# Spatial Data Visualization and Analytics

A modern introduction to working with spatial datasets

*Ujaval Gandhi*

## Contents

**Introduction** **4**

Why do we care about location? . . . . . 4

Location data is everywhere . . . . . 4

Spatial Data Model . . . . . 4

Spatial Data Formats . . . . . 5

Spatial Data Types . . . . . 5

**Map Projections** **5**

**Introduction to QGIS** **5**

Plugins . . . . . 5

**Points** **6**

Exercise: Mapping Air Quality . . . . . 6

**Lines** **28**

Exercise: Visualize GPS Tracks . . . . . 28

**Polygons** **40**

Exercise: Mapping Census Data . . . . . 40

**Point Clouds** **51**

Exercise: View Point Cloud from an Aerial Survey . . . . . 52

**Raster - Photos** **54**

Exercise: View Drone Imagery . . . . . 54

**Raster - Satellite Images** **57**

Exercise: View Sentinel-2 Image and Create Composites . . . . . 57

**Raster - Elevation Data** **61**

Exercise: Analyze Metro Rail Accesiblity . . . . . 61

Exercise: Uber Movement Analysis . . . . . 61

**Data Credits** **77**

**License** **78**

---

**This course is also offered as an in-person class. If you would like to attend one of my workshops, visit www.spatialthoughts.com/events to know details of upcoming classes. You may also sign up for my mailing list to know when new sessions are scheduled.**

---

# Introduction

## Why do we care about location?

> "Everything is related to everything else, but near things are more related than distant things." - Waldo Tobler's First Law of Geopraphy

When modeling and analyzing our world, location is a critical factor. A non-spatial model cannot accurately reflect the processes and interactions happening in our world. Take this example - predicting housing prices - where a spatial prediction model performed much better than a purely non-spatial one.

## Location data is everywhere

Today the availability of location data - both for individuals and businesses - has exploded. Spatial data adds another dimension to data, and reveals patterns that are otherwise not obvious.

Individuals - with GPS sensors on their smartphones - have the ability to tag their data with location. Photos taken with smartphones have the location embedded in it. If opted-in, one can store and access their location history on an ongoing basis.

Most businesses have location data in one form or the other. Customer addresses, IP-locations of website visitors, sales territories, supply chain routes and so on. For other businesses - such as taxi aggregators, food delivery, logistics - generate huge amounts of location data that can be mined for intelligence.

IoT (Internet-Of-Things) devices are collecting location data continuously alongside with sensor data.

Governments are also increasing collecting and sharing location based data. Data relating to urban infrastructure, census, LIDAR and aerial imagery etc. are being collected at massive scale. Many governments have implemented open data sharing policies - making this data available to individuals and businesses to use.

## Spatial Data Model

geometry + properties

shape + table

geometry = coordinates + reference properties = type + data

A GeoJSON representation

```
{
     "type": "Feature",
     "geometry": {
       "type": "Point",
       "coordinates": [ 77.58270263671875, 12.963074139604124]
     },
     "properties": {
```

```
        "name": "Bengaluru"
      }
  }
```

### Spatial Data Formats

| Type | Non-Spatial Data | Spatial Data |
|---|---|---|
| Text | csv, json, xml | csv, geojson, gml, kml |
| Binary/Compressed | xlsx, zip | shapefile, geopackage |
| Images | tiff, jpg, png | geotiff, jpeg2000 |
| Databases | SQLite, PostgreSQL, Oracle | Spatialite, PostGIS, Oracle Spatial |

### Spatial Data Types

| Type | Sub Types | Examples |
|---|---|---|
| Vector | Point | |
| | Line | |
| | Polygons | |
| | Point Cloud | LIDAR surveys |
| Raster | Photos | Aerial and Drone Photos |
| | Grids | Satellite Imagery, Elevation Data, Thematic Data |
| Mesh | Mesh | |
| Tiles | Raster Tile Layers | TMS / WMS |
| | Vector Tile Layers | |

## Map Projections

https://bl.ocks.org/syntagmatic/ba569633d51ebec6ec6e

EPSG: 7755 WGS 84 / India NSF LCC NATIONAL SPATIAL FRAMEWORK

https://giscourses.cfans.umn.edu/sites/giscourses.cfans.umn.edu/files/5480chapter3_projections_excerpt.pdf

## Introduction to QGIS

### Plugins

- QuickMapServices
- TimeManager
- QuickOSM

# Points

The simplest representation of spatial data can be done using a table. A place can be represented using a pair of coordinates - Latitude and Longitude - with other attribute information about the place. Many spatial data source come in this form. Excel sheets, CSV files, database tables etc.

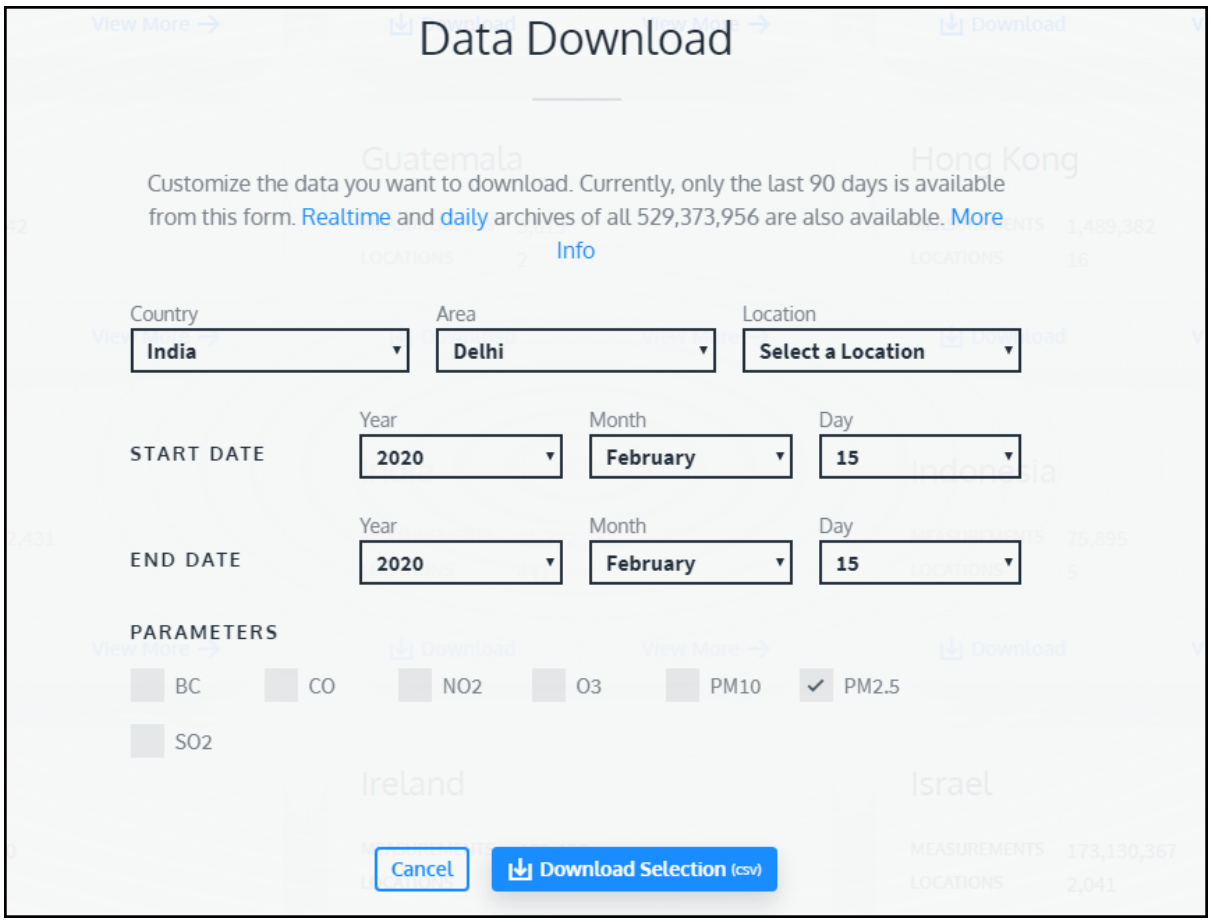
## Exercise: Mapping Air Quality

Worsening air quality is a severe problem in many countries around the world. India - particularly - Delhi suffers from acute problems of high pollution levels.One of the first steps to better understand the problem, is to have continuous monitoring of air quality across the cities. Many organizations have stepped up and setup such sensors that collect air quality data and make it publicly available. OpenAQ is a platform that collects this data from all public sources and makes it available in an easy to use form.

> If you are interested in this topic, Urban Emissions has a lot of relevant information and datasets for India.

We will take the sensor data for PM2.5 concentrations 1 day and map it. The aim is to turn this tabular data info an informative spatial data visualization.

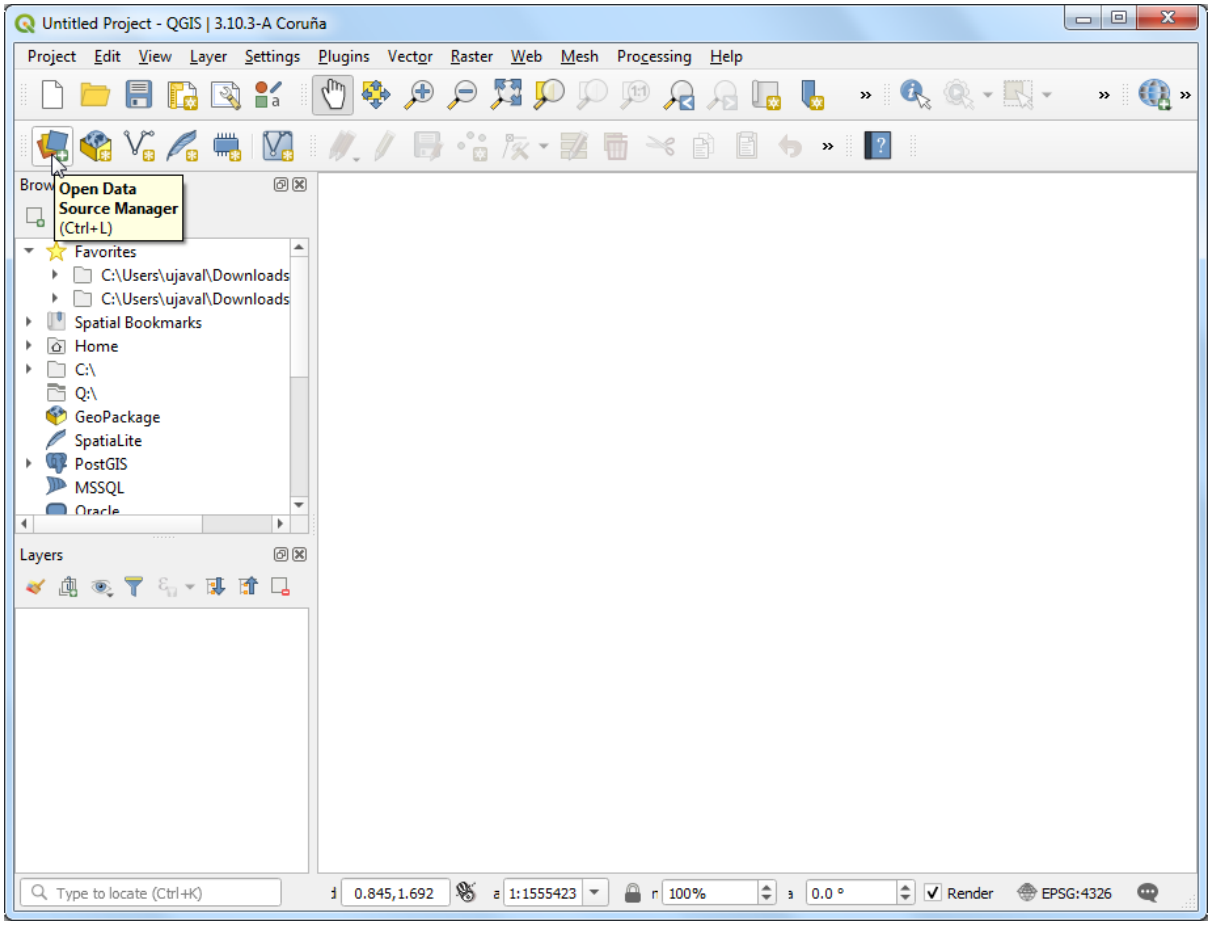
For this exercise, we are using daily average data for Delhi, India for February 15, 2020. This data was downloaded from OpenAQ Data Download



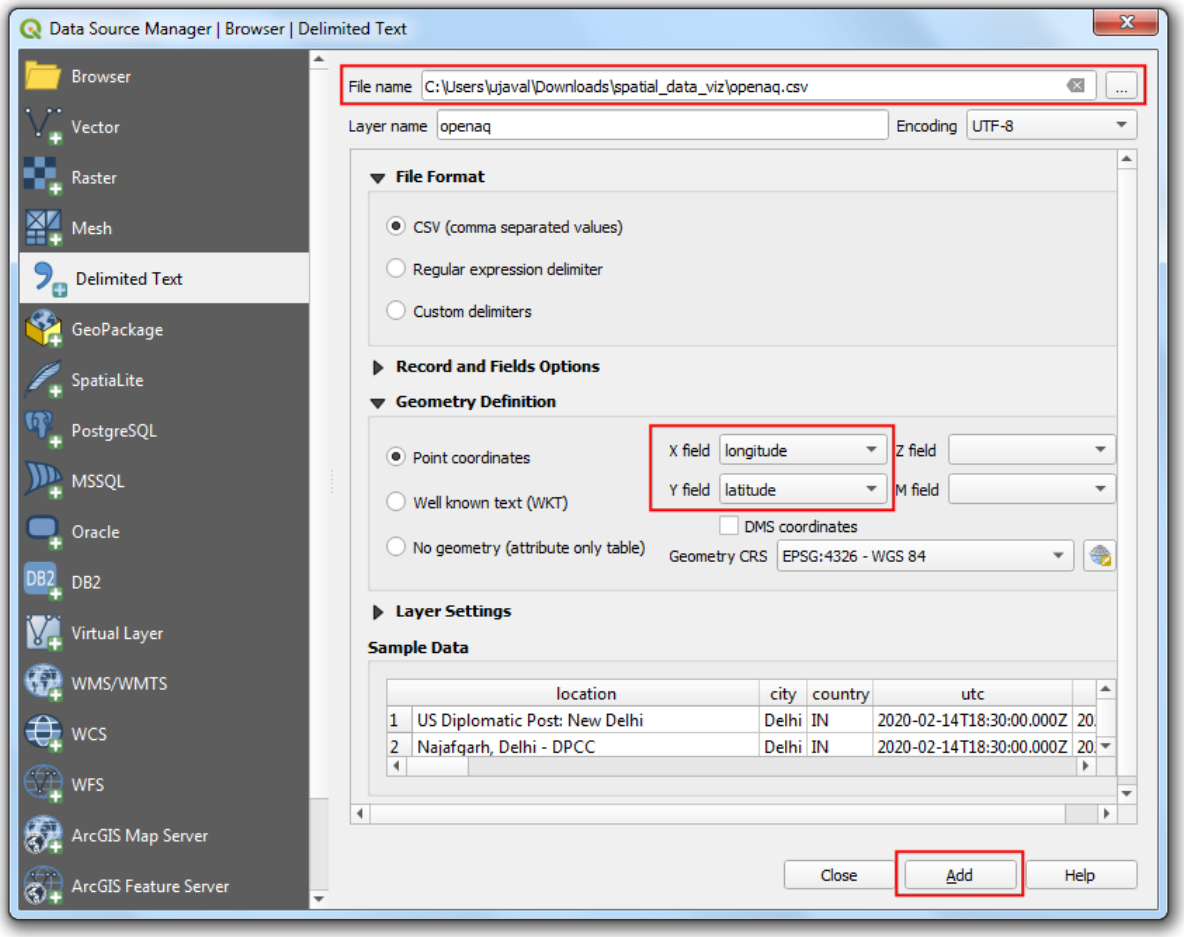
Open the `openaq.csv` file in a text editor and examine it. Each row of data contains data from 1 monitoring station. The `latitude` and `longitude` column contain the coordinates of the station and the `value` contains the daily average PM2.5 concentration

| location | city | country | utc | local | parameter | value | unit | latitude | longitude | attribution |
|---|---|---|---|---|---|---|---|---|---|---|
| US Diplomatic Post: New Delhi | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 67 | µg/m³ | 28.63576 | 77.22445 | {"name":"EPA AirNow I |
| Najafgarh, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 195 | µg/m³ | 28.570173 | 76.933762 | {"name":"Central Pollu |
| Alipur, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 85 | µg/m³ | 28.815329 | 77.15301 | {"name":"Central Pollu |
| Mundka, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 68 | µg/m³ | 28.684678 | 77.076574 | {"name":"Central Pollu |
| Sri Aurobindo Marg, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 85 | µg/m³ | 28.531346 | 77.190156 | {"name":"Central Pollu |
| Narela, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 101 | µg/m³ | 28.822836 | 77.101981 | {"name":"Central Pollu |
| Vivek Vihar, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 84 | µg/m³ | 28.672342 | 77.31526 | {"name":"Central Pollu |
| Okhla Phase-2, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 102 | µg/m³ | 28.530785 | 77.271255 | {"name":"Central Pollu |
| Ashok Vihar, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 74 | µg/m³ | 28.695381 | 77.181665 | {"name":"Central Pollu |
| Dr. Karni Singh Shooting Range, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 68 | µg/m³ | 28.498571 | 77.26484 | {"name":"Central Pollu |
| Punjabi Bagh, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 63 | µg/m³ | 28.674045 | 77.131023 | {"name":"Central Pollu |
| R K Puram, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 72 | µg/m³ | 28.563262 | 77.186937 | {"name":"Central Pollu |
| Mandir Marg, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 70 | µg/m³ | 28.636429 | 77.201067 | {"name":"Central Pollu |
| Bawana, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 150 | µg/m³ | 28.7762 | 77.051074 | {"name":"Central Pollu |
| Sonia Vihar, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 77 | µg/m³ | 28.710508 | 77.249485 | {"name":"Central Pollu |
| Patparganj, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 52 | µg/m³ | 28.623748 | 77.287205 | {"name":"Central Pollu |
| Nehru Nagar, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 59 | µg/m³ | 28.56789 | 77.250515 | {"name":"Central Pollu |
| Jawaharlal Nehru Stadium, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 57 | µg/m³ | 28.58028 | 77.233829 | {"name":"Central Pollu |
| Jahangirpuri, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 106 | µg/m³ | 28.73282 | 77.170633 | {"name":"Central Pollu |
| Dwarka-Sector 8, Delhi - DPCC | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 103 | µg/m³ | 28.5710274 | 77.0719006 | {"name":"Central Pollu |
| NSIT Dwarka, Delhi - CPCB | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 118.51 | µg/m³ | 28.60909 | 77.0325413 | {"name":"Central Pollu |
| ITO, Delhi - CPCB | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 53 | µg/m³ | 28.628624 | 77.24106 | {"name":"Central Pollu |
| DTU, Delhi - CPCB | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 133.45 | µg/m³ | 28.7500499 | 77.1112615 | {"name":"Central Pollu |
| Shadipur, Delhi - CPCB | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 63.41 | µg/m³ | 28.6514781 | 77.1473105 | {"name":"Central Pollu |
| IHBAS, Dilshad Garden, Delhi - CPCB | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 52.4 | µg/m³ | 28.6811736 | 77.3025234 | {"name":"Central Pollu |
| North Campus, DU, Delhi - IMD | Delhi | IN | 2020-02-14T18:30:00.000Z | 2020-02-15T00:00:00+05:30 | pm25 | 37.95 | µg/m³ | 28.6573814 | 77.1585447 | {"name":"Central Pollu |

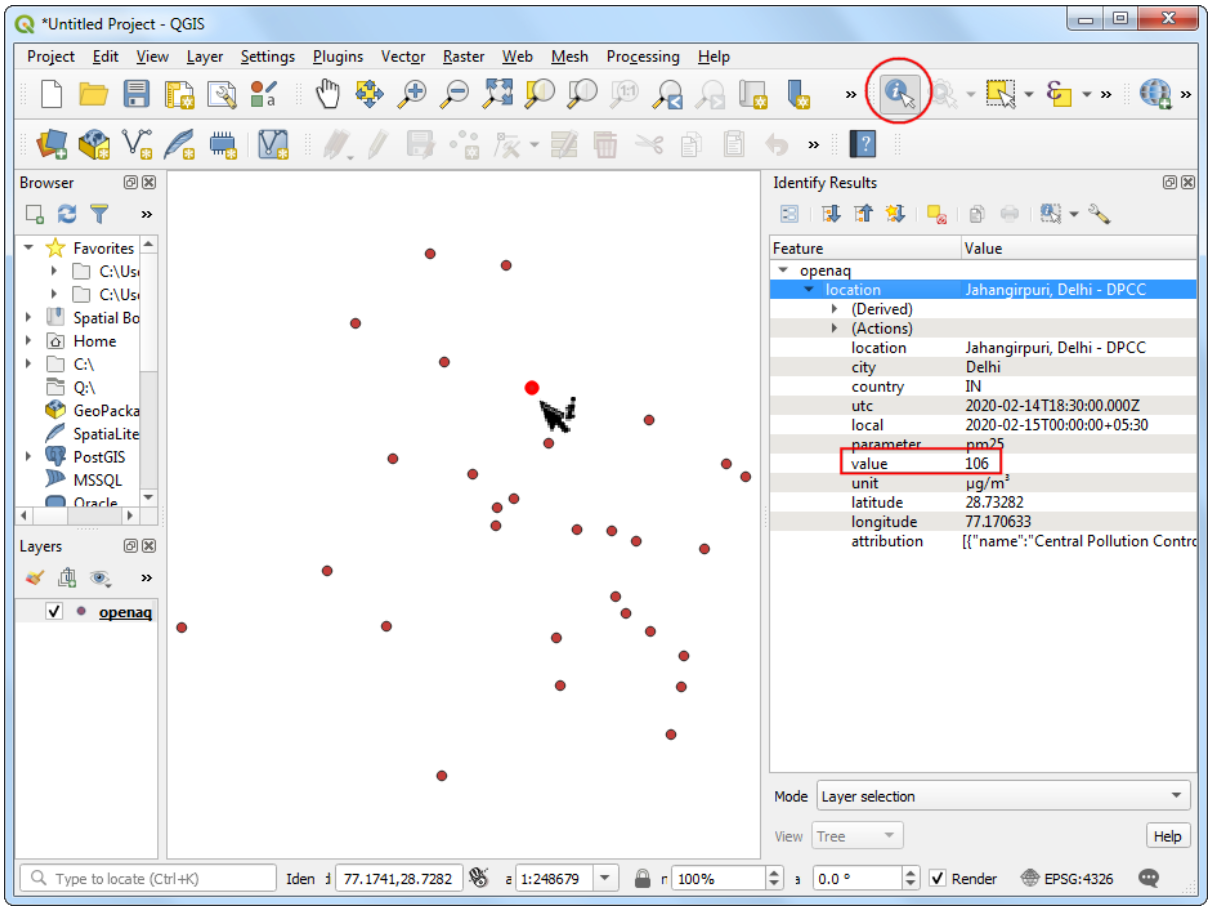
Tabular data in text files fall into the category of **Delimited Text** files, such as this can be imported in QGIS via *Data Source Manager*. Click *Open Data Source Manager* button.



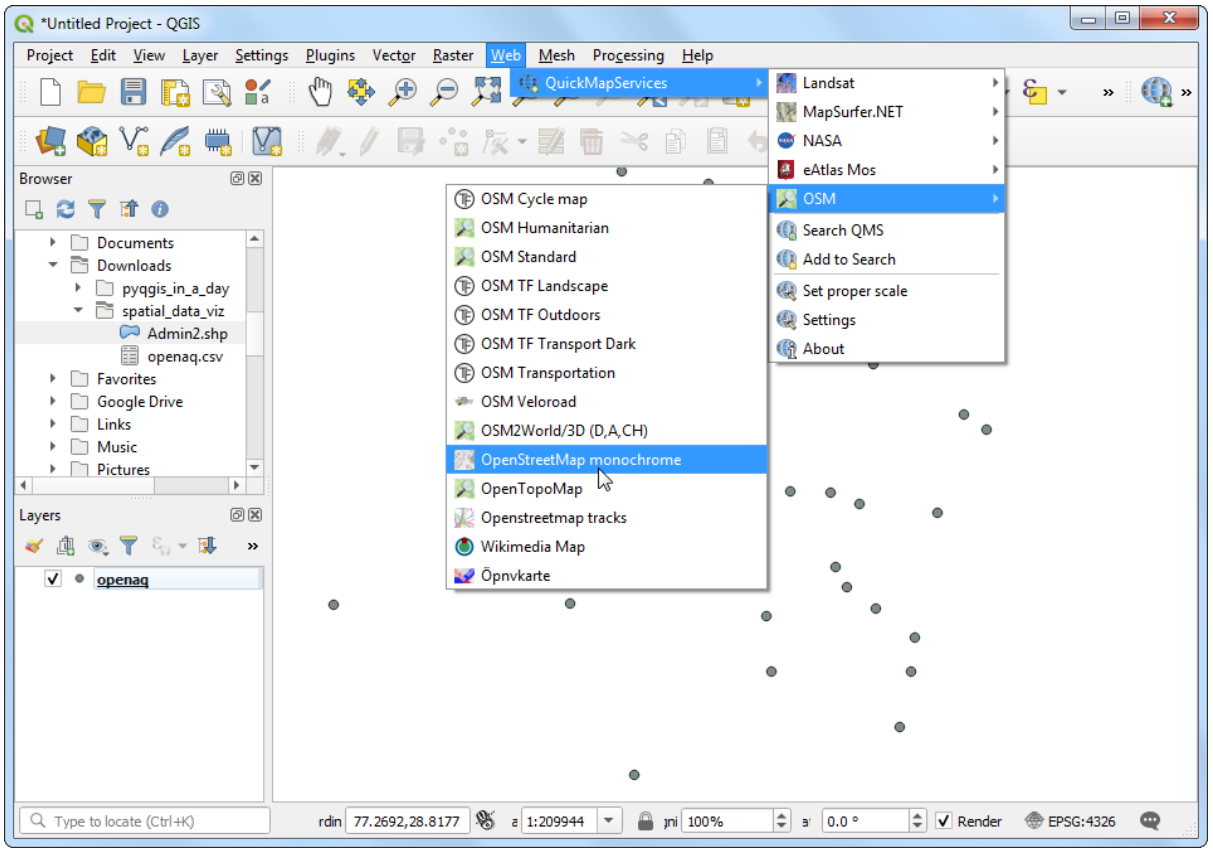
Browse to the `openaq.csv` file and open it. As we want to import this file as points, select *Point coordinates*. Choose `longitude` as *X Field* and `latitude` as *Y Field*. Click *Add*.



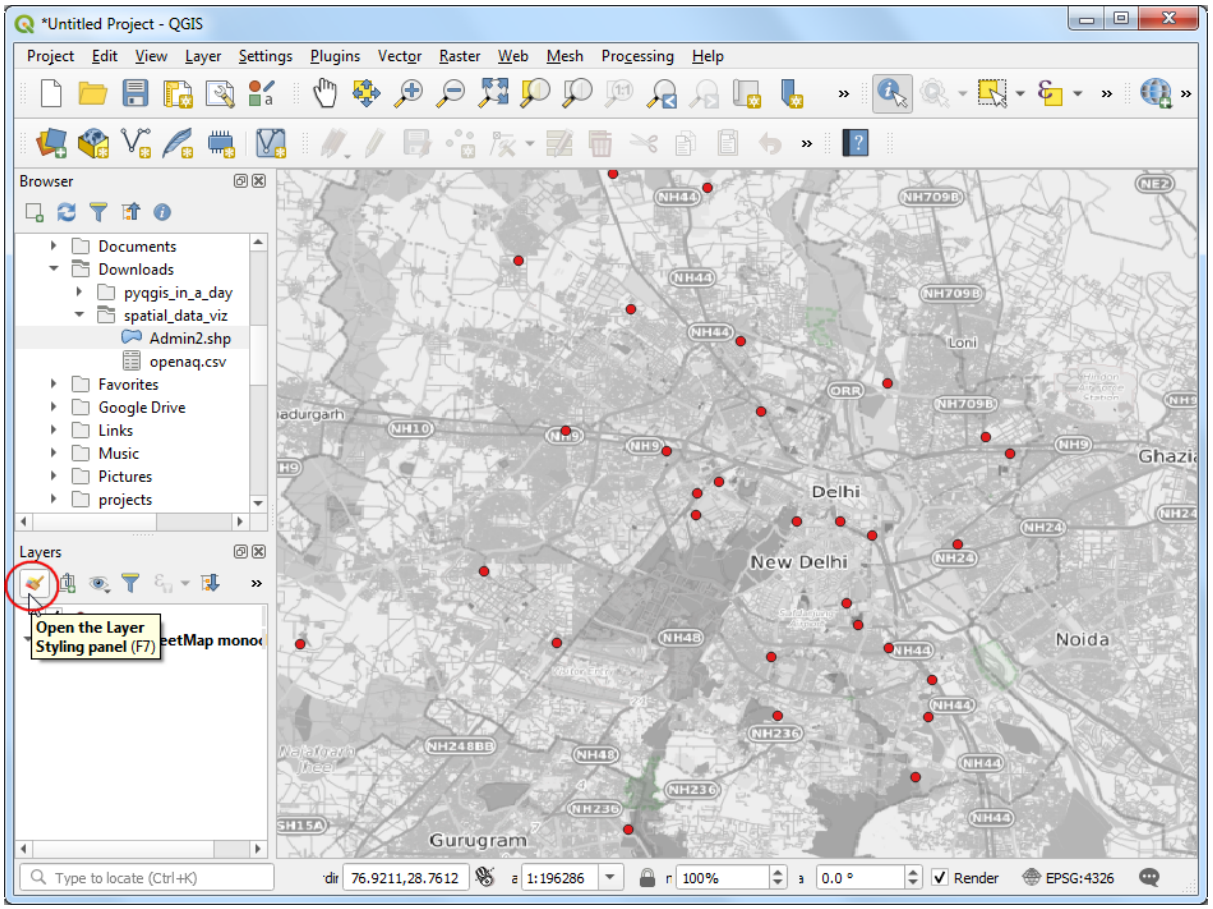

You will see the tabular data now loaded in QGIS canvas as a spatial data layer. Use the *Identify* button and click on any of the point. You will see the attribute data that is attached to each point.




Through we can see the point distribution is across the city of Delhi, we are missing the context on where is point is located. A base-map layer will help us understand this data better. The **QuickMapServices** plugin gives us ready access to many different types of base-maps. Go to **Web → QuickMapServices → OSM → OpenStreetMap monochrome** layer.

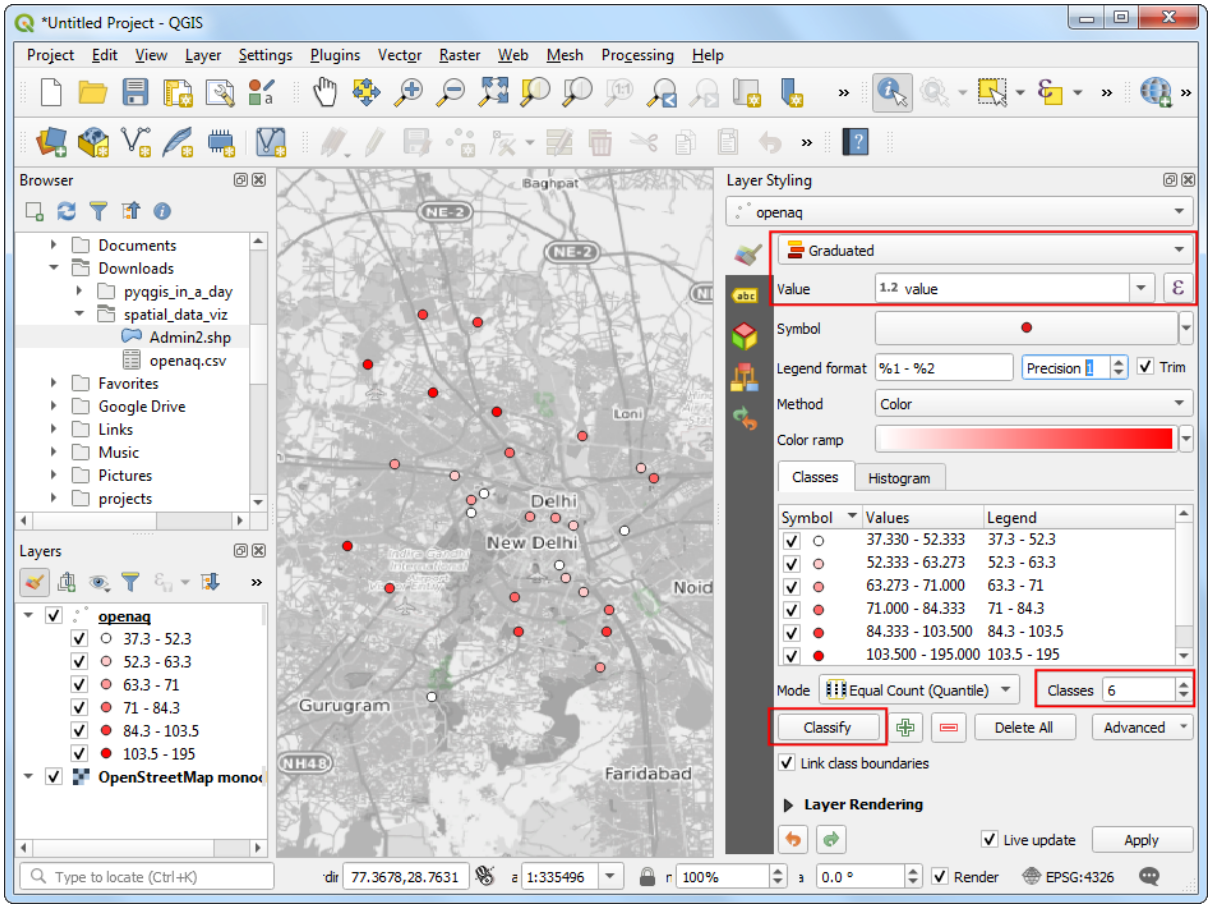



A new layer will get added to the *Layers* panel and the *Canvas*. Now you can see the points in the context of the city and surroundings. Let's style the point layer better now. Click *Open the Layer Styling Panel*.




