometry columns also.

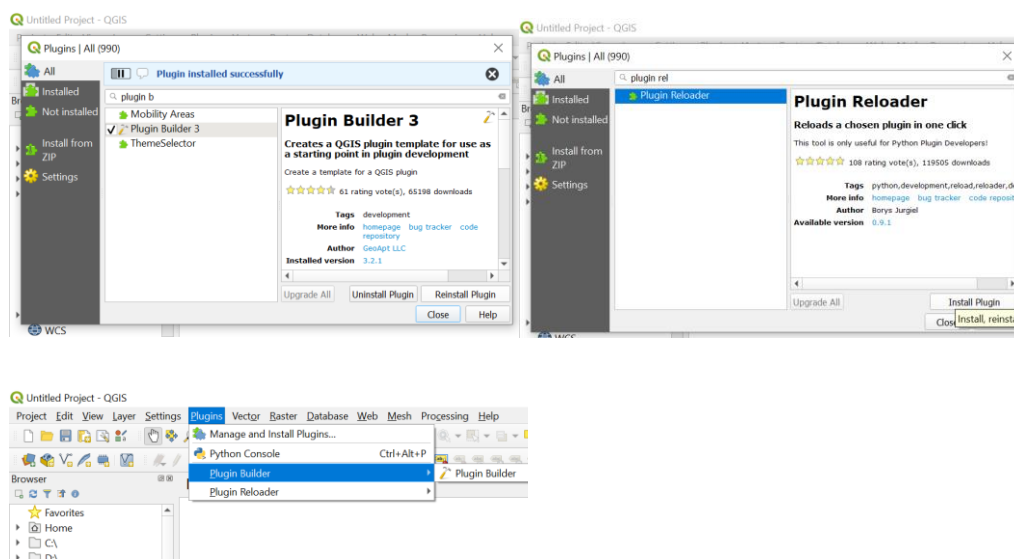
Procedure:

- Open QGIS. Go to *Plugins - Plugin Builder - Plugin Builder*

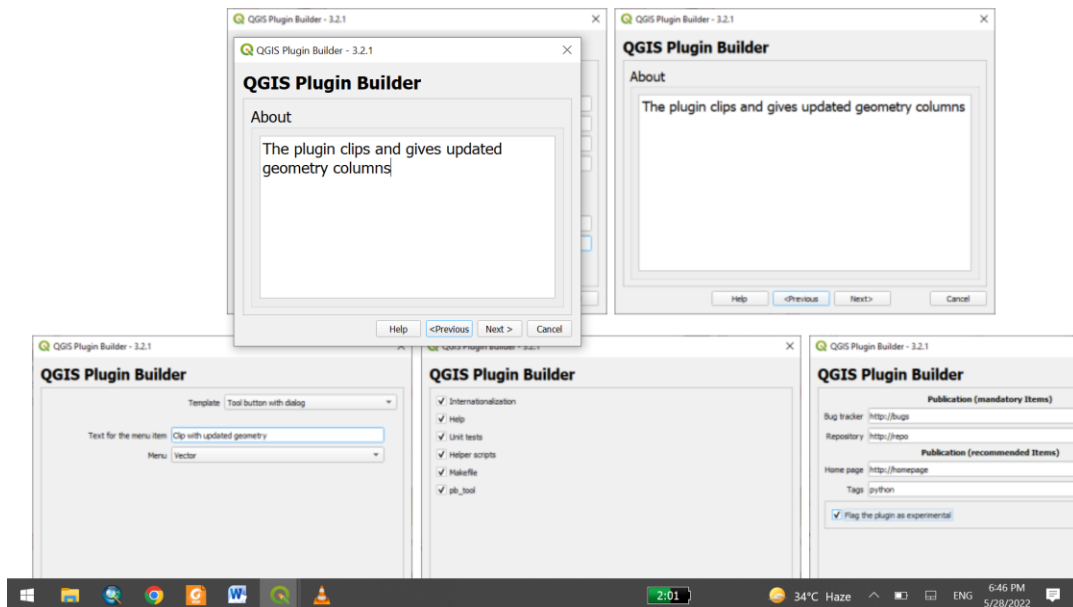


- You will see the QGIS Plugin Builder dialog with a form. You can fill the form with details relating to our plugin. The Class name will be the name of the Python Class containing the logic of the plugin. This will also be the name of the folder containing all the plugin files. Enter ClipWithGeometry as the class name. The Plugin name is the name under which your plugin will appear in the Plugin Manager. Enter the name as "Clip With Geometry". Add a description in the Description field. The Module name will be the name of the main python file for

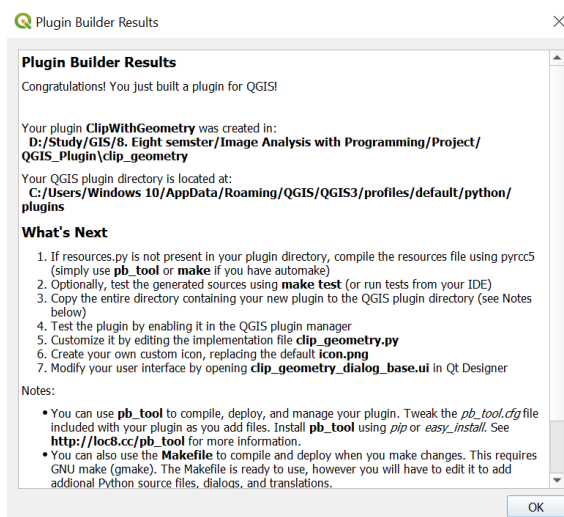
the plugin. Enter it as clip_geometry. Leave the version numbers as they are. Enter your name and email address in the appropriate fields. Click on Next. Enter some text in the “About” box and click on Next. Select the “Tool button with dialog” from the Template selector. The Text for menu item value will be how the users will find your plugin in QGIS menu. Enter it as “Clip with Update Geometry”. The Menu field will decide where your plugin item is added in QGIS. Since our plugin is for vector data, select Vector. Click on Next. Plugin builder will prompt you for the type of files to generate. Keep the default selection and click Next. Check the Flag the plugin as experimental box at the bottom. Click Next

