

Writing Python Scripts for Processing Framework (QGIS3)

QGIS Tutorials and Tips



Author
Ujal Gandhi
<http://www.spatialthoughts.com>

Writing Python Scripts for Processing Framework (QGIS3)

One can write standalone pyqgis scripts that can be run via the Python Console in QGIS. With a few tweaks, you can make your standalone scripts run via the Processing Framework. This has several advantages. First, taking user input and writing output files is far easier because Processing Framework offers standardized user interface for these. Second, having your script in the Processing Toolbox also allows it to be part of any Processing Model or be run as a Batch job with multiple inputs. This tutorial will show how to write a custom python script that can be part of the Processing Framework in QGIS.

Note

The Processing API was completely overhauled in QGIS3. Please refer to [this guide](#) for best practices and tips.

Overview of the task

Our script will perform a dissolve operation based on a field chosen by the user. It will also sum up values of another field for the dissolved features. In the example, we will dissolve a world shapefile based on a `CONTINENT` attribute and sum up `POP_EST` field to calculate total population in the dissolved region.

Get the data

We will use the [Admin 0 - Countries](#) dataset from Natural Earth.

Download the [Admin 0 - countries shapefile..](#)

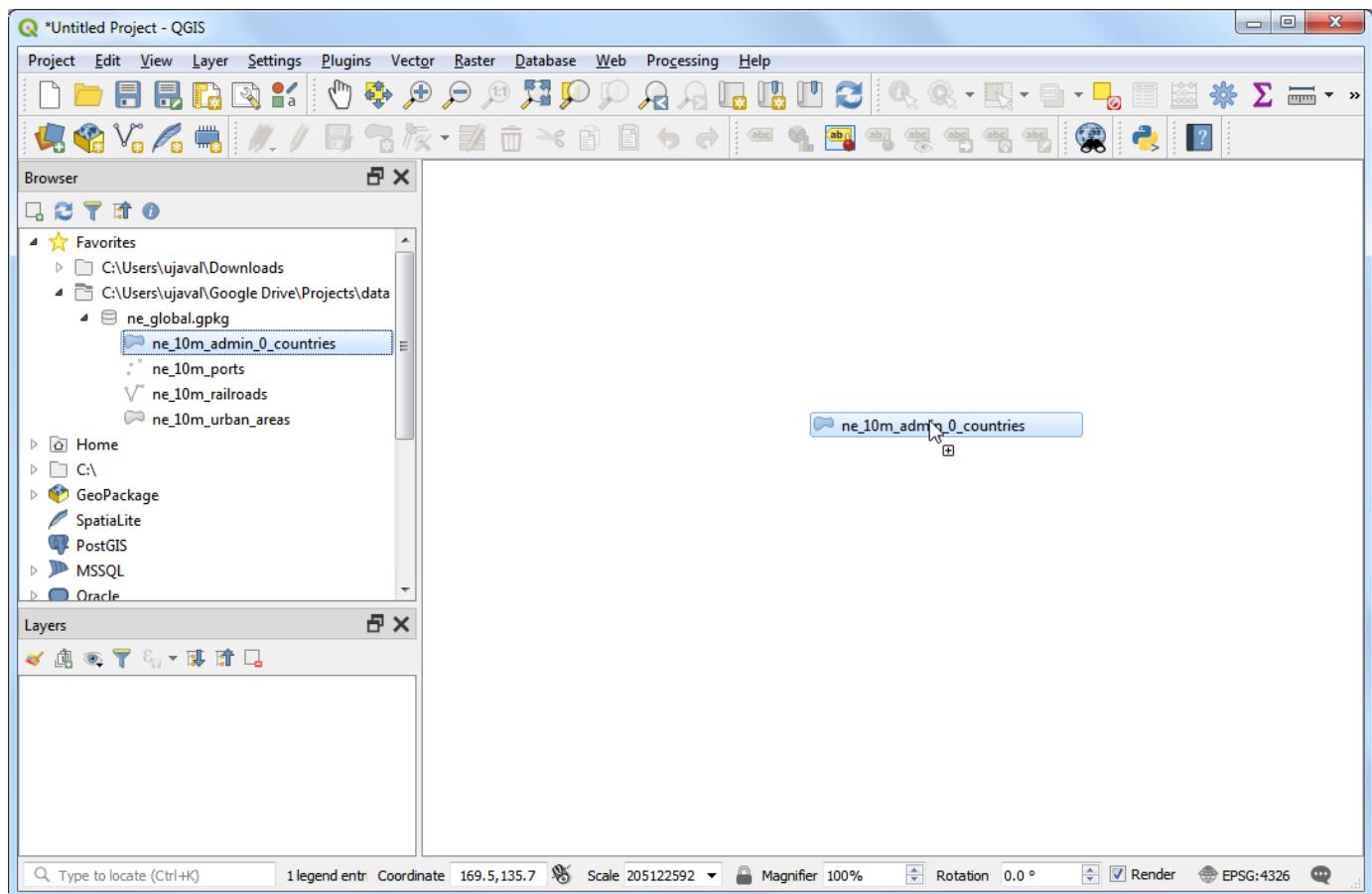
Data Source [NATURALEARTH]

For convenience, you may directly download a geopackage containing the above layer from below:

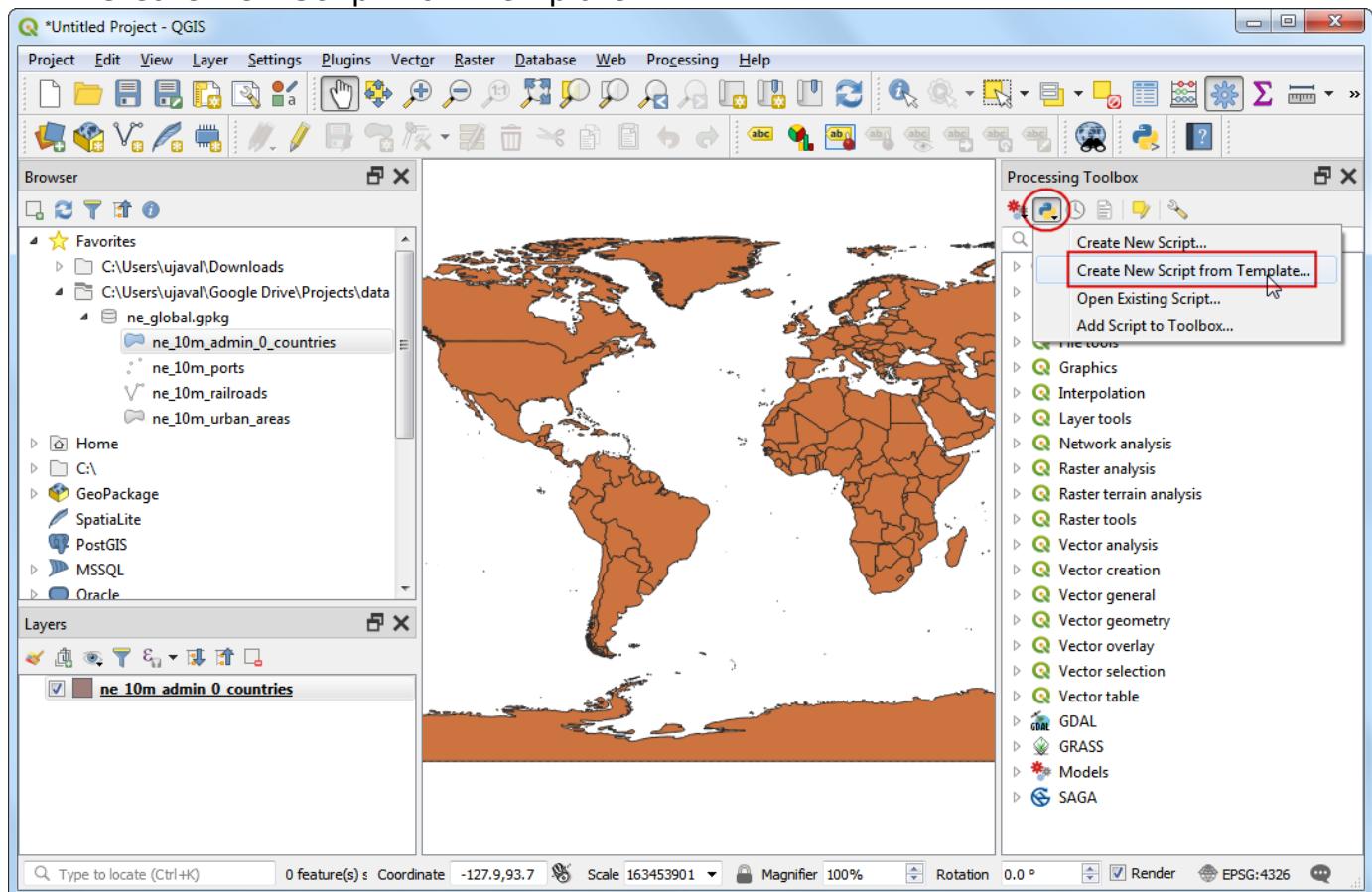
[ne_global.gpkg](#)

Procedure

1. In the QGIS Browser Panel, locate the directory where you saved your downloaded data. Expand the `zip` or the `gpkg` entry and select the `ne_10m_admin_0_countries` layer. Drag the layer to the canvas.



