UNIVERSITE ABDELMALEK ESSAADI
FACULTE DES SCIENCES ET TECHNIQUE DE TANGER
FILIERE INGENIEUR GEOINFORMATION

Rapport TP QGIS Plugin

Partie Attributaire et Partie Spatiale

Réalisé par :

TIHTIH Reda

Demandé par :

Mr. YAZIDI ALAOUI Otmane

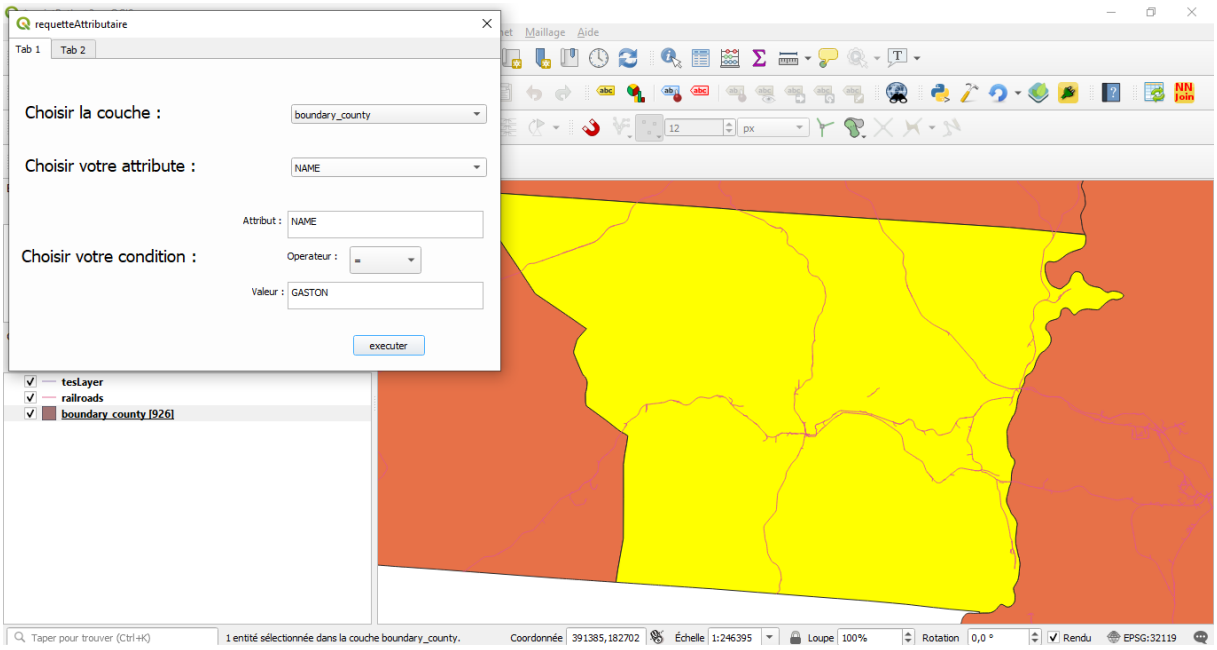
Année universitaire 2020/2021

Le code de la réalisation du plugin est à la fin du ce rapport

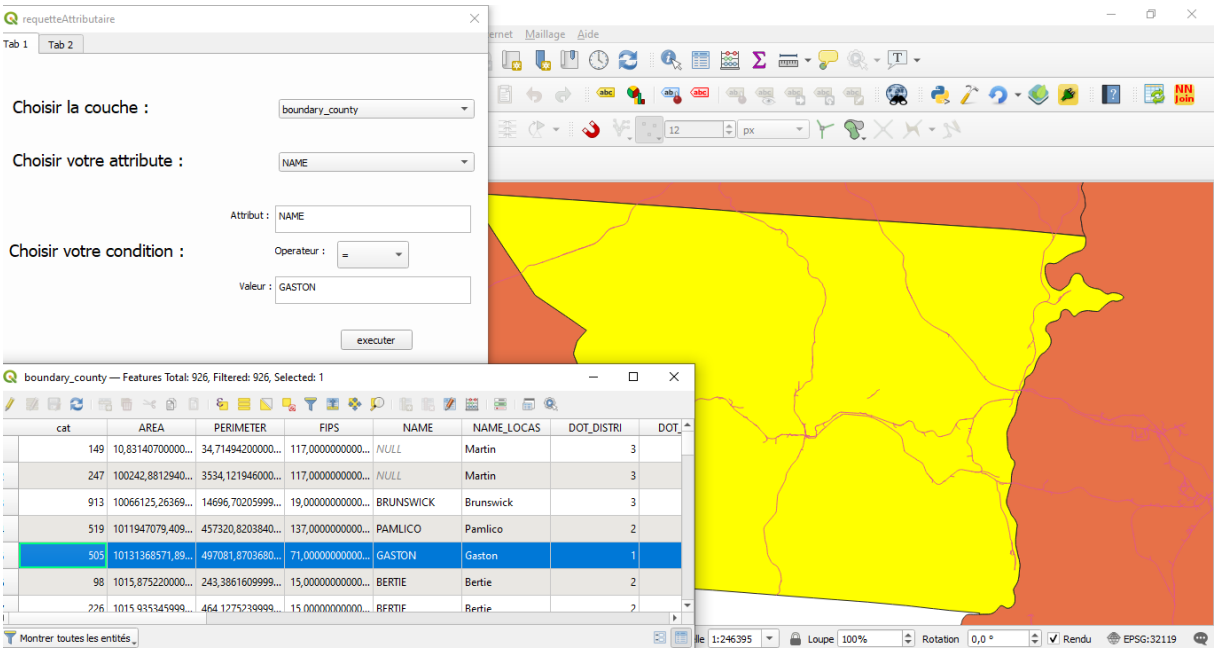
I. La sélection et le zoom sur les entités sélectionnées :