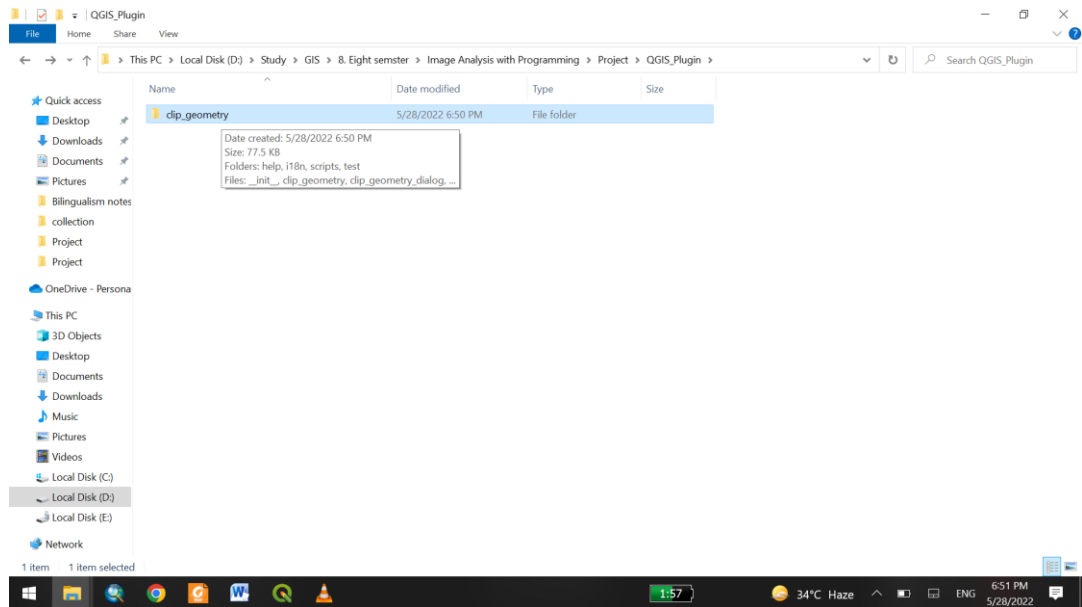


- Next, you will be prompted to choose a directory for your plugin. You need to browse to the QGIS python plugin directory on your computer and select Select Folder. Provide the path of today's folder: (Replace username with your login name). Click on Generate. You may get a warning saying pyrcc5 is not found in the path. You can ignore the message.



4. You will see a confirmation dialog once your plugin template is created. Note the path to the plugin folder





5. Before we can use the newly created plugin, we need to compile the resources.qrc file that was created by Plugin Builder. Launch the OSGeo4W Shell on windows start menu or running the file C:\Program Files\QGIS 3.6\OSGeo4W.bat.

The screenshot shows an 'OSGeo4W Shell' terminal window. The text inside the terminal is as follows:

```
run o-help for a list of available commands
C:\>C:\Users\Windows 10\Desktop\clip_geometry
'C:\Users\Windows 10\Desktop\clip_geometry' is not recognized as an internal or external command,
operable program or batch file.

C:\>cd Users\Windows 10\Desktop\clip_geometry
C:\Users\Windows 10\Desktop\clip_geometry>echo off
call "C:\Program Files\QGIS 3.6\bin\qt5_env.bat"
call "C:\Program Files\QGIS 3.6\bin\qt5_env.bat"
call "C:\Program Files\QGIS 3.6\bin\py3_env.bat"
```

 At the bottom of the terminal window, a command is being typed: `pyrcc5 -o resources.py resources.qrc`.

6. Once you are in the directory, type the following commands one by one. This will run the pyrcc4 command that we had installed as part of Qt bindings for Python

```
OSGeo4W Shell
run o-help for a list of available commands
C:\>C:\Users\Windows 10\Desktop\clip_geometry
'C:\Users\Windows 10\Desktop\clip_geometry' is not recognized as an internal or external command,
operable program or batch file.

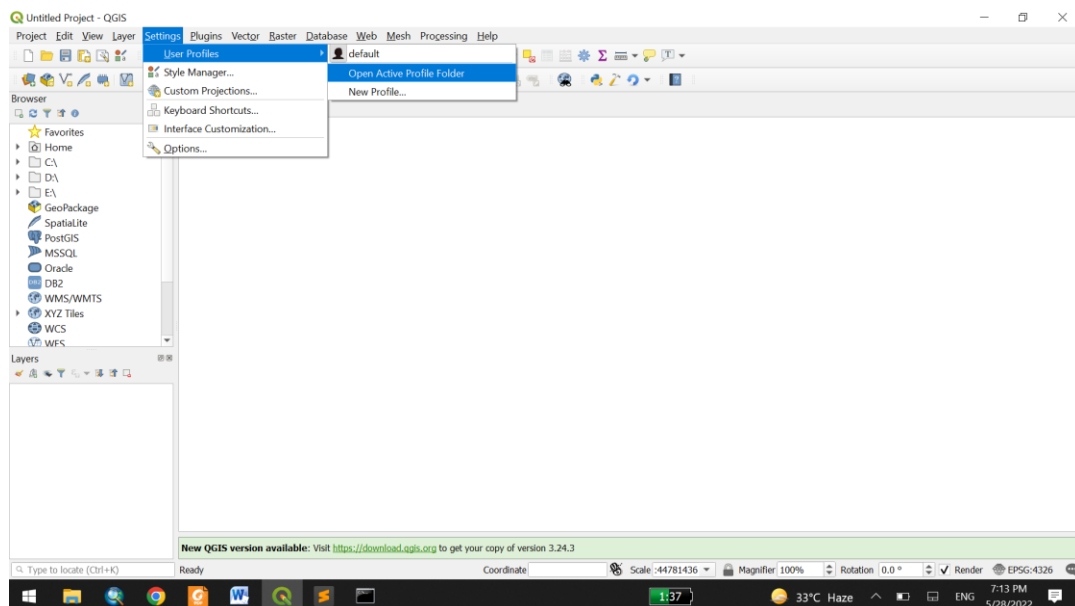
C:\>cd Users\Windows 10\Desktop\clip_geometry

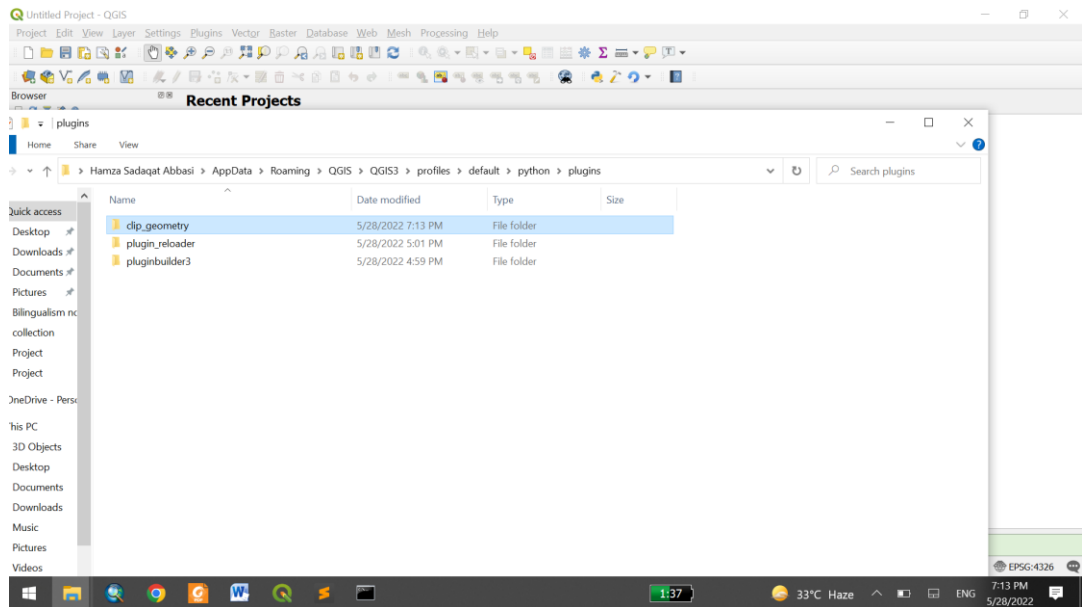
C:\Users\Windows 10\Desktop\clip_geometry>@echo off
call "C:\Program Files\QGIS 3.6\bin\o4w_env.bat"
call "C:\Program Files\QGIS 3.6\bin\qt5_env.bat"
call "C:\Program Files\QGIS 3.6\bin\py3_env.bat"
@echo on

C:\Users\Windows 10\Desktop\clip_geometry>pyrcc5 -o resources.py resources.qrc

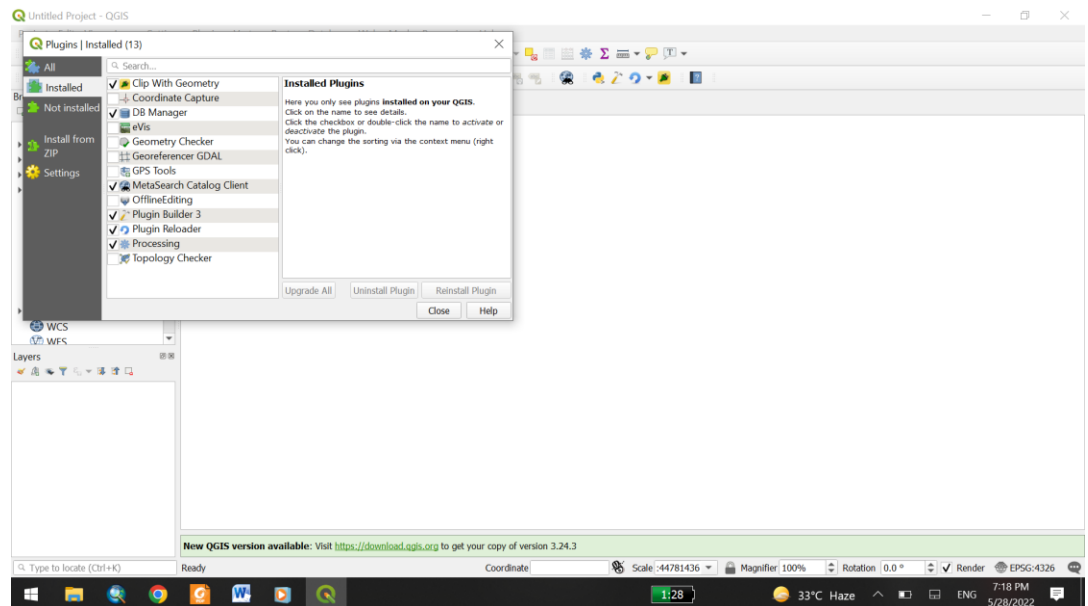
C:\Users\Windows 10\Desktop\clip_geometry>
```