2. Go to Processing ▶ Toolbox. Click the Scripts button in the toolbar and select Create New Script from Template.



3. The template contains all the boilerplate code that is required for the Processing Framework to recognize it as a Processing script, and manage inputs/outputs. Let's start customizing the example template to our needs. First change the class name from `ExampleProcessingAlgorithm` to `DissolveProcessingAlgorithm`. This name also needs to be updated in the `createInstance` method. Add a docstring to the class that explains what the algorithm does.

```

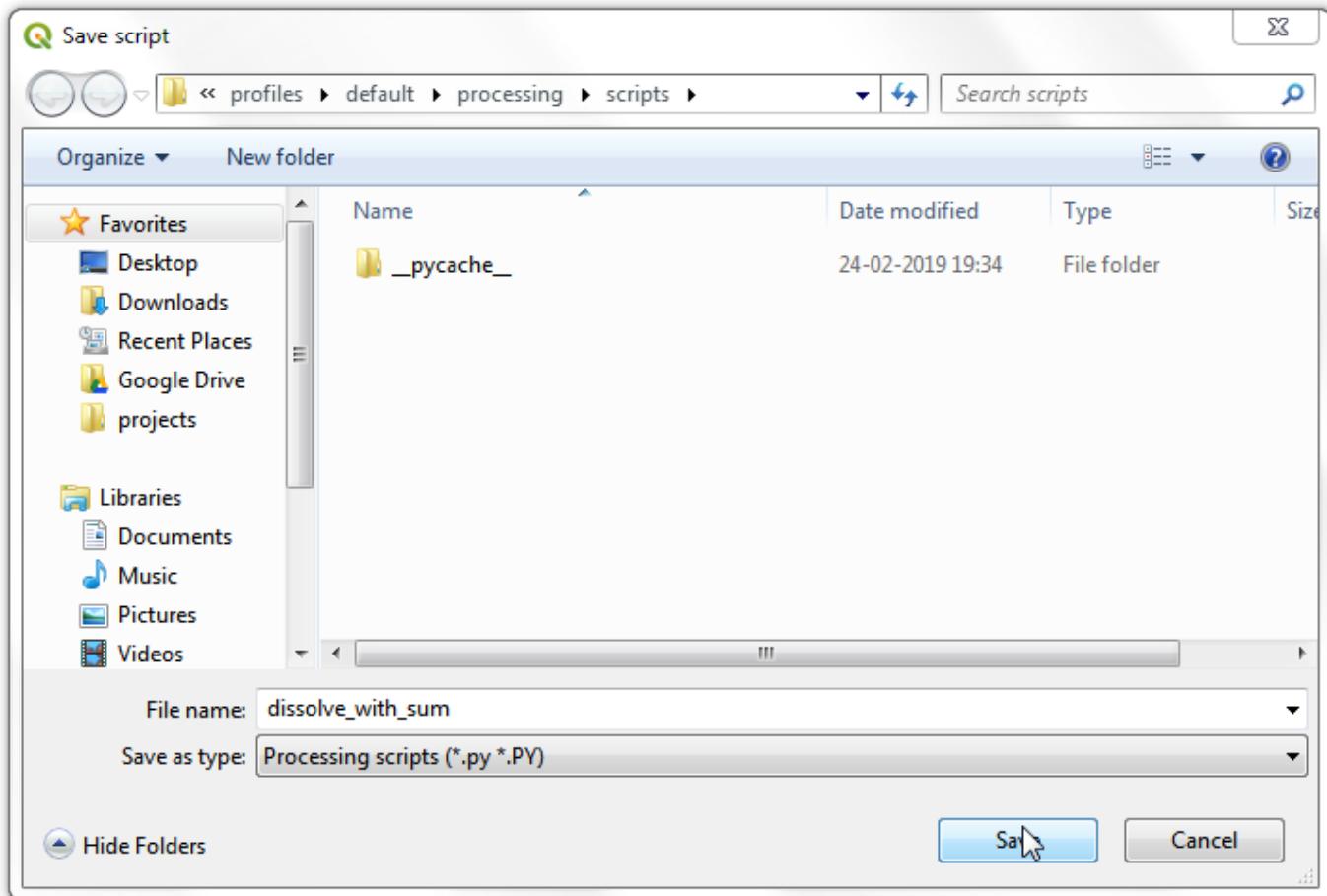
22     QgsProcessingParameterFeatureSink,
23     )
24
25
26
27 - class DissolveProcessingAlgorithm(QgsProcessingAlgorithm):
28     """
29     Dissolve algorithm that dissolves features based on selected
30     attribute and summarizes the selected field by computing the
31     sum of dissolved features.
32     """
33     INPUT = 'INPUT'
34     OUTPUT = 'OUTPUT'
35
36 -     def tr(self, string):
37         """
38             Returns a translatable string with the self.tr() function.
39         """
40         return QCoreApplication.translate('Processing', string)
41
42 -     def createInstance(self):
43         return DissolveProcessingAlgorithm()
44
45 -     def name(self):
46         """
47             Returns the algorithm name, used for identifying the algorithm. This
48             string should be fixed for the algorithm, and must not be localised.
49             The name should be unique within each provider. Names should contain

```

4. As you scroll down, you will see methods that assign name, group, description etc. to the script. Change the return values for `name` method to be `dissolve_with_sum`, `displayName` method to `Dissolve with Sum`, `group` method and `groupId` method to `scripts`. Change the return value of `shortHelpString` method to a description that will appear to the user. Click the Save button.

```
47 -     def name(self):
48 -         """
49 -             Returns the algorithm name, used for identifying the algorithm. This
50 -             string should be fixed for the algorithm, and must not be localised.
51 -             The name should be unique within each provider. Names should contain
52 -             lowercase alphanumeric characters only and no spaces or other
53 -             formatting characters.
54 -         """
55 -         return 'dissolve_with_sum'
56 -
57 -     def displayName(self):
58 -         """
59 -             Returns the translated algorithm name, which should be used for any
60 -             user-visible display of the algorithm name.
61 -         """
62 -         return self.tr('Dissolve with Sum')
63 -
64 -     def group(self):
65 -         """
66 -             Returns the name of the group this algorithm belongs to. This string
67 -             should be localised.
68 -         """
69 -         return self.tr('scripts')
70 -
71 -     def groupId(self):
72 -         """
73 -             Returns the unique ID of the group this algorithm belongs to. This
74 -             string should be fixed for the algorithm, and must not be localised.
75 -             The group id should be unique within each provider. Group id should
76 -             contain lowercase alphanumeric characters only and no spaces or other
77 -             formatting characters.
78 -         """
79 -         return 'scripts'
80 -
81 -     def shortHelpString(self):
82 -         """
83 -             Returns a localised short helper string for the algorithm. This string
84 -             should provide a basic description about what the algorithm does and the
85 -             parameters and outputs associated with it..
86 -         """
87 -         return self.tr("Dissolves selected features and creates and sums values of fea")
88 -
89 -     def initAlgorithm(self, config=None):
90 -         """
```

5. Name the script `dissolve_with_sum` and save it at the default location under `profiles > default > processing > scripts` folder.