Le principe c'est on va choisir la couche qu'on veut sélectionnée à partir d'un ComboBox qui va contenir l'ensemble des couches qui existe dans QGIS, ensuite les attributs de la couche sélectionnée vont être afficher automatiquement dans un autre combobox, puis une fois on a cliqué sur l'attribut il va automatiquement s'afficher dans la case ou on va écrire notre requête SQL,

Voilà le résultat :



Et pour vérifier on va ouvrir la table attributaire et voilà :



On vérifier bien que a partie attributaire à bien fonctionnées.

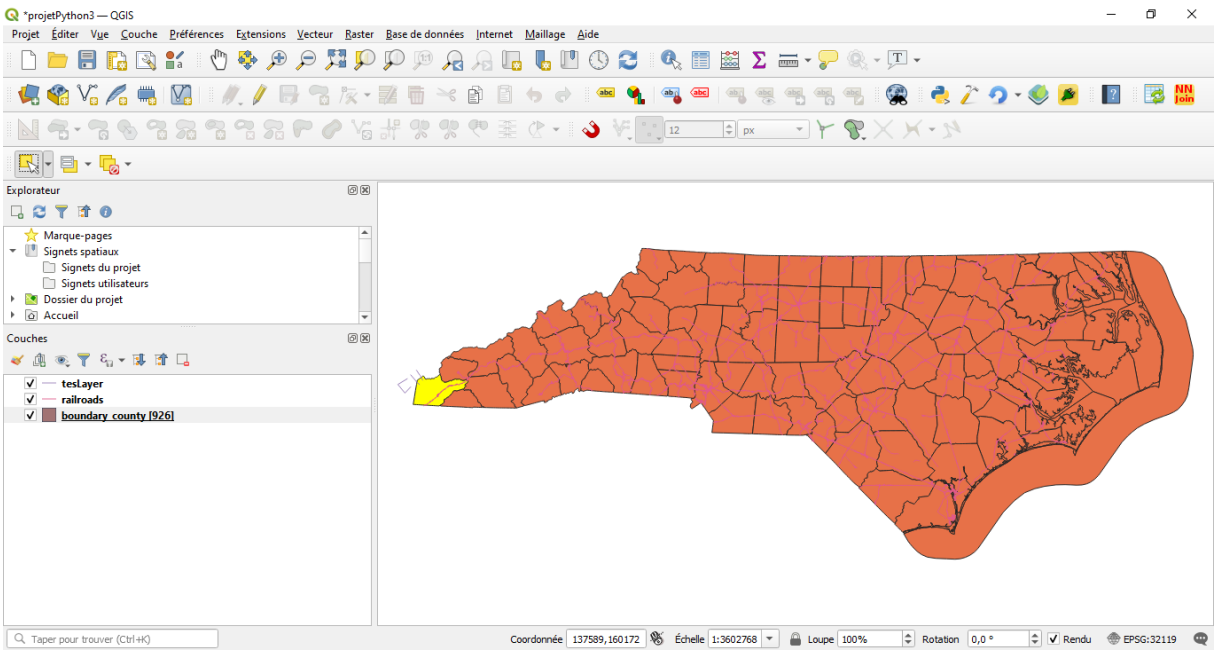
Maintenant on va voir la partie spatiale du TP.