7. Plugins in QGIS are stored in a special folder. We must copy our plugin directory to that folder before it can be used. In QGIS, locate your current profile folder by going to Settings ▸ User Profiles ▸ Open Active Profile Folder. In the profile folder, copy the plugin folder to python ▸ plugins subfolder

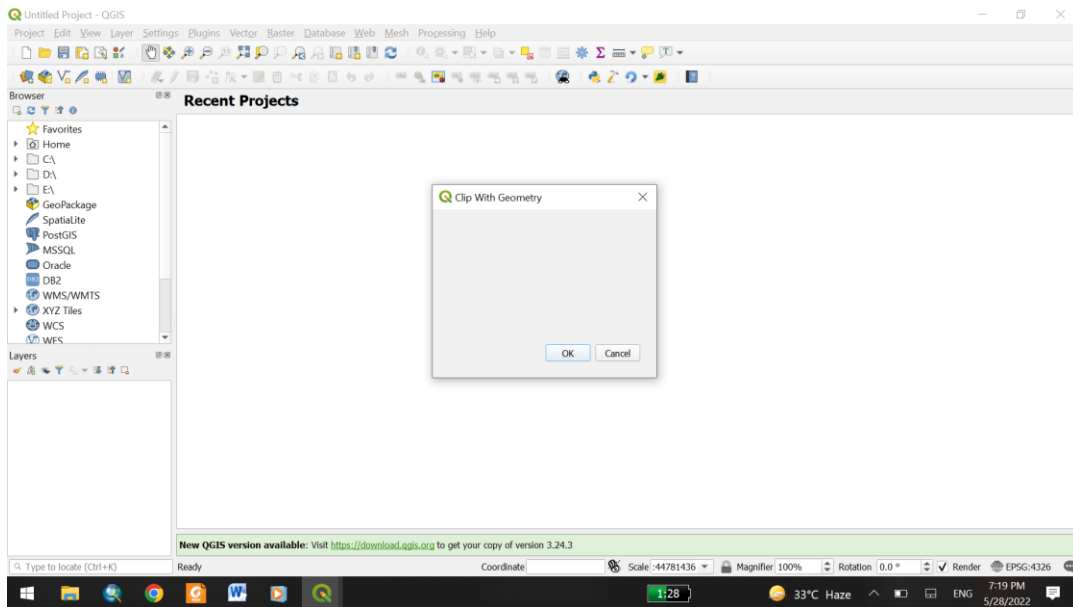




8. Now we are ready to have a first look at the brand new plugin we created. Close QGIS and launch it again. Go to Plugins - Manage and Install plugins and enable the “Clip With Geometry” plugin in the Installed tab. You will notice that there is a new icon in the toolbar and a new menu entry under Vector - Clip With Geometry - Clip with Update Geometry. Select it to launch the plugin dialog.