6. Now we will define the inputs for the script. The template already contains a definition of an INPUT Vector Layer and a OUTPUT layer. We will add 2 new inputs which allows the user to select a DISSOLVE_FIELD and a SUM_FIELD. Add a new import at the top and the following code in the initAlgorithm method. Click the Run button to preview the changes.

```
from qgis.core import QgsProcessingParameterField

self.addParameter(
    QgsProcessingParameterField(
        self.DISSOLVE_FIELD,
        'Choose Dissolve Field',
        '',
        self.INPUT))

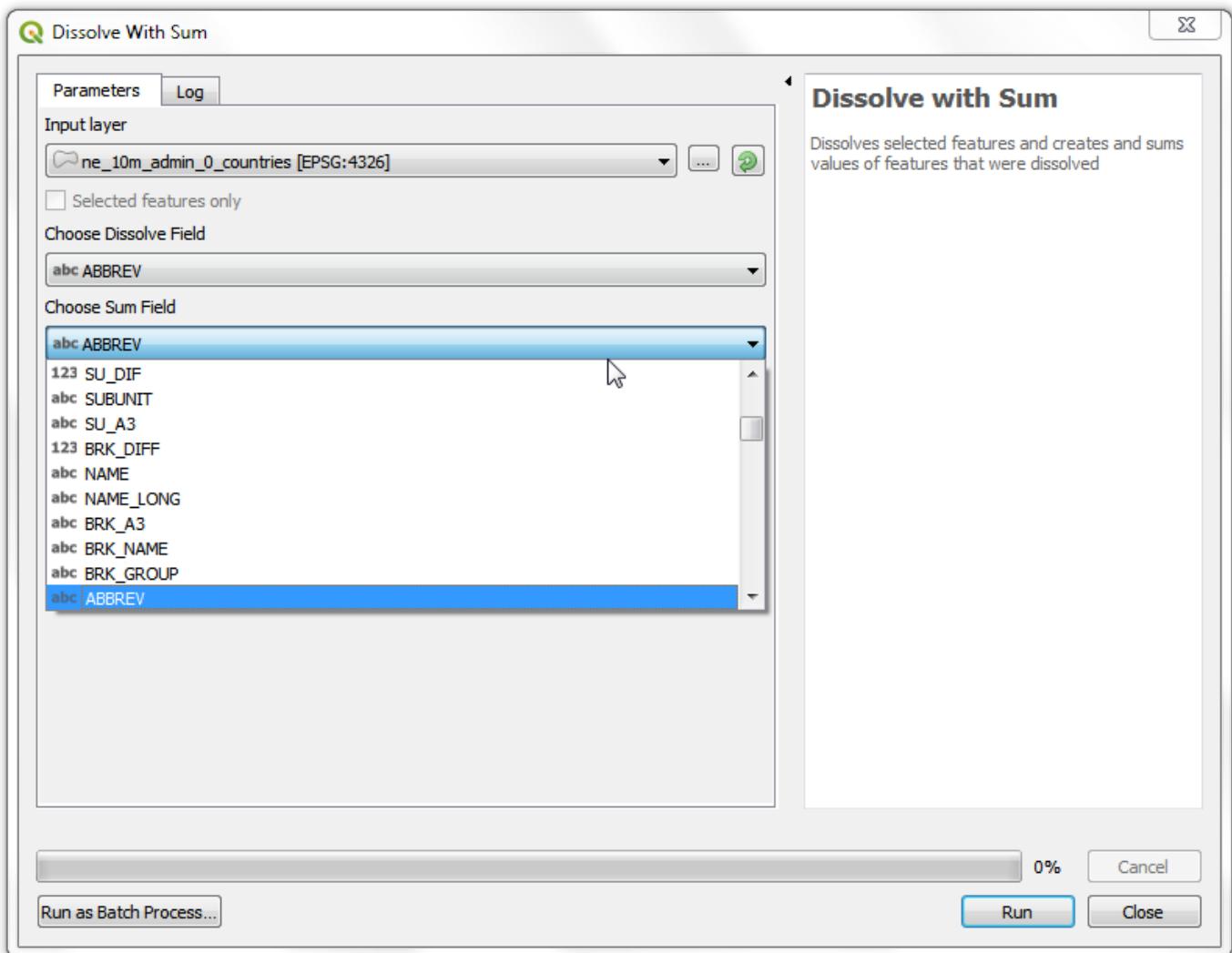
self.addParameter(
    QgsProcessingParameterField(
        self.SUM_FIELD,
        'Choose Sum Field',
        '',
        self.INPUT))
```

The screenshot shows the Processing Script Editor window with the file 'dissolve_with_sum.py' open. The code implements a 'DissolveProcessingAlgorithm' class that extends 'QgsProcessingAlgorithm'. It imports several modules from PyQt5.QtCore and qgis.core, including QCoreApplication, QgsFeatureSink, QgsProcessingException, QgsProcessingAlgorithm, QgsProcessingParameterFeatureSource, QgsProcessingParameterFeatureSink, and QgsProcessingParameterField. The class has a docstring explaining it dissolves features based on a selected attribute and summarizes the dissolved features by computing the sum of the dissolved features. It defines constants for INPUT ('INPUT'), OUTPUT ('OUTPUT'), DISSOLVE_FIELD ('dissolve_field'), and SUM_FIELD ('sum_field'). A red box highlights the 'QgsProcessingParameterField' import and the constant definitions. Another red box highlights the play button icon in the toolbar, which is circled in red.

```
13
14  from PyQt5.QtCore import QCoreApplication
15 - from qgis.core import (QgsProcessing,
16   QgsFeatureSink,
17   QgsProcessingException,
18   QgsProcessingAlgorithm,
19   QgsProcessingParameterFeatureSource,
20   QgsProcessingParameterFeatureSink,
21   QgsProcessingParameterField)
22 import processing
23
24
25 -class DissolveProcessingAlgorithm(QgsProcessingAlgorithm):
26     """
27         Dissolve algorithm that dissolves features based on selected
28         attribute and summarizes the selected field by computing the
29         sum of dissolved features.
30     """
31     INPUT = 'INPUT'
32     OUTPUT = 'OUTPUT'
33     DISSOLVE_FIELD = 'dissolve_field'
34     SUM_FIELD = 'sum_field'
35
36 -     def tr(self, string):
```

```
85     return self.tr("Dissolves selected features and creates and sums values of features")
86
87     def initAlgorithm(self, config=None):
88         """
89             Here we define the inputs and output of the algorithm, along
90             with some other properties.
91         """
92
93         # We add the input vector features source. It can have any kind of
94         # geometry.
95         self.addParameter(
96             QgsProcessingParameterFeatureSource(
97                 self.INPUT,
98                 self.tr('Input layer'),
99                 [QgsProcessing.TypeVectorAnyGeometry]
100            )
101        )
102        self.addParameter(
103            QgsProcessingParameterField(
104                self.DISSOLVE_FIELD,
105                'Choose Dissolve Field',
106                '',
107                self.INPUT))
108        self.addParameter(
109            QgsProcessingParameterField(
110                self.SUM_FIELD,
111                'Choose Sum Field',
112                '',
113                self.INPUT))
```

7. You will see a Dissolve with Sum dialog with our newly defined inputs. Select the `ne_10m_admin_0_countries` layer as Input layer. As both Dissolve Field and Sum Fields are filtered based on the input layer, they will be pre-populated with existing fields from the input layer. Click the Close button.



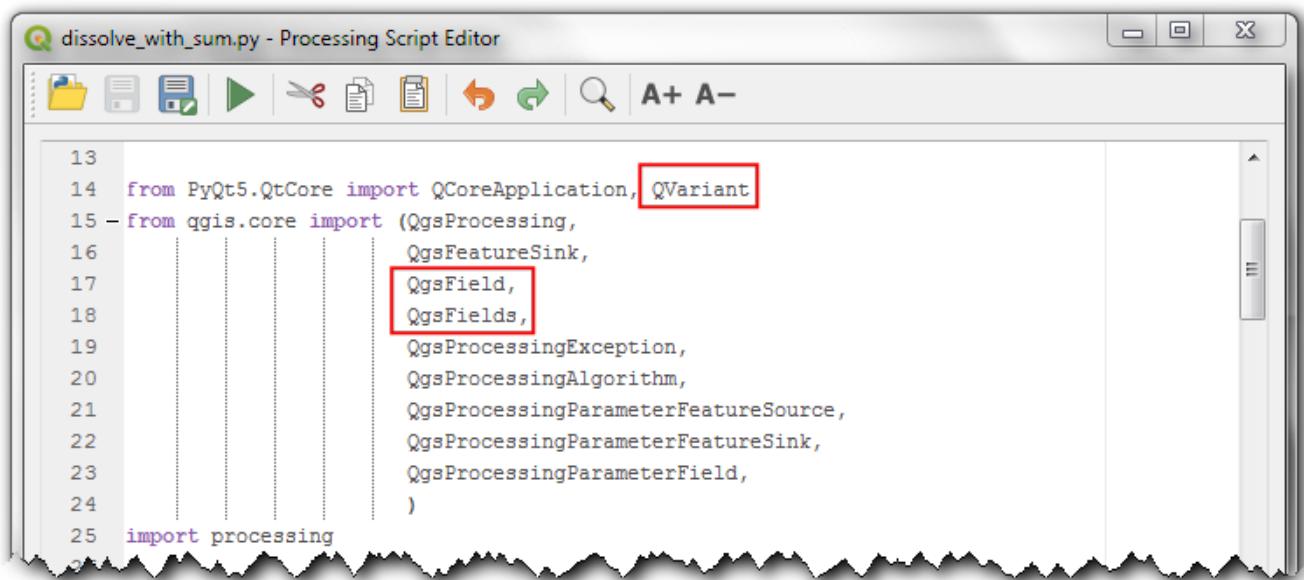
8. Now we define our custom logic for processing data in the `processAlgorithm` method. This method gets passed a dictionary called `parameters`. It contains the inputs that the user has selected. There are helper methods that allow you to take these inputs and create appropriate objects. We first get our inputs using `parameterAsSource` and `parameterAsString` methods. Next we want to create a feature sink where we will write the output. QGIS3 has a new class called `QgsFeatureSink` which is the preferred way to create objects that can accept new features. The output needs only 2 fields - one for the value of dissolved field and another for the sum of the selected field.

```
from PyQt5.QtCore import QVariant
from qgis.core import QgsField, QgsFields

source = self.parameterAsSource(
    parameters,
    self.INPUT,
    context)
dissolve_field = self.parameterAsString(
    parameters,
    self.DISSOLVE_FIELD,
    context)
sum_field = self.parameterAsString(
    parameters,
    self.SUM_FIELD,
    context)
```

```
fields = QgsFields()
fields.append(QgsField(dissolve_field, QVariant.String))
fields.append(QgsField('SUM_' + sum_field, QVariant.Double))

(sink, dest_id) = self.parameterAsSink(
    parameters,
    self.OUTPUT,
    context, fields, source.wkbType(), source.sourceCrs())
```



```

125
126     def processAlgorithm(self, parameters, context, feedback):
127         """
128             Here is where the processing itself takes place.
129         """
130         source = self.parameterAsSource(
131             parameters,
132             self.INPUT,
133             context
134         )
135         dissolve_field = self.parameterAsString(
136             parameters,
137             self.DISSOLVE_FIELD,
138             context)
139         sum_field = self.parameterAsString(
140             parameters,
141             self.SUM_FIELD,
142             context)
143
144         fields = QgsFields()
145         fields.append(QgsField(dissolve_field, QVariant.String))
146         fields.append(QgsField('SUM_' + sum_field, QVariant.Double))
147
148         (sink, dest_id) = self.parameterAsSink(
149             parameters,
150             self.OUTPUT,
151             context, fields, source.wkbType(), source.sourceCrs())