II. *La partie spatiale*, on clique sur l'entité graphique et on applique les relations spatiales sur l'entité sélectionné :

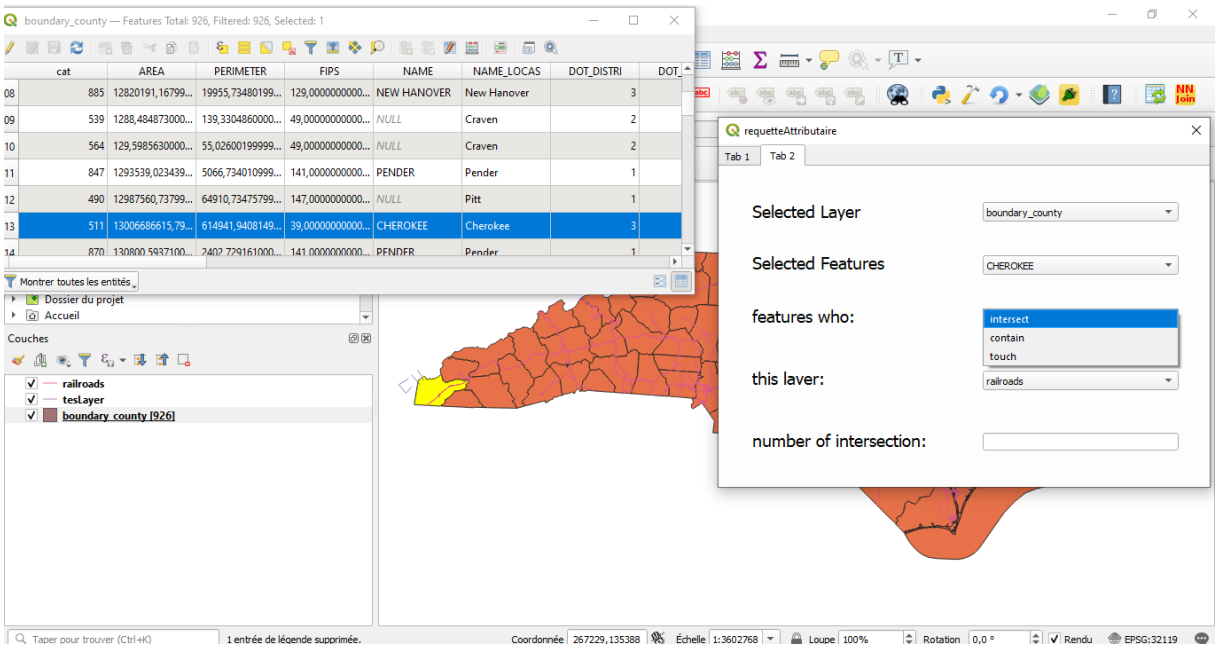
Dans cette partie on va sélectionnée l'entité graphique qu'on veut par la souris et on suite après l'ouverture du plugin on va voire la couche sélectionnée puis exactement l'entité sélectionnée, ensuite dans un combo box on va appliquer la relation spatiale qu'on veut appliquer par exemple (intersect, touched, containe…) dans notre cas on va utiliser la relation intersect et contain qui vont etre appliquer sur les autres couche (dans un autre combo box on va afficher les autre couche sauf la couche qu'on a sélectionnée) et finalement on va voir comme résultat le calcule du nombre des intersection qui existe et on plus on va exporter les entitées qui intersect notre entité qu'on a sélectionnée et on va les afficher dans QGIS,

Voila le résultat pour la relation intersect :

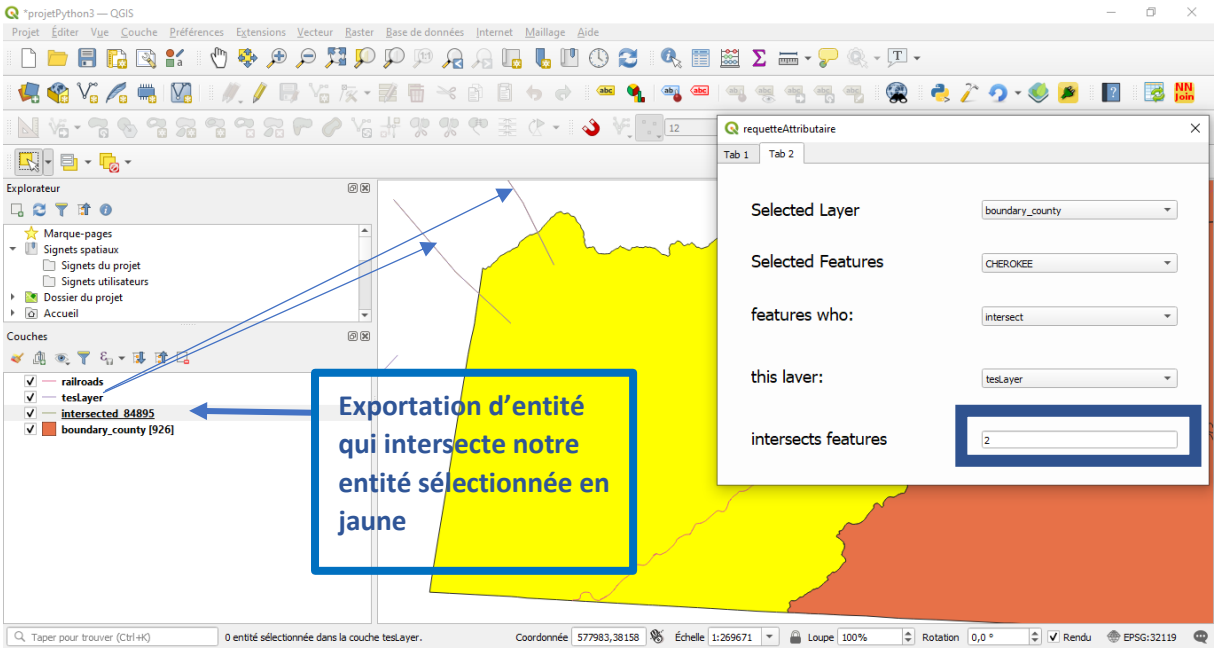
1 d'abord on sélection l'entité :