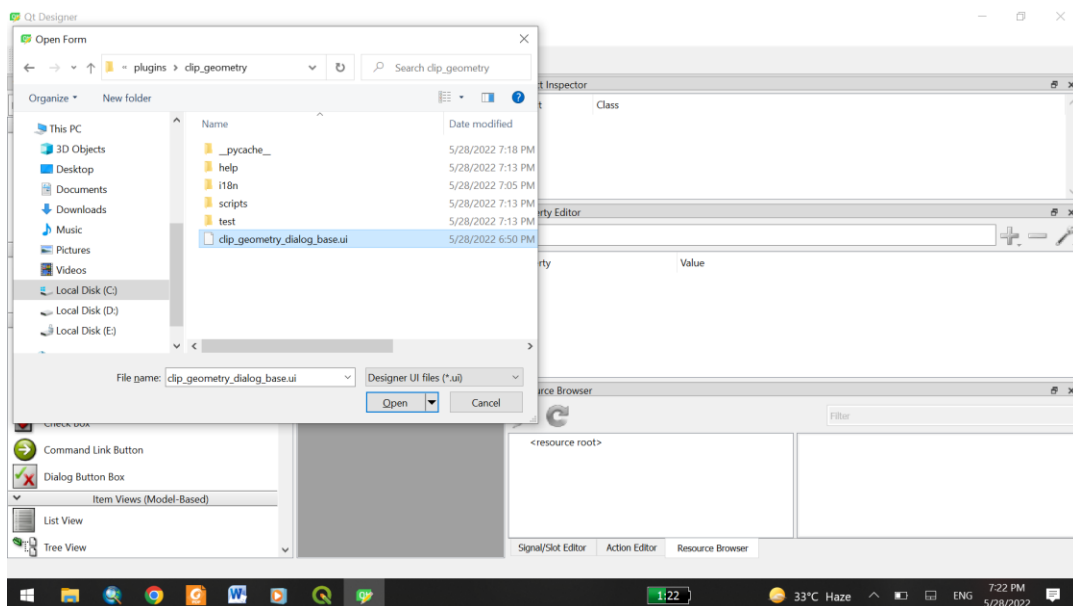
9. You will notice a new blank dialog named Clip With Geometry. Close this dialog



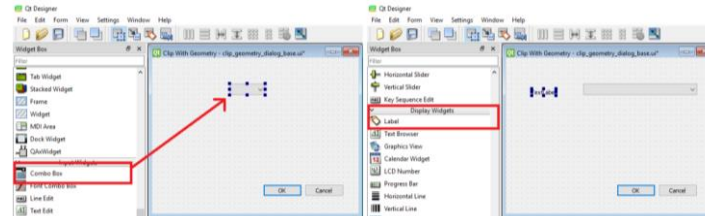
10. We will now design our dialog box and add some user interface elements to it.

Open the QtDesigner program and to to File -> Open File or Project....

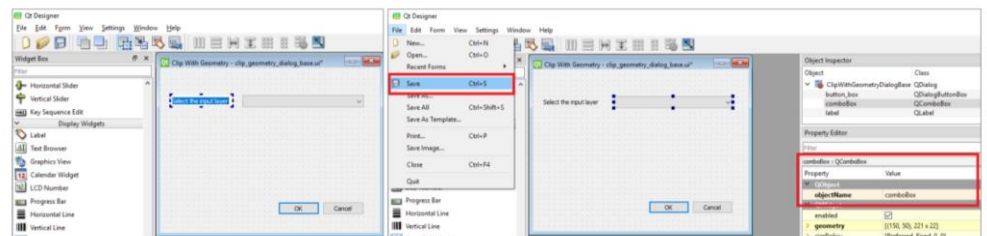
11. Browse to the plugin directory (in my case it is “C:\Users\prasun\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\clip_geometry”) and select the clip_geometry_dialog_base.ui file. Click Open



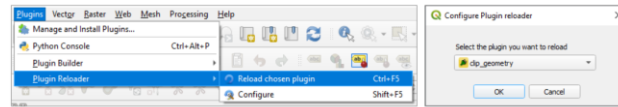
11. You will see the blank dialog from the plugin. You can drag-and-drop elements from the left-hand panel on the dialog. We will add a Combo Box type of Input Widget. Drag it to the plugin dialog



12. Resize the combo box and adjust its size. Now drag a Label type Display Widget on the dialog.
13. Click on the label text and enter “Select the input layer”.



14. Save this file by going to File - Save “clip_geometry_dialog_base.ui”. Note the name of the combo box object is comboBox. To interact with this object using python code, we will have to refer to it by this name.
15. Let’s reload our plugin so we can see the changes in the dialog window. Go to Plugin – Plugin Reloader - Choose a plugin to be reloaded.
16. Select ClipWithGeometry in the Configure Plugin reloader dialog
17. Now click the button. You will see the newly designed dialog box.



18. Let's add some logic to the plugin that will populate the combo box with the layers loaded in QGIS. Go to the plugin directory and load the file `clip_geometry.py` in a text editor. First, insert at the top of the file with the other imports

 The image shows a Sublime Text editor window with the file `clip_geometry.py` open. The file path in the title bar is `C:\Users\Windows 10\Desktop\clip_geometry\clip_geometry.py - Sublime Text (UNREGISTERED)`. The editor has a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a toolbar. The file content is as follows:


```

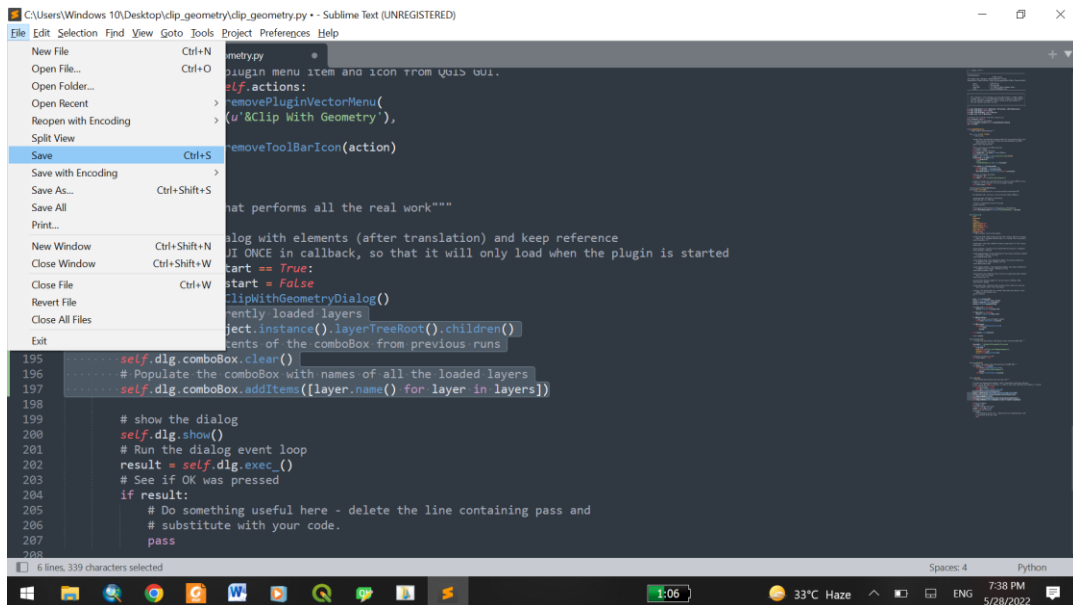
1  -*- coding: utf-8 -*-
2
3  /*****
4  ClipWithGeometry
5
6  A QGIS plugin
7  The plugin clips and gives updated geometry columns
8  Generated by Plugin Builder; http://g-sherman.github.io/Qgis-Plugin-Builder/
9
10     begin                : 2022-05-28
11     git sha               : $Format:%H$
12     copyright             : (C) 2022 by Hamza Sadaqat Abbasi
13     email                 : hamzasdt786@gmail.com
14
15     *****/
16
17  * This program is free software; you can redistribute it and/or modify *
18  * it under the terms of the GNU General Public License as published by *
19  * the Free Software Foundation; either version 2 of the License, or *
20  * (at your option) any later version. *
21  *
22  *****/
23
24  from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
25  from qgis.PyQt.QtGui import QIcon
26  from qgis.PyQt.QtWidgets import QAction
27
28  # Initialize Qt resources from file resources.py
29  from .resources import *
30  # Import the code for the dialog
31  from .clip_geometry_dialog import ClipWithGeometryDialog
32  import os.path
  