```

9. Now we will ready the input features and create a dictionary to hold the unique values from the `dissolve_field` and the sum of the values from the `sum_field`. Note the use of `feedback.pushInfo()` method to communicate the status with the user.

```

feedback.pushInfo('Extracting unique values from dissolve_field and computing sum')

features = source.getFeatures()
unique_values = set(f[dissolve_field] for f in features)

# Get Indices of dissolve field and sum field
dissolveIdx = source.fields().indexFromName(dissolve_field)
sumIdx = source.fields().indexFromName(sum_field)

# Find all unique values for the given dissolve_field and
# sum the corresponding values from the sum_field
sum_unique_values = {}
attrs = [{dissolve_field: f[dissolveIdx], sum_field: f[sumIdx]} for f in source.getFeatures()]
for unique_value in unique_values:
    val_list = [f_attr[sum_field] for f_attr in attrs if f_attr[dissolve_field] == unique_value]
    sum_unique_values[unique_value] = sum(val_list)

```

```

145     fields.append(QgsField(dissolve_field, QVariant.String))
146     fields.append(QgsField('SUM_' + sum_field, QVariant.Double))
147
148 -     (sink, dest_id) = self.parameterAsSink(
149         parameters,
150         self.OUTPUT,
151         context, fields, source.wkbType(), source.sourceCrs())
152
153     # Create a dictionary to hold the unique values from the
154     # dissolve_field and the sum of the values from the sum_field
155     feedback.pushInfo('Extracting unique values from dissolve_field and computing sum')
156     features = source.getFeatures()
157     unique_values = set(f[dissolve_field] for f in features)
158     # Get Indices of dissolve field and sum field
159     dissolveIdx = source.fields().indexFromName(dissolve_field)
160     sumIdx = source.fields().indexFromName(sum_field)
161
162     # Find all unique values for the given dissolve_field and
163     # sum the corresponding values from the sum_field
164     sum_unique_values = {}
165     attrs = [{dissolve_field: f[dissolveIdx], sum_field: f[sumIdx]}
166               for f in source.getFeatures()]
167     for unique_value in unique_values:
168         val_list = [f_attr[sum_field]
169                     for f_attr in attrs if f_attr[dissolve_field] == unique_value]
170         sum_unique_values[unique_value] = sum(val_list)
171
172     return {self.OUTPUT: dest_id}

```

10. Next, we will call the built-in processing algorithm `native:dissolve` on the input layer to generate the dissolved geometries. Once we have the dissolved geometries, we iterate through the output of the dissolve algorithm and create new features to be added to the output. At the end we return the `dest_id` FeatureSink as the output. Now the script is ready. Click the Run button.

Note

Notice the use of `parameters[self.INPUT]` to fetch the input layer from the parameters dictionary directly without defining it as a source. As we are passing the input object to the algorithm without doing anything with it, it's not necessary to define it as a source.

```

from qgis.core import QgsFeature

# Running the processing dissolve algorithm
feedback.pushInfo('Dissolving features')
dissolved_layer = processing.run("native:dissolve", {
    'INPUT': parameters[self.INPUT],
    'FIELD': dissolve_field,
})

```

```

    'OUTPUT': 'memory:'
}, context=context, feedback=feedback)['OUTPUT']

# Read the dissolved layer and create output features
for f in dissolved_layer.getFeatures():
    new_feature = QgsFeature()
    # Set geometry to dissolved geometry
    new_feature.setGeometry(f.geometry())
    # Set attributes from sum_unique_values dictionary that we had computed
    new_feature.setAttributes([f[dissolve_field], sum_unique_values[f[dissolve_field]]])
    sink.addFeature(new_feature, QgsFeatureSink.FastInsert)

return {self.OUTPUT: dest_id}