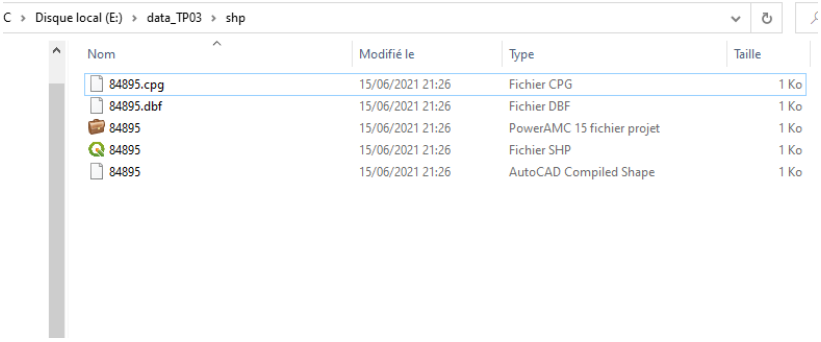
Puis on ouvre notre plugin et on se déplace vers tab2 et voilà ce qu'on va trouver :



Puis on va sélectionnée une couche depuis **THIS LAYER** et automatiquement on va voir un zoom sur la couche sélectionnée et on va voire le nombre des intersections avec la couche qu'on a choisie avec l'exportation des éléments qui intersecte notre entité sélectionnée au départ et voilà le résultat :

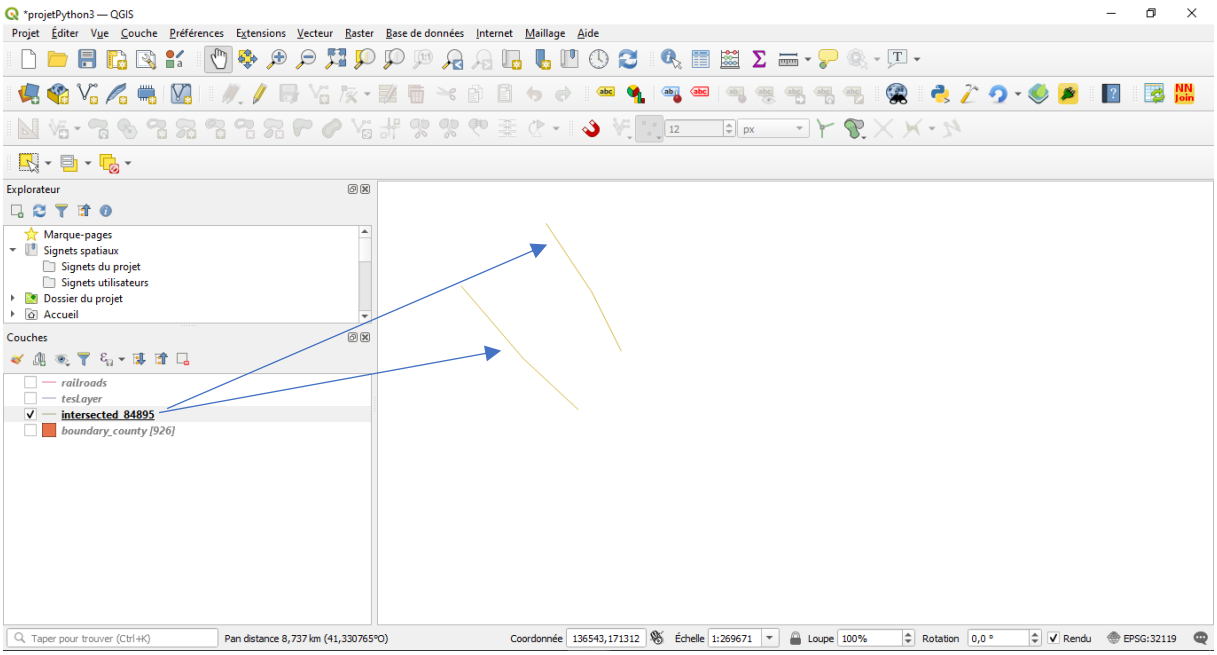


Et voilà les épreuves :