```

 The status bar at the bottom shows 'Line 1, Column 1: Detect Indentation: Setting indentation to 4 spaces', 'Spaces: 4', 'Python', and system information like '1:12', '33°C Haze', and '7:32 PM 5/28/2022'.


```
15 /
16 *
17 * This program is free software; you can redistribute it and/or modify
18 * it under the terms of the GNU General Public License as published by
19 * the Free Software Foundation; either version 2 of the License, or
20 * (at your option) any later version.
21 *
22 *****/
23
24 from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
25 from qgis.PyQt.QtGui import QIcon
26 from qgis.PyQt.QtWidgets import QDialog, QVBoxLayout, QComboBox, QPushButton
27 from qgis.core import QgsProject
28
29 # Initialize Qt resources from file resources.py
30 from .resources import *
31 # Import the code for the dialog
32 from .clip_geometry_dialog import ClipWithGeometryDialog
33 import os.path
34
35
36 class ClipWithGeometry:
37     """QGIS Plugin Implementation."""
38
39     def __init__(self, iface):
40         """Constructor.
41
42         :param iface: An interface instance that will be passed to this class
43                       which provides the hook by which you can manipulate the QGIS
44                       application at run time.
45         :type iface: QgsInterface
46         """
47         # Save reference to the QGIS interface
```

19. Scroll down and find the run(self) method. This method will be called when you click the toolbar button or select the plugin menu item. Add the following code at the beginning of the method. This code gets the layers loaded in QGIS and adds it to the comboBox object from the plugin dialog. Save the file

```
176 removes the plugin menu item and icon from QGIS GUI.
177 for action in self.actions:
178     self.iface.removePluginVectorMenu(
179         self.tr(u'Clip With Geometry'),
180         action)
181     self.iface.removeToolBarIcon(action)
182
183
184 def run(self):
185     """Run method that performs all the real work"""
186
187     # Create the dialog with elements (after translation) and keep reference
188     # Only create GUI ONCE in callback, so that it will only load when the plugin is started
189     if self.first_start == True:
190         self.first_start = False
191         self.dlg = ClipWithGeometryDialog()
192         # Fetch the currently loaded layers
193         layers = QgsProject.instance().layerTreeRoot().children()
194         # Clear the contents of the comboBox from previous runs
195         self.dlg.comboBox.clear()
196         # Populate the comboBox with names of all the loaded layers
197         self.dlg.comboBox.addItem([layer.name() for layer in layers])
198
199     # show the dialog
200     self.dlg.show()
201     # Run the dialog event loop
202     result = self.dlg.exec_()
203     # See if OK was pressed
204     if result:
205         # Do something useful here - delete the line containing pass and
206         # substitute with your code.
207         pass
208
```

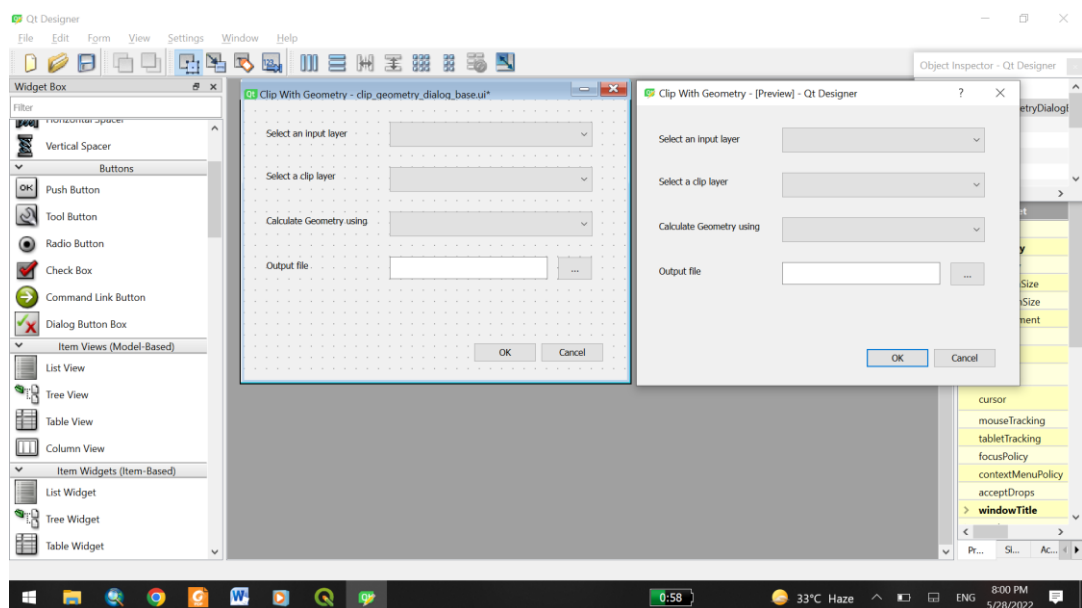


20. . Back in the main QGIS window, reload the plugin by going to Plugins – Plugin Reloader – Reload plugin: ClipWithGeometry. Alternatively, you can just press Ctrl+F5
21. To test this new functionality, we must load some layers in QGIS. Add the two layers provided (Swat_Snowcover and Swat).
22. After you load the data, launch the plugin by going to Vector - Clip With Geometry - Clip with Updated Geometry. You will see that our combo box is now populated with the layer names that are loaded in QGIS