```

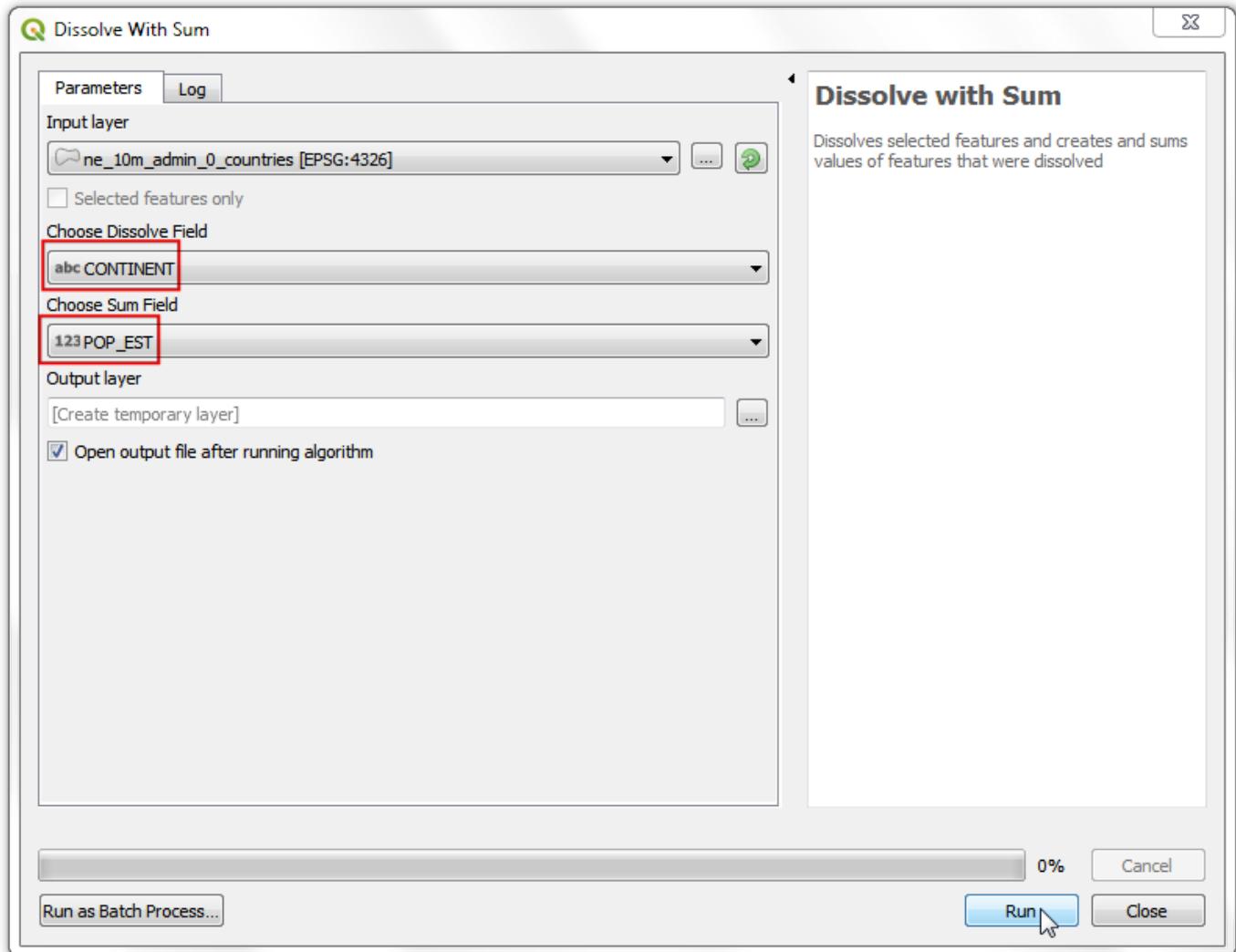
The screenshot shows the Processing Script Editor window with the script 'dissolve_with_sum.py' open. The code is as follows:

```

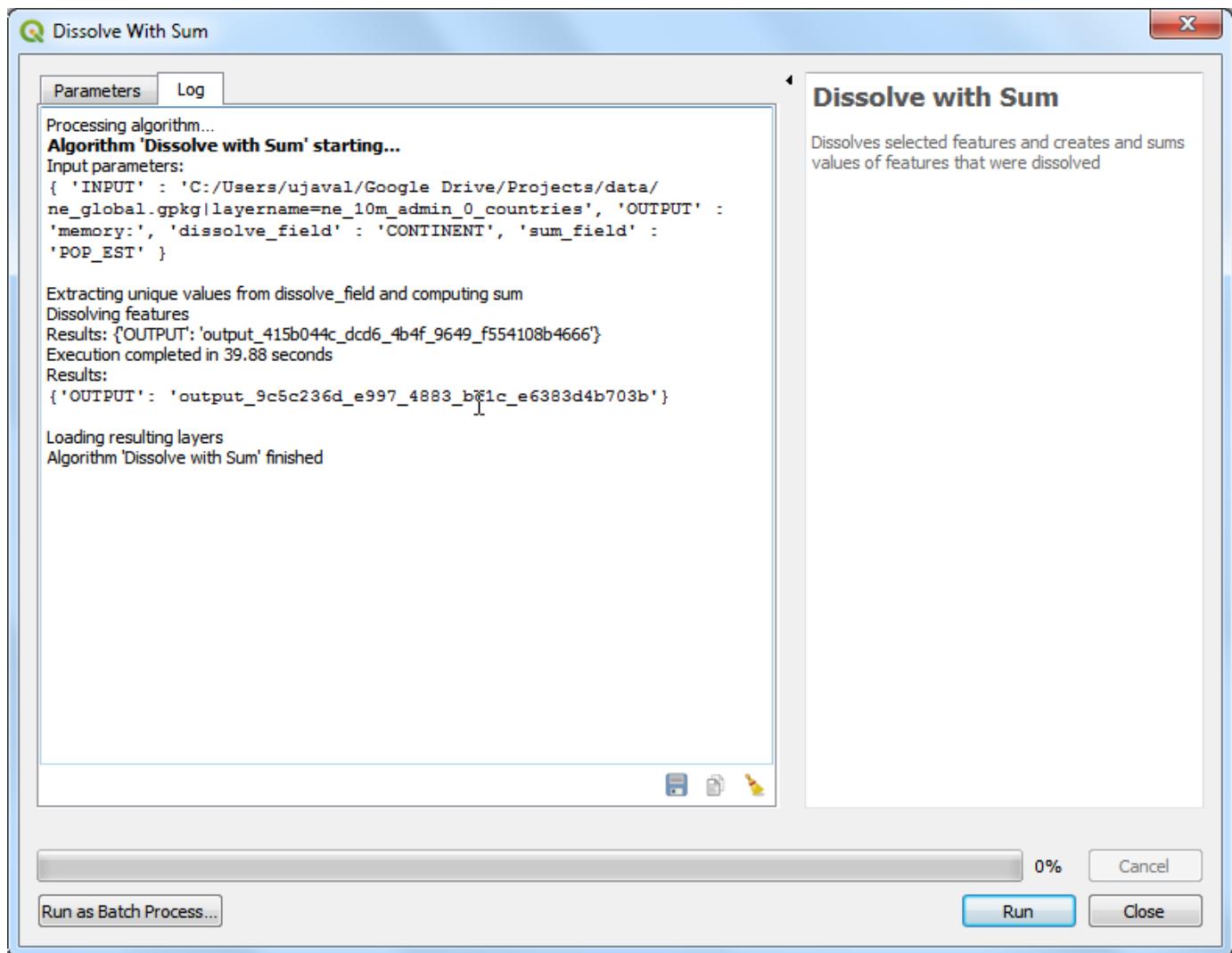
13
14 from PyQt5.QtCore import QCoreApplication, QVariant
15 - from qgis.core import (QgsProcessing,
16                           QgsFeatureSink,
17                           QgsFeature,
18                           QgsField,
19                           QgsFields,
20                           QgsProcessingException,
21                           QgsProcessingAlgorithm,
22                           QgsProcessingFeatureSource
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999

```

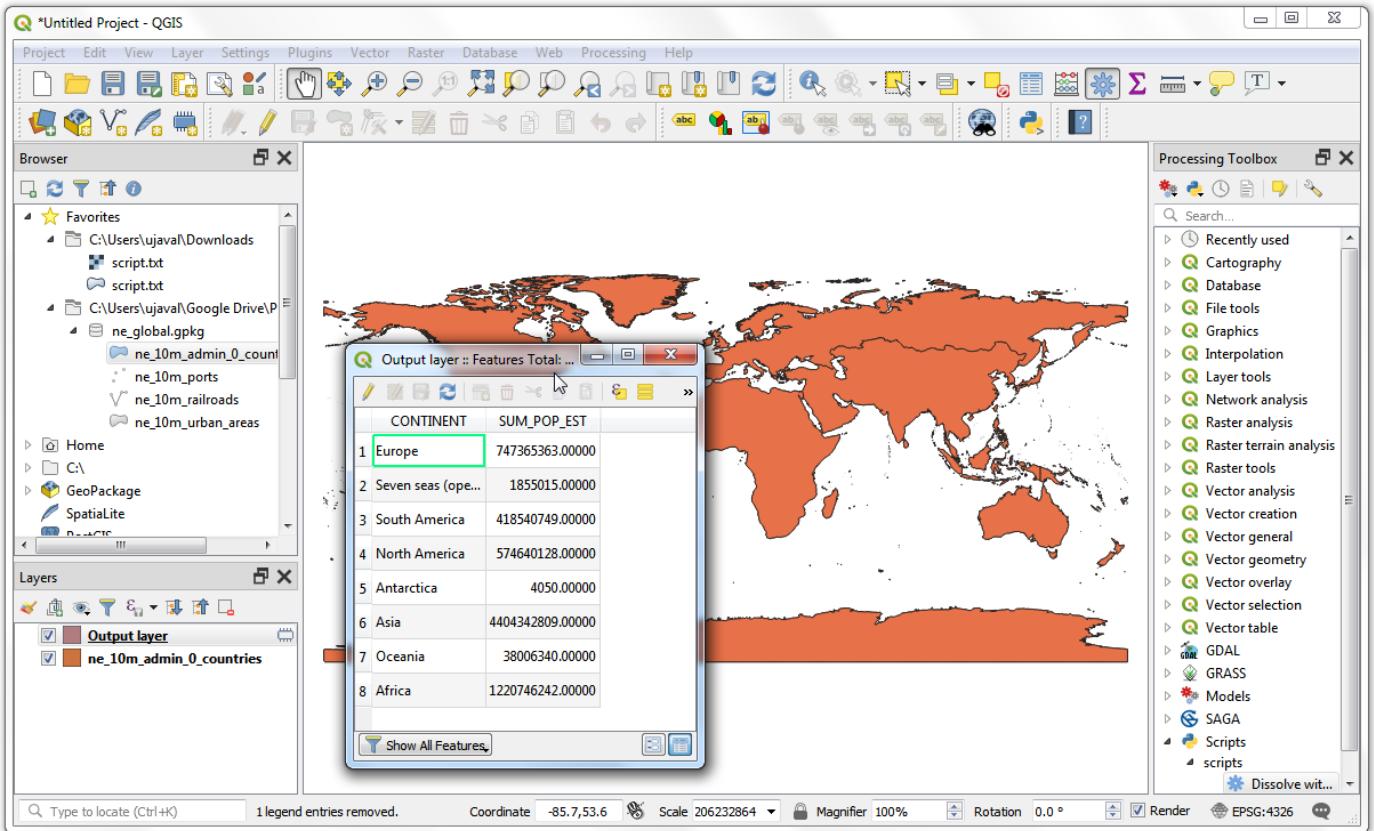
11. In the Dissolve with Sum, dialog, select `ne_10m_admin_0_countries` as the Input layer, `CONTINENT` as the Dissolve field and `POP_EST` as the Sum field. Click Run.