We will color each point according to the observed PM2.5 value. Choose `Graduated`

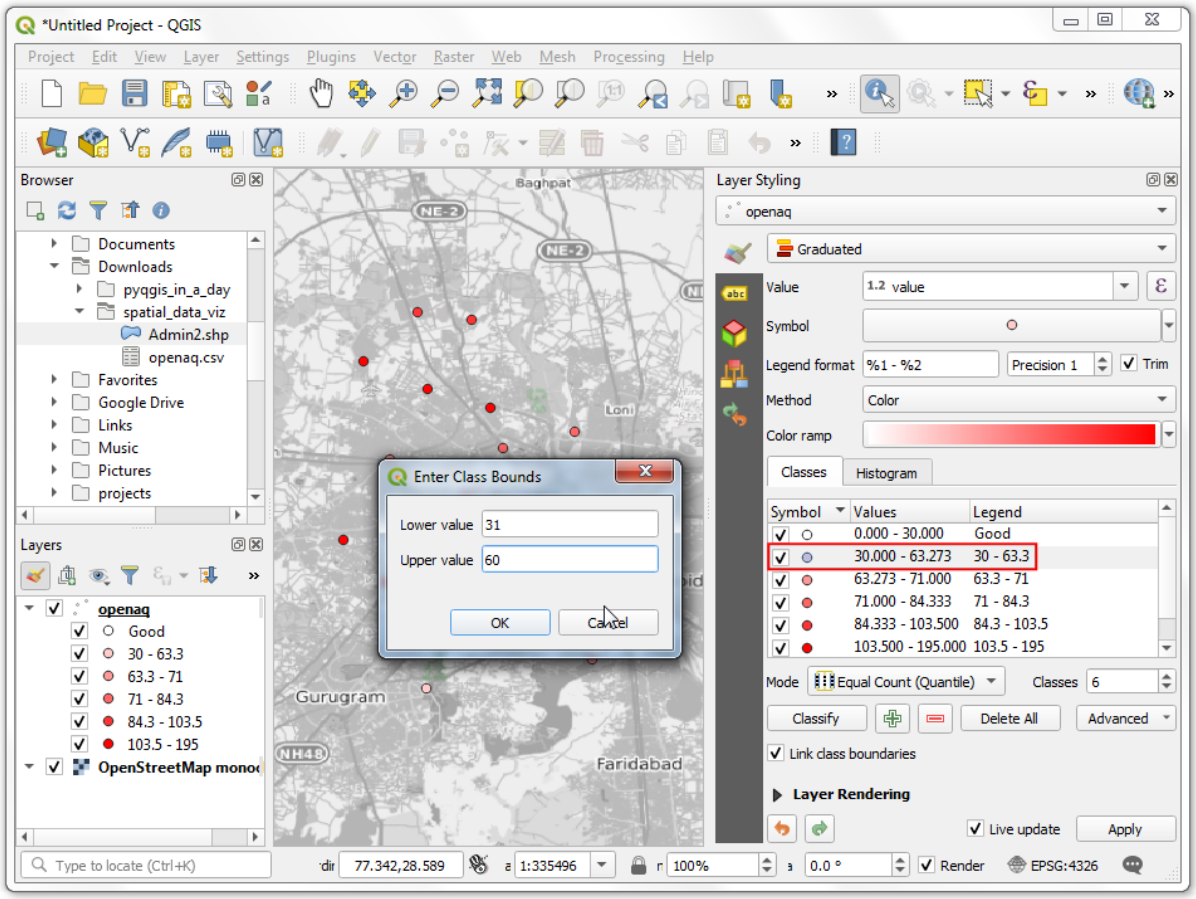
renderer and `value` as the *Value* column. Set the number of *Classes* to `6` and click *Classify*.



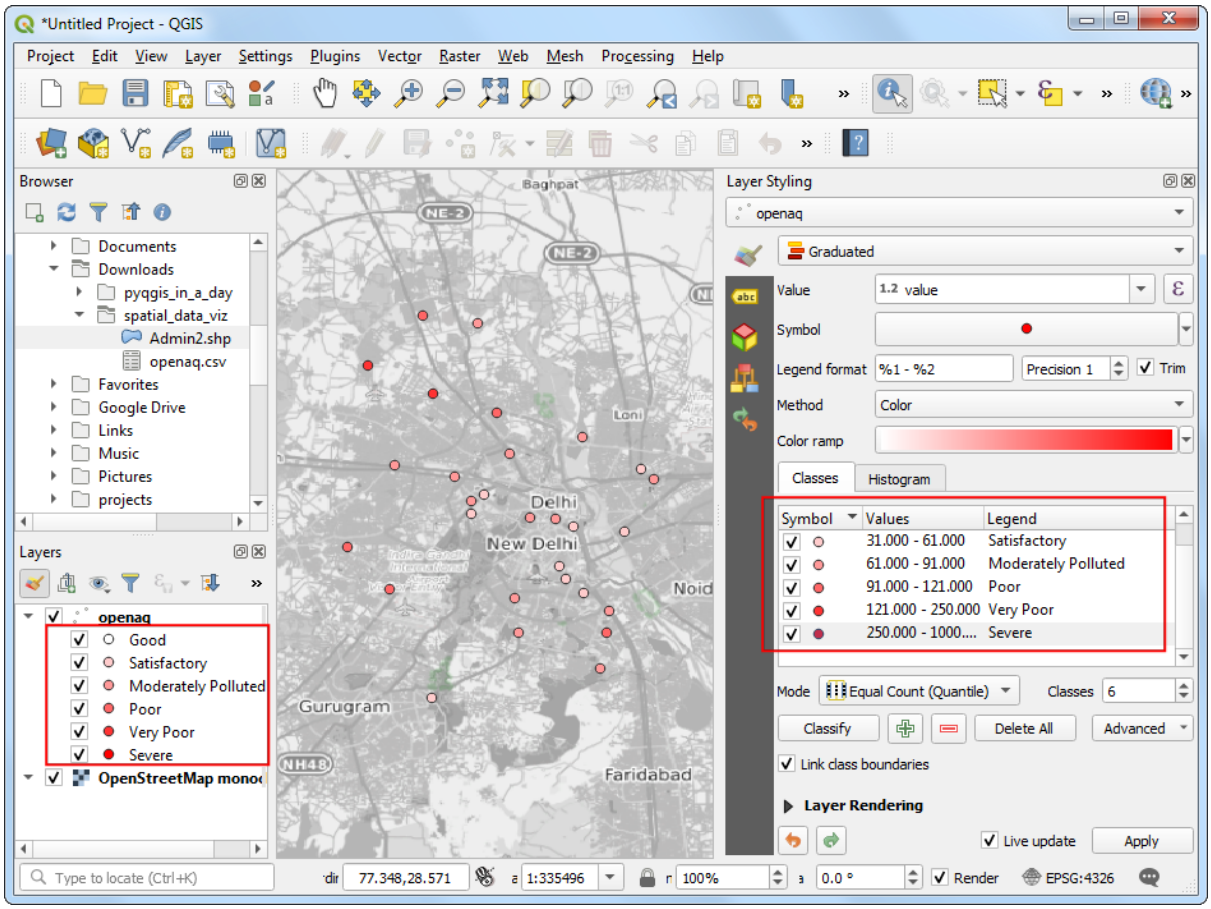

For the class ranges to have some meaning, we need to link them to the commonly used scale. India has adopted National Air Quality Index with the following definitions.

| AQI Category, Pollutants and Health Breakpoints | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **AQI Category (Range)** | **$PM_{10}$ 24-hr** | **$PM_{2.5}$ 24-hr** | **$NO_2$ 24-hr** | **$O_3$ 8-hr** | **CO 8-hr ($mg/m^3$)** | **$SO_2$ 24-hr** | **$NH_3$ 24-hr** | **Pb 24-hr** |
| **Good (0-50)** | **0-50** | **0-30** | **0-40** | **0-50** | **0-1.0** | **0-40** | **0-200** | **0-0.5** |
| **Satisfactory (51-100)** | **51-100** | **31-60** | **41-80** | **51-100** | **1.1-2.0** | **41-80** | **201-400** | **0.5 – 1.0** |
| **Moderately polluted (101-200)** | **101-250** | **61-90** | **81-180** | **101-168** | **2.1- 10** | **81-380** | **401-800** | **1.1-2.0** |
| **Poor (201-300)** | **251-350** | **91-120** | **181-280** | **169-208** | **10-17** | **381-800** | **801-1200** | **2.1-3.0** |
| **Very poor (301-400)** | **351-430** | **121-250** | **281-400** | **209-748*** | **17-34** | **801-1600** | **1200-1800** | **3.1-3.5** |
| **Severe (401-500)** | **430 +** | **250+** | **400+** | **748+*** | **34+** | **1600+** | **1800+** | **3.5+** |

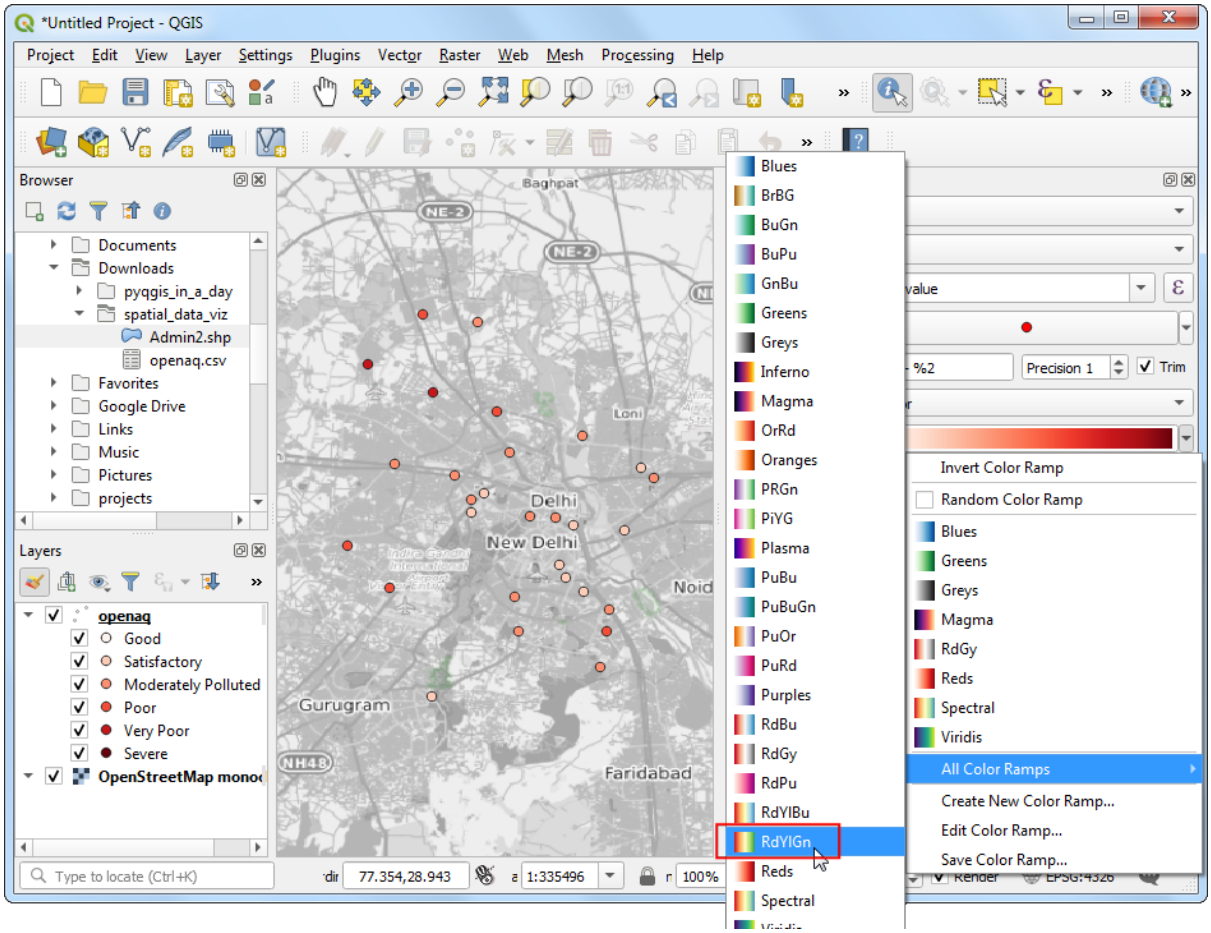
Let's adjust the class values to match those defined in the National Air Quality Index. We can also change the *Legend* labels to the human-readable category names. You can double-click each class range and edit it.



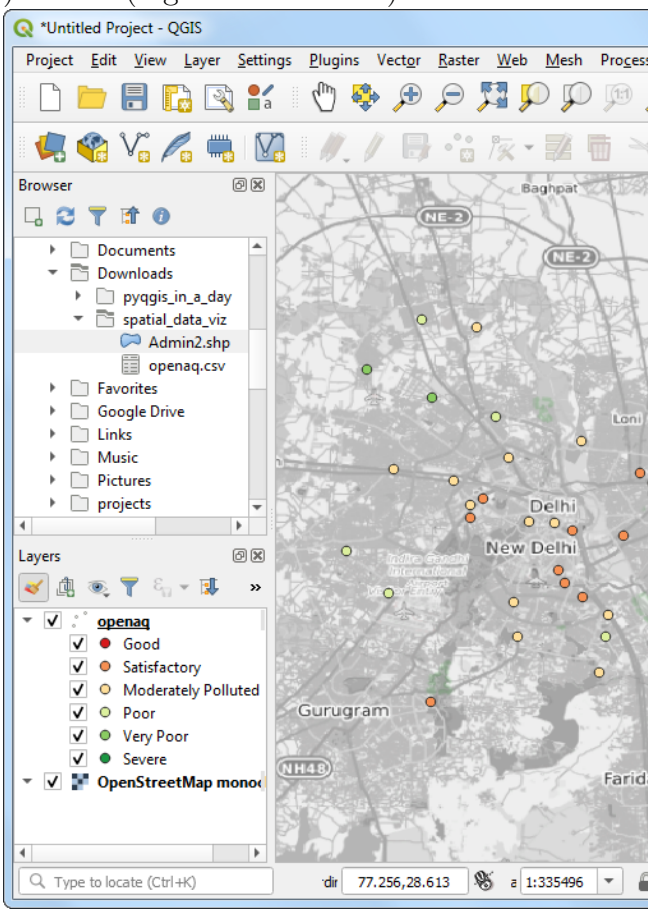

As you edit the categories, the map visualization will change accordingly. The layer legend will also show the legend labels now.

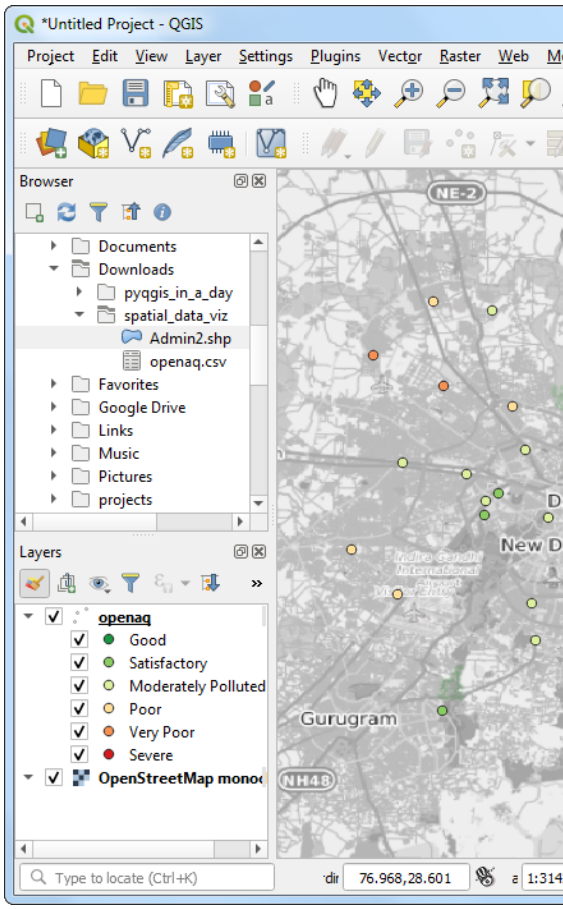



We can change the color also to match those defined in the index. Select the dropdown next to the color ramp and select the `RdYlGn` (Red-Yellow-Green) ramp.



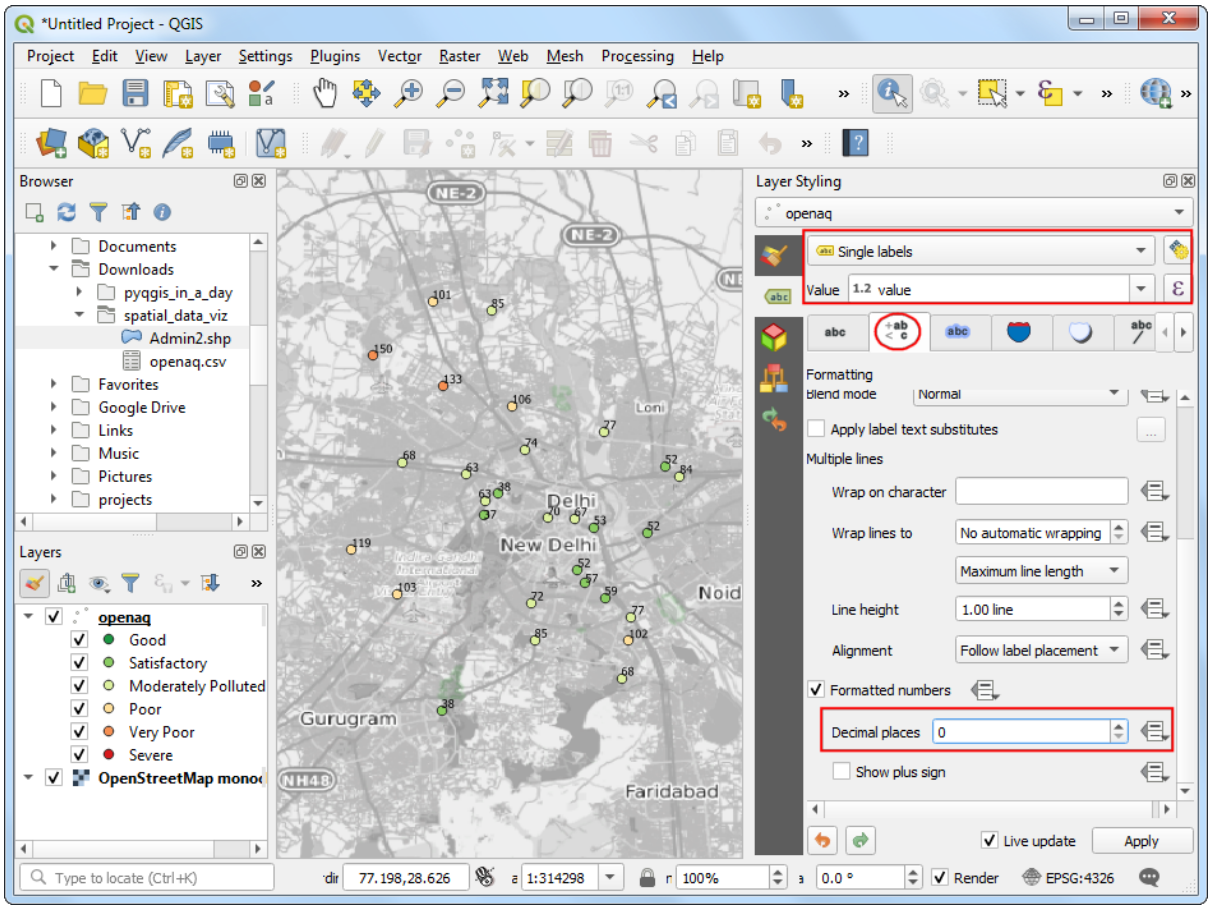
We want the ramp to go from Green (low PM2.5 values) to Red (high PM2.5 values) - so click the dropdown again and select *Invert Color Ramp*.



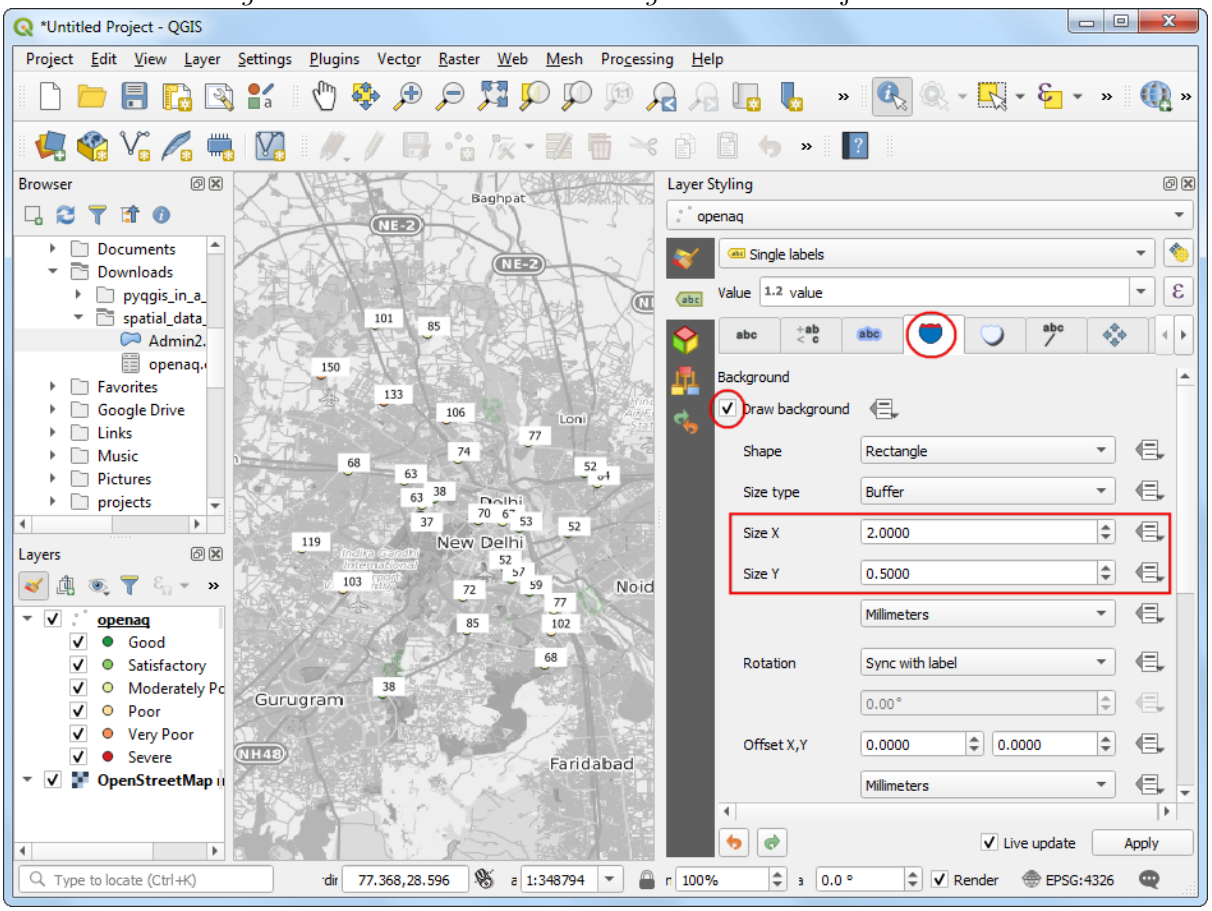




We will add labels to the points now. Switch to the *Labels* tab.

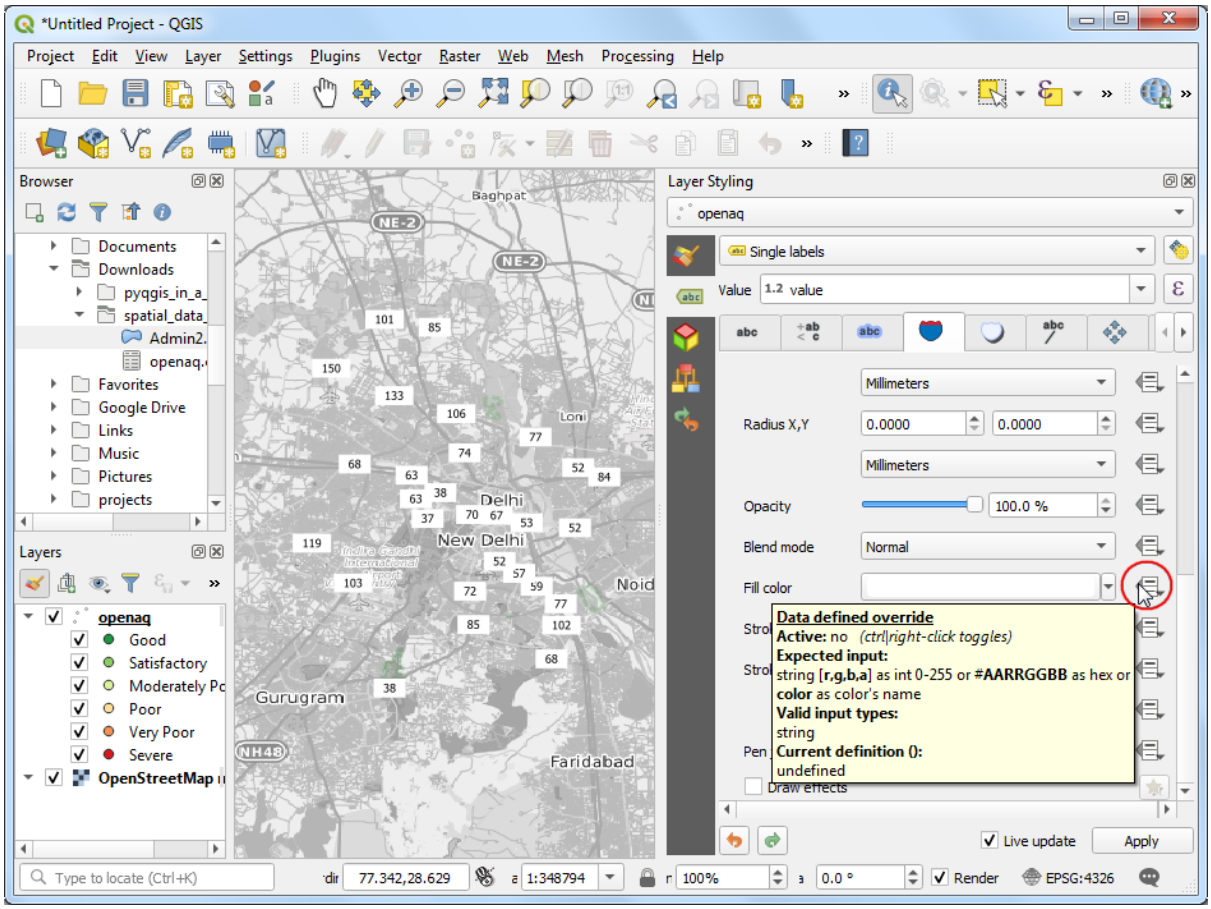
Choose `Single labels` and `value` as *Value*. Scroll down and check *Formatted numbers* and change the *Decimal places* to `0`.



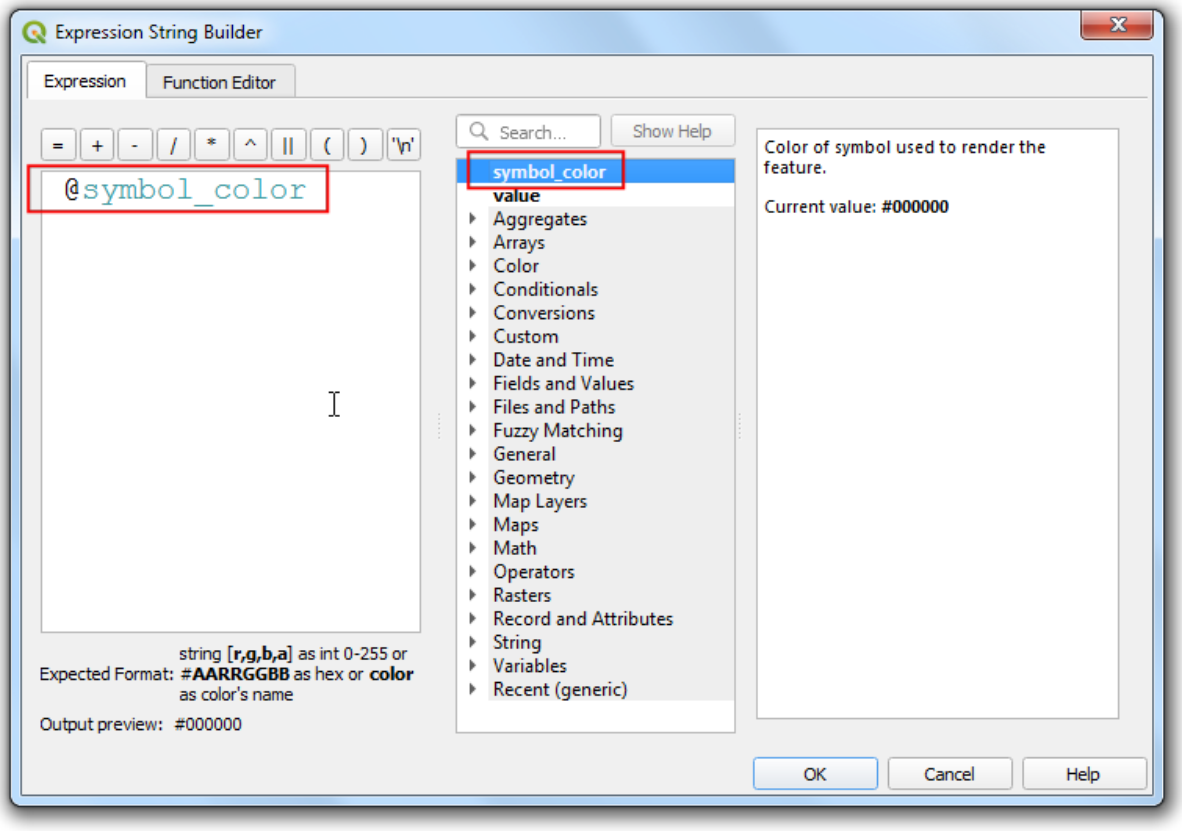

Next, switch to the *Background* tab. Check *Draw background* and adjust the *Size X* and *Size Y*.



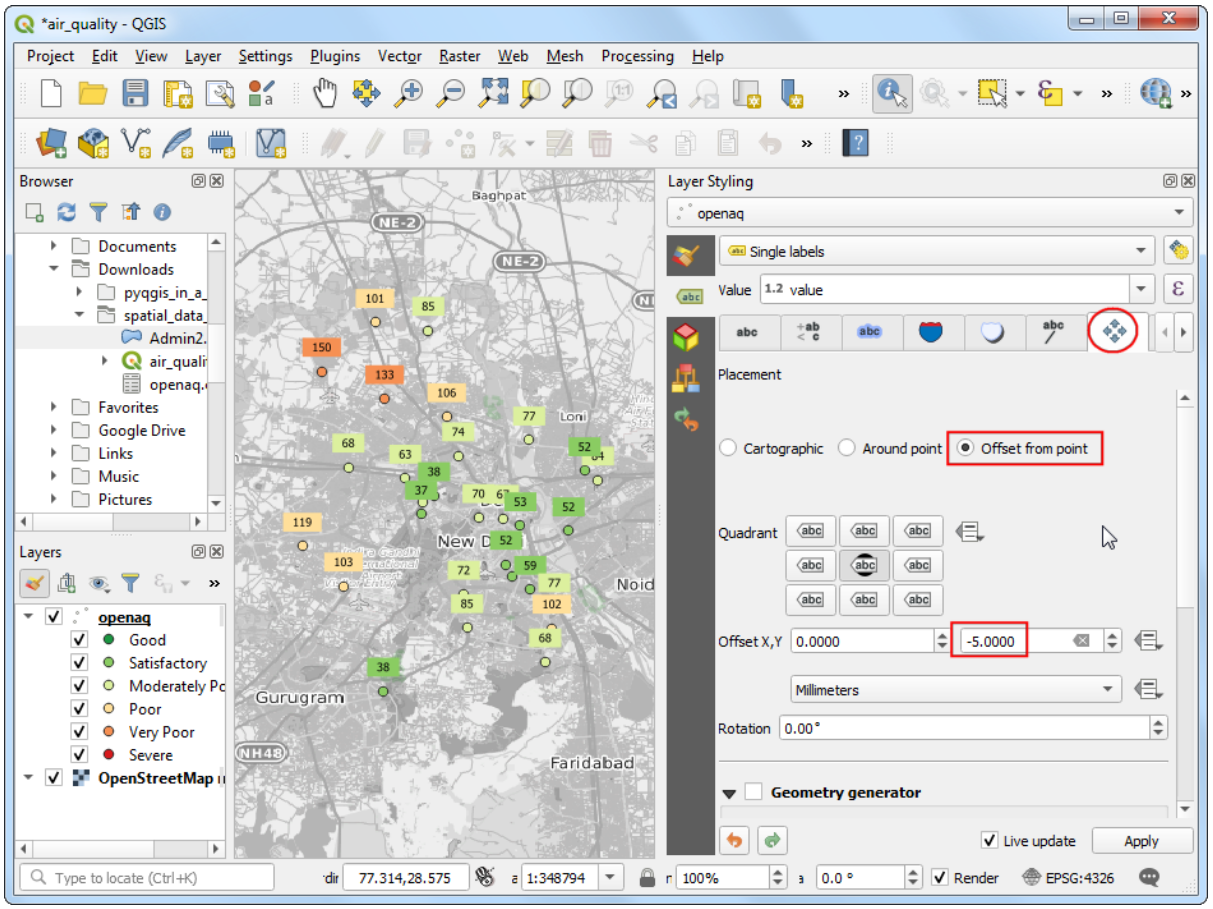

Scroll down to the *Fill color* section. We want the fill color for each box to match the color of the associated point. Click the *Data defined override* button and choose *Edit*.