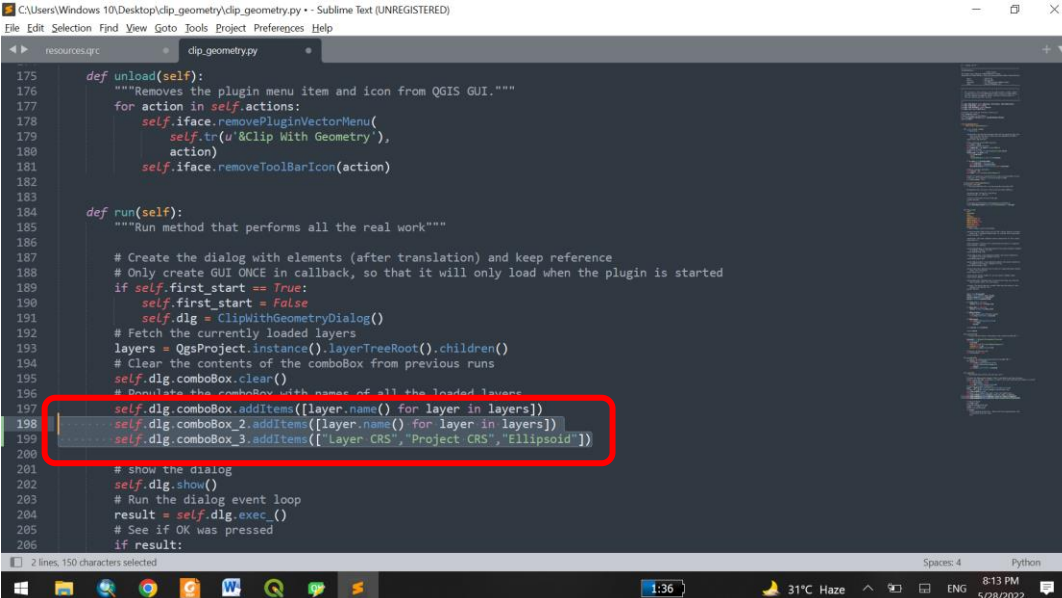


23. Let's add remaining user interface elements. Switch back to Qt Creator and load theclip_geometry.ui file.

- Add a Label *Display Widget* and change the text to “Select the clip layer”.
 - Add another Combo Box type *Input Widget* that will show the clip layer (objectName =comboBox_2).
 - Add a Label *Display Widget* and change the text to “Calculate Geometry using”.
 - Add another Combo Box type *Input Widget* that will show the method used to calculate the Geometry values (objectName = comboBox_3).
 - Add a Label Display Widget and change the text to “Select the output file”.
 - Add a LineEdit type *Input Widget* that will show the output file path that the user has chosen (objectName = lineEdit).
 - Next, add a Push Button type *Button* and change the button label to “...”
- Note the object names of the widgets that we will have to use to interact with them (objectName = pushButton). Save the file

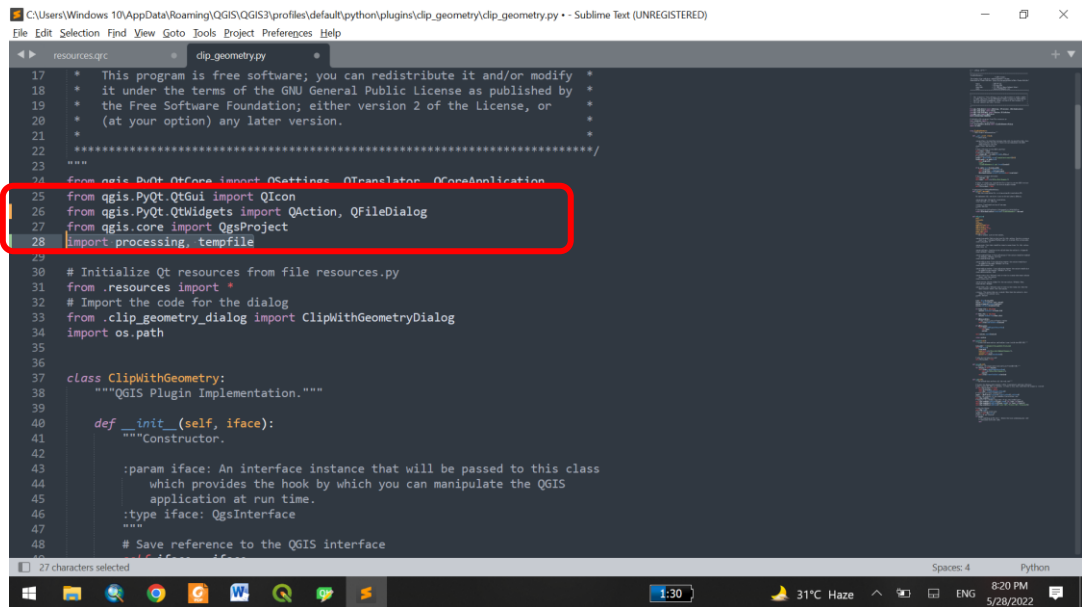


24. Open the clip_geometry.py file in a text editor. Scroll down and find the run(self) method. Add the following code lines as shown. The 1st line gets the layers loaded in QGIS and adds it to the comboBox_2 object from the plugin dialog; and the 2nd line indicates the options available to the user to Calculate Geometry value



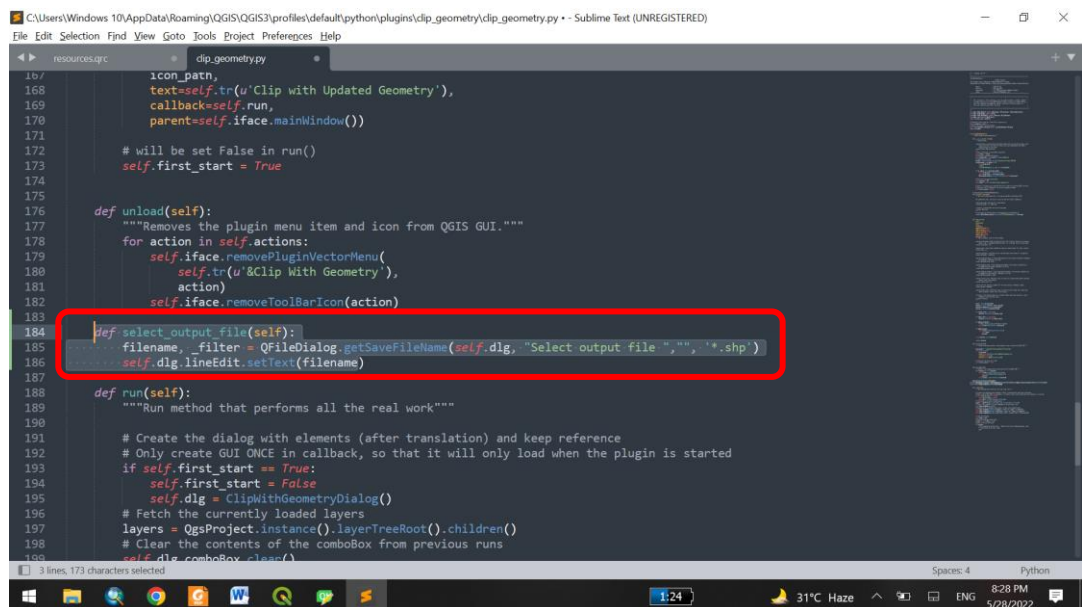
```
175 def unload(self):
176     """Removes the plugin menu item and icon from QGIS GUI."""
177     for action in self.actions:
178         self.iface.removePluginVectorMenu(
179             self.tr(u'Clip With Geometry'),
180             action)
181         self.iface.removeToolBarIcon(action)
182
183
184 def run(self):
185     """Run method that performs all the real work"""
186
187     # Create the dialog with elements (after translation) and keep reference
188     # Only create GUI ONCE in callback, so that it will only load when the plugin is started
189     if self.first_start == True:
190         self.first_start = False
191         self.dlg = ClipWithGeometryDialog()
192     # Fetch the currently loaded layers
193     layers = QgsProject.instance().layerTreeRoot().children()
194     # Clear the contents of the comboBox from previous runs
195     self.dlg.comboBox.clear()
196     # Populate the comboBox with names of all the loaded layers
197     self.dlg.comboBox.addItem(layer.name() for layer in layers)
198     self.dlg.comboBox_2.addItem(layer.name() for layer in layers)
199     self.dlg.comboBox_3.addItem('Layer CRS', 'Project CRS', 'Ellipsoid')
200
201     # show the dialog
202     self.dlg.show()
203     # Run the dialog event loop
204     result = self.dlg.exec_()
205     # See if OK was pressed
206     if result:
```