Juste dans l'affichage dans Qgis j'ai ajouté le nom intersected + 84895

Voila si on désélection les autres fichiers il reste seulement les entités exporter (celle qui intersecte l'entité qu'on a sectionnée :




```
#Voilà le code pour l'exportation et l'importation dans QGIS

folder ='E:/data_TP03/shp/'+str(i)+'.shp'
        writer = QgsVectorFileWriter.writeAsVectorFormat(selectedLayer,folder,
 'utf-8',driverName='ESRI Shapefile',onlySelected = True)

        #imporation des shp exprté vers QGIS
        self.iface.addVectorLayer(folder,'intersected ','ogr')
```

Pour les autre relation Contain et touch il sont le même principe que la relation intersect

Il sufit just de changer

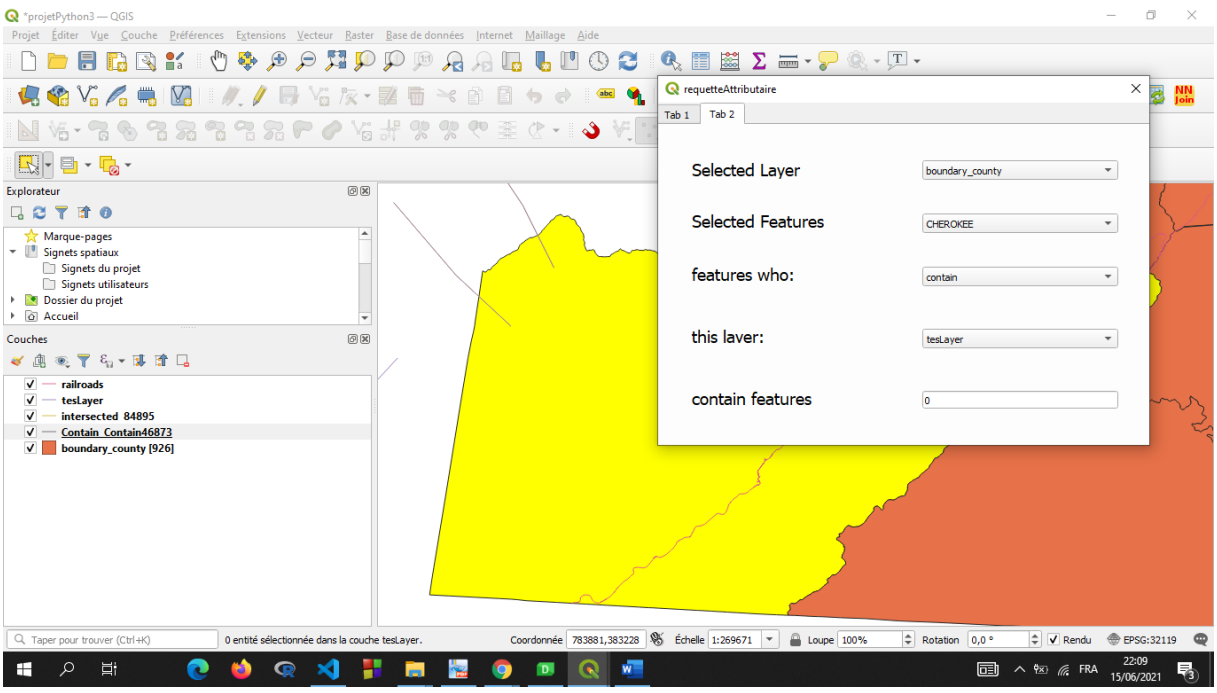
```
sel.geometry().intersects(f.geometry())
```