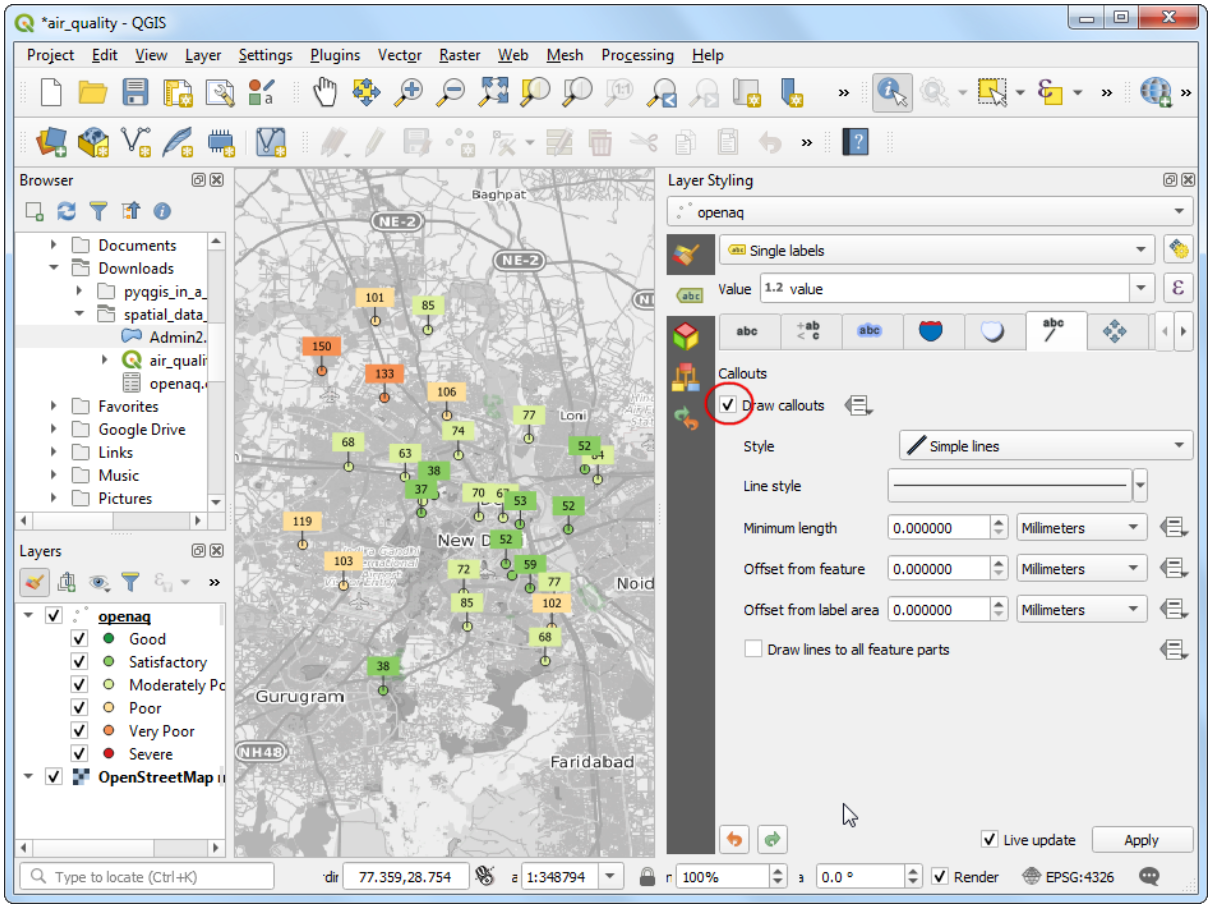

Double-click the `@symbol_color` variable to add it to the expression. Click OK.




You will see the map change and the label shields will have the color matching the category based on the values. We will move the labels slightly above so the points are also seen. Switch to the *Placement* tab and select *Offset from point*. Check the *Offset Y* to `-5`.

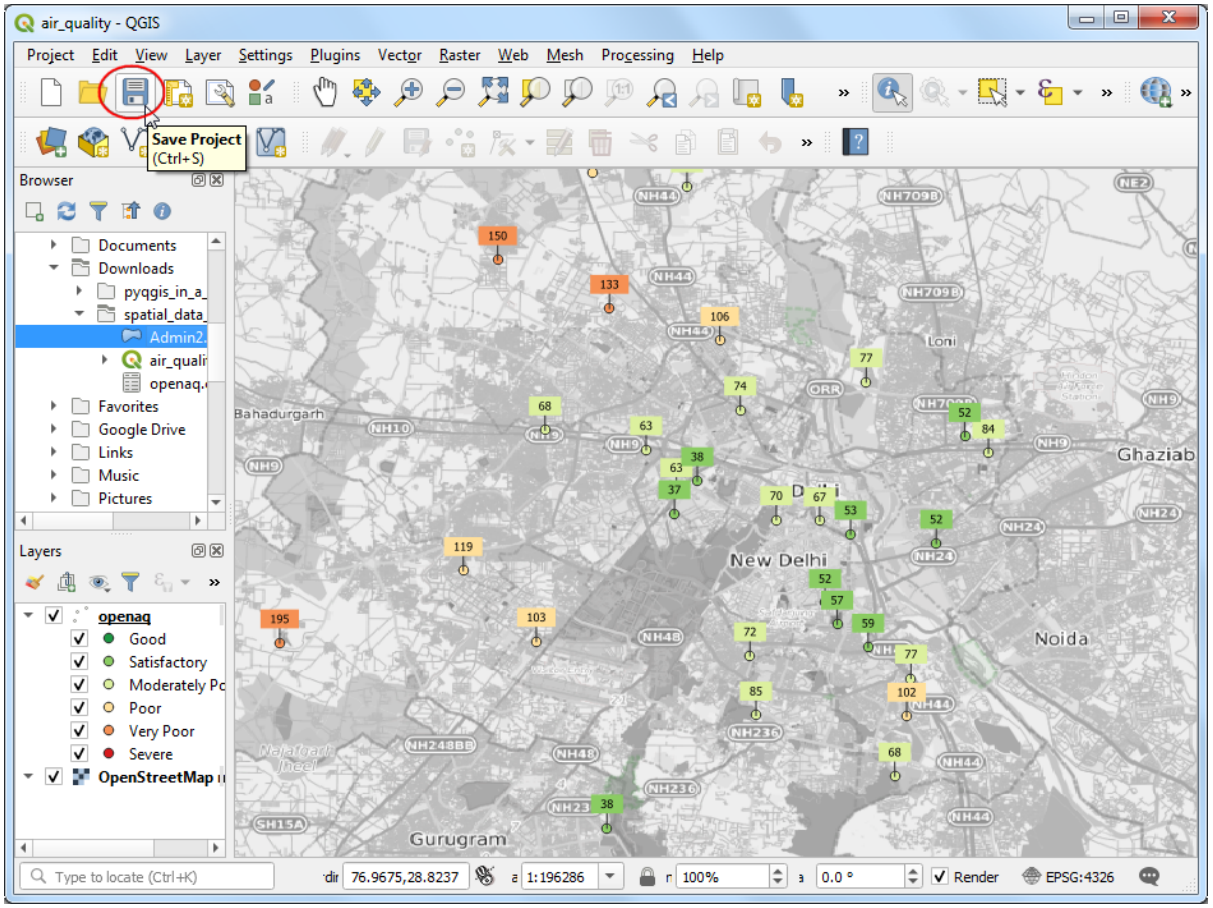



Lastly, go the *Callouts* tab and check *Draw callouts* to make it easy to see which labels belong to which point.

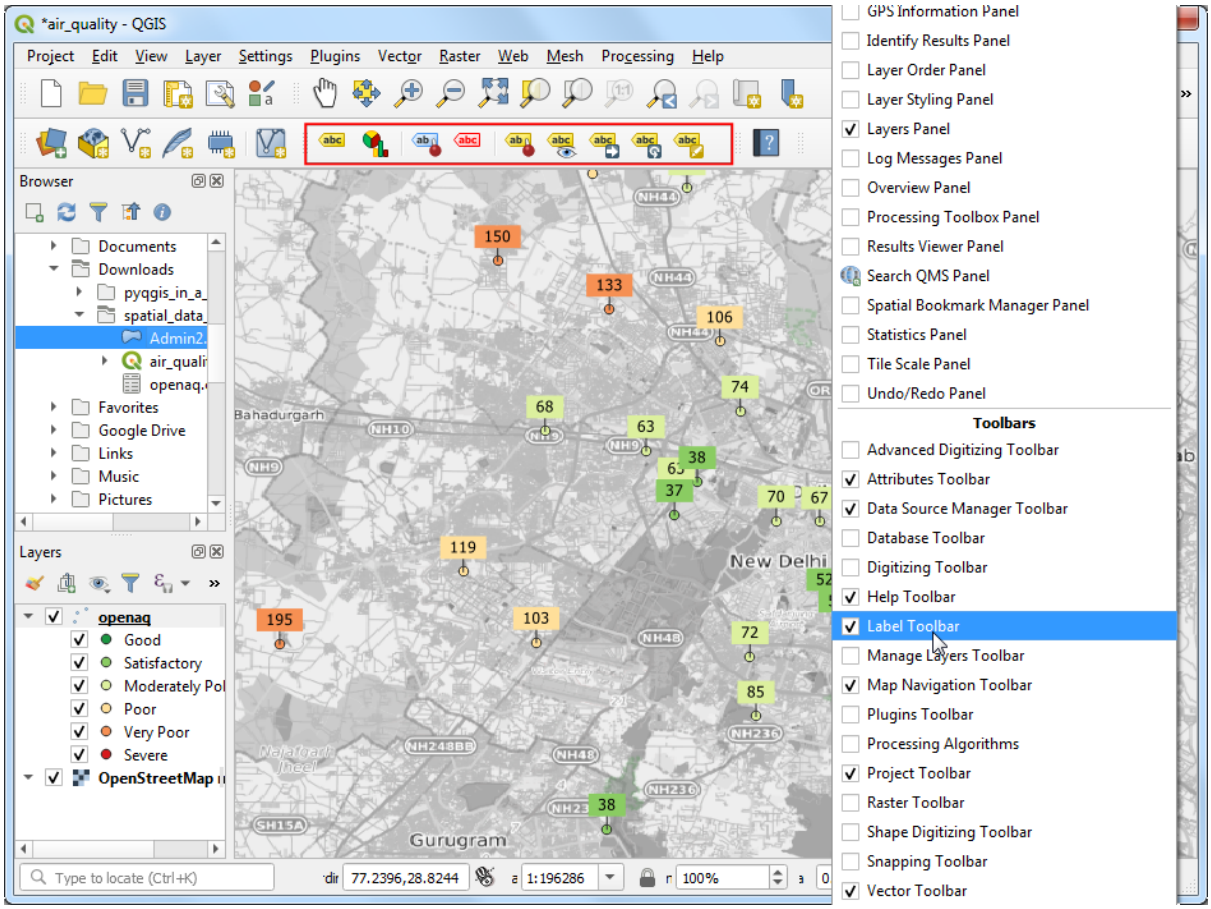



It is a good idea to save out work. Click the *Save Project* button and save it as

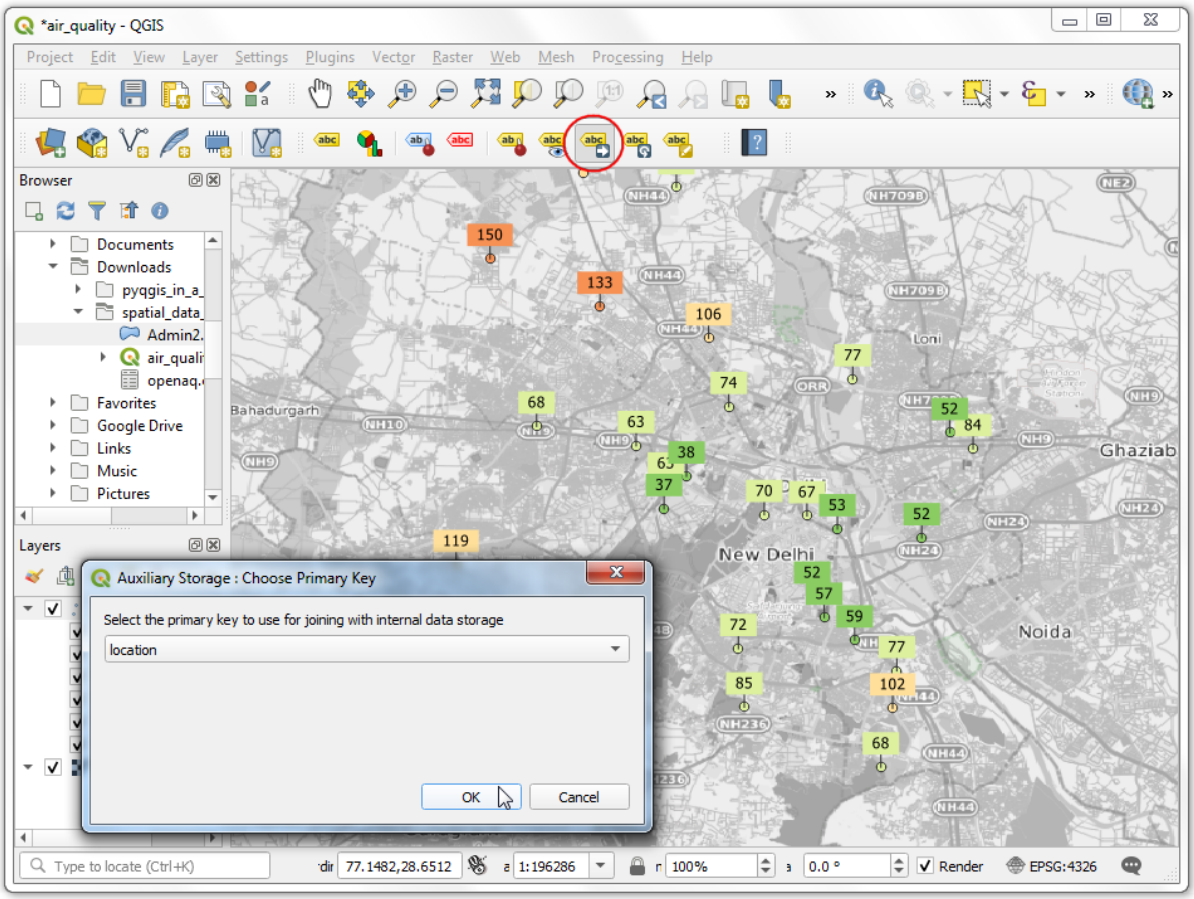
`air_quality.qgz`.




You can see most of the labels are legible, but some are too close and feel a bit cluttered. We can fix them by manually adjusting the placement. Right-click anywhere on the toolbar area and enable the *Label Toolbar*.

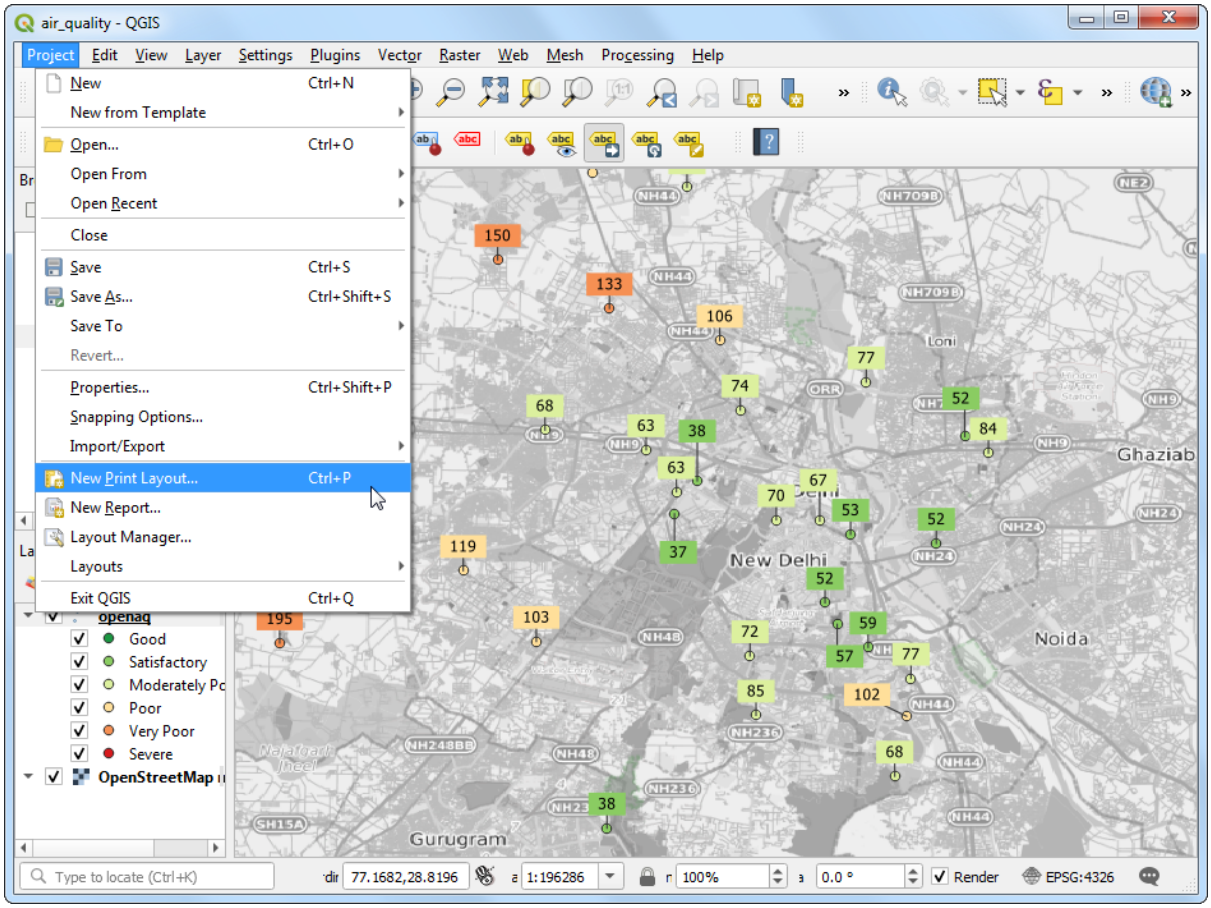



Click the *Move a Label* button. Click *OK* on the *Auxiliary Storage* prompt.

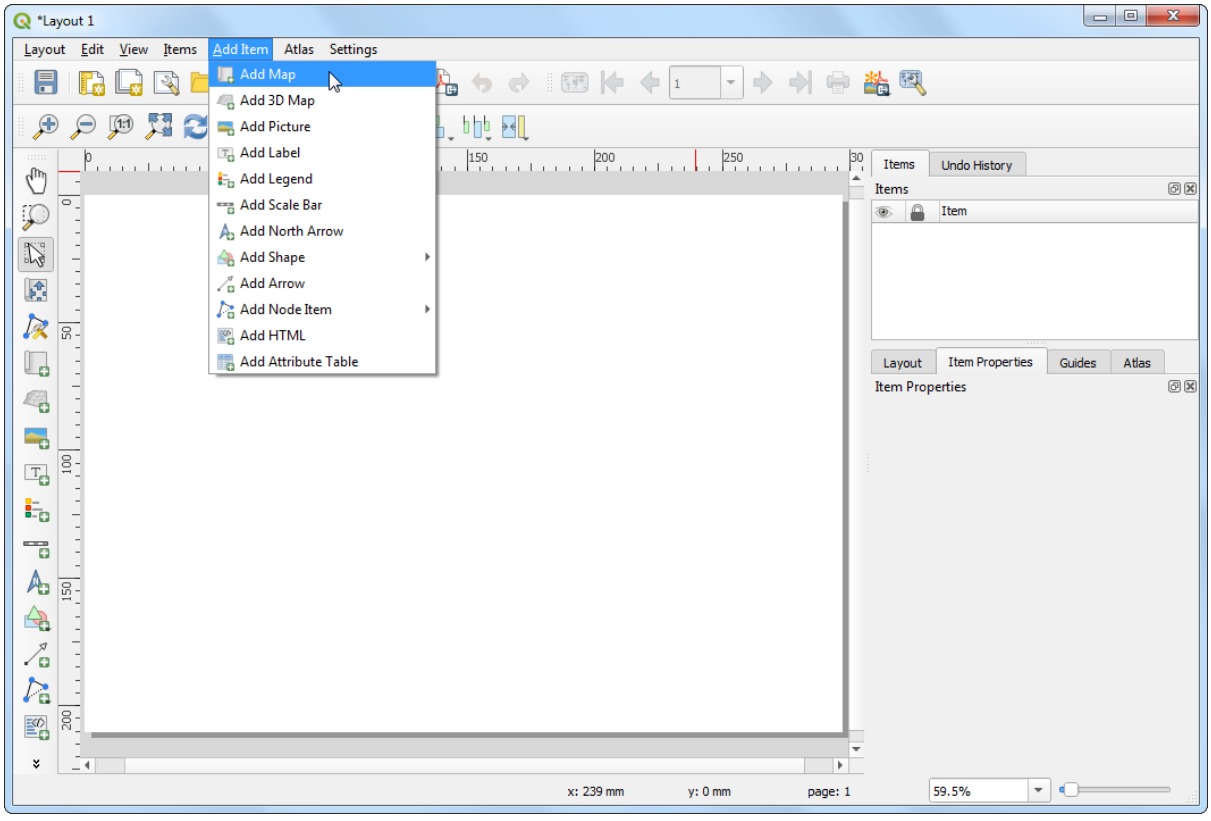



With the *Move a Label* button active, click on a any label to select it. Click at the place you want to move it to, and it will be placed there. Once you are satisfied, save the

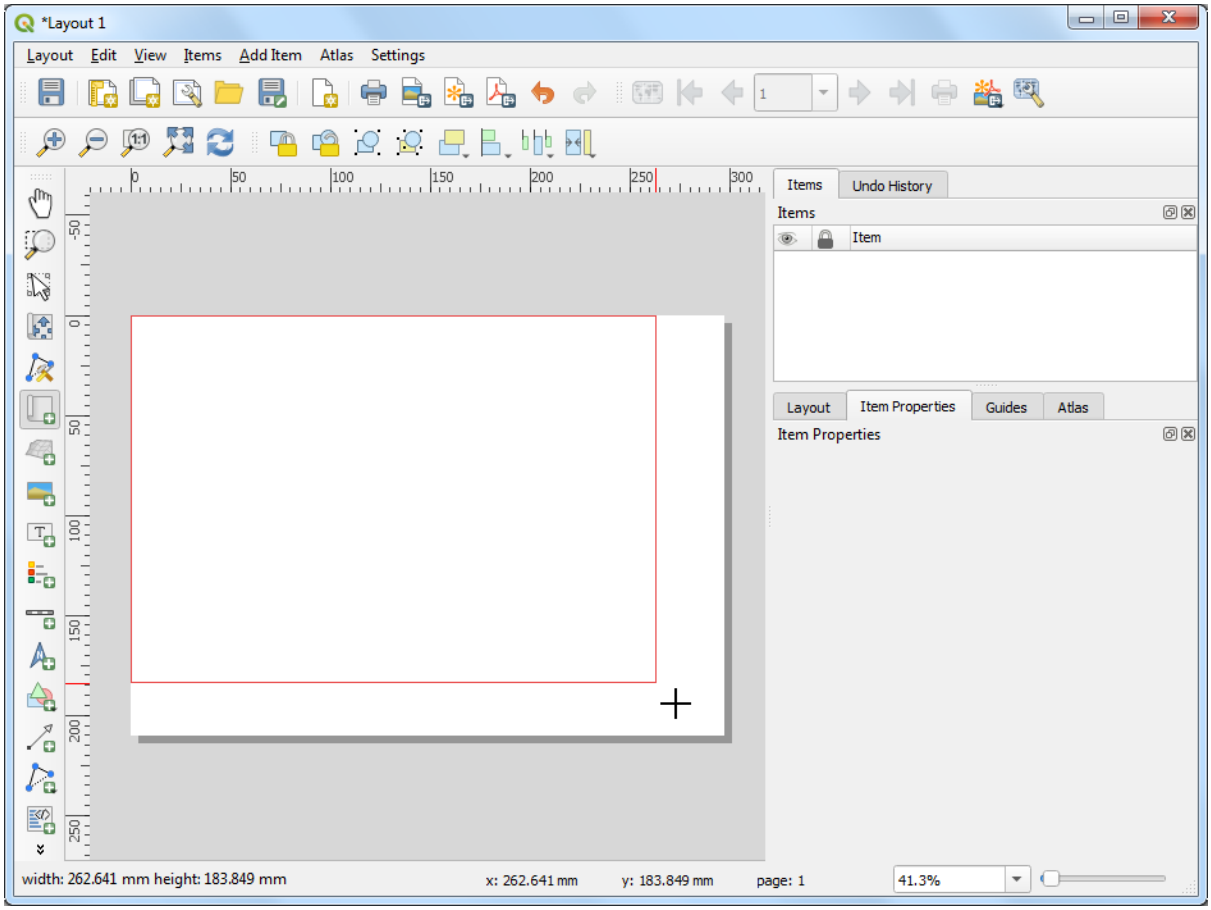
project. It is time to export our map. Go to **Project → New Print Layout...**. Leave the name empty and click *OK*.



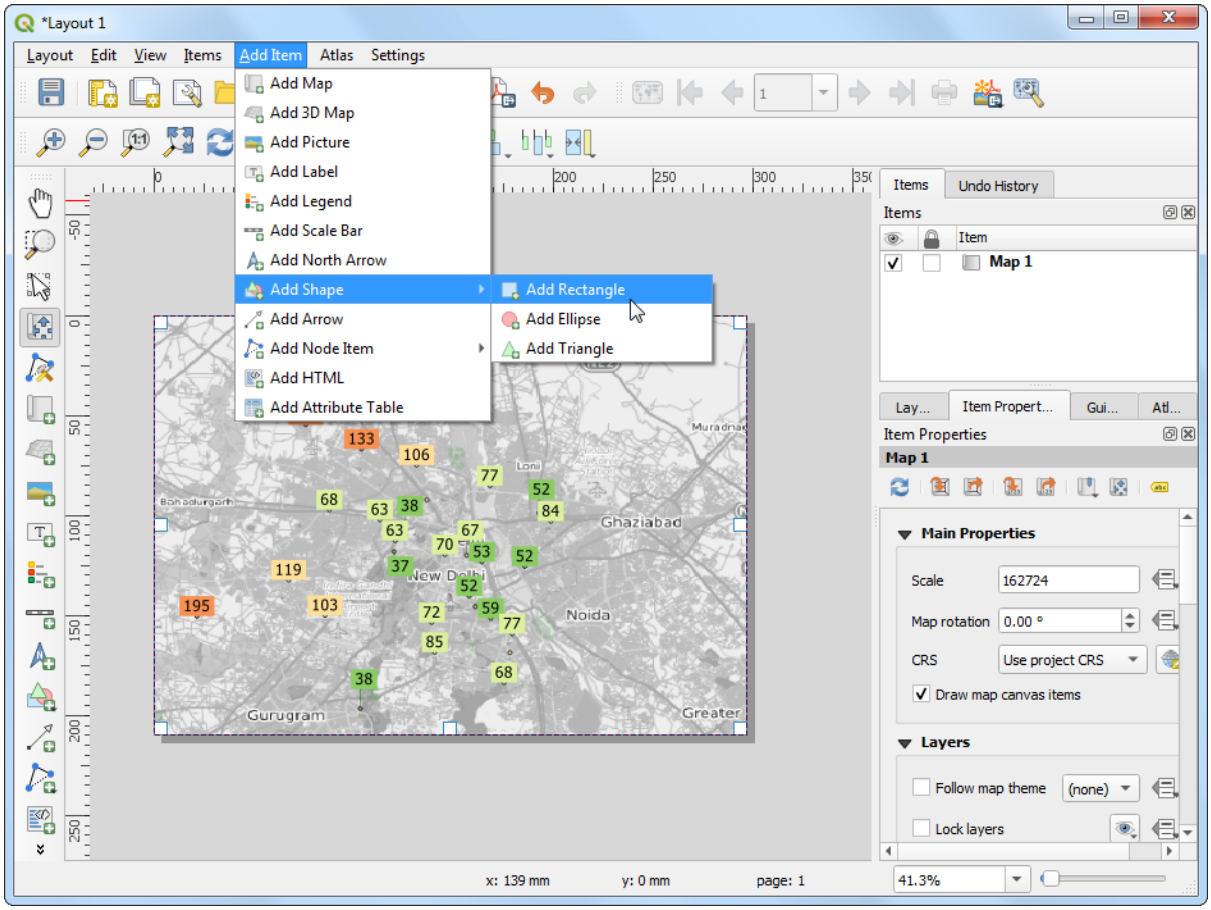

Print Layouts are a way to compose a static map with various elements such as labels, legends, north arrow, scale bar etc. Go to **Add Item → Add Map**.



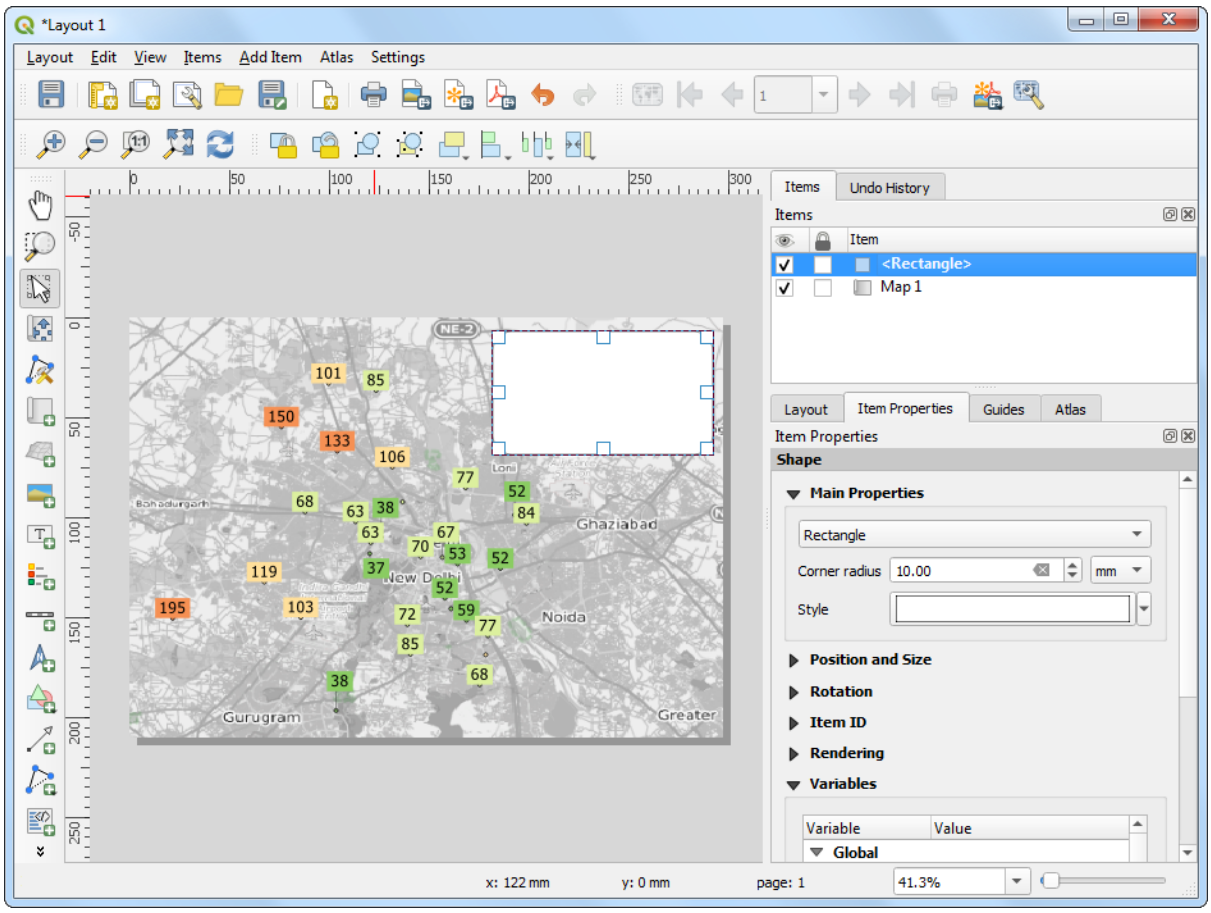
Drag a rectangle where you want the map to be rendered. Once you leg go of you your mouse button, the map from the main canvas will be loaded in the region.



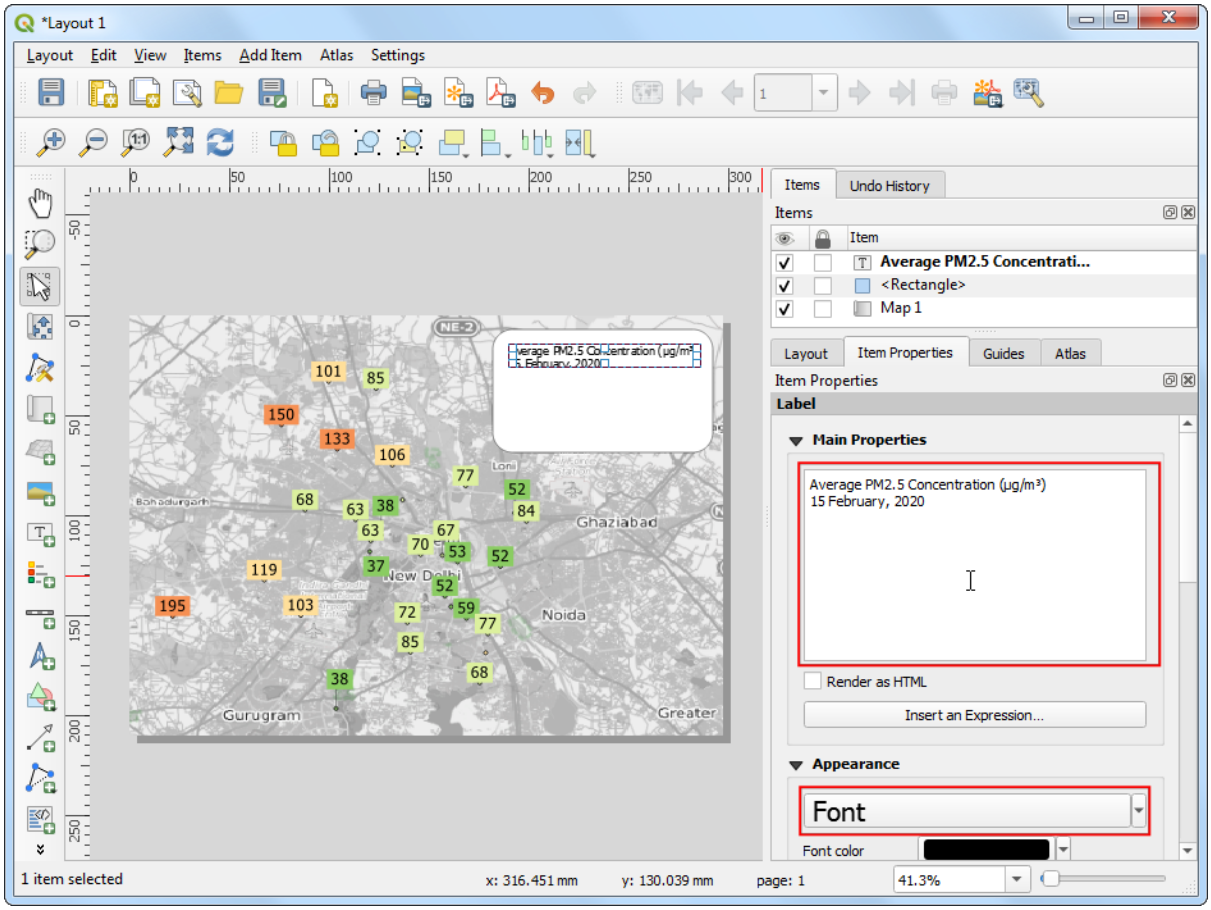

Next, we will add a *Rectangle* which will hold the tile, legend and attribution.