25. We will now add python code to open a file browser when the user clicks the “...” push button and show the select path in the line edit widget. Add QFileDialog, processing and tempfile to our list of imports at the top of the clip_geometry.py file



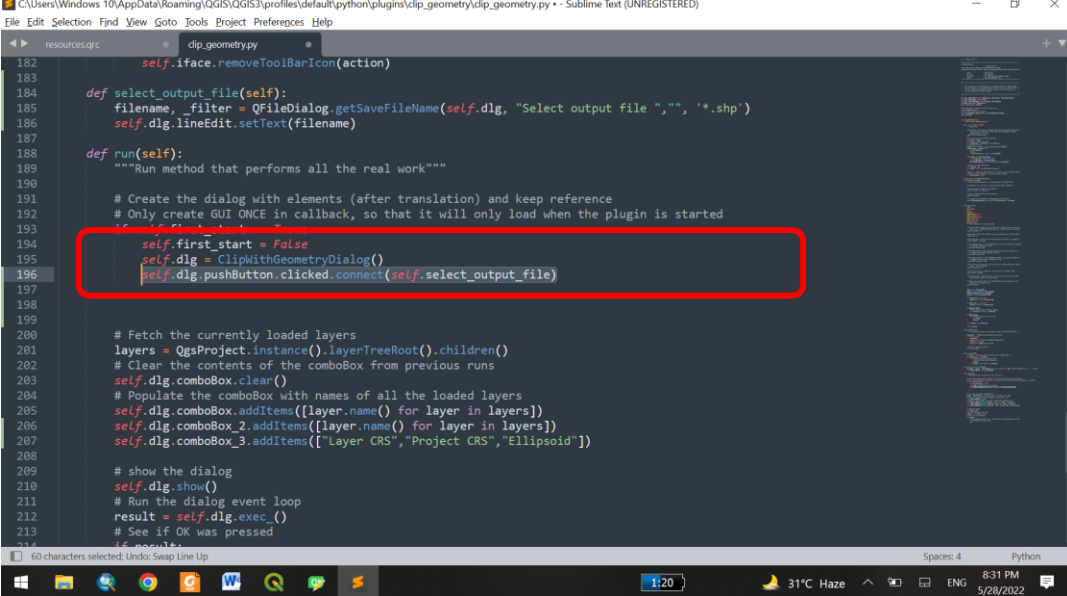
```
17 * This program is free software; you can redistribute it and/or modify *
18 * it under the terms of the GNU General Public License as published by *
19 * the Free Software Foundation; either version 2 of the License, or *
20 * (at your option) any later version. *
21 *
22 *****/
23
24 from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
25 from qgis.PyQt.QtGui import QIcon
26 from qgis.PyQt.QtWidgets import QAction, QFileDialog
27 from qgis.core import QgsProject
28 import processing, tempfile
29
30 # Initialize Qt resources from file resources.py
31 from .resources import *
32 # Import the code for the dialog
33 from .clip_geometry_dialog import ClipWithGeometryDialog
34 import os.path
35
36
37 class ClipWithGeometry:
38     """QGIS Plugin Implementation."""
39
40     def __init__(self, iface):
41         """Constructor.
42
43         :param iface: An interface instance that will be passed to this class
44                       which provides the hook by which you can manipulate the QGIS
45                       application at run time.
46         :type iface: QgsInterface
47         """
48         # Save reference to the QGIS interface
```

26. Add a new method called `select_output_file` with the following code. This code will open a file browser and populate the line edit widget with the path of the file that the user chose



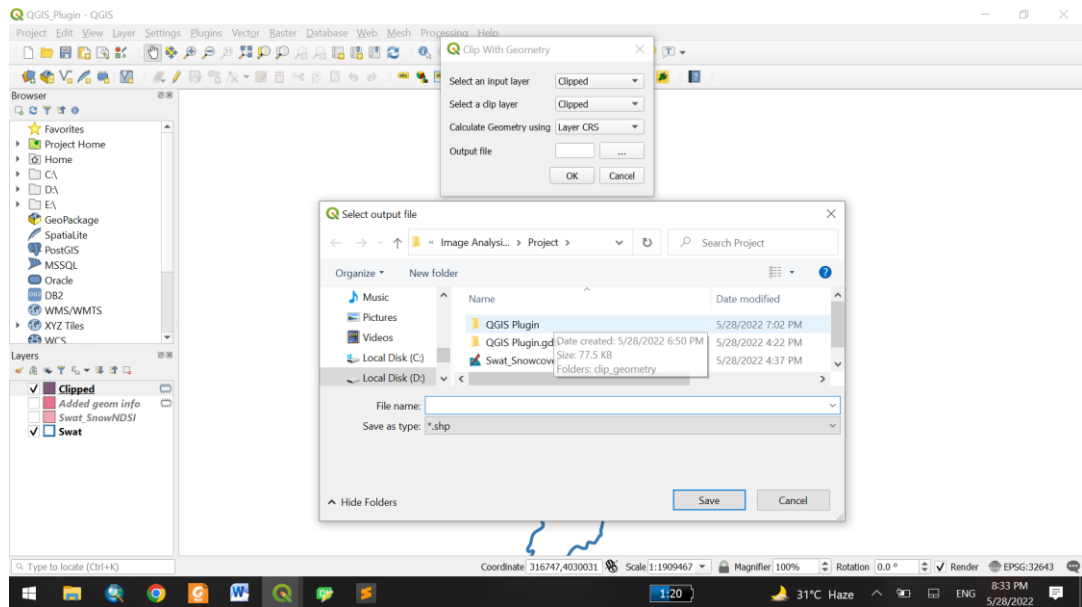
```
167         icon_path,
168         text=self.tr('Clip with Updated Geometry'),
169         callback=self.run,
170         parent=self.iface.mainWindow())
171
172     # Will be set False in run()
173     self.first_start = True
174
175
176     def unload(self):
177         """Removes the plugin menu item and icon from QGIS GUI."""
178         for action in self.actions:
179             self.iface.removePluginVectorMenu(
180                 self.tr(u'Clip With Geometry'),
181                 action)
182             self.iface.removeToolBarIcon(action)
183
184     def select_output_file(self):
185         filename, _filter = QFileDialog.getSaveFileName(self.dlg, "Select output file", "", "*.shp")
186         self.dlg.lineEdit.setText(filename)
187
188     def run(self):
189         """Run method that performs all the real work"""
190
191         # Create the dialog with elements (after translation) and keep reference
192         # Only create GUI ONCE in callback, so that it will only load when the plugin is started
193         if self.first_start == True:
194             self.first_start = False
195             self.dlg = ClipWithGeometryDialog()
196         # Fetch the currently loaded layers
197         layers = QgsProject.instance().layerTreeRoot().children()
198         # Clear the contents of the comboBox from previous runs
199         self.dlg.comboBox.clear()
```