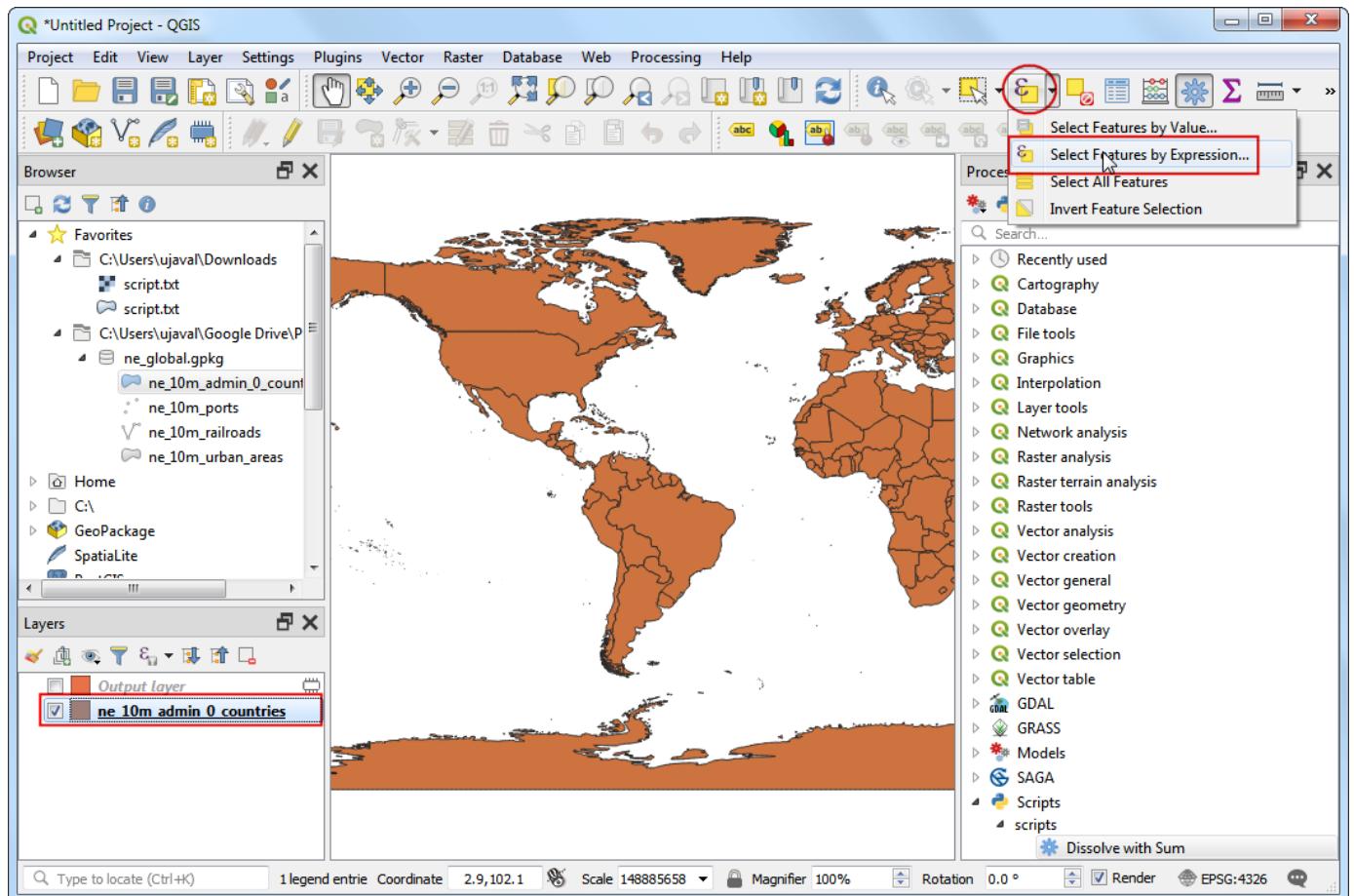
12. Once the processing is finished, click the Close button and switch to the main QGIS window.



13. You will see the dissolved output layer with one feature for every continent and the total population summed from the individual countries belonging to that continent.

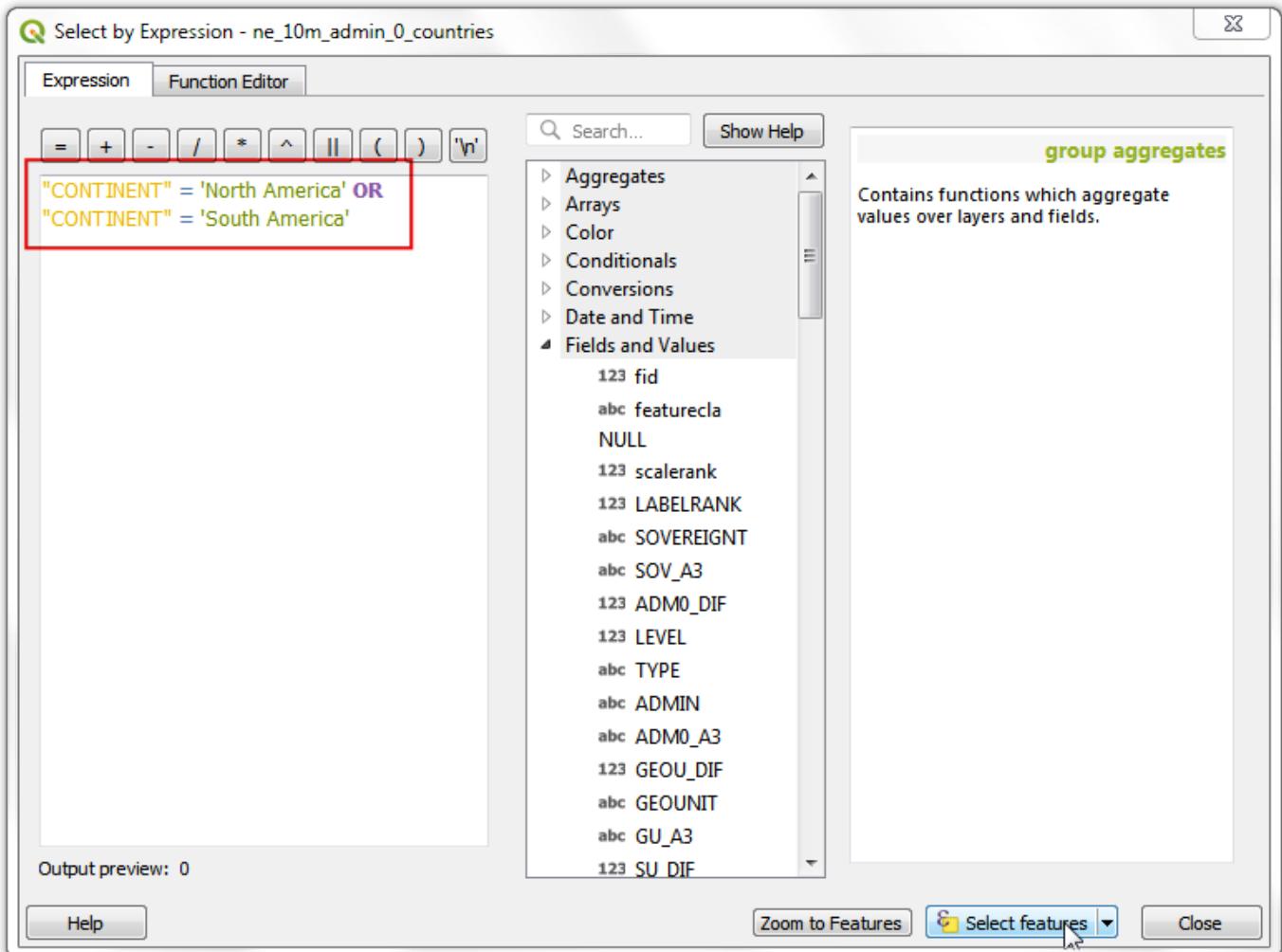


14. One another advantage of writing processing script is that the methods within the Processing Framework are aware of layer selection and automatically filter your inputs to use only the selected features. This happens because we are defining our input as a `QgsProcessingParameterFeatureSource`. A feature source allows use of ANY object which contains vector features, not just a vector layer, so when there are selected features in your layer and ask Processing to use selected features, the input is passed on to your script as a `QgsProcessingFeatureSource` object containing selected features and not the full vector layer. Here's a quick demonstration of this functionality. Let's say we want to dissolve only certain continents. Let's create a selection using Select feature by Expression tool.