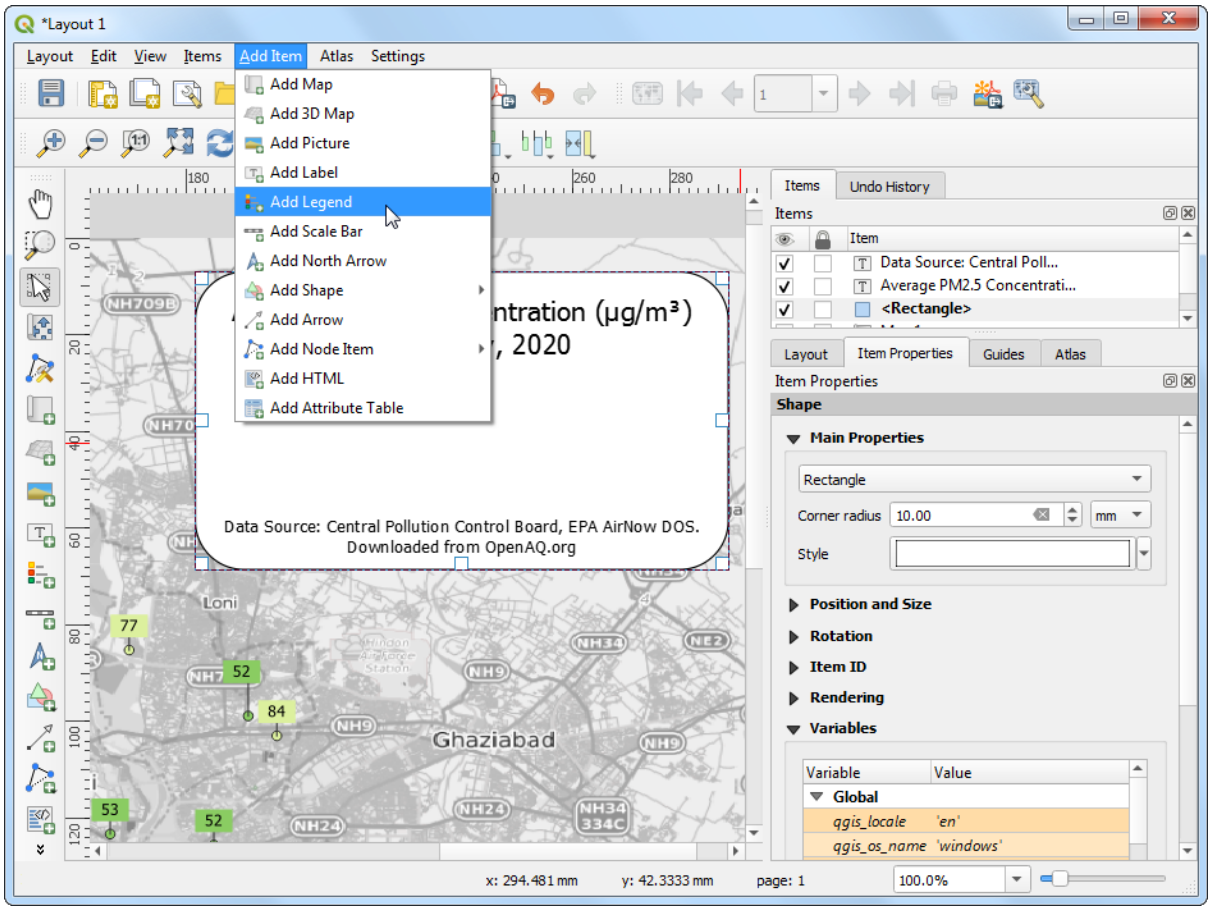
Place the rectangle on the top-right hand corner and change the *Corner radius* to `10`.



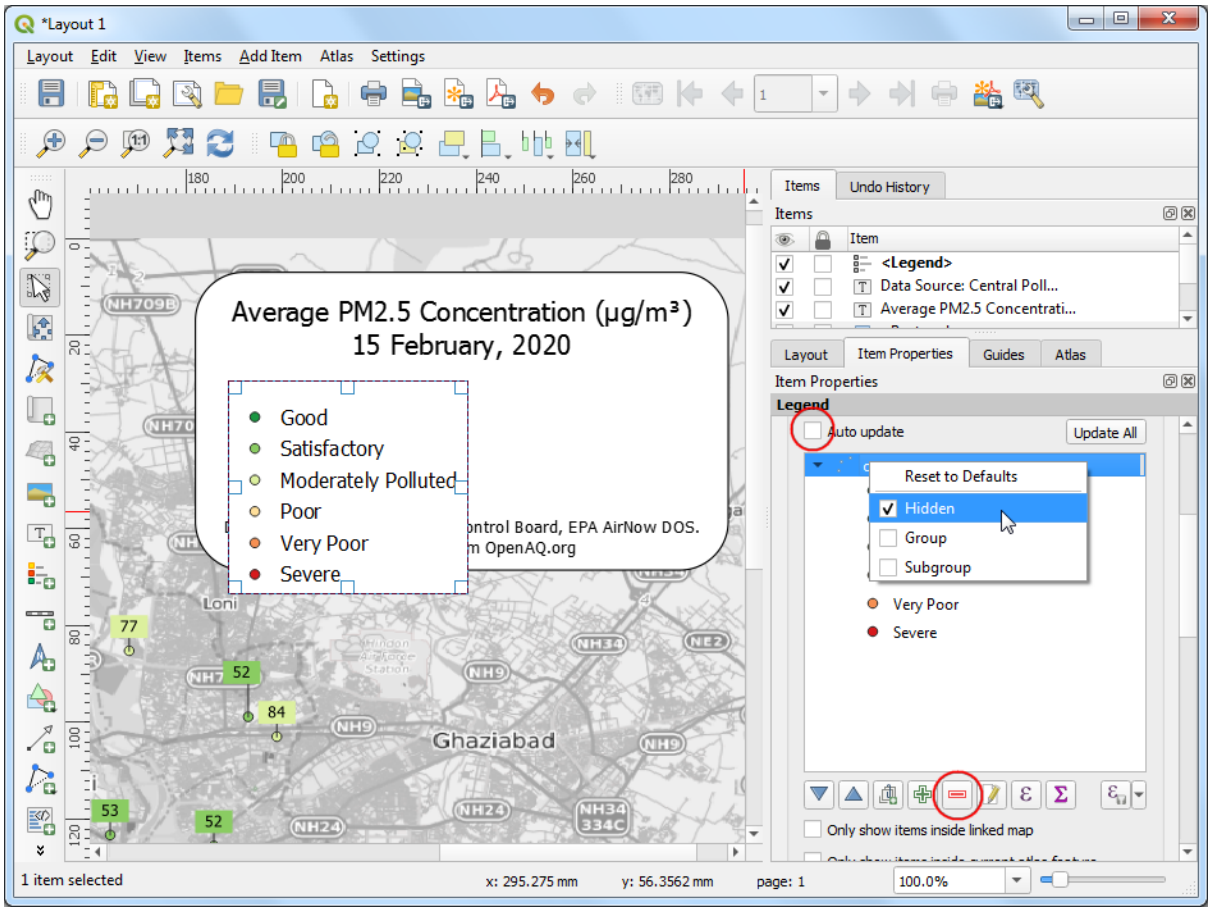

Next, add a *Label* with the title `Average PM2.5 Concentration (µg/m3)` and date `15 Bebruary, 2020`.



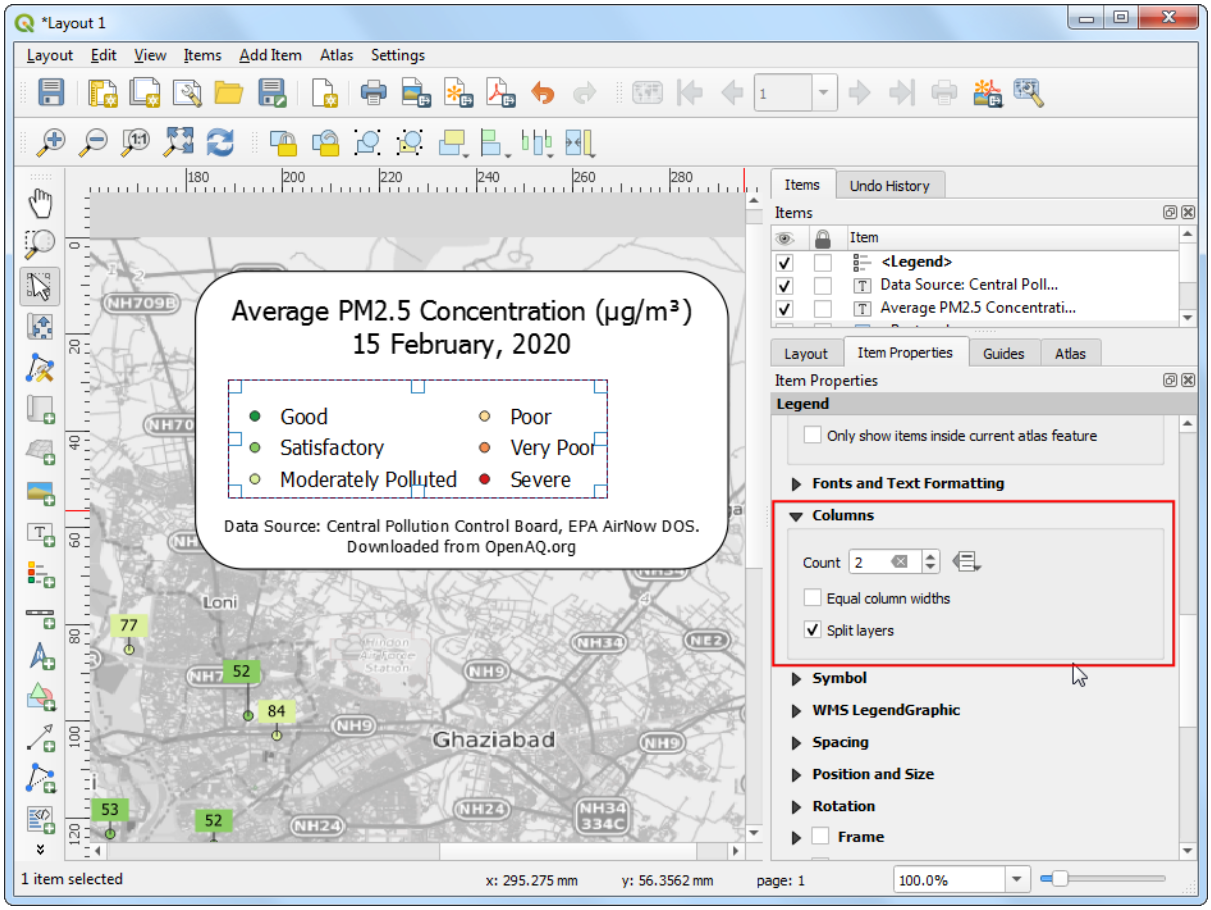
Add another *Label* with the attribution text `Data Source: Central Pollution Control Board, EPA AirNow DOS. Downloaded from OpenAQ.org`. Now we will add a legend, so our users know how to interpret various colors on the map. Go to **Add Item → Add Legend**



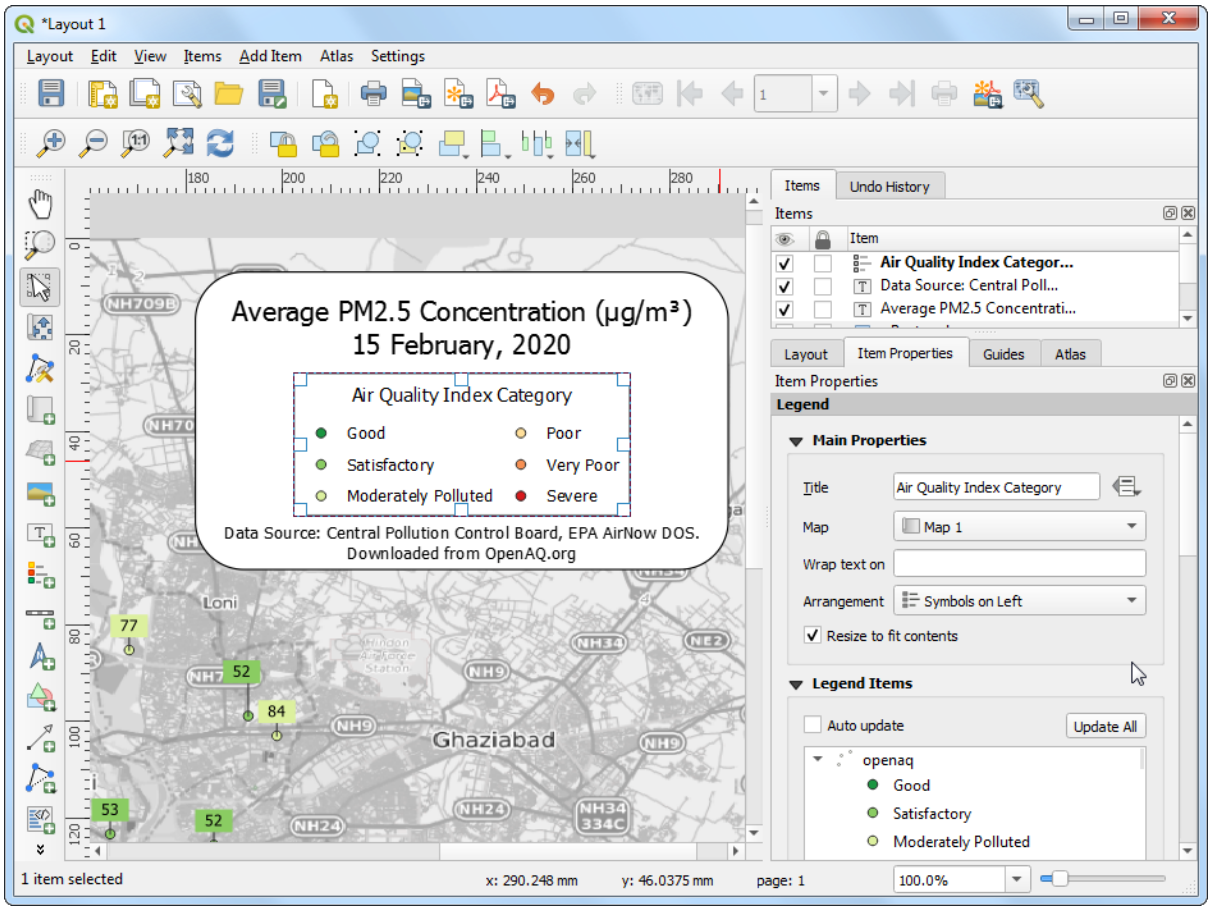

Drag a box to insert the legend. Go to the *Item Properties* tab and turn-off *Auto update*. Remove the `OpenStreetMap monochrome` layer and set the `openaq` group label to *Hidden*.




The default legend placement is vertical - in a single column. We can change it to be in multiple columns to make it span horizontal. Scroll down to the *Columns* section and change the *Count* to 2 and check *Split layers* button.

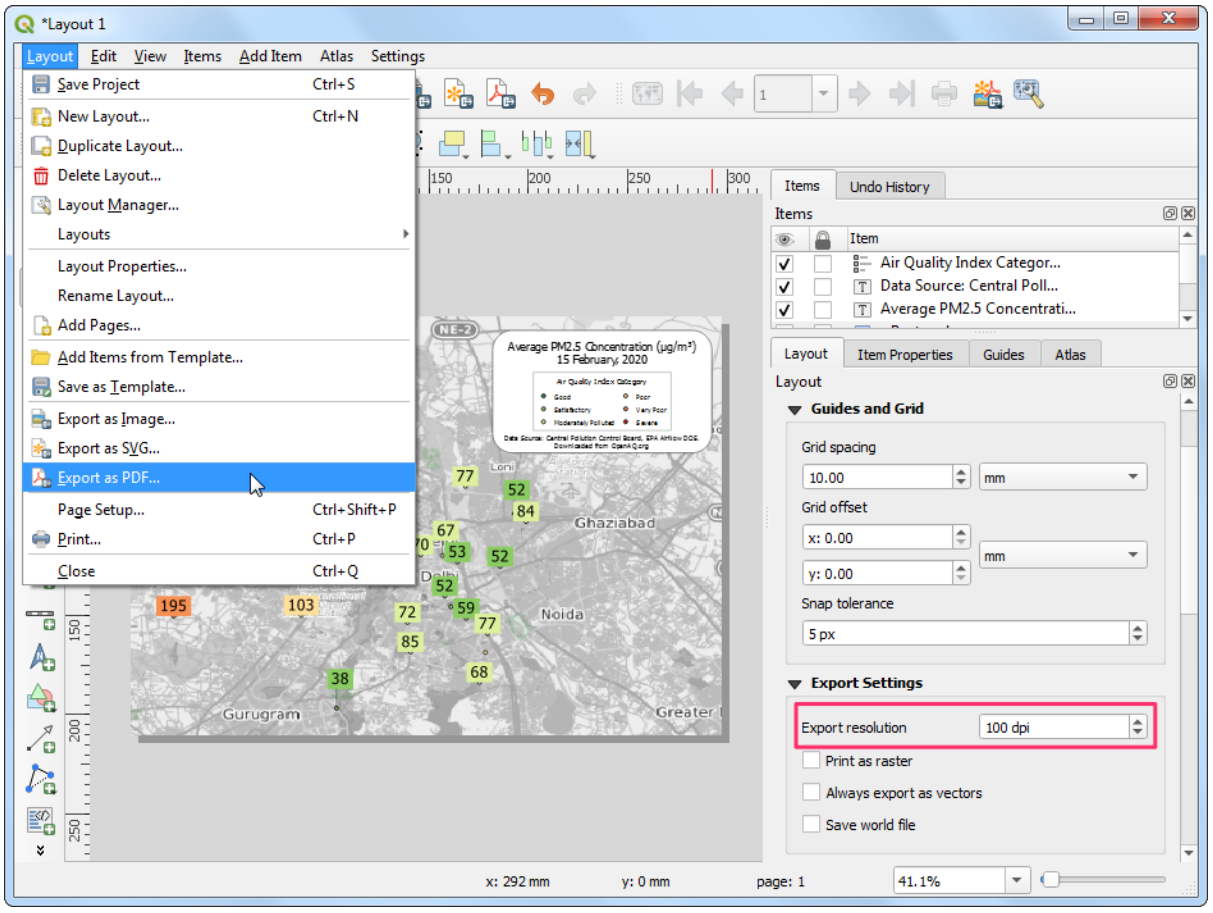


In the *Main Properties* section, enter a *Title* for the legend as `Air Quality Index Category`.

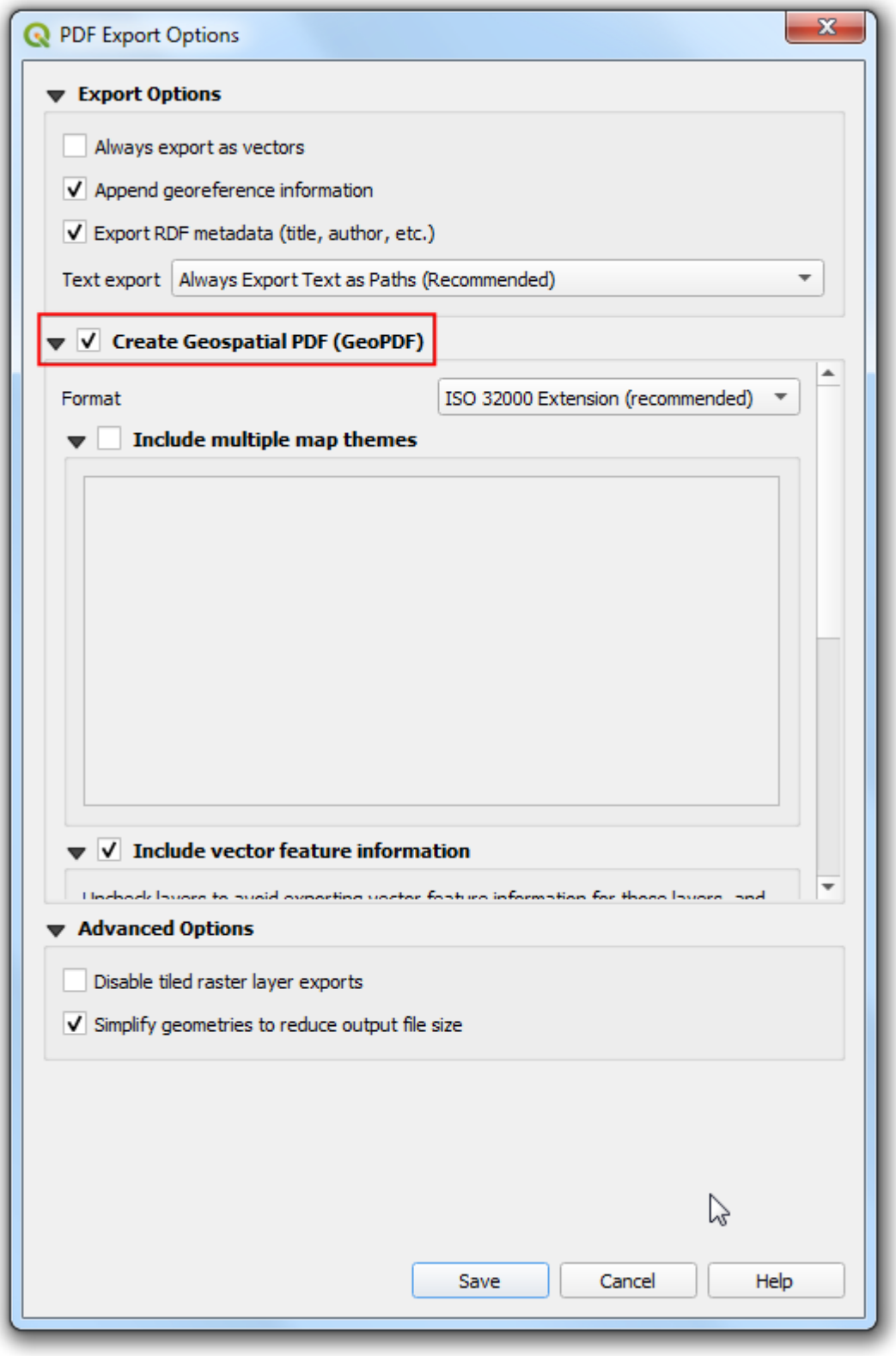



Once you are satisfied with the map, we can export the map. You can export the composition as an **Image** if you wanted to use it in a website, email, slideshow etc. You can also export it to as an **SVG** so it can be further edited in a graphics program such as InkScape. But the most common format for maps is still a PDF.

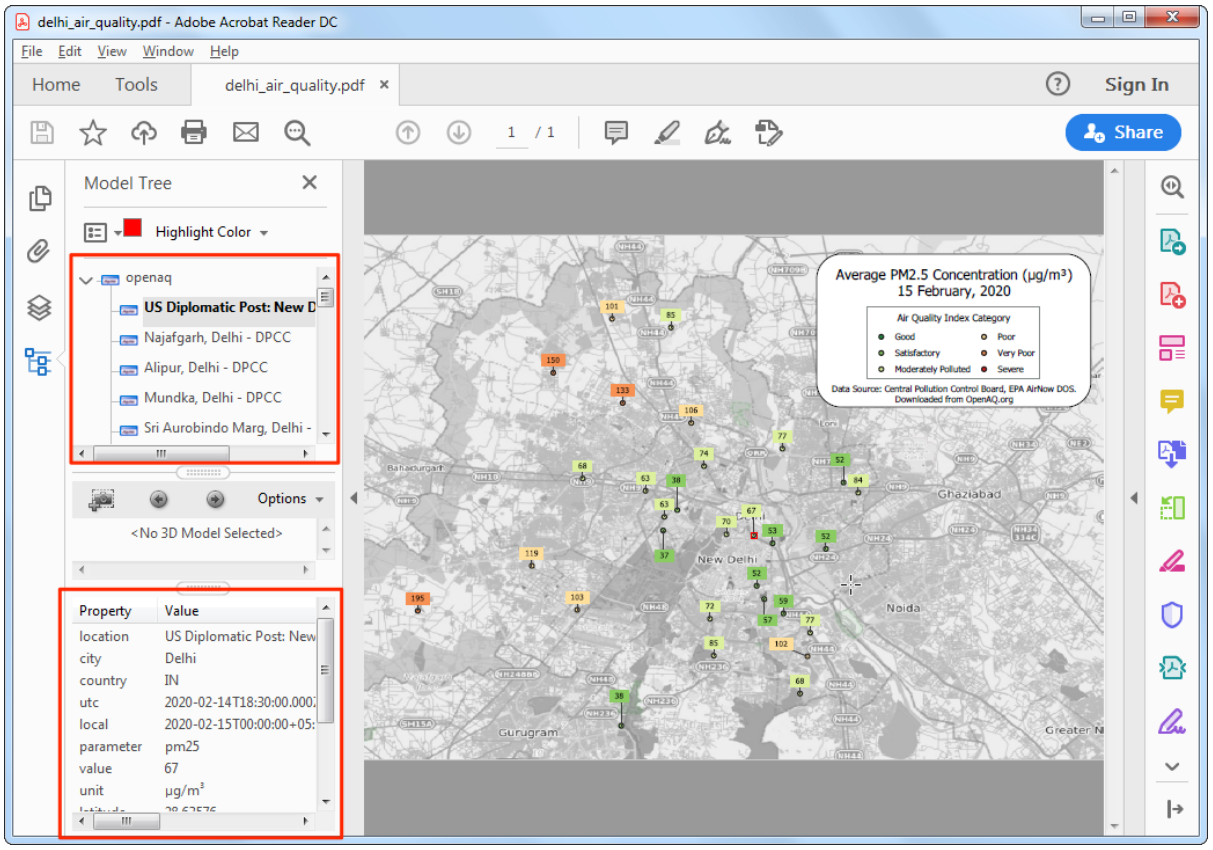

So let's export our map to a PDF. Before exporting, switch to the *Layout* tab and set the *Export resolution* to 100 dpi. Go to **Layout → Export as PDF**.




In the *PDF Export Options*, you can check *Create Geospatial PDF (GeoPDF)*. GeoPDF is an enhanced PDF format that is spatially aware and preserves layer and attribute information. Click *Save* and save the output as `delhi_air_quality.pdf`.




If you open the resulting PDF in a compatible reader such as Adobe Acrobat, you can toggle layer visibility, query attributes by features, measure distances and so on.




# Lines

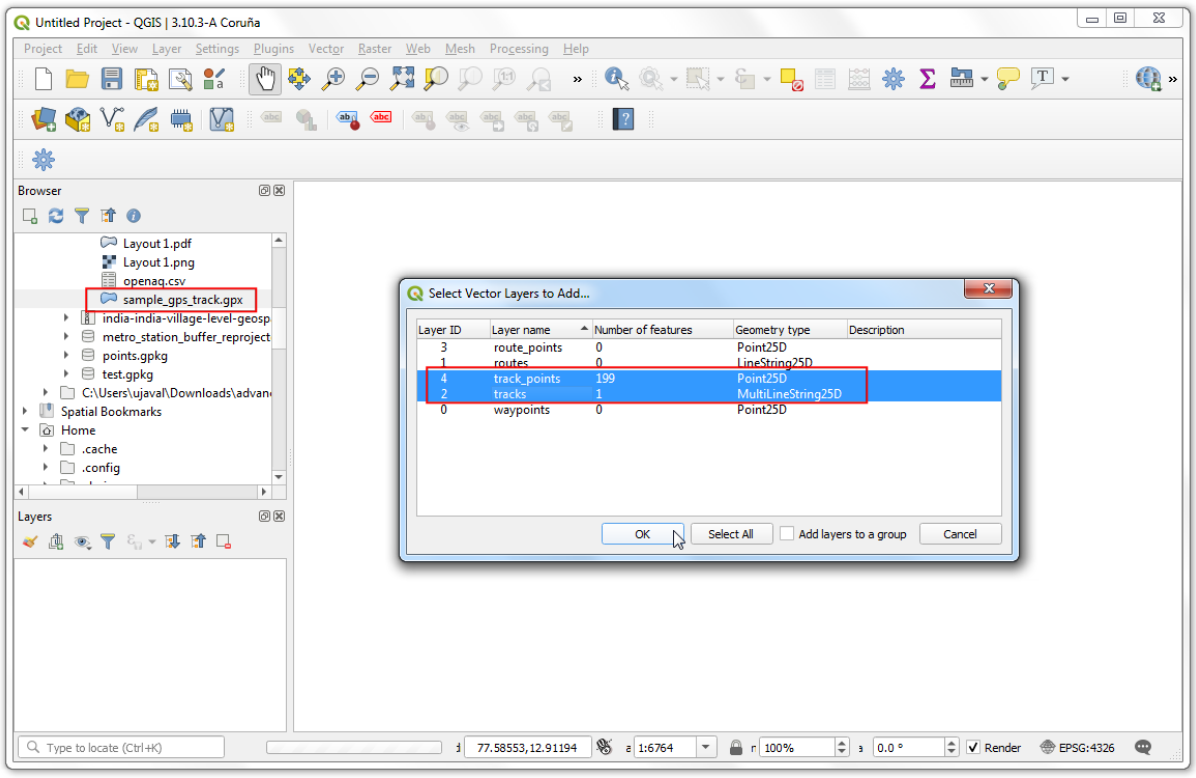
Many of our transportation infrastructure such as roads, bridges, railways etc. as well as natural features such as rivers, streams etc. can be modeled as lines. Other abstract concepts, such as contours and trajectories are also modeled using linear features. Shapefiles, GeoJSON, GPX are commonly used file formats for storing line datasets.

## Exercise: Visualize GPS Tracks

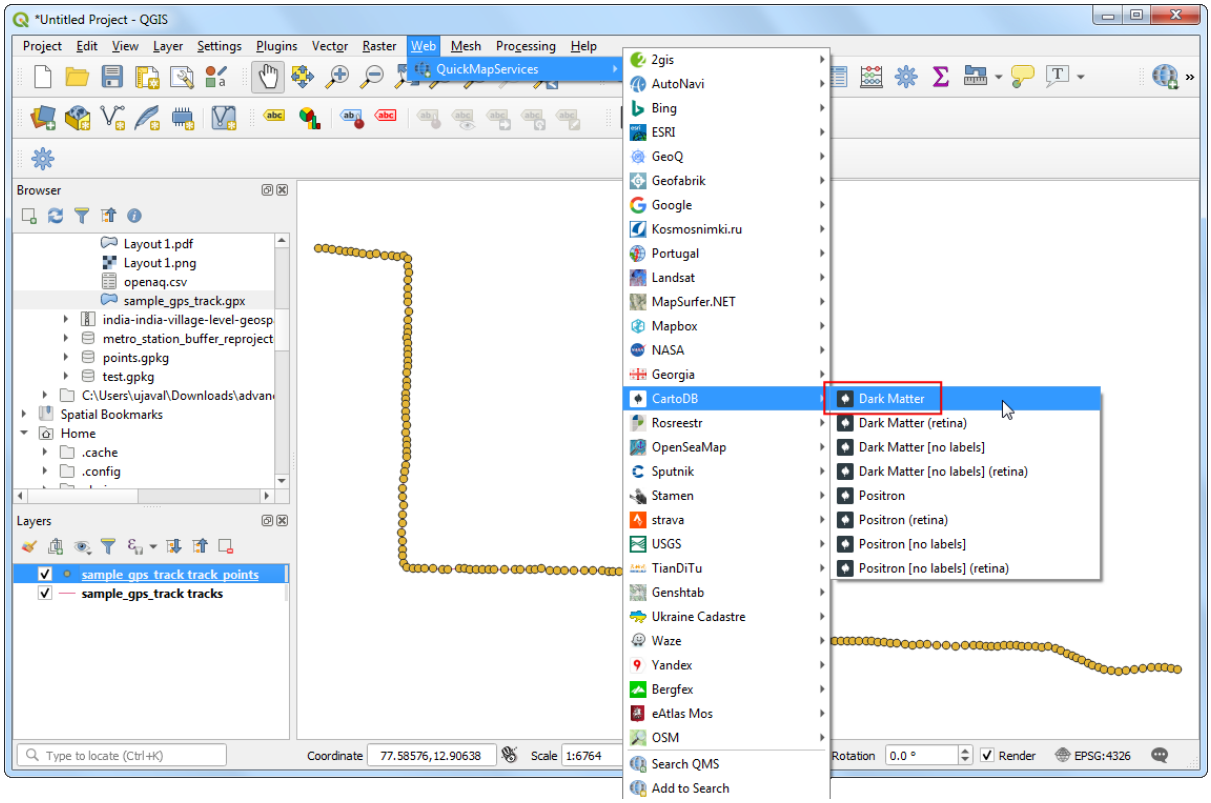
GPS tracks have become ubiquitous in modern life. With GPS built-into most phones, many of us capture the tracks while running or biking outdoors. Cab companies use GPS tracks collected during the trip to determine fares. Delivery and logistics companies store and analyze millions of GPS tracks from their assets to derive location intelligence.