Par

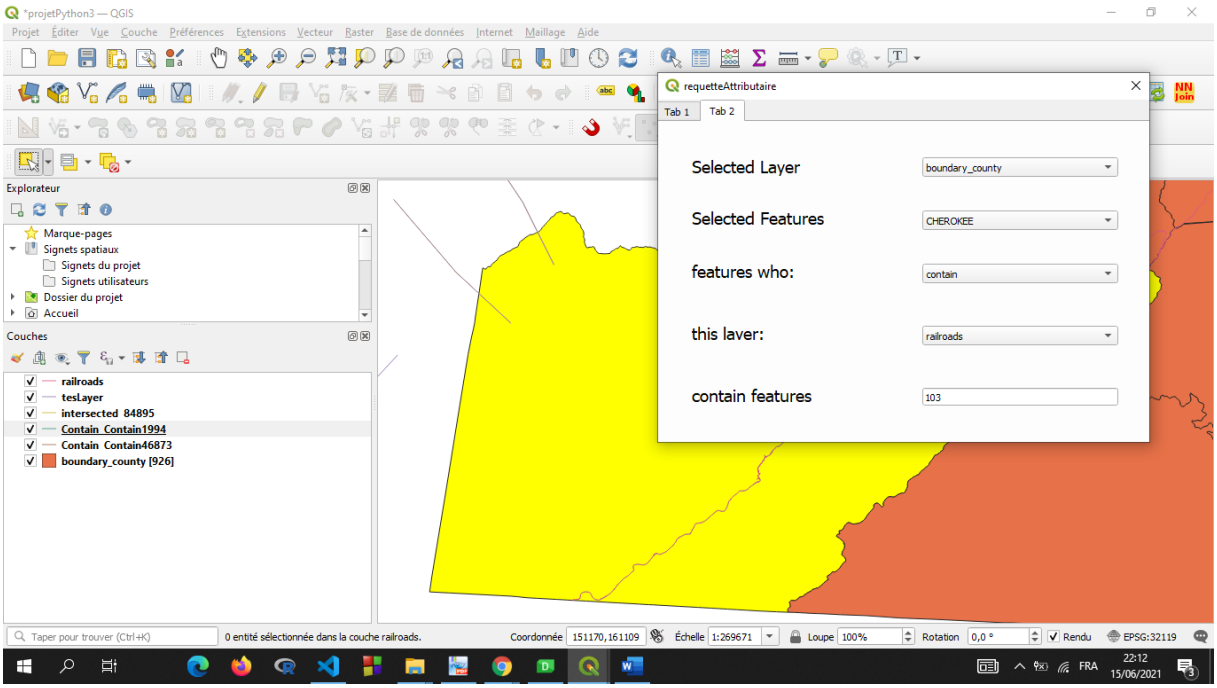
```
sel.geometry().contains(f.geometry())
```

Voilà la relation Contain() qui va nous exporter et calculer les entités a l'intérieur de l'entité qu'on a sélectionné

Voilà le résultat :



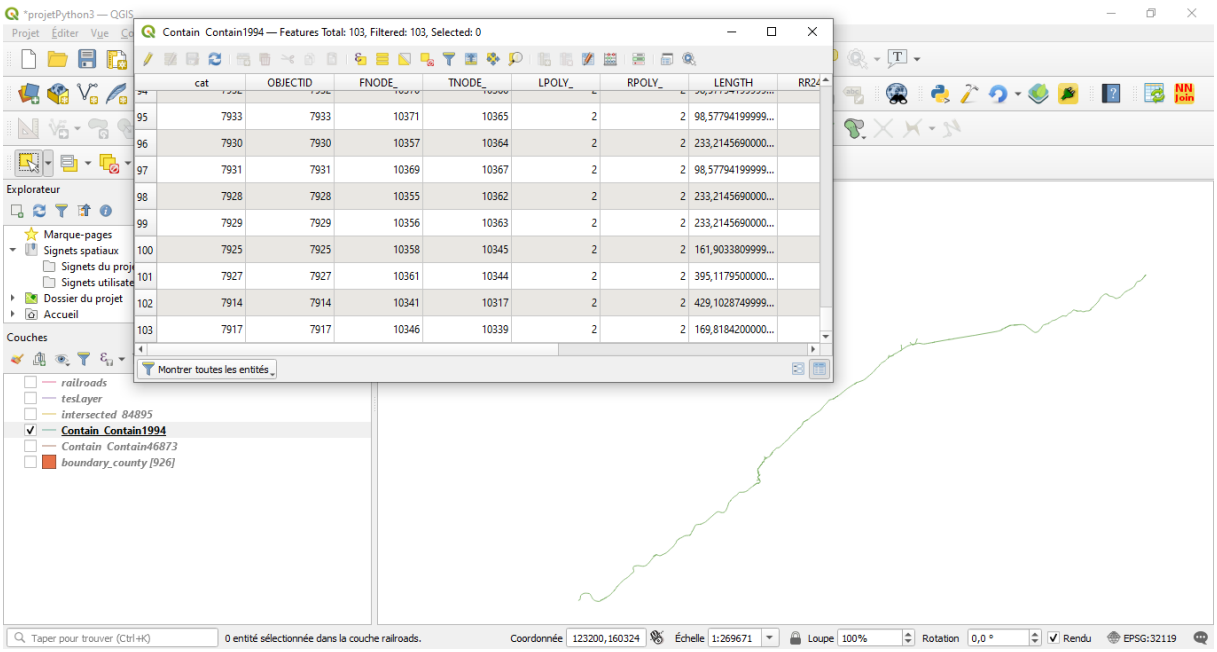
On va tester par une autre couche pour vérifier :