27. Now we need to add code so that when the ... button is clicked, select_output_file method is called. Scroll down to the run method and add the following line in the block where the dialog is initialized. This code will connect the select_output_file method to the clicked signal of the push button widget

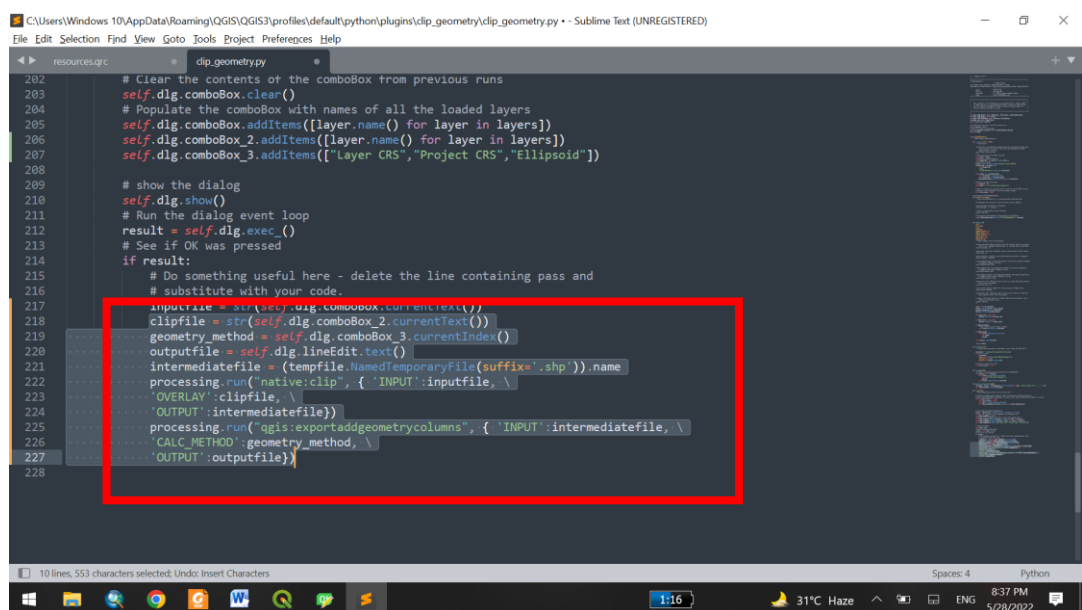


```
182 self iface.removeToolBarIcon(action)
183
184 def select_output_file(self):
185     filename, _filter = QFileDialog.getSaveFileName(self.dlg, "Select output file", "", "*.shp")
186     self.dlg.lineEdit.setText(filename)
187
188 def run(self):
189     """Run method that performs all the real work"""
190
191     # Create the dialog with elements (after translation) and keep reference
192     # Only create GUI ONCE in callback, so that it will only load when the plugin is started
193
194     self.first_start = False
195     self.dlg = ClipWithGeometryDialog()
196     self.dlg.pushButton.clicked.connect(self.select_output_file)
197
198
199
200 # Fetch the currently loaded layers
201 layers = QgsProject.instance().layerTreeRoot().children()
202 # Clear the contents of the comboBox from previous runs
203 self.dlg.comboBox.clear()
204 # Populate the comboBox with names of all the loaded layers
205 self.dlg.comboBox.addItem(layer.name() for layer in layers)
206 self.dlg.comboBox_2.addItem(layer.name() for layer in layers)
207 self.dlg.comboBox_3.addItem(["Layer CRS", "Project CRS", "Ellipsoid"])
208
209 # show the dialog
210 self.dlg.show()
211 # Run the dialog event loop
212 result = self.dlg.exec_()
213 # See if OK was pressed
214 if result:
```

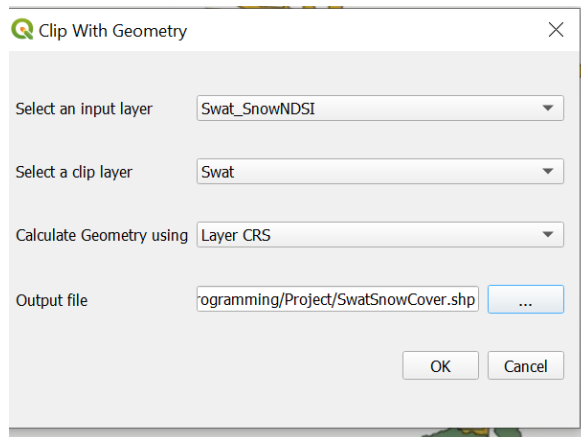
28. Back in QGIS, reload the plugin and open the Clip With Geometry dialog. If all went fine, you will be able to click the ... button and select an output text file from your disk



27. When you click OK on the plugin dialog, nothing happens. That is because we have not added the logic to pull attribute information from the layer and write it to the text file. We now have all the pieces in place to do just that. Find the place in the run method where it says pass. Replace it with the code below.



29. Now our plugin is ready. Reload the plugin and try it out. You will find that the output shape file will have the updated geometry columns after the clip operation.



3.1 RESULTS AND DICUSSION:

Here is the comparison of both shapefiles. On the right, there is attribute table of file that is generated using clip tool of the QGIS. While, on the left side this shapefile is clipped using GetGeometryPlugin. Now, we can clip updated geometry using this plugin.

Id	area	perimeter
1	0.020418065741296	0.762552009003600
2	0.017846625826678	0.628886541322613
3	0.018914955706578	0.571518575978321
4	0.019022871168090	0.589816704750202

Id	area	perimeter
1	0.036604413762689	0.910860181973019
2	0.045606277883053	0.949947363325254
3	0.024726955220103	0.643007364423191
4	0.024411695078015	0.697476049660455

DEVELOPED BY:

Hamza Sadaqat Abbasi & Tayyab Hanif & Shahzad Sohail, Institute of Geoinformatics and Earth Observation, PMAS-AAUR (hamzasdt786@gmail.com)

DATA SOURCES:

I have uploaded Plugin and tutorial on my Github account. Anyone can access to the plugin using following link and can download and install in QGIS.

<https://github.com/Hamza-Sadaqat/QGIS-Plugin-Development>