We will use a GPS track I collected using the open-source GPS Logger app while cycling to work. The default format for storing GPS tracks is GPS Exchange Format (GPX). It is a XML-based text format that allows storing points, tracks and routes in a single file. We will use the data in `sample_gps_track.gpx` file and create an animated GIF showing the trip.

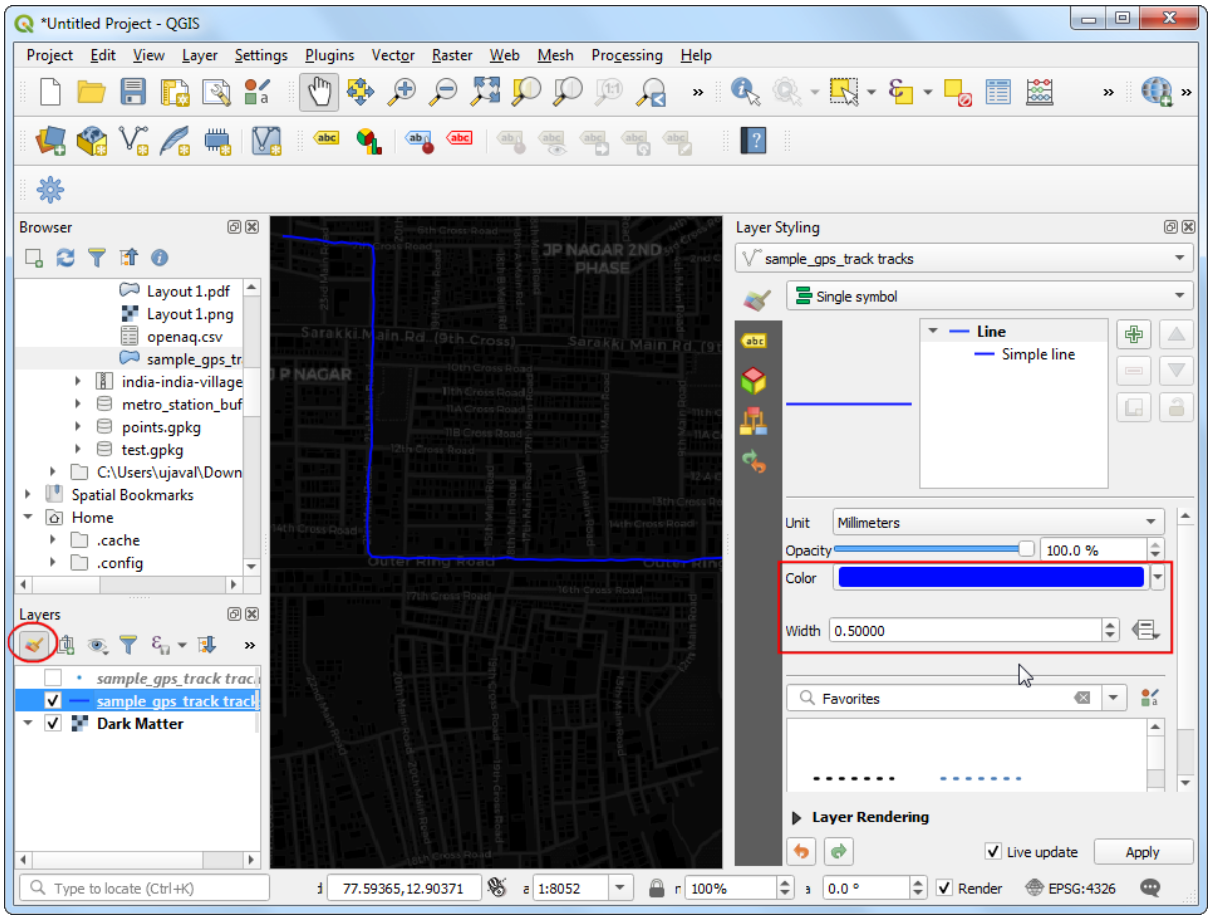
1. Locate the`sample_gps_track.gpx` file and drag it to the canvas. As the file contains multiple data types, a pop-up will ask us to select the layers to add. Hold the *Shift* key and select both `track_points` and `tracks` layers. Click *OK*.



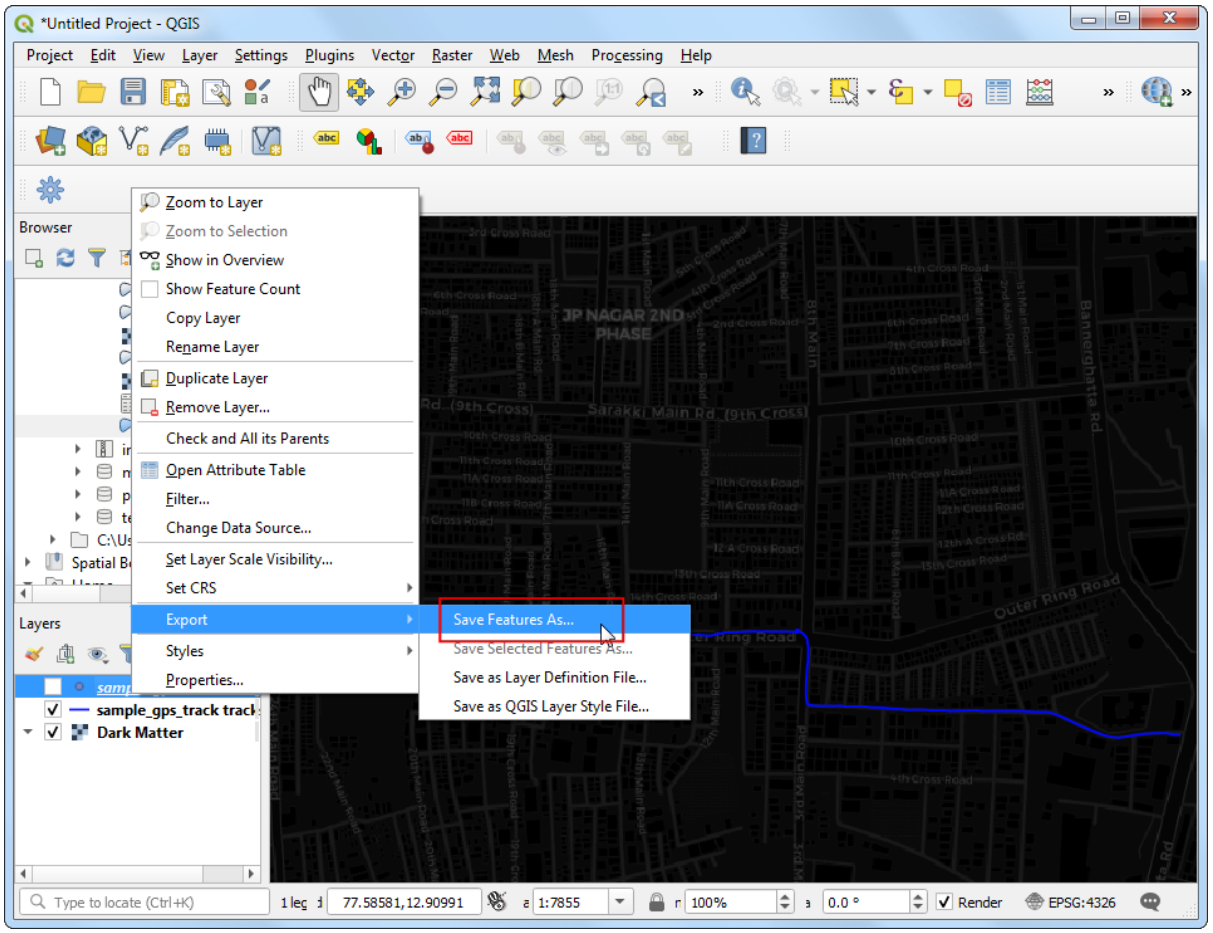

2. To add some context to the map, we should add a basemap. A dark background map works best for the visualization we want to create. Go to **Web → QuickMapServices → CartoDB → Dark Matter** layer.



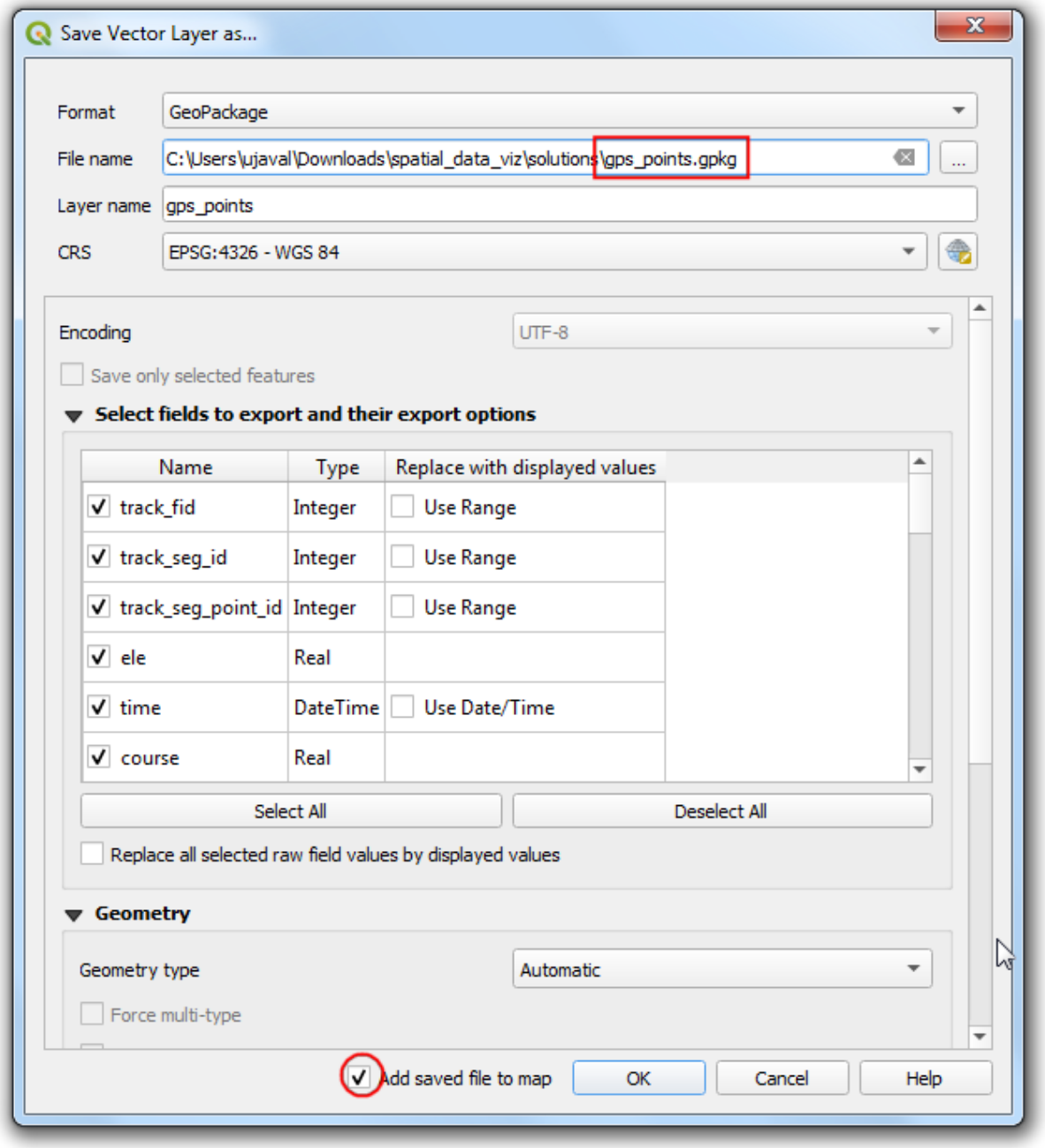

3. Select the `sample_gps_track tracks` layer and click *Open the Layer Styling Panel*. You can change the line *Color* to `Blue` and *Width* to `0.5`.



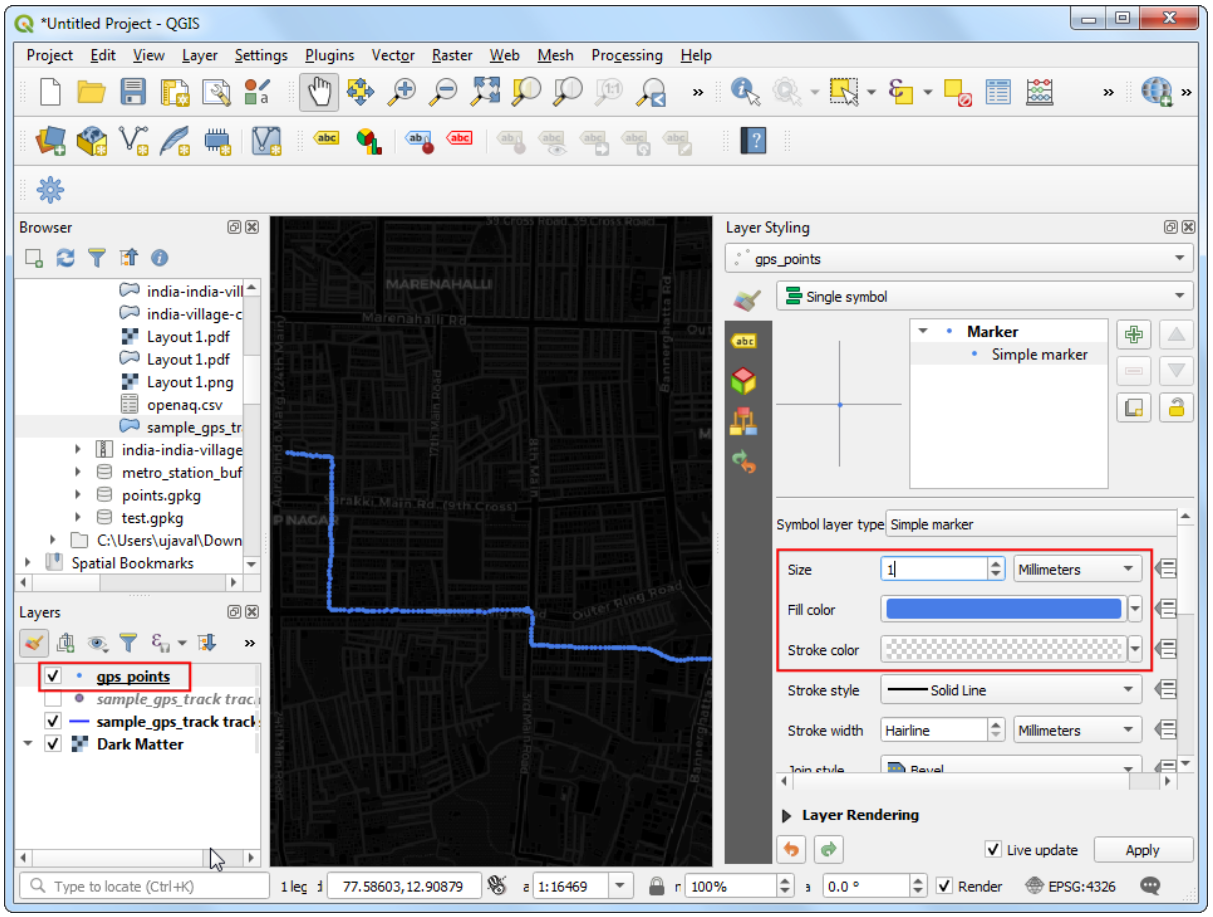

4. Before styling the points, we will first export the layer to a different format. Animating the points require applying filters on the data layer which doesn't work well with the GPX format, so we will export it to a GeoPackage. Right-click the `sample_gps_tracks track_points` layer and go to **Export → Save Feature As**.



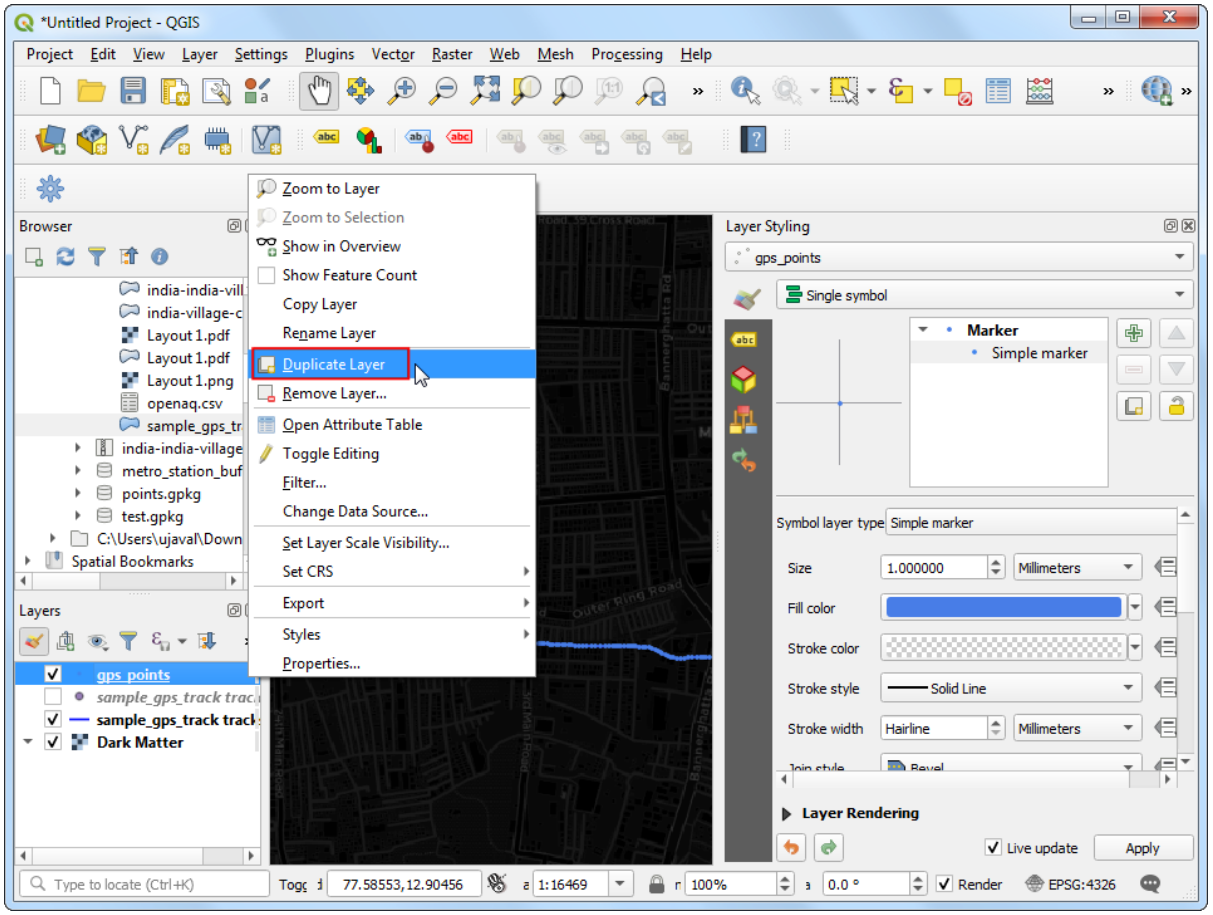

5. Select the output *File name* as `gps_points.gpkg`. Make sure the *Add saved file to map* button is checked and click *OK*.



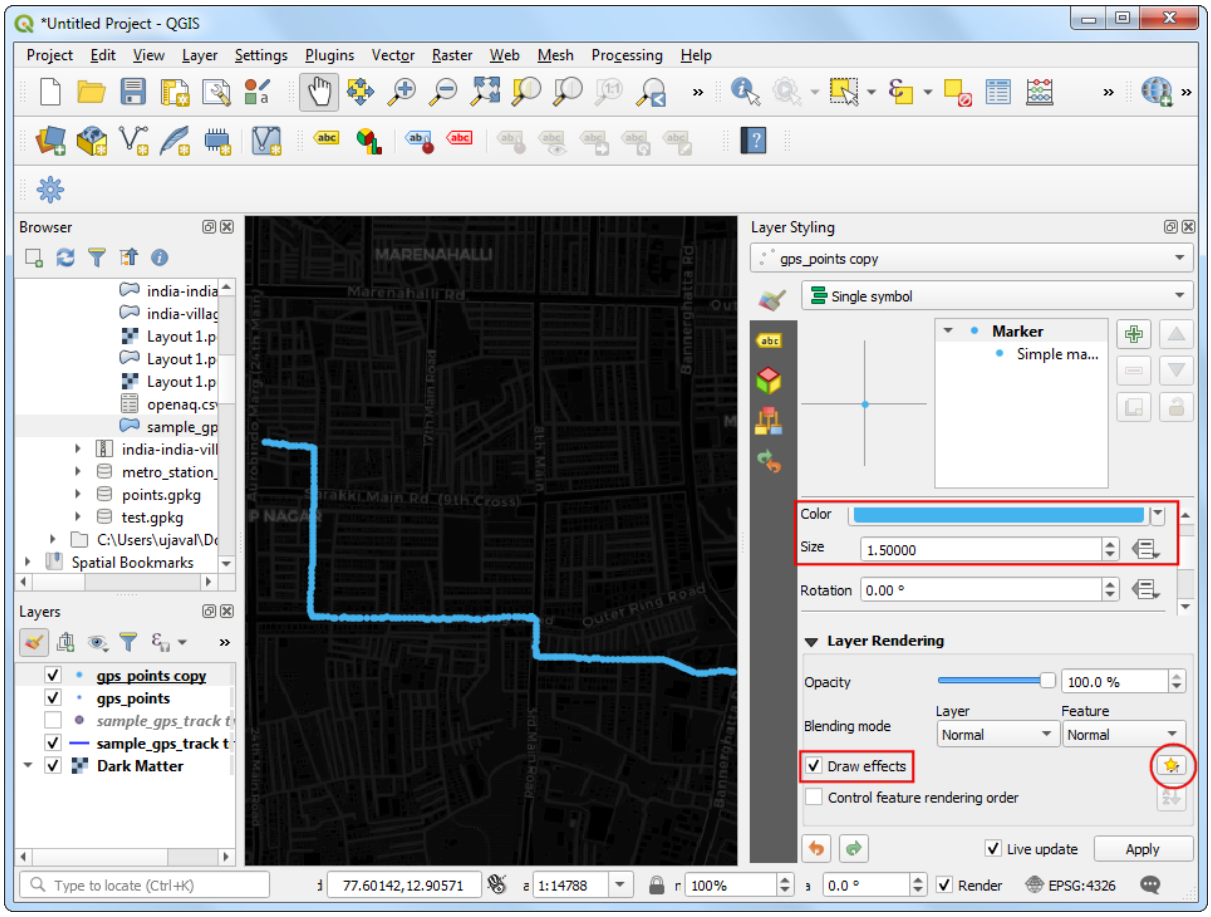

6. A new layer `gps_points` will be added to the *Layers* panel. In the *Layer Styling Panel*, select *Simple marker* symbol. Change the point *Size* to `1` . Choose a lighter shade of Blue as the *Fill color* and a `Transparent Stroke` as *Stroke color*.



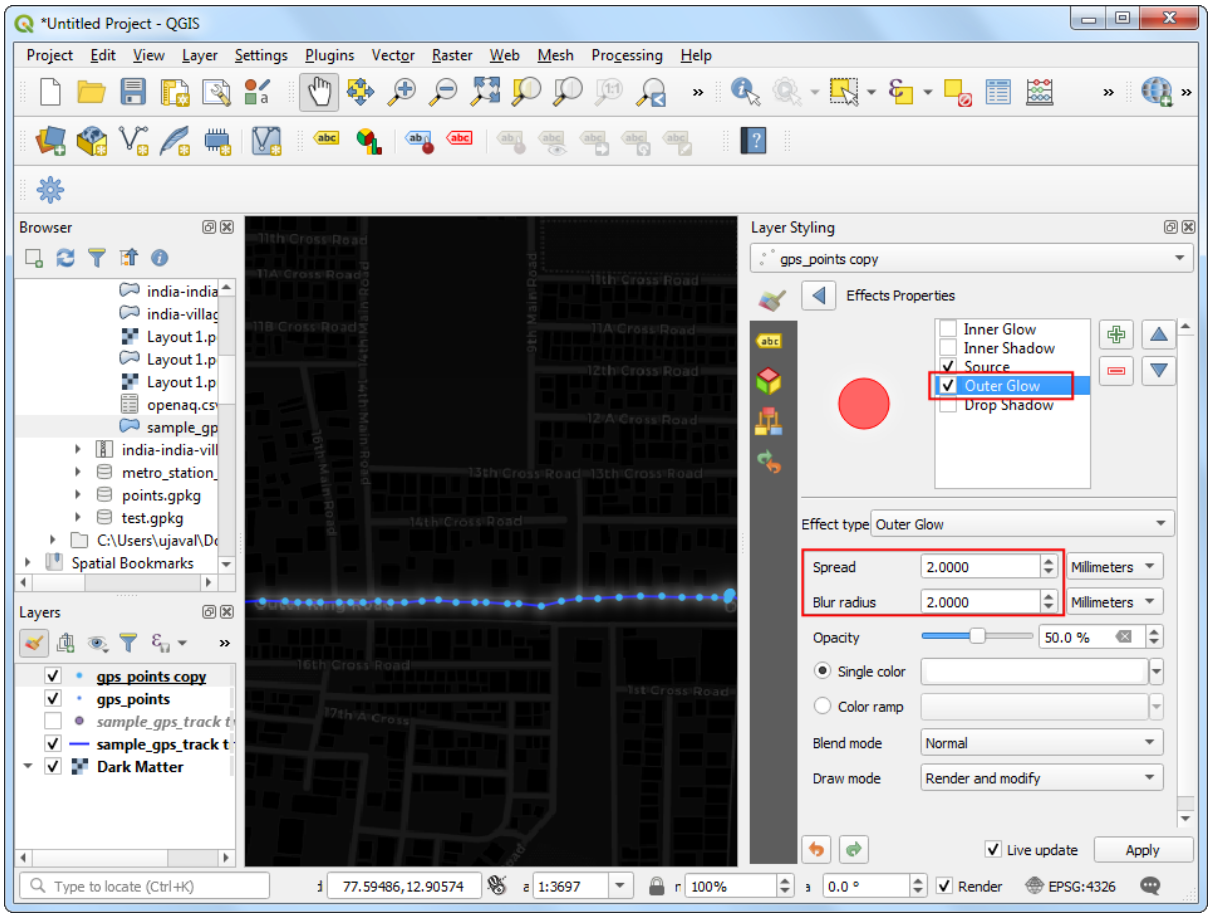

7. We want to give a glow effect to the points as it is animating. Right-click the `gps_points` layer and choose *Duplicate Layer*.



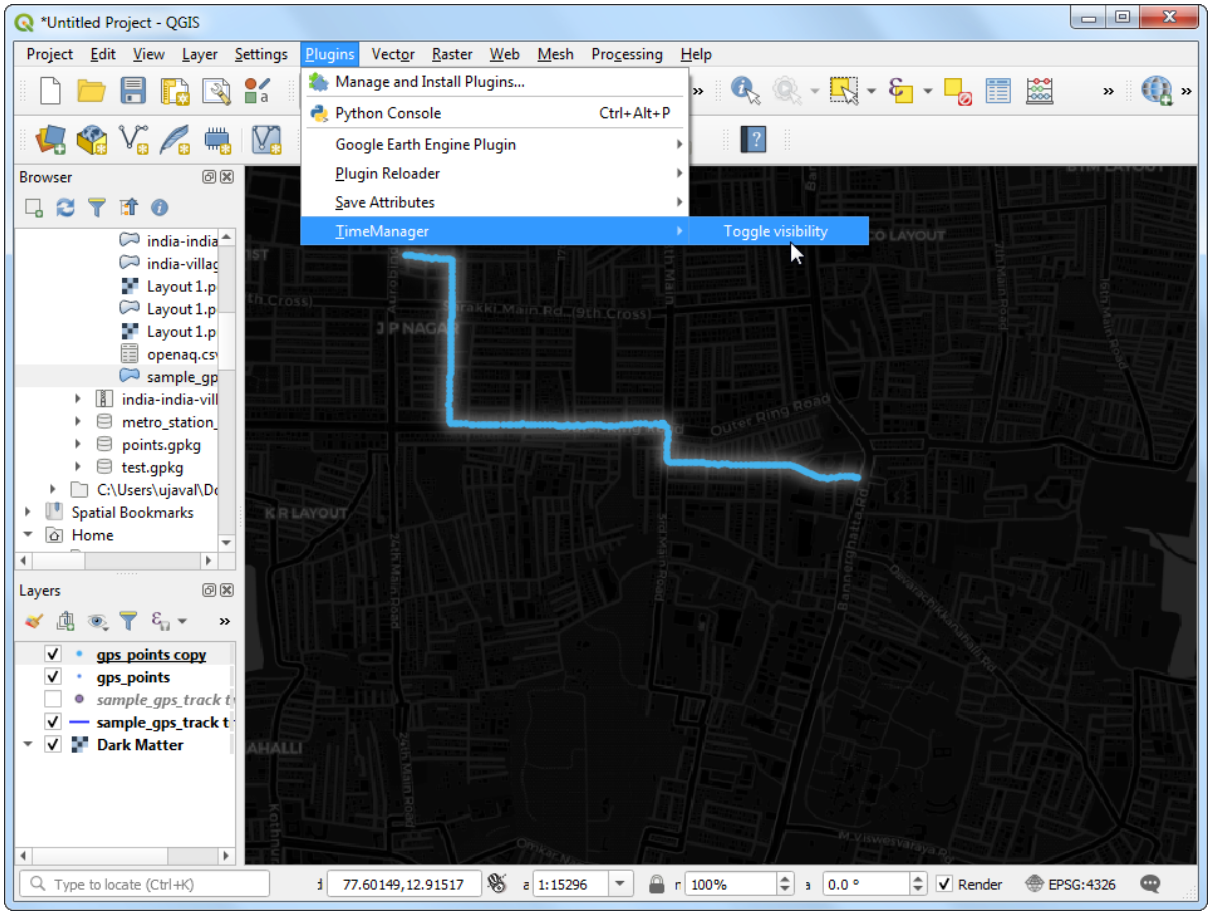
8. In the *Layer Styling Panel* for the duplicated `gps_points_copy` layer, choose bright neon as the *Color* and increase the size to `1.5`. Check the *Draw Effects* option and click the *Effects* button next to it.