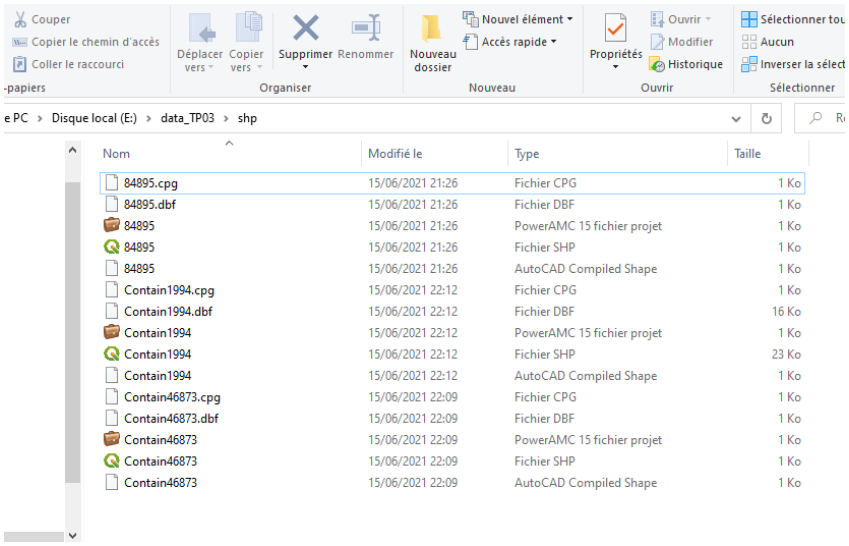
On destingue qu'il a trouver vraiment 103 entité de la couche railroads et ila crée leur layer qui s'appelle **Contain Contain 1994**

Verification

Voila la table attributaire de la couche exporté qui existe a l'intérieur de la couche selectionné au début et vraiment il y a 103 entités



Voila le dossier où se trouve les shapefiles exporté :