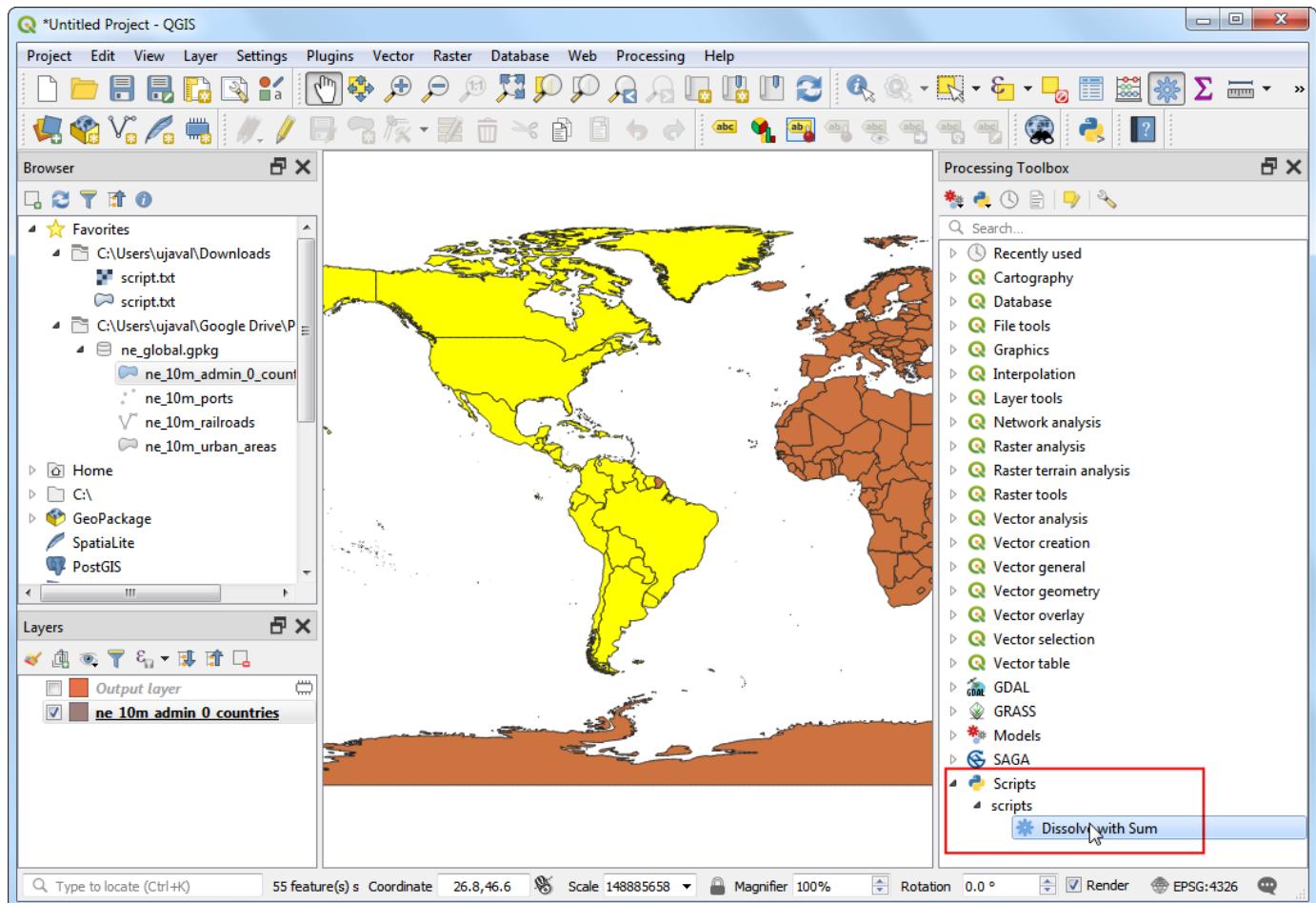


15. Enter the following expression to select features from North and South America and click Select.

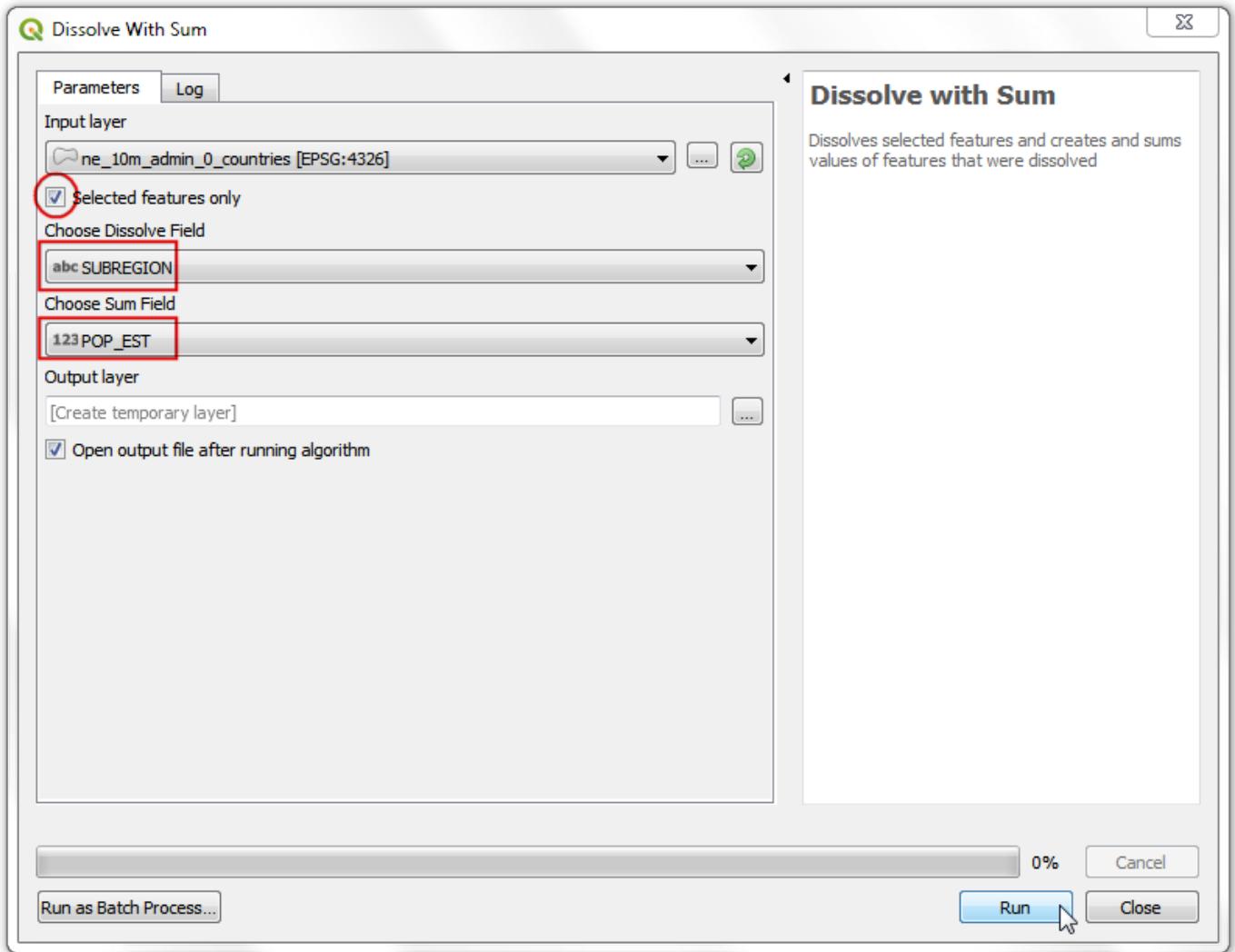
```
"CONTINENT" = 'North America' OR "CONTINENT" = 'South America'
```