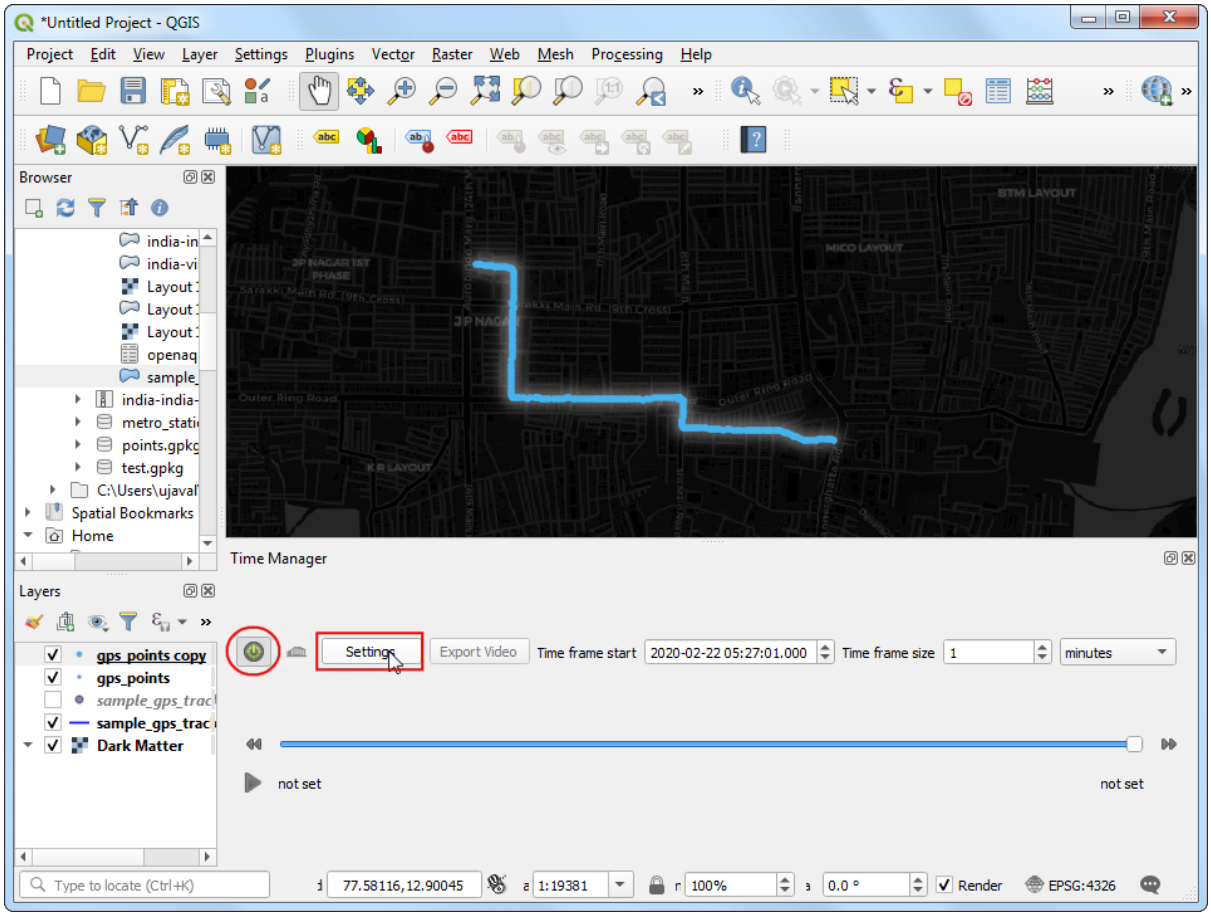

9. In the *Effects Properties* panel, check *Outer Glow*. Select `2.0` for both *Spread* and *Blue radius*.



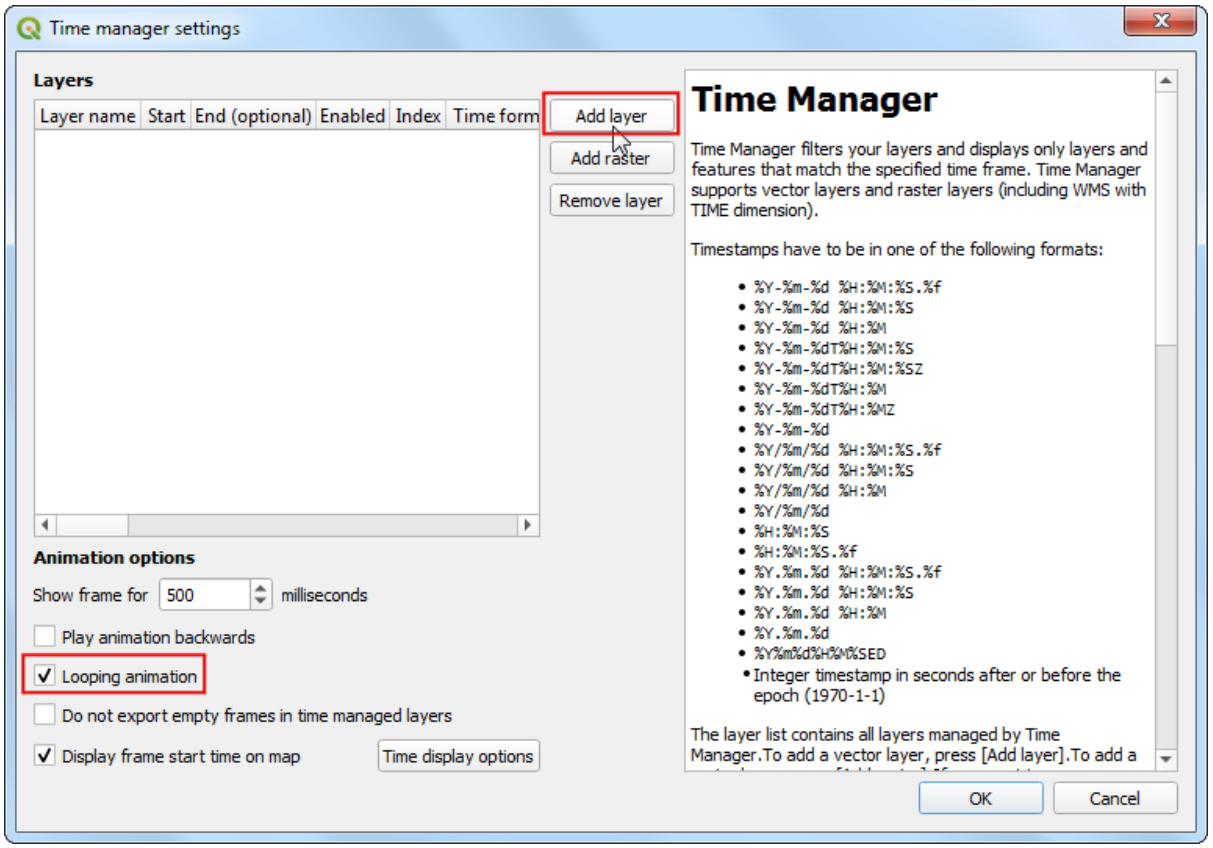

10. Now we are ready to animate the points. Go to **Plugins → Time Manager → Toggle visibility**.



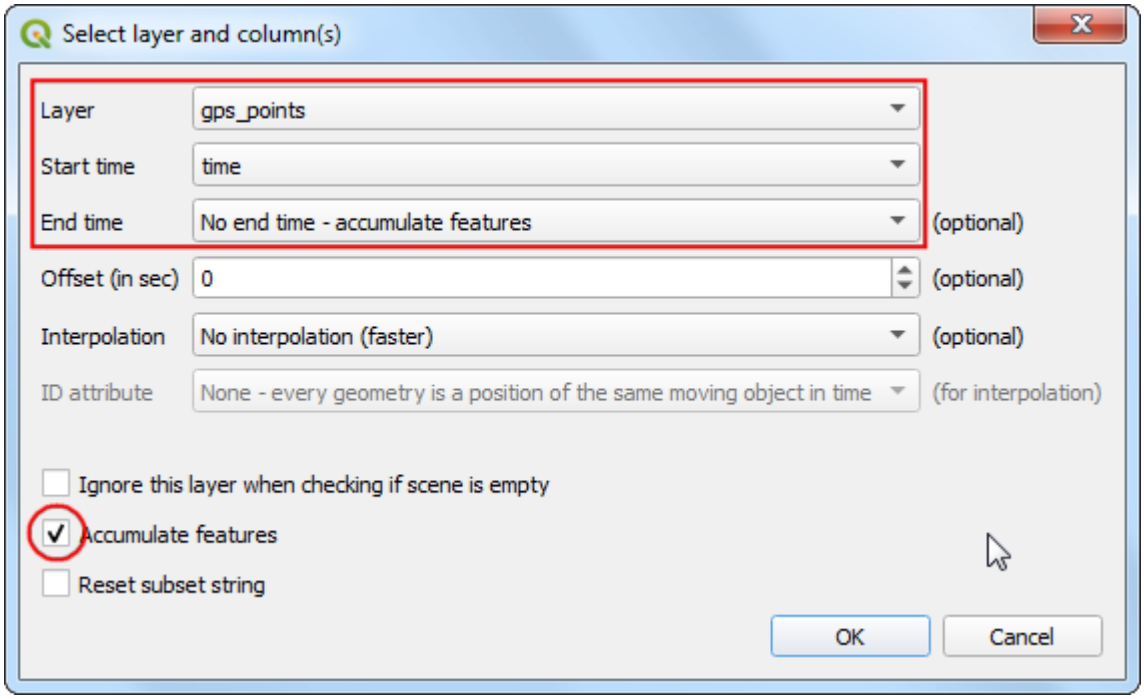
11. In the *Time Manager* panel, click the *Power* button to activate the plugin. Click *Settings*.



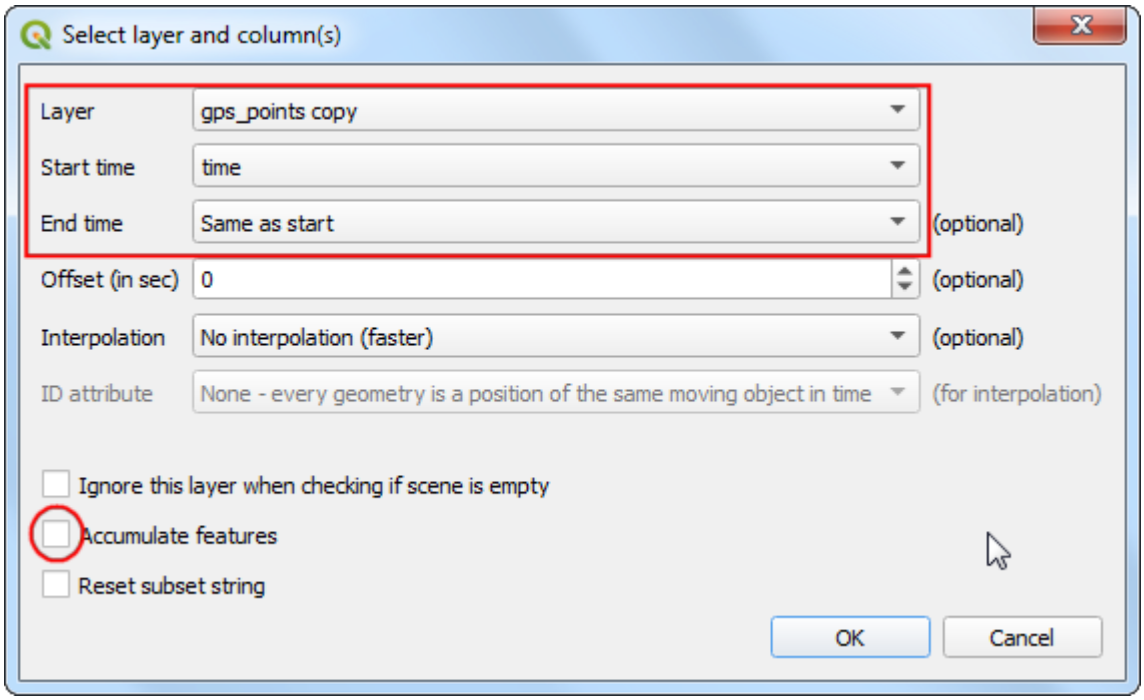

12. In the *Time Manager Settings* window, check the *Lopping animation* button. Next, click the *Add layer* button.



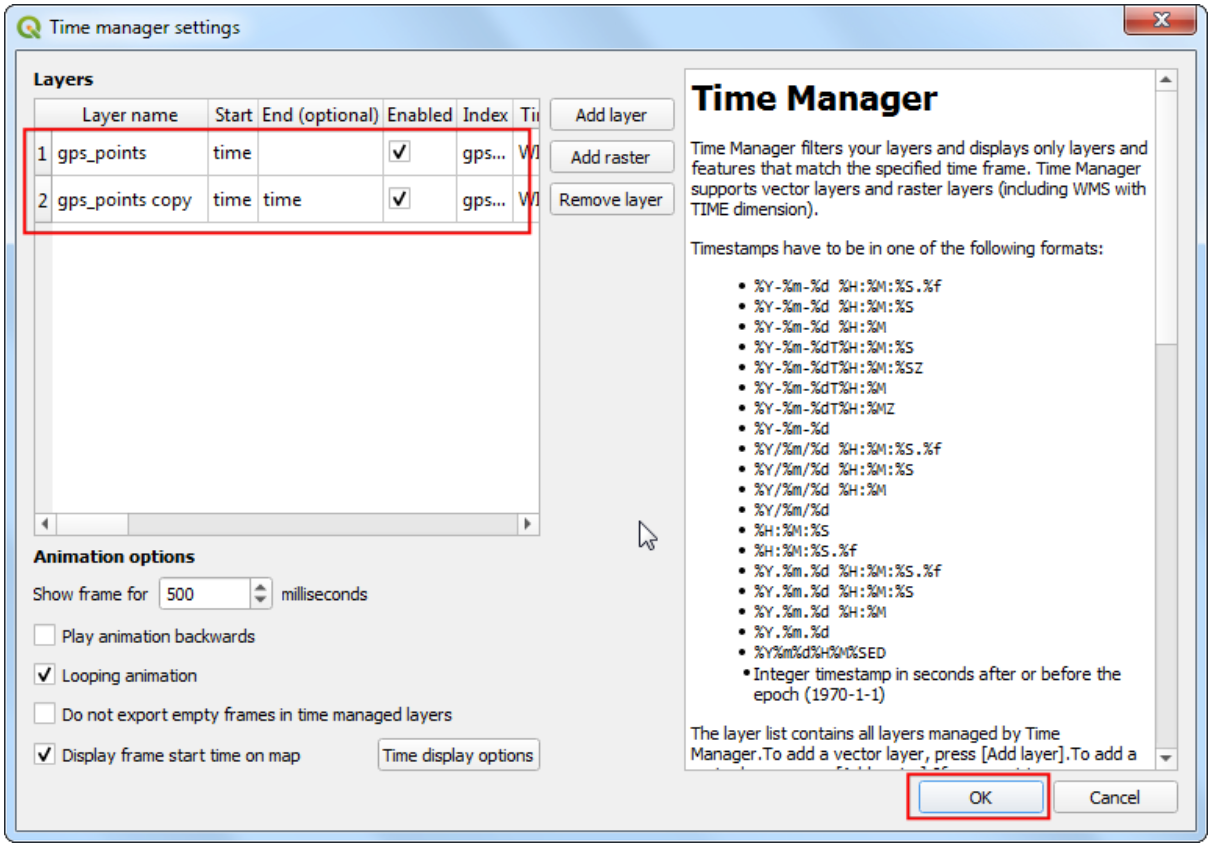

13. To achieve the effect we want, we will animate both `gps_points` and the `gps_points_copy` layers. First select `gps_points` layer. Select `time` as the *Start time* and `No end time - accumulate features` as the *End time*. Make sure to check the *Accumulate features* box. Click *OK*.



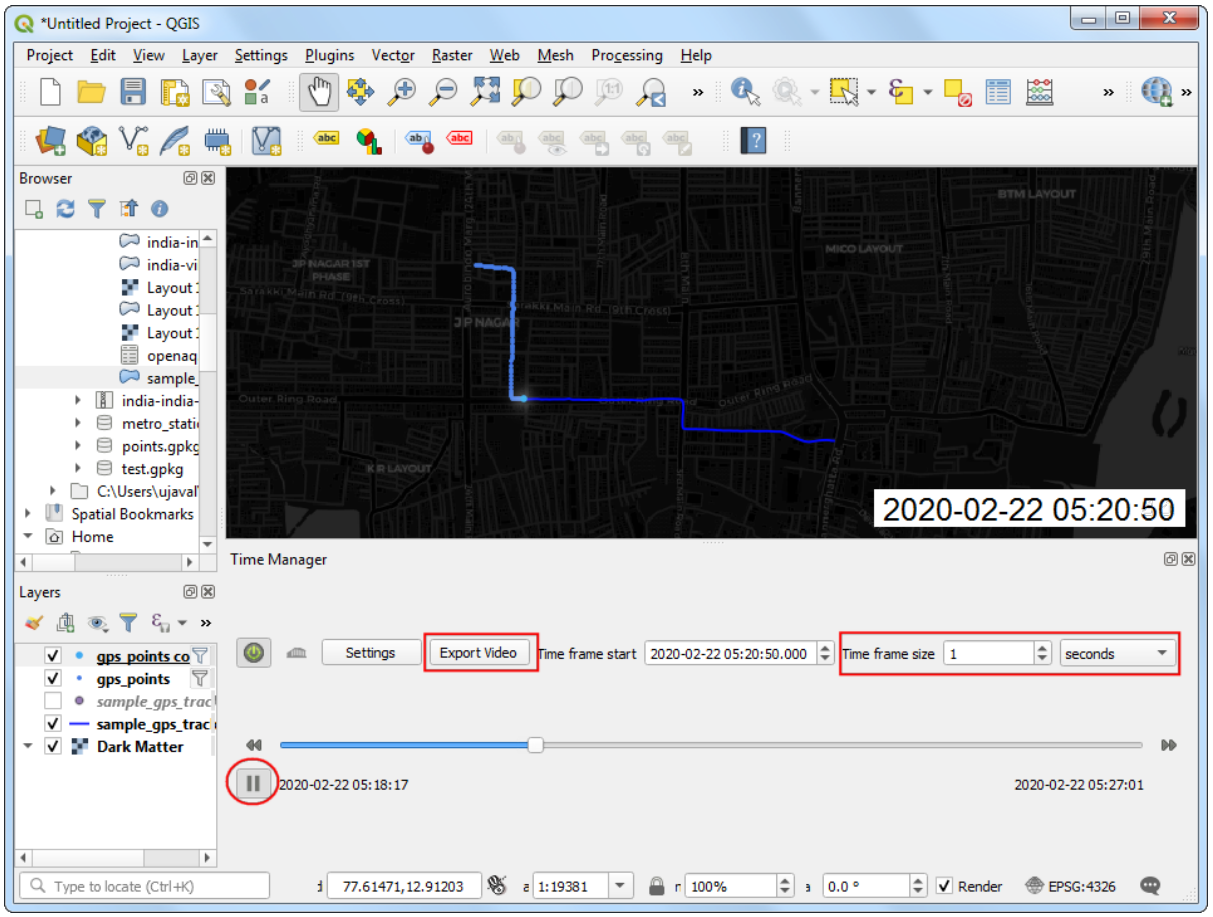

14. Click *Add layer* again and this time select `gps_points_copy` layer. Select `time` as the *Start time* and `Same as start` as the *End Time*. Uncheck the *Accumulate features* box.



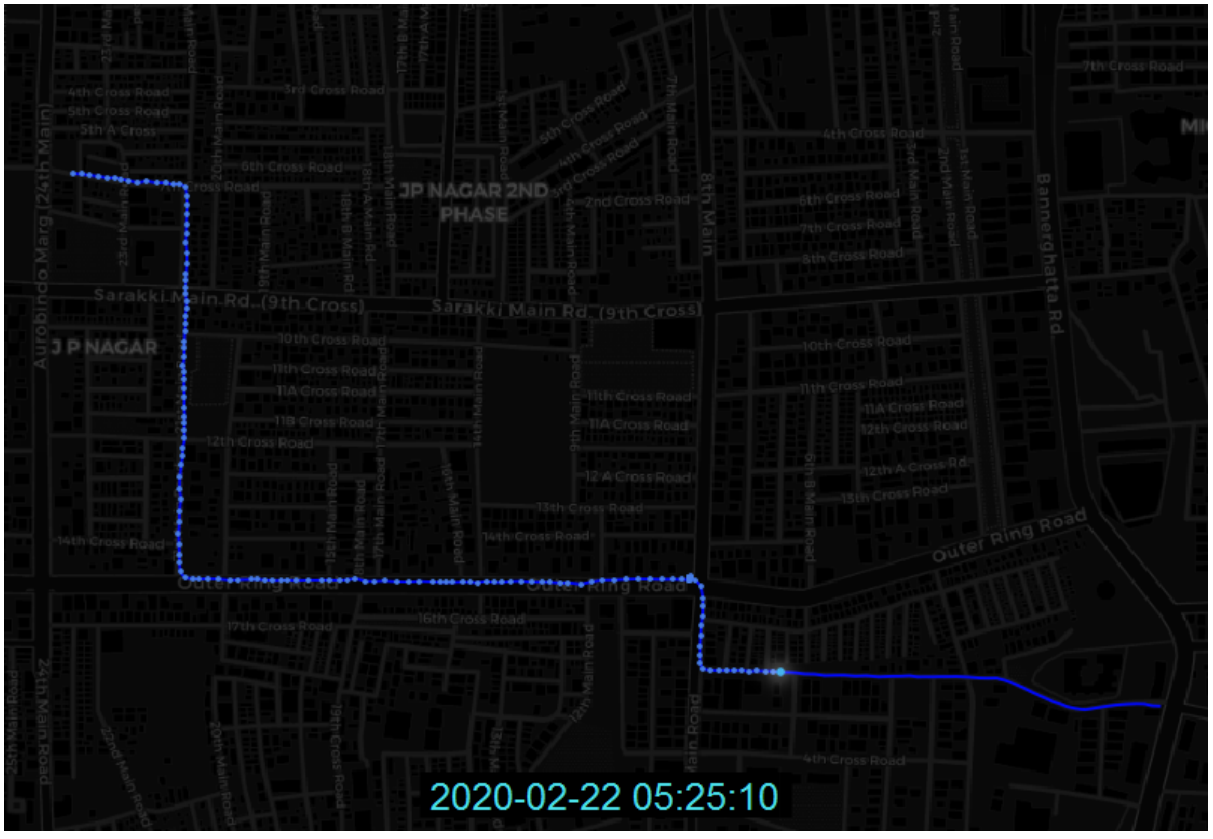

15. Once both the layers are configured, click *OK*.




16. Back in the main QGIS window, change the *Time frame size* to `1 second`. Click *Play*. You will see the points animate and show the position along with the timestamp. You can click *Export Video* to export individual frames to a folder.




17. Once you have the individual frames, you can use a program such as ezgif.com to create an animated GIF from them. The result would look as below.



# Polygons

Regions are modeled as polygons. Polygons are most commonly used to model administrative areas, buildings, land parcels etc. Polygon geometry is represented as a series of coordinates. Since the shapes can be complex, polygons have a more verbose geometry descriptions and seldom come in a CSV files. GeoJSON and shapefile are the most commonly used file formats for storing polygon datasets.

## Exercise: Mapping Census Data

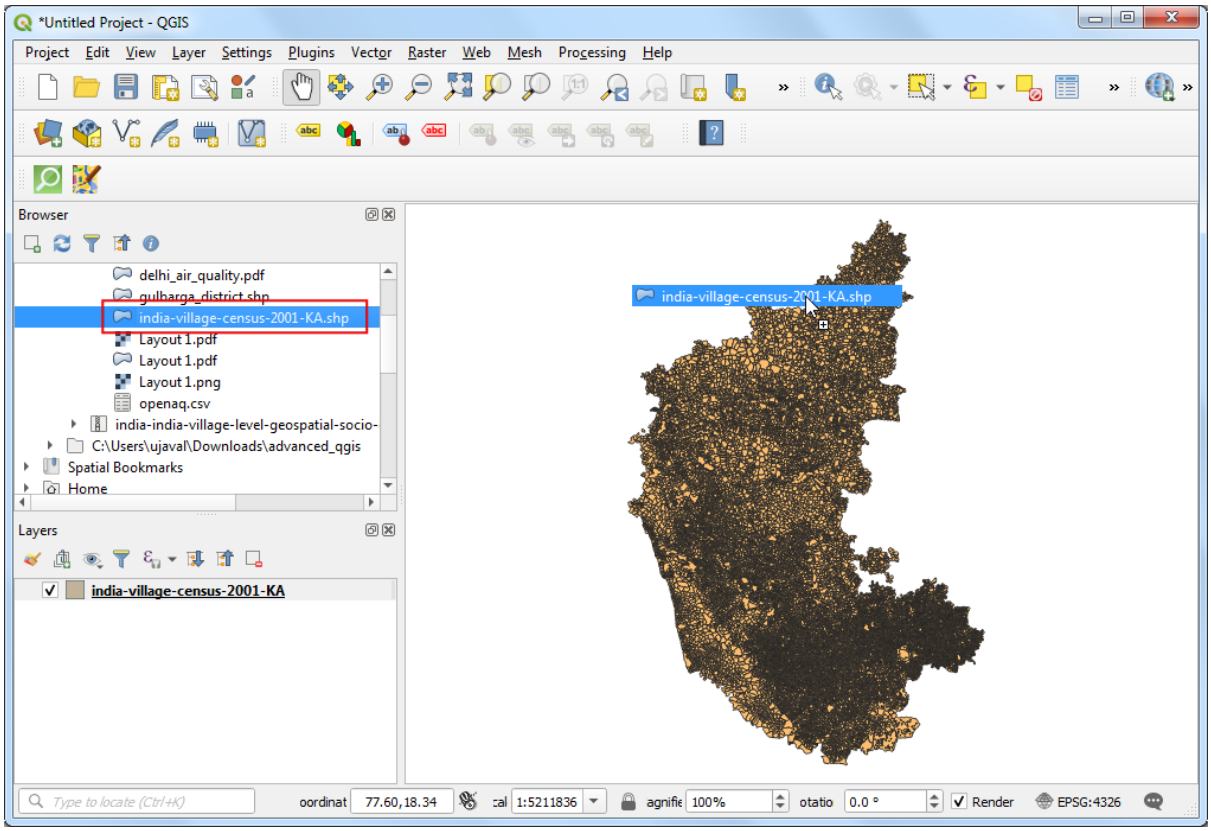
Census data is one of the major sources of secondary data available in a country. Many types of spatial analysis requires detailed demographic information that is available from the census data.

Census data is usually published as tables by aggregating the raw numbers to an administrative region - typically a *census block*. To map these tables, one needs to know the *goemetry* of these regions - which are supplied separately as boundary files. Both of these can be joined to create a polygon layer that can be visualized and mapped. See this tutorial on how this process is carried out in QGIS.

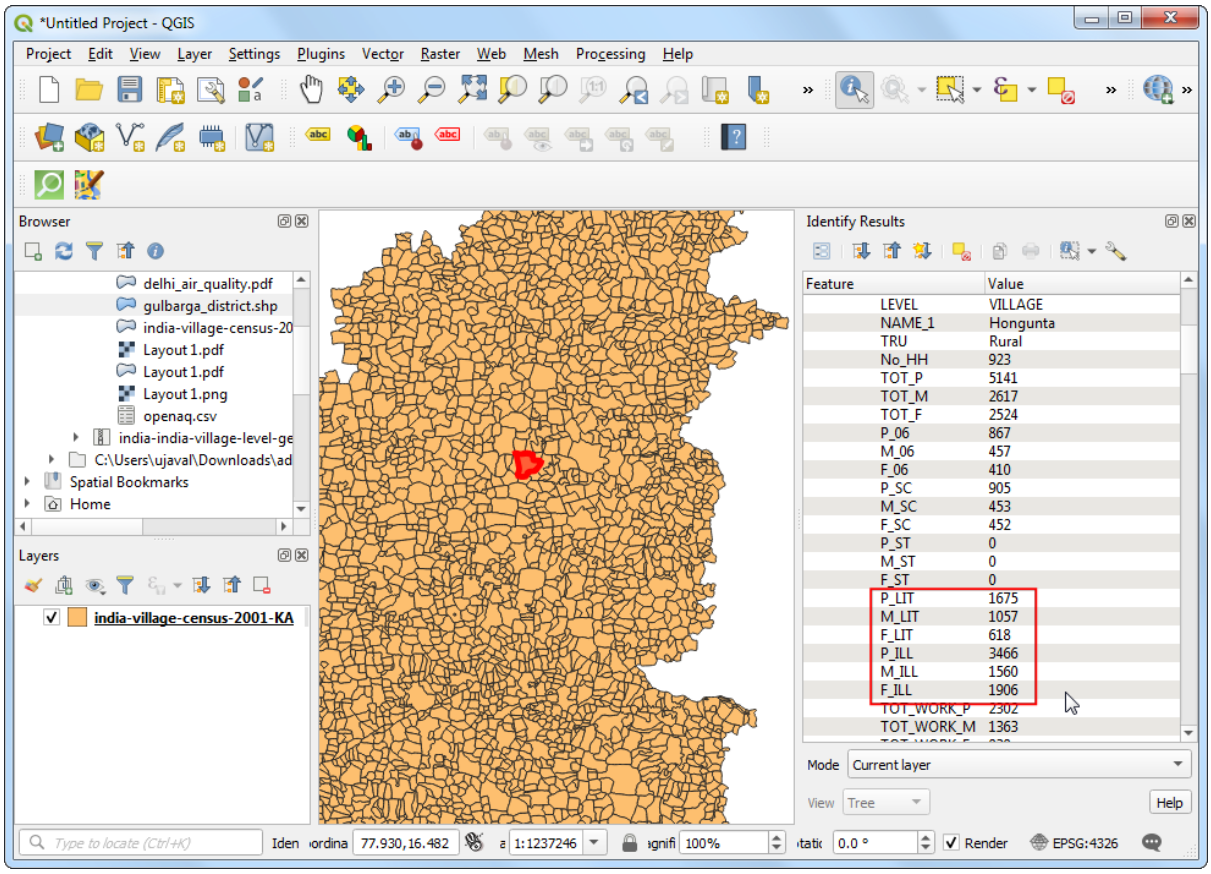
We will use India Village-Level Geospatial Socio-Economic Data Set published by NASA Socioeconomic Data and Applications Center (SEDAC). This dataset combines the village/town level boundaries with Primary Census Abstract (PCA) and Village Directory (VD) data series of the Indian census. It is distributed as shapefiles.

For this exercise, we will be using the shapefile for the state of Karnataka and map the literacy rate in the Gulbarga district.

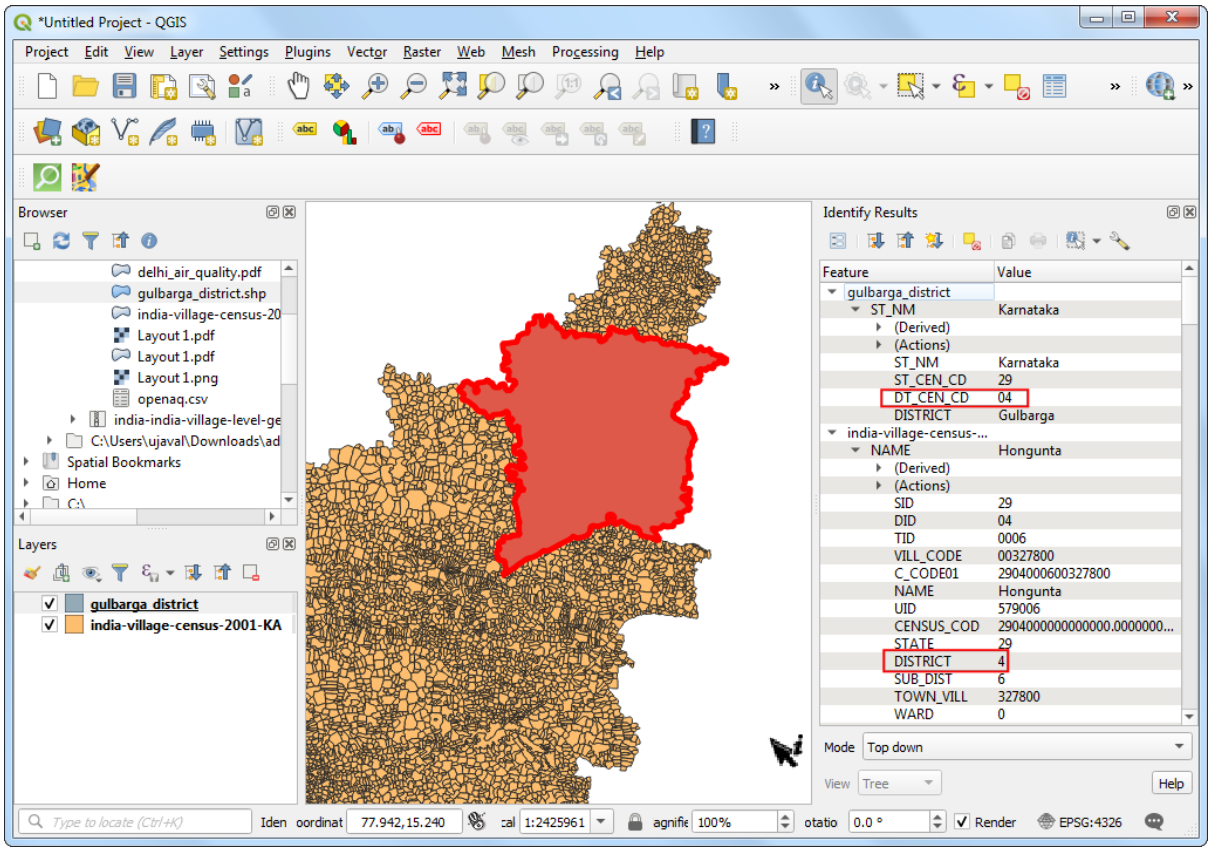
Locate the `india-village-census-2001-KA.shp` file in the *Browser* panel and drag it to QGIS canvas.



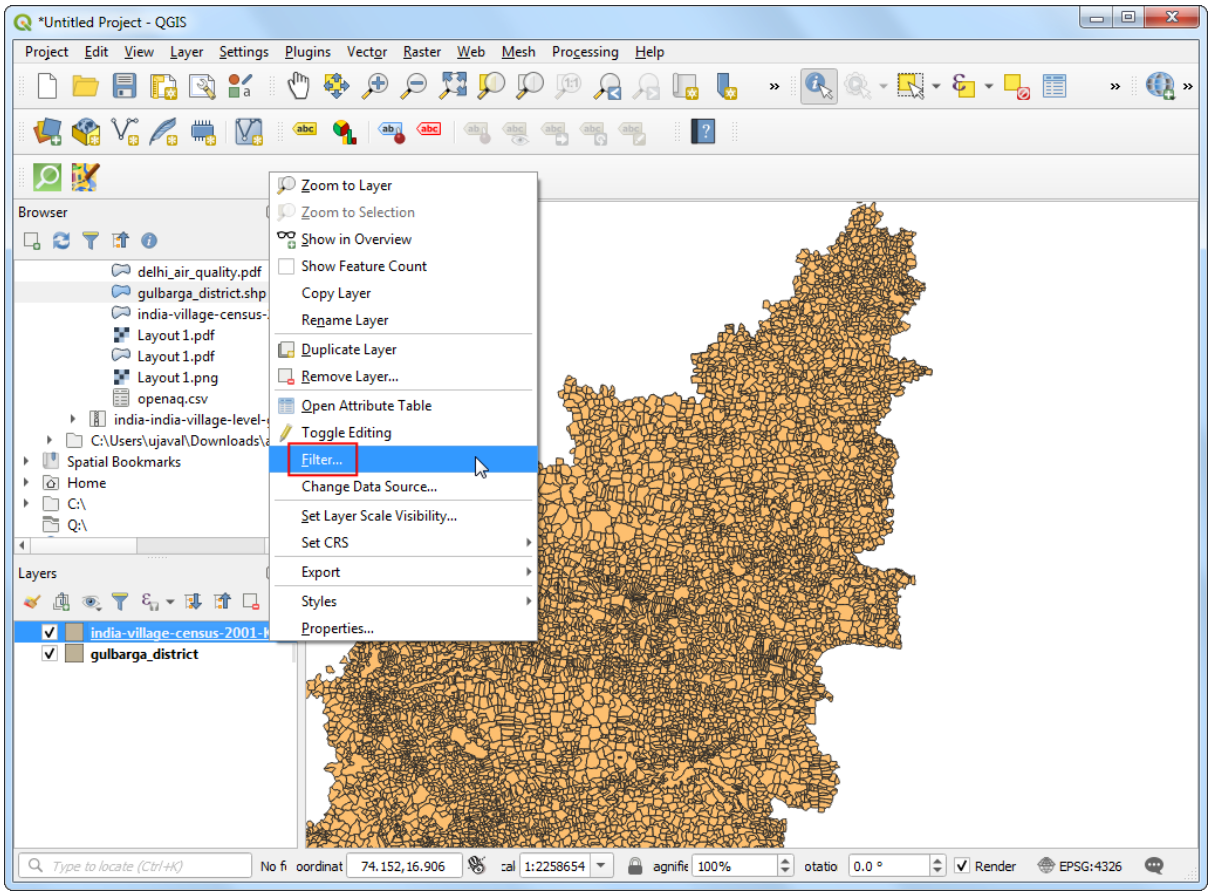
A new layer `india-village-census-2001-KA` will be added to the *Layers* panel. Use the *Identify* tool to click on any polygon are explore the attributes. The definitions of each column is contained in the documentation that is supplied with the data. As we are looking to map the literacy levels, the attributes with `_LIT` suffix are useful for our purpose. The `P_LIT` column refers to *Person Literates* and `TOT_P` refers to *Total Population* that we will use to calculate and map literacy rate.