Voila le code de la réalisation du ce plugin :

```
    def test(self,layers):
        selectedLayerIndex = self.dlg.comboLayer.currentIndex()
        selectedLayer = layers[selectedLayerIndex].layer()
        fieldnames = [field.name() for field in selectedLayer.fields()]
        #print(fieldnames)
        self.dlg.comboAttribute.clear()
        # Populate the comboBox with names of all the loaded layers
        self.dlg.comboAttribute.addItems(fieldnames)


    def test2(self,selectedAttributeIndex):
        self.dlg.textCondition.setText(self.dlg.comboAttribute.currentText()+'
 ')

    def queryAttribute(self,layers):
        selectedLayerIndex = self.dlg.comboLayer.currentIndex()
        selectedLayer = layers[selectedLayerIndex].layer()
        self.iface.setActiveLayer(selectedLayer)
        attrName = self.dlg.textCondition.toPlainText()
        operateur = self.dlg.comboOperateur.currentText()
        valeur = self.dlg.textValeur.toPlainText()
        print(attrName+""+operateur+" "+"'"+valeur+"'")

        selectedLayer.selectByExpression(attrName+""+operateur+" "+"'"+valeur+
"'")
```

```
        self.iface.actionZoomToSelected().trigger()

    def changeLabel(self):
        relation = self.dlg.spatialCondition.currentText()

        if relation == "intersect":
            self.dlg.dLabel.setText("intersects features")

        elif relation == "contain":
            self.dlg.dLabel.setText("selected area")
            print(1)
        elif relation == "touch":
            self.dlg.dLabel.setText("touched features")
            print(2)
        else:
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("this relation doesn't exist yet")
            msg.setWindowTitle("Error")
            msg.exec_()


    def spatialFunction(self,layers1,selection):

        relation = self.dlg.spatialCondition.currentText()


        if relation == "intersect":
            self.dlg.dLabel.setText("intersects features")
            self.intersects(layers1,selection)
        elif relation == "contain":
            self.dlg.dLabel.setText("contain features")
            self.contain(layers1,selection)
            print(1)
        elif relation == "touch":
            self.dlg.dLabel.setText("touched features")
            print(2)
        else:
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("this relation doesn't exist yet")
            msg.setWindowTitle("Error")
            msg.exec_()
```

```
    def intersects(self,layers1,selection):
        selectedLayerIndex = self.dlg.nonActiveLayer.currentIndex()
        selectedLayer = layers1[selectedLayerIndex].layer() #layer qui n'est p
as selectionnée au debut

        #fieldnames = [field.name() for field in selectedLayer.fields()]
        features=[]
        i=random.randint(1,100000)
        count=0
        for sel in selection:

            for f in selectedLayer.getFeatures():

                if sel.geometry().intersects(f.geometry()):
                    count=count+1
                    features.append(f.id())
                else:
                    pass
        print(features)
        #on va selectionnées les features qui intersect le feature qu'on a sel
ectionnée au départ
        selFeat = selectedLayer.select(features)
        self.iface.actionZoomToSelected().trigger()

        #exportation des features selectionné
        folder ='E:/data_TP03/shp/'+str(i)+'.shp'
        writer = QgsVectorFileWriter.writeAsVectorFormat(selectedLayer,folder,
 'utf-8',driverName='ESRI Shapefile',onlySelected = True)

        #imporation des shp exprté vers QGIS
        self.iface.addVectorLayer(folder,'intersected ','ogr')
        selectedLayer.deselect(features)


        self.dlg.countIntersects.setText(str(count))
        count=0


    def contain(self,layers1,selection):
        selectedLayerIndex = self.dlg.nonActiveLayer.currentIndex()
        selectedLayer = layers1[selectedLayerIndex].layer() #layer qui n'est p
as selectionnée au debut

        #fieldnames = [field.name() for field in selectedLayer.fields()]
        features=[]
        i=random.randint(1,100000)
        count=0
        for sel in selection:
```

```
            for f in selectedLayer.getFeatures():

                if sel.geometry().contains(f.geometry()):
                    count=count+1
                    features.append(f.id())
                else:
                    pass
        print(features)
        #on va selectionnées les features qui intersect le feature qu'on a sel
ectionnée au départ
        selFeat = selectedLayer.select(features)
        self.iface.actionZoomToSelected().trigger()

        #exportation des features selectionné
        folder ='E:/data_TP03/shp/Contain'+str(i)+'.shp'
        writer = QgsVectorFileWriter.writeAsVectorFormat(selectedLayer,folder,
 'utf-8',driverName='ESRI Shapefile',onlySelected = True)

        #imporation des shp exprté vers QGIS
        self.iface.addVectorLayer(folder,'Contain ','ogr')
        selectedLayer.deselect(features)


        self.dlg.countIntersects.setText(str(count))
        count=0


    def run(self):
        """Run method that performs all the real work"""

        # Create the dialog with elements (after translation) and keep referen
ce
        # Only create GUI ONCE in callback, so that it will only load when the
 plugin is started
        if self.first_start == True:
            self.first_start = False
            self.dlg = requette_attributaireDialog()

        # Fetch the currently loaded layers
        layers = QgsProject.instance().layerTreeRoot().children()
        # Clear the contents of the comboBox from previous runs
        self.dlg.comboLayer.clear()
        # Populate the comboBox with names of all the loaded layers
        self.dlg.comboLayer.addItems([layer.name() for layer in layers])

        self.dlg.comboLayer.currentIndexChanged.connect(lambda : self.test(lay
ers))
```

```
        selectedAttributeIndex = self.dlg.comboAttribute.currentIndex()

        self.dlg.comboAttribute.activated.connect(lambda: self.test2(selectedA
ttributeIndex))

        self.dlg.btn.clicked.connect(lambda: self.queryAttribute(layers))



        layer2 = self.iface.activeLayer()
        """
        layer2=self.iface.layerTreeView().selectedLayers()
        self.dlg.layerSelected.clear()
        self.dlg.layerSelected.addItem(layer2)
        """
        self.dlg.layerSelected.clear()
        self.dlg.layerSelected.addItem(layer2.name())




        selection = layer2.selectedFeatures()
        self.dlg.featureSelected.clear()
        selectedfeatures = [str(feat['NAME'])  for feat in selection]


        self.dlg.featureSelected.addItems(selectedfeatures)


        self.dlg.nonActiveLayer.clear()
        layers1=[]
        for layer in layers:
            if layer.name()==layer2.name():
                pass
            else:
                layers1.append(layer)
                self.dlg.nonActiveLayer.addItem(layer.name())

        self.dlg.nonActiveLayer.currentIndexChanged.connect(lambda : self.spat
ialFunction(layers1,selection))




        # show the dialog
        self.dlg.show()
        # Run the dialog event loop
        result = self.dlg.exec_()
```

```
    # See if OK was pressed
    if result:
        # Do something useful here - delete the line containing pass and
        # substitute with your code.


        pass
```