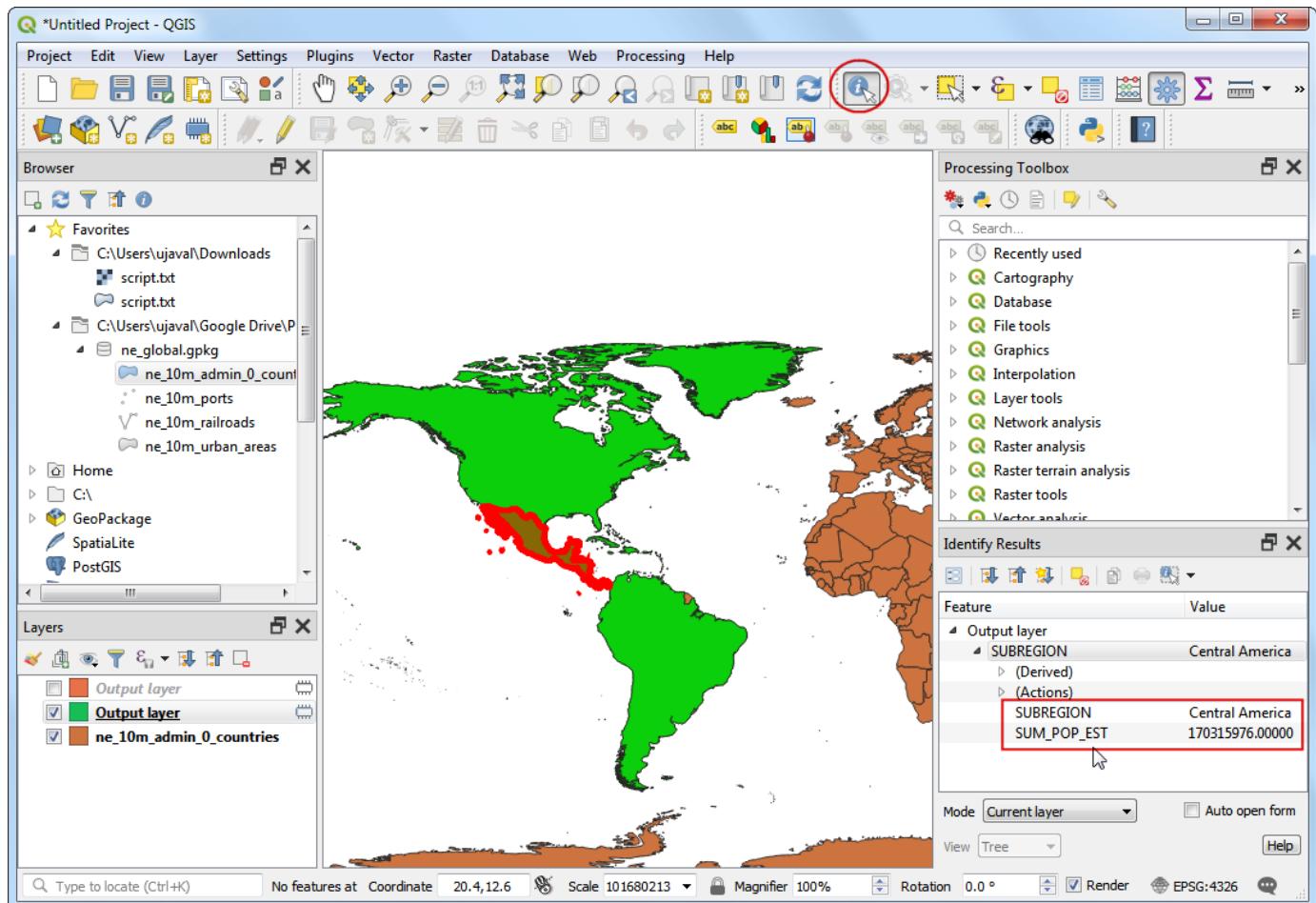
16. You will see the selected features highlighted in yellow. Locate the `dissolve_with_sum` script and double-click it to run it.



17. In the Dissolve with Sum dialog, select the ne_10m_admin_0_countries as the Input layer. This time, make sure you check the Selected features only box. Choose SUBREGION as the Dissolve field and POP_EST as the Sum field.



18. Once the processing finishes, click Close and switch back to the main QGIS window. You will notice a new layer with only the selected features dissolved. Click Identify button and click on a feature to inspect and verify that the script worked correctly.



Below is the complete script for reference. You may modify it to suit your needs.

```
# -*- coding: utf-8 -*-

"""
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
"""

from PyQt5.QtCore import QApplication, QVariant
from qgis.core import (QgsProcessing,
                       QgsFeatureSink,
                       QgsFeature,
                       QgsField,
                       QgsFields,
                       QgsProcessingException,
                       QgsProcessingAlgorithm,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterFeatureSink,
                       QgsProcessingParameterField,
                       )
import processing
```

```

class DissolveProcessingAlgorithm(QgsProcessingAlgorithm):
    """
    Dissolve algorithm that dissolves features based on selected
    attribute and summarizes the selected field by computing the
    sum of dissolved features.
    """
    INPUT = 'INPUT'
    OUTPUT = 'OUTPUT'
    DISSOLVE_FIELD = 'dissolve_field'
    SUM_FIELD = 'sum_field'

    def tr(self, string):
        """
        Returns a translatable string with the self.tr() function.
        """
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        return DissolveProcessingAlgorithm()

    def name(self):
        """
        Returns the algorithm name, used for identifying the algorithm. This
        string should be fixed for the algorithm, and must not be localised.
        The name should be unique within each provider. Names should contain
        lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return 'dissolve_with_sum'

    def displayName(self):
        """
        Returns the translated algorithm name, which should be used for any
        user-visible display of the algorithm name.
        """
        return self.tr('Dissolve with Sum')

    def group(self):
        """
        Returns the name of the group this algorithm belongs to. This string
        should be localised.
        """
        return self.tr('scripts')

    def groupId(self):
        """
        Returns the unique ID of the group this algorithm belongs to. This
        string should be fixed for the algorithm, and must not be localised.
        The group id should be unique within each provider. Group id should
        contain lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return 'scripts'

    def shortHelpString(self):
        """
        Returns a localised short helper string for the algorithm. This string
        should provide a basic description about what the algorithm does and the
        parameters and outputs associated with it..
        """
        return self.tr("Dissolves selected features and creates and sums values of features")

```

```

def initAlgorithm(self, config=None):
    """
    Here we define the inputs and output of the algorithm, along
    with some other properties.
    """

    # We add the input vector features source. It can have any kind of
    # geometry.
    self.addParameter(
        QgsProcessingParameterFeatureSource(
            self.INPUT,
            self.tr('Input layer'),
            [QgsProcessing.TypeVectorAnyGeometry]
        )
    )
    self.addParameter(
        QgsProcessingParameterField(
            self.DISSOLVE_FIELD,
            'Choose Dissolve Field',
            '',
            self.INPUT))
    self.addParameter(
        QgsProcessingParameterField(
            self.SUM_FIELD,
            'Choose Sum Field',
            '',
            self.INPUT))
    # We add a feature sink in which to store our processed features (this
    # usually takes the form of a newly created vector layer when the
    # algorithm is run in QGIS).
    self.addParameter(
        QgsProcessingParameterFeatureSink(
            self.OUTPUT,
            self.tr('Output layer')
        )
    )

def processAlgorithm(self, parameters, context, feedback):
    """
    Here is where the processing itself takes place.
    """

    source = self.parameterAsSource(
        parameters,
        self.INPUT,
        context
    )
    dissolve_field = self.parameterAsString(
        parameters,
        self.DISSOLVE_FIELD,
        context)
    sum_field = self.parameterAsString(
        parameters,
        self.SUM_FIELD,
        context)

    fields = QgsFields()
    fields.append(QgsField(dissolve_field, QVariant.String))
    fields.append(QgsField('SUM_' + sum_field, QVariant.Double))

    (sink, dest_id) = self.parameterAsSink(
        parameters,

```

```

    self.OUTPUT,
    context, fields, source.wkbType(), source.sourceCrs())

# Create a dictionary to hold the unique values from the
# dissolve_field and the sum of the values from the sum_field
feedback.pushInfo('Extracting unique values from dissolve_field and computing sum')
features = source.getFeatures()
unique_values = set(f[dissolve_field] for f in features)
# Get Indices of dissolve field and sum field
dissolveIdx = source.fields().indexFromName(dissolve_field)
sumIdx = source.fields().indexFromName(sum_field)

# Find all unique values for the given dissolve_field and
# sum the corresponding values from the sum_field
sum_unique_values = {}
attrs = [{dissolve_field: f[dissolveIdx], sum_field: f[sumIdx]}]
    for f in source.getFeatures()]
for unique_value in unique_values:
    val_list = [f_attr[sum_field]
        for f_attr in attrs if f_attr[dissolve_field] == unique_value]
    sum_unique_values[unique_value] = sum(val_list)

# Running the processing dissolve algorithm
feedback.pushInfo('Dissolving features')
dissolved_layer = processing.run("native:dissolve", {
    'INPUT': parameters[self.INPUT],
    'FIELD': dissolve_field,
    'OUTPUT': 'memory:'
}, context=context, feedback=feedback)['OUTPUT']

# Read the dissolved layer and create output features
for f in dissolved_layer.getFeatures():
    new_feature = QgsFeature()
    # Set geometry to dissolved geometry
    new_feature.setGeometry(f.geometry())
    # Set attributes from sum_unique_values dictionary that we had computed
    new_feature.setAttributes([f[dissolve_field], sum_unique_values[f[dissolve_field]]])
    sink.addFeature(new_feature, QgsFeatureSink.FastInsert)

return {self.OUTPUT: dest_id}

```