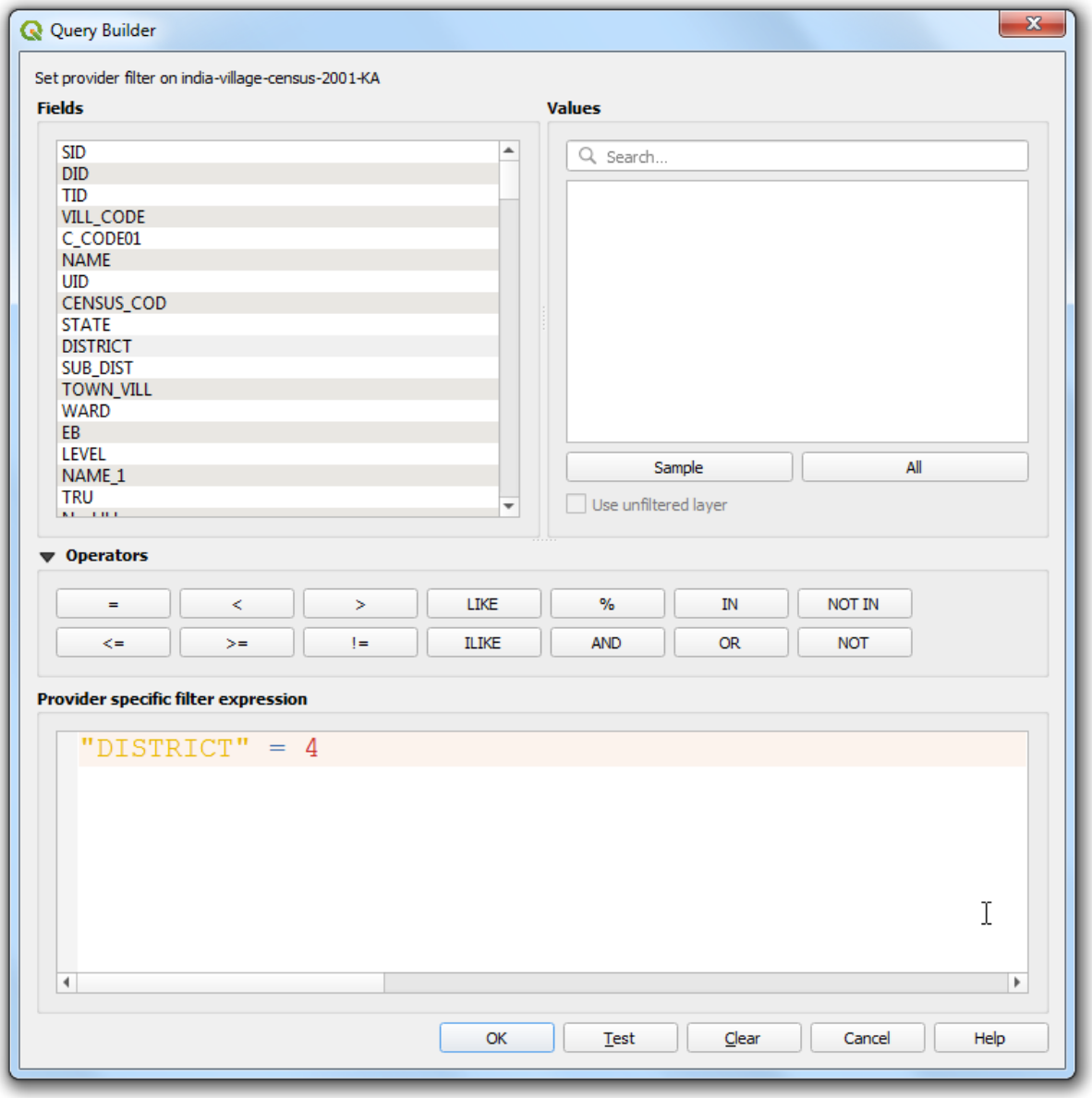

We want to select a subset of polygons from this layer belonging to the *Gulbarga* district. Load the `gulbarga_district.shp` layer that has been extracted from the Districts shapefile supplied by DataMeet. The column `DT_CEN_CD` contains the district id for this particular district. We can use this to filter the polygon layer.



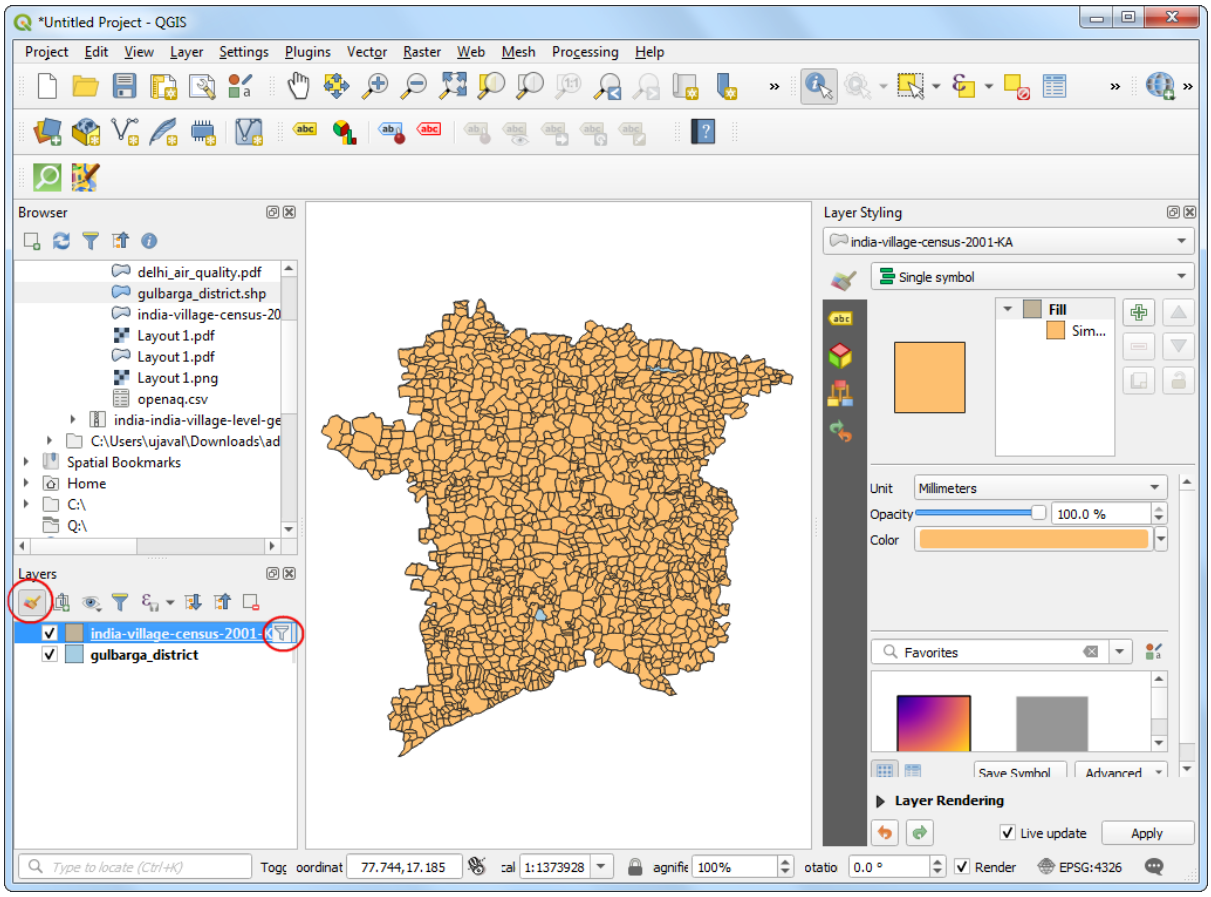

In the *Layers* panel, drag the `gulbarga_district` layer below the `india-village-census-2001-KA` layer. Right-click the `india-village-census-2001-KA` layer and select *Filter*.



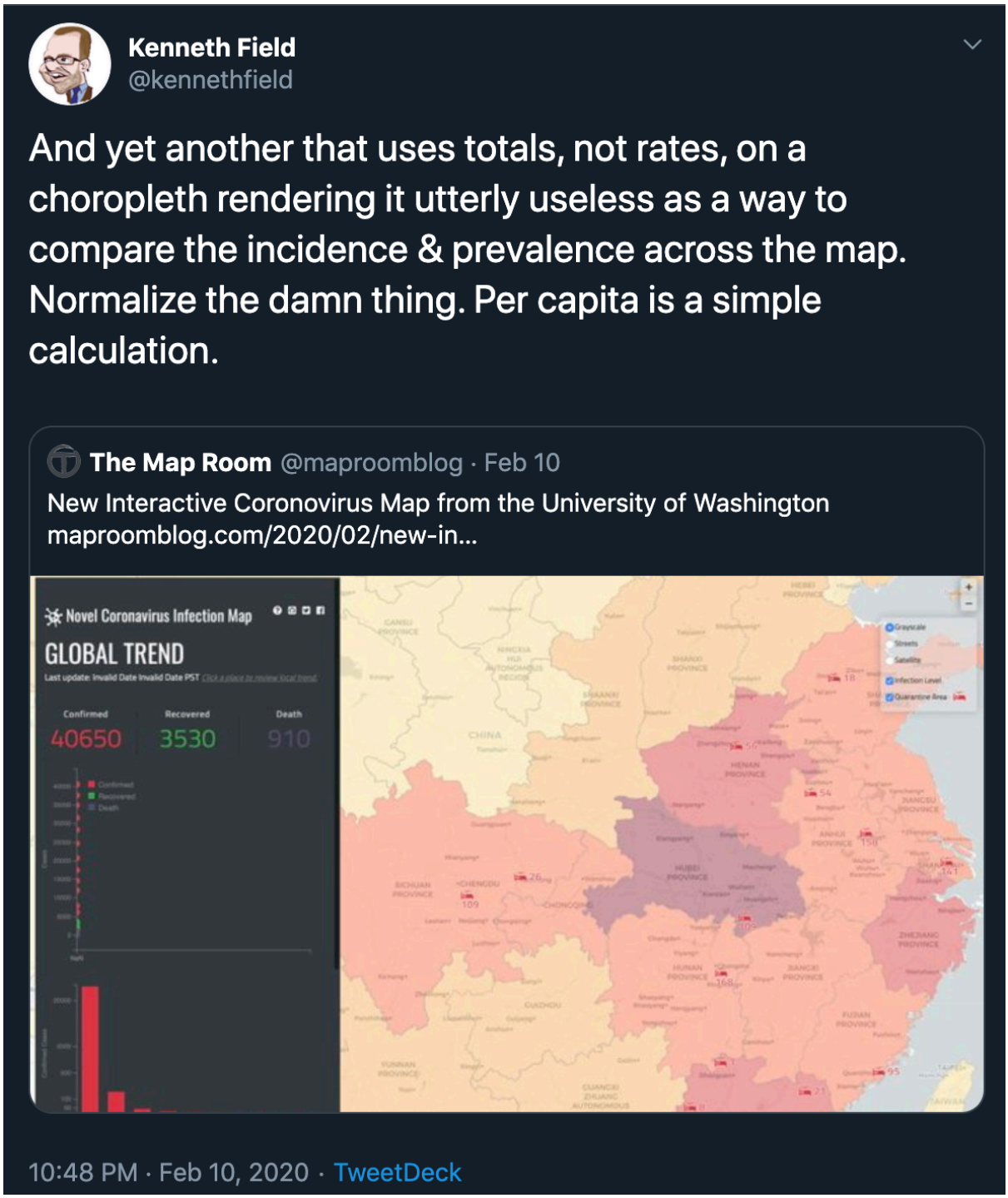

Enter the expression `DISTRICT = 4` to select all villages and towns from our chosen district. Click *OK*.



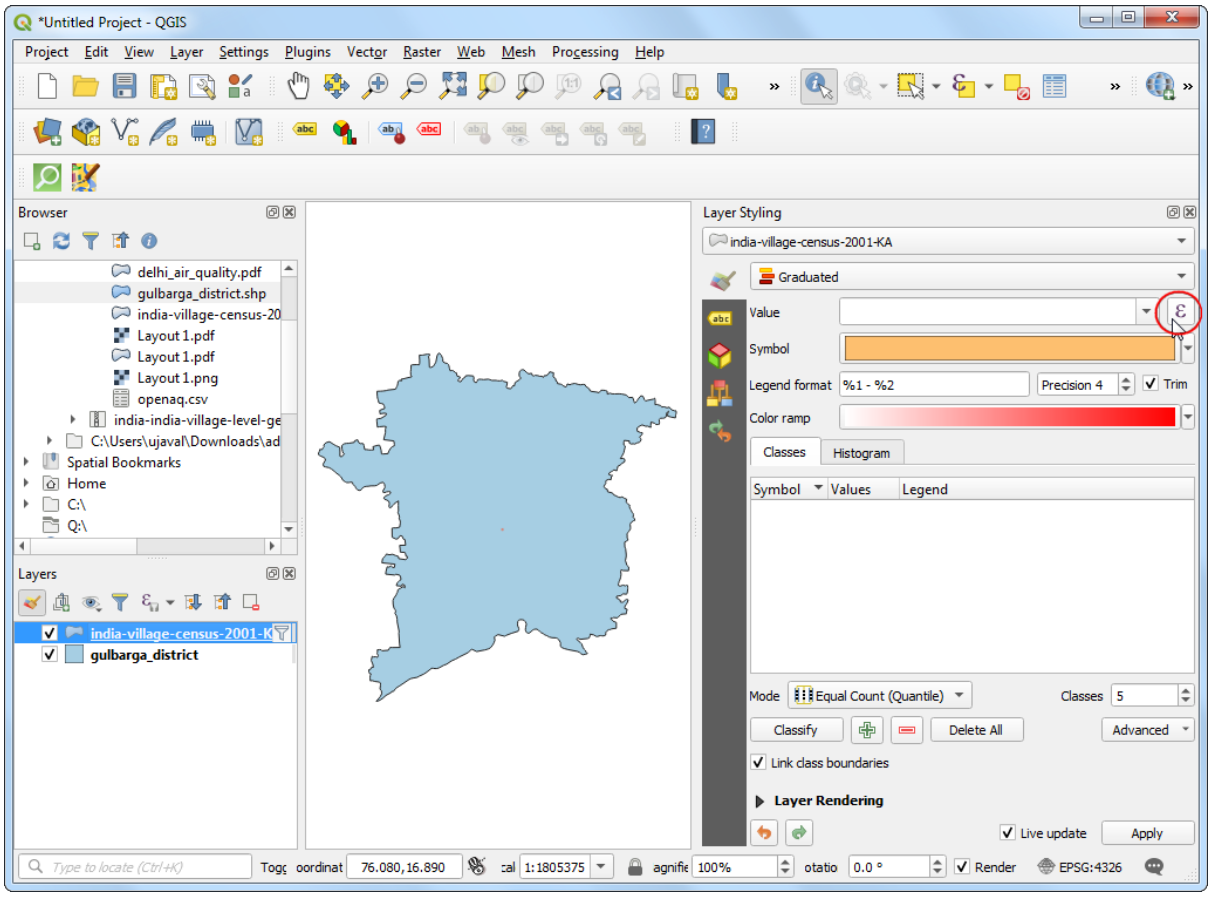

You will see a filter icon next to the `india-village-census-2001-KA` layer indicating that a filter is applied to the layer. The map canvas will update to show only the polygons belonging to the district.




Now we will create a *thematic* map showing the literacy rate in the district. When creating a thematic (choropleth) map such as this, it is important to normalize the values you are mapping. A common mistake is to use totals instead of rates in such map.

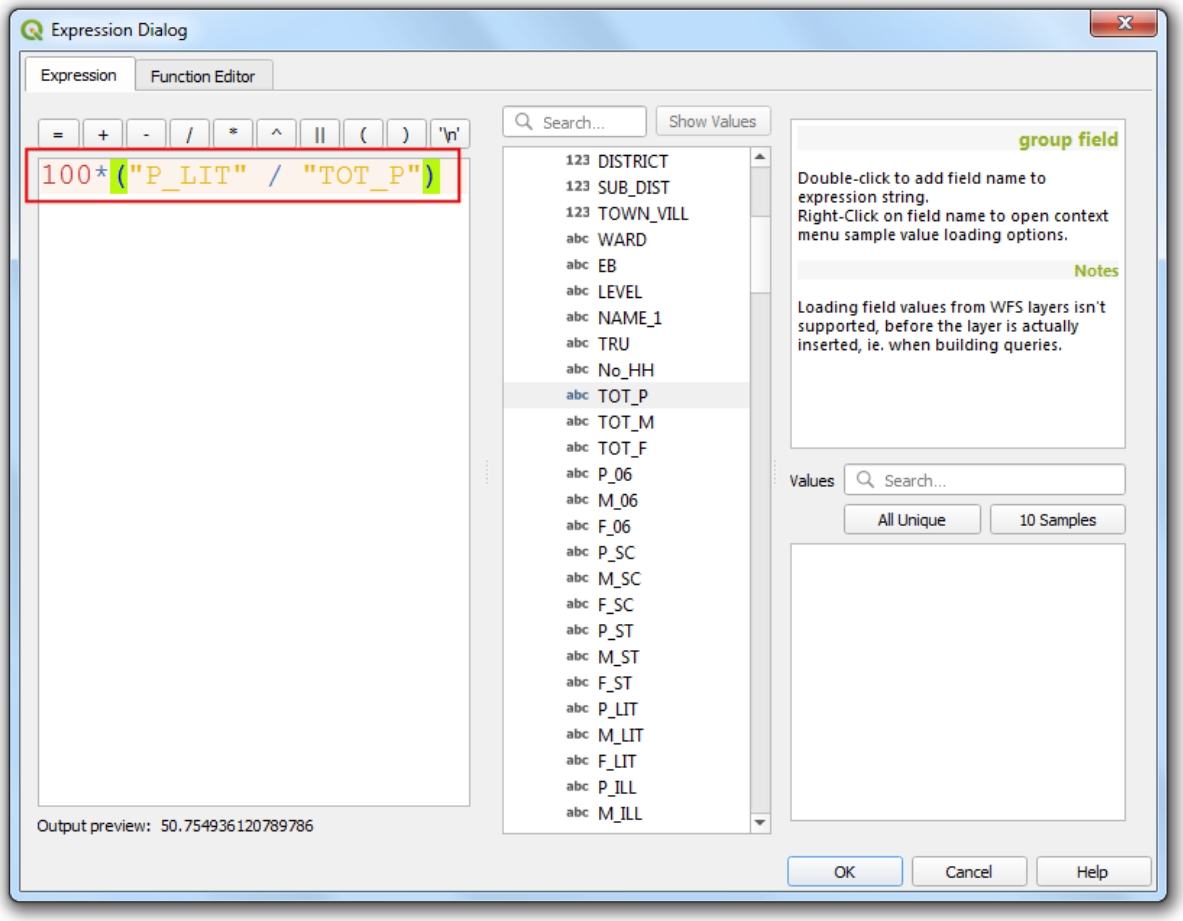



Click *Open the Layer Styling Panel*. Select *Graduated* renderer. In the *Value* column, click the *Expression* button.

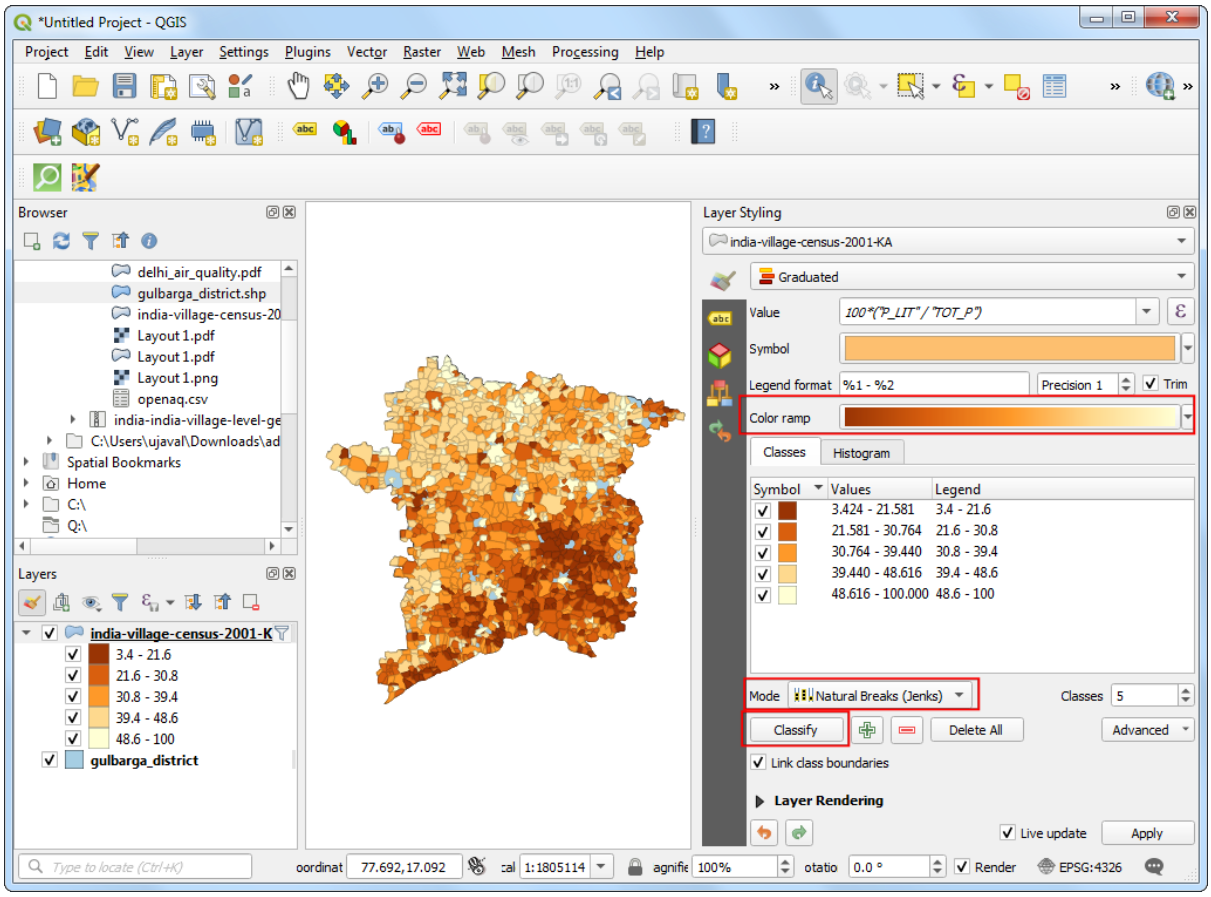



Enter the following expression. As we want to map literacy rate, we can normalize the total literate persons by dividing with the total population.

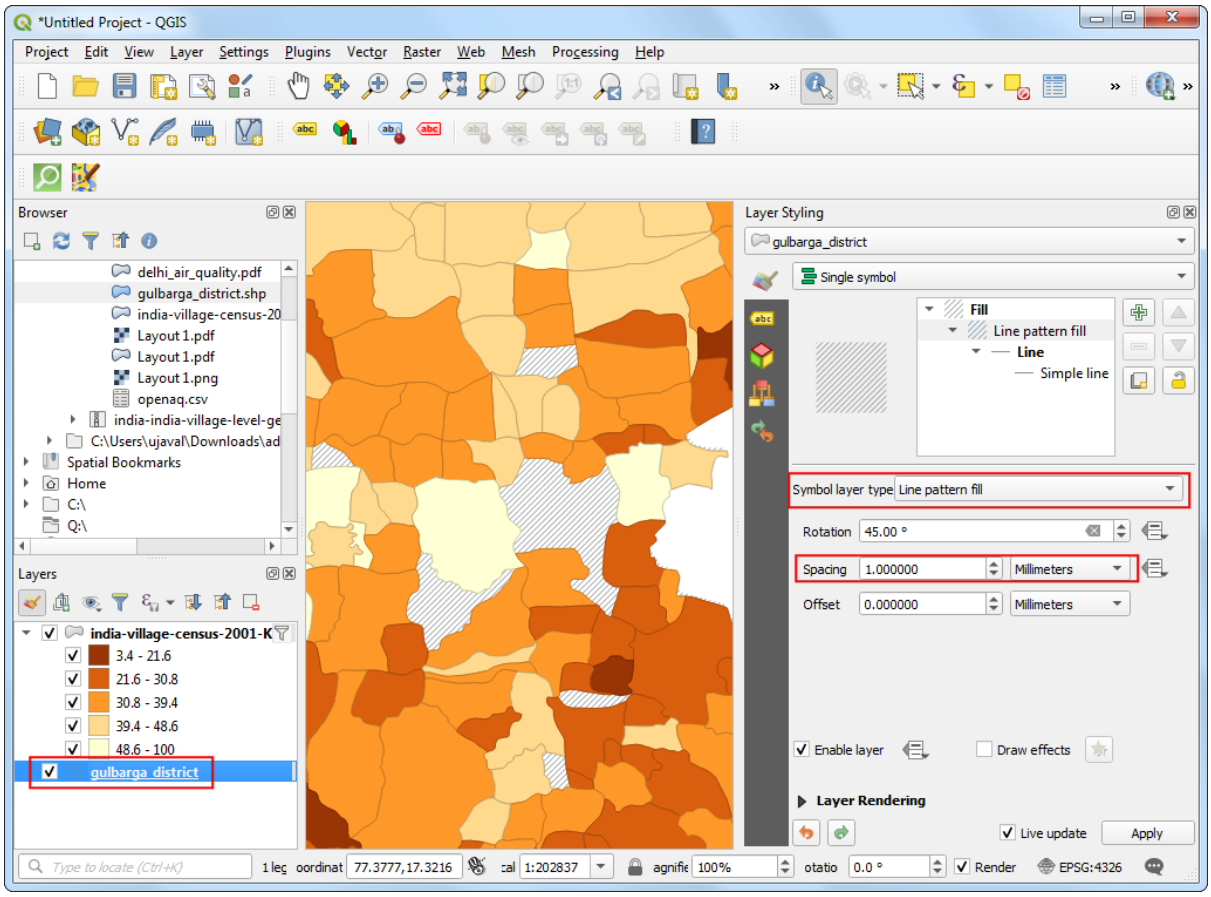
```
100*("P_LIT"/"TOT_P")
```



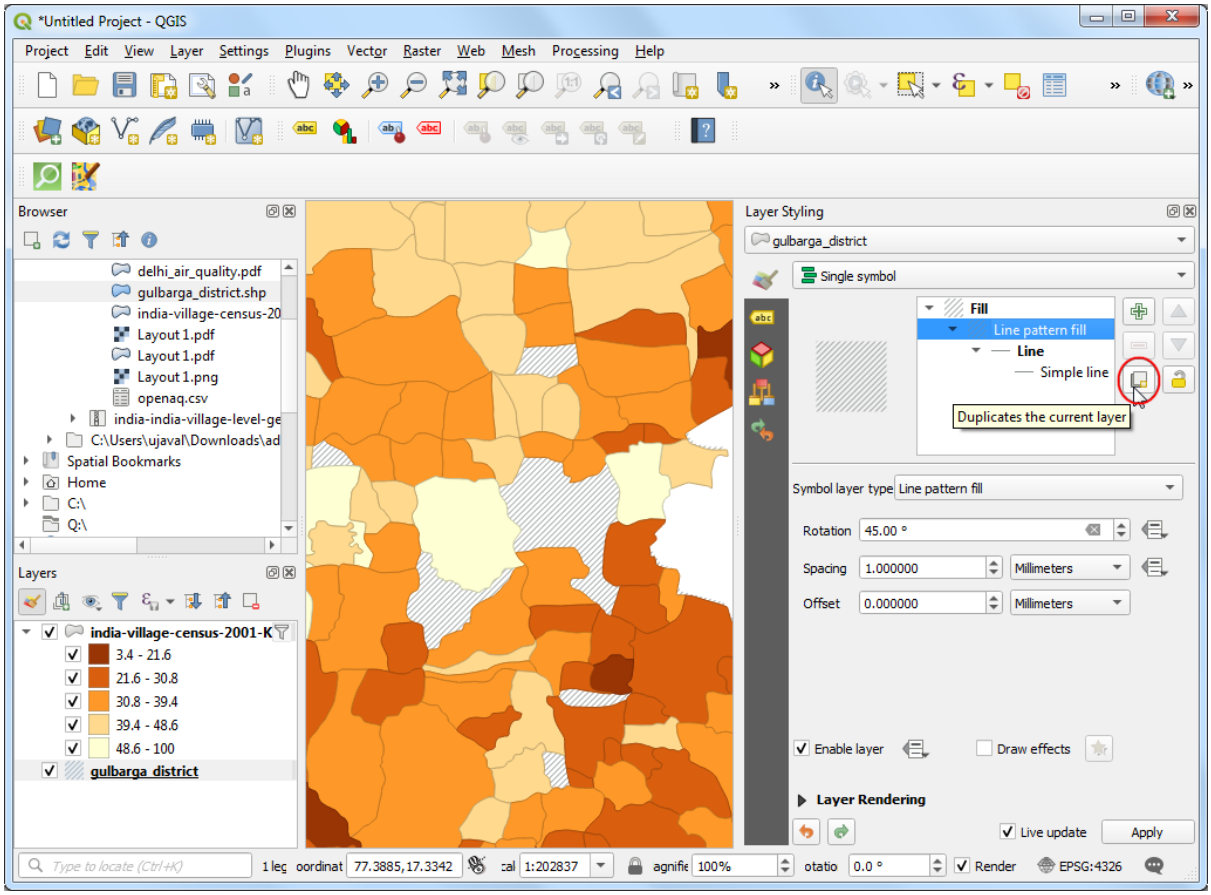

Choose a color ramp and the *Mode* of your choice and click *Classify*. You will see the polygons colored according to the literacy rates. Mapping this makes is much easier to see the pattern that villages to the south of the district have much lower literacy rates than their northern counterparts.



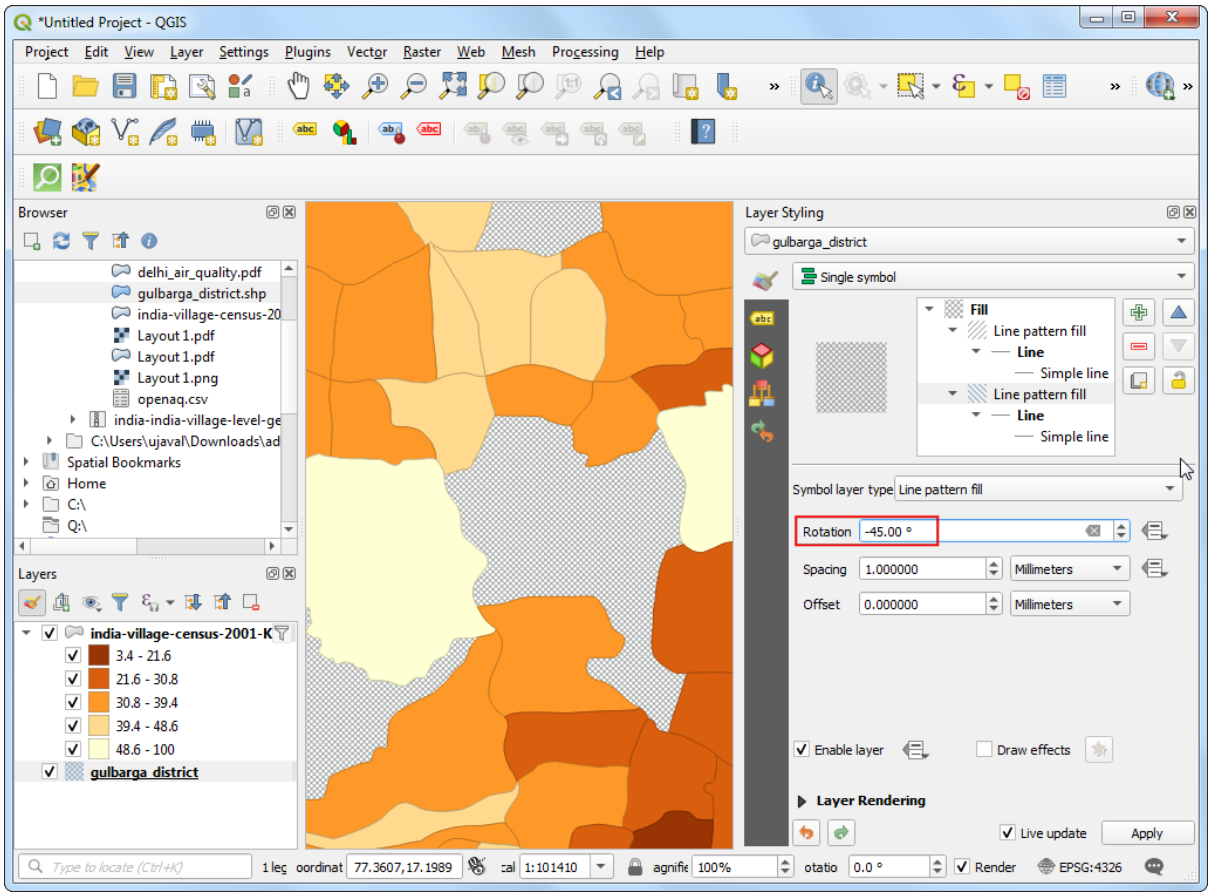

If you zoom in - you will notice gaps in the polygon layer. These are the areas with cities which do not have Village Directory (VD) data tables and thus excluded in this data layer. Instead of a hole, we can style is better to indicate no-data values. Select the `gulbarga_district` layer. Change the *Symbol layer type* to be `Line pattern fill`. Change the spacing as per your liking.



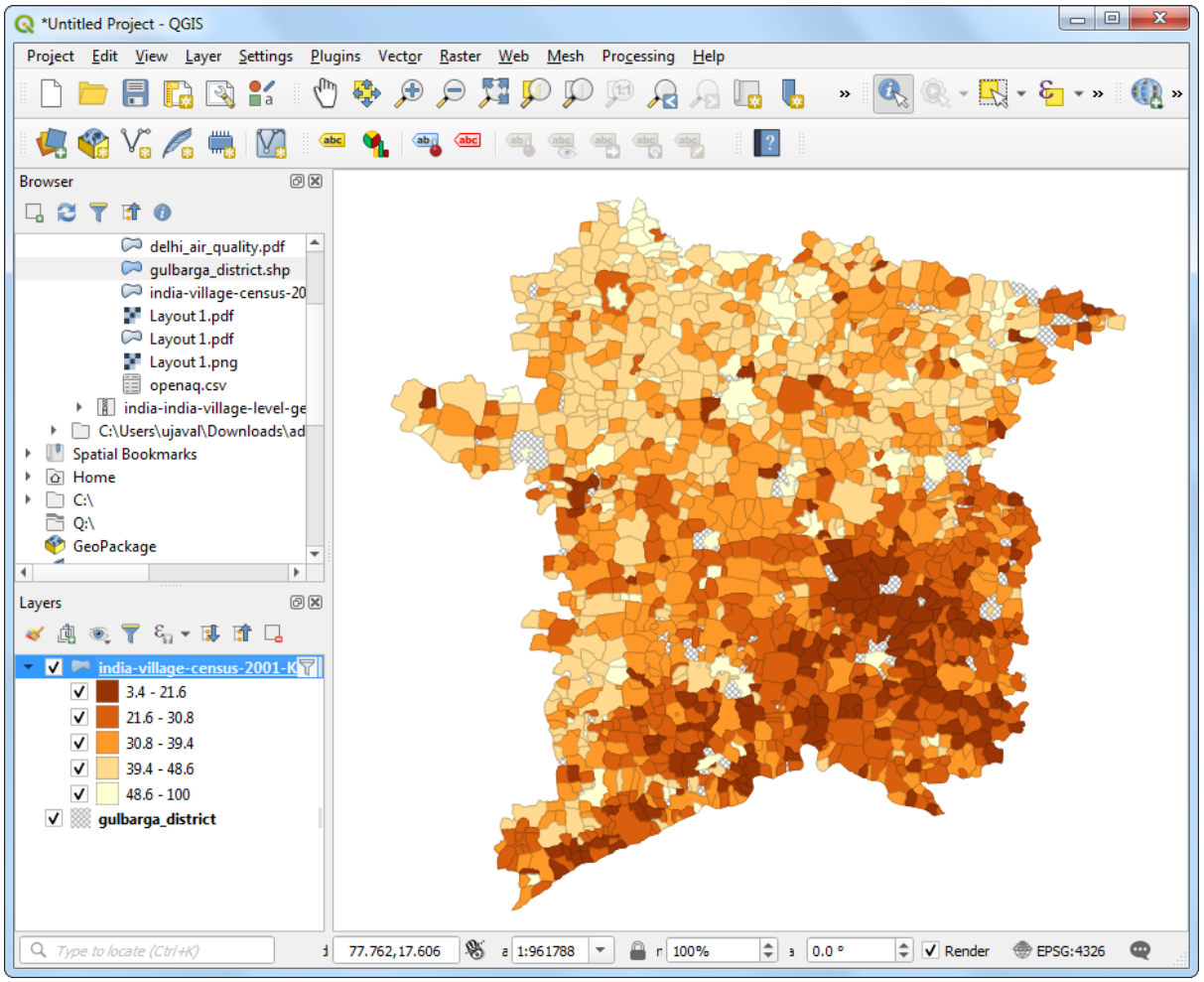

You will notice that the options for fill do not have a cross pattern fill. That is because it can be easily created by combining 2 line pattern fills of opposite direction. Click the *Duplicate the current layer* button in the *Layer Styling Panel*.



Change the *Rotation* value to `-45.00` degrees. You will see the gaps now rendered with a cross-pattern fill.




You have a visualization of the literacy rate in the district.



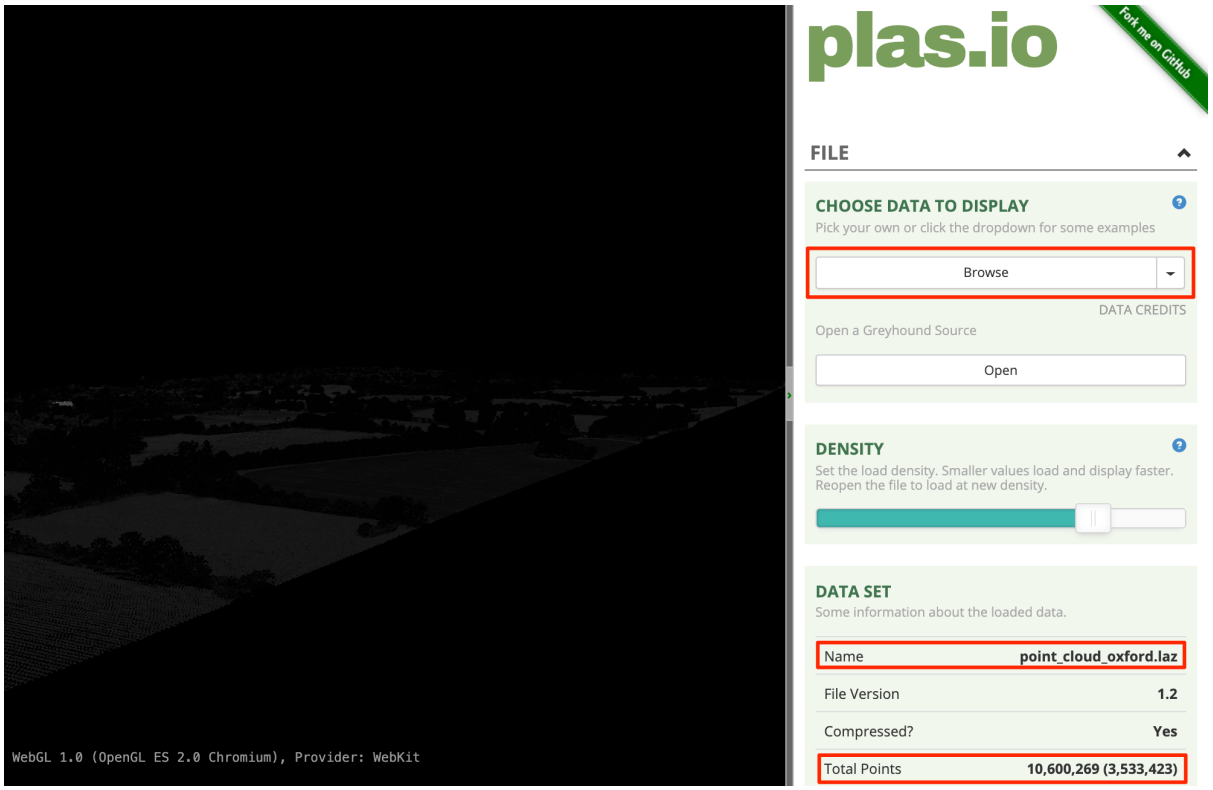

# Point Clouds

Modern mapping technology includes doing aerial surveys using a LiDAR sensor. LiDAR stands for "Light Detection and Ranging". This sensor uses light pulses to determine the distance to the objects of the ground. For each light-pulse that is sent out the system computes X,Y and Z coordinates of the object. This data representation is not new for spatial data - but since survey of even a small area can result in millions of such points - standard tools for viewing and processing points do not work. Such a *Point Cloud* is typically stored in the LAS or LAZ formats.

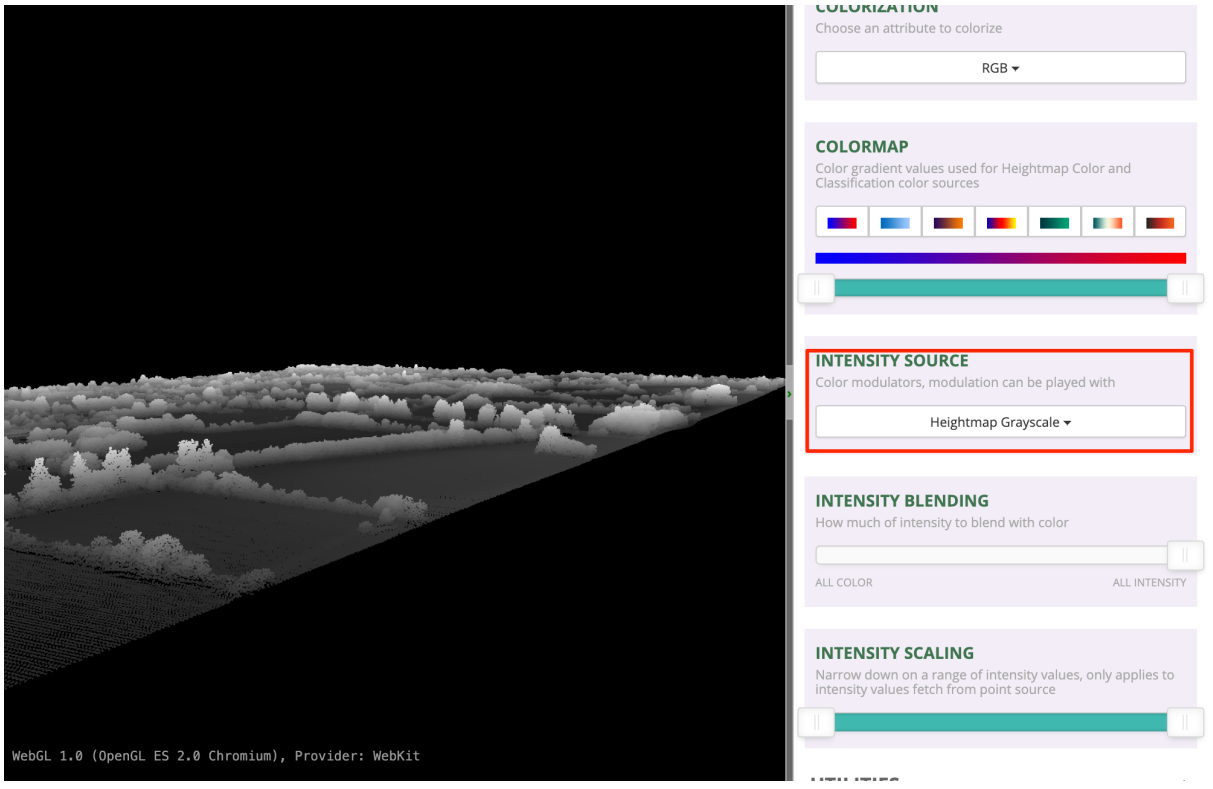
## Exercise: View Point Cloud from an Aerial Survey

UK's Department of Environment Food & Rural Affairs (DEFRA) provides country-wide LiDAR data and products via the Defra Data Services Platform under an open license. We will use a point cloud dataset for Oxford University available as a LAZ file.

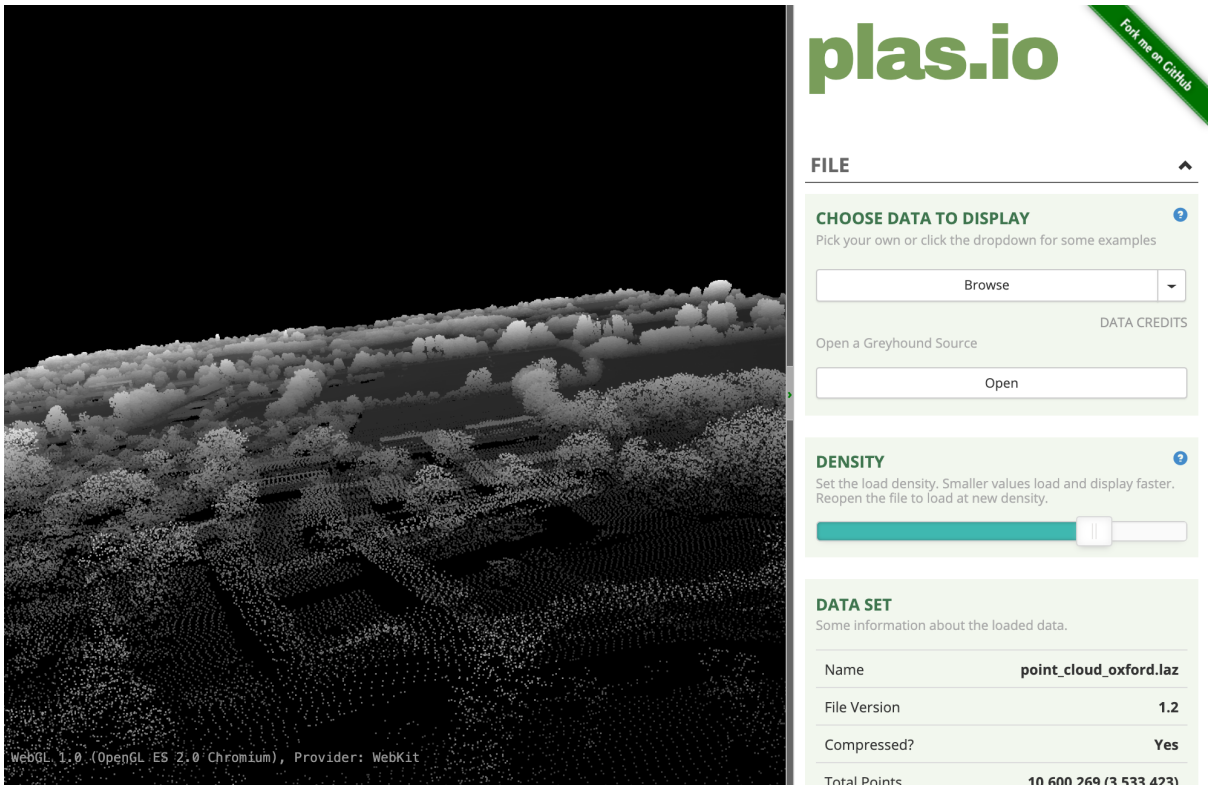
Plasio is a browser-based LAS/LAZ point cloud viewer. Visit plas.io. Click *Browse* and locate the `point_cloud_oxford.laz` file. Click *Open*. Note that this small region is rendered with over 3.5M source points.




Once the data is loaded, scroll down in the right-hand panel and locate the *Intensity Source* section. Select `Heightmap Grayscale` option from the drop-down selector.




Use the left-mouse button and scroll-wheel to zoom/pan around and explore the dataset.




# Raster - Photos

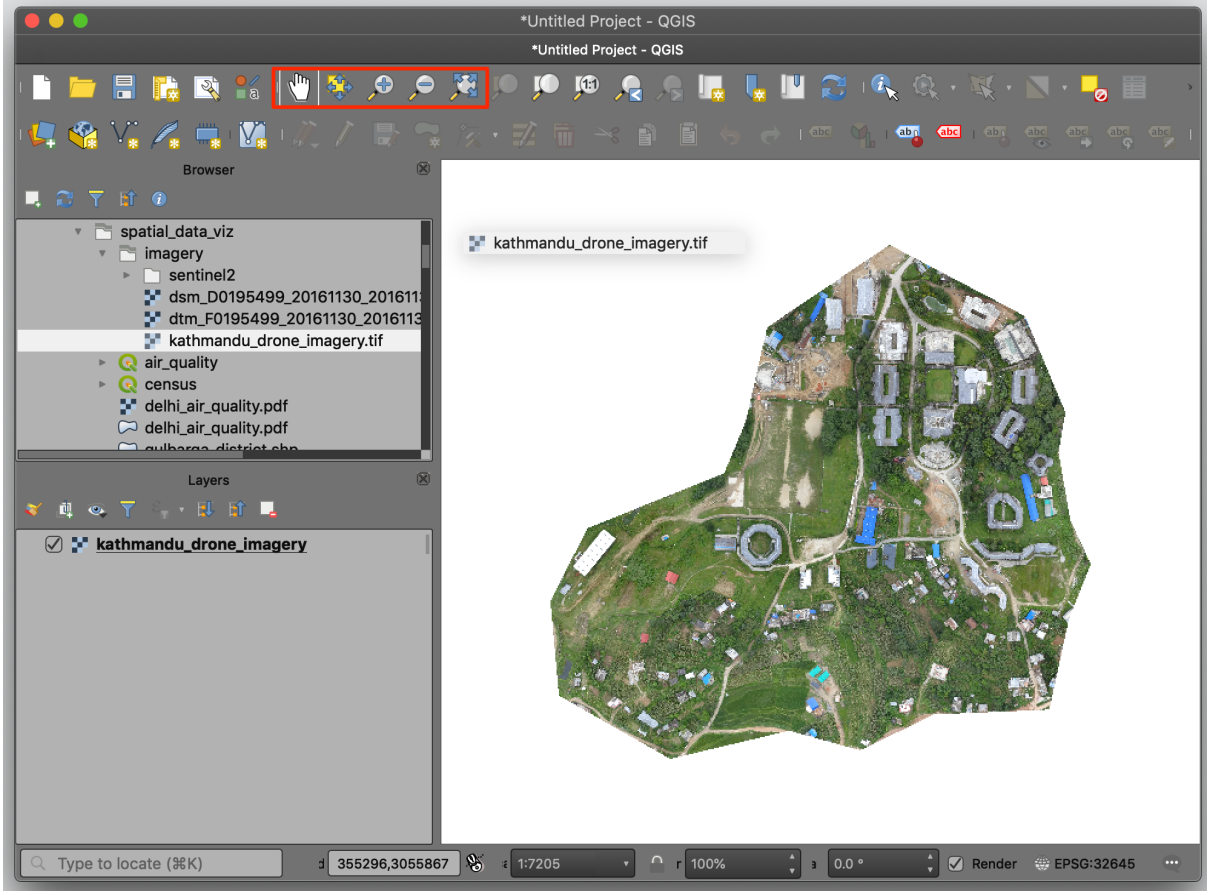
Photos collected from airborne sensors - such as kites, hot-air balloons, planes, helicopters and more recently UAVs - are useful source of information for mapping. They often act as a basemap - providing context for other spatial data. They are also used to extract feature information that are modeled as vector data.

The most common format for imagery is GeoTiff. A geotiff file contains additional metadata that allow us to convert pixel location (row/column) to a real-world location (latitude/longitude). A regular photo can be converted to a spatially-aware raster through a process known as GeoReferencing.

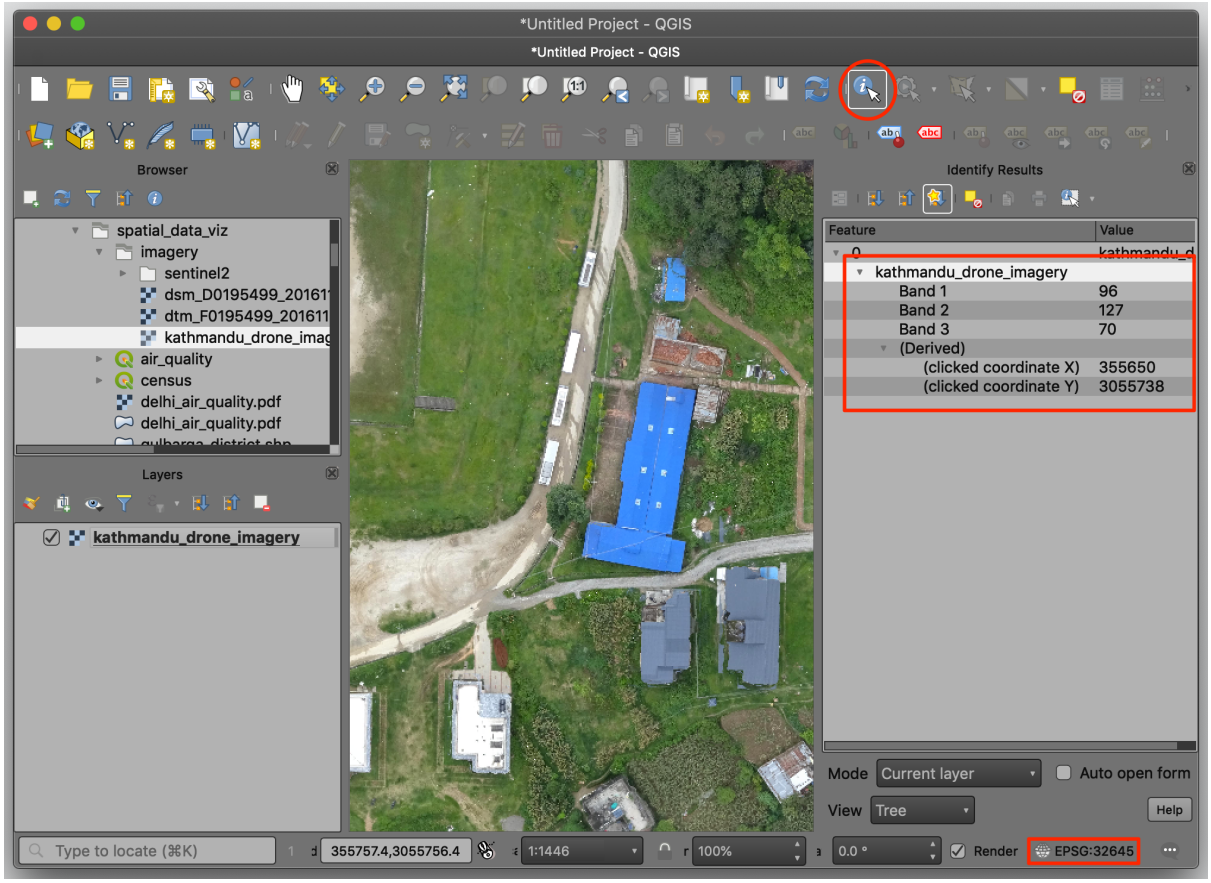
## Exercise: View Drone Imagery

OpenAerialMap is an open service to share and download overhead imagery. We will use an image of Kathmandu University Grounds shared by WeRobotics.

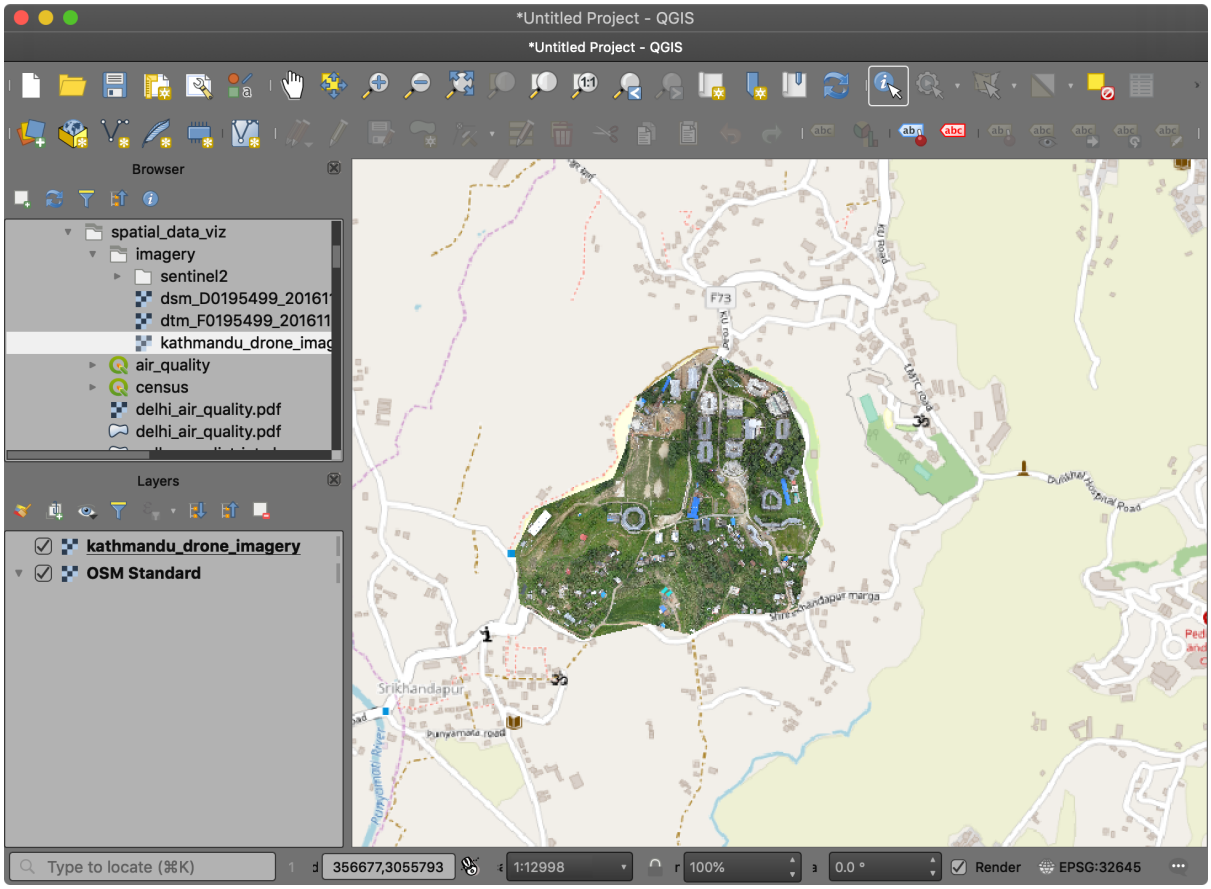
Browse to the `kathmandu_drone_imagery.tif` file and drag it to QGIS. Use the Zoom/Pan tools to explore the imagery.




At the bottom right, notice that the CRS is EPSG:32645 which refers to the UTM Zone 45N. Select the *Identify* tool and click anywhere on the image. You will see that the image contains 3 bands - one each for Red, Green and Blue. The coordinates are projected coordinates - not geographic coordinates. These are referred as *X (Easting)* and *Y (Northing)*.




As the image is also georeferenced, we can see it in context of other spatial layers. Go to **Web → QuickMapServices → OSM → OSM Standard**. A tile layer will be added to the *Layers* panel and you can see both line up with each other since both layers are spatially aware.



# Raster - Satellite Images

There are hundreds of Earth Observation Satellites in space continuously capturing images of the earth. Many space agencies around the world make this data available freely. These datasets are immensely valuable to scientists, researchers, governments and businesses.

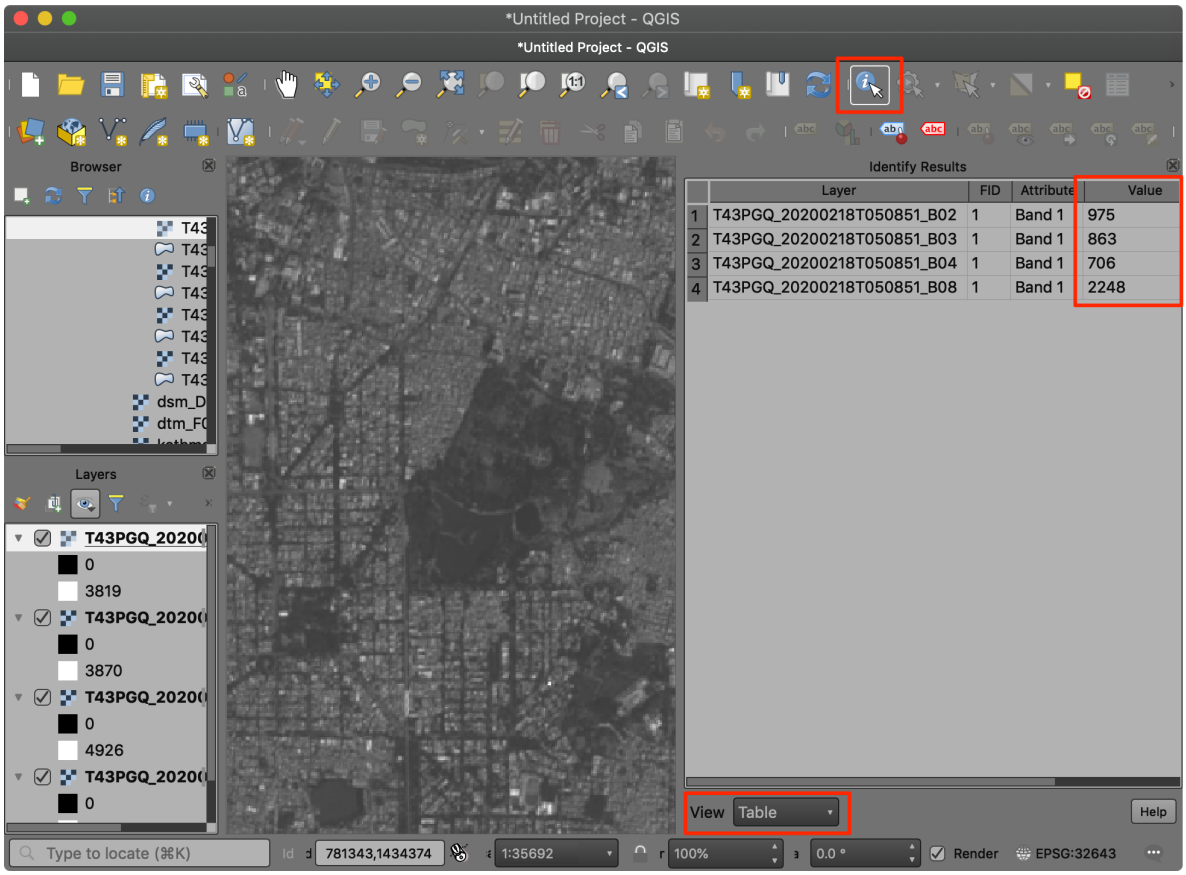
The satellite images are different than regular photos because they contain information in across many bands of wavelengths - not just - Red, Green and Blue. This rich information allows machine learning models to easily distinguish different objects. For example, an astroturf and real lawn may both look green, but reflect the infrared light very differently. So one can easily differentiate between these using the additional information contained in a multi-spectral image.

## Exercise: View Sentinel-2 Image and Create Composites

Sentinel-2 is a European Space Agency (ESA) mission with 2 satellites. The resolution of each pixel in the image is 10 meters. This is lower than drone or aerial imagery resolution, but still good enough for city and region level analysis. More importantly, the data is captured in 12 different bands - making it very useful for scientific applications. These mission capture every location on the earth every 5 days, allowing for continuous monitoring of the whole each. ESA also makes all the data from this mission freely available.

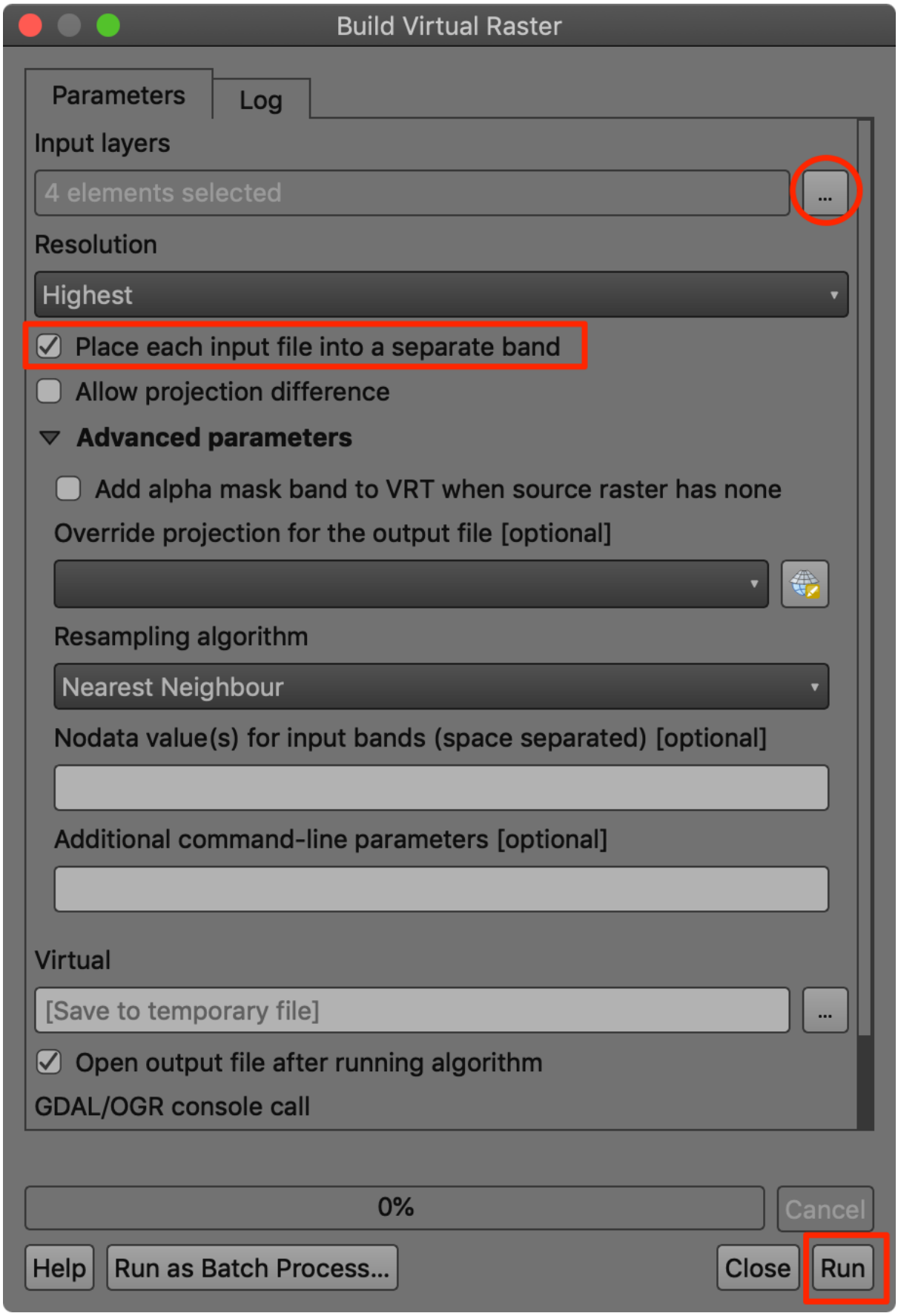
We will load a Sentinel-2 image for Bangalore, India captured on 18 February, 2020. The data package directory `imagery/sentinel-2` 4 files in JPEG2000 format. The 4 files are for bands Red (B4), Green (B3), Blue (B2) and Near Infrared (B8).

Drag all 4 images to the QGIS canvas. Zoom in to any area and explore how the same object has different reflectance in different bands.




We can combine multiple bands into a single image and visualize it. These are called *Color Composites*. To merge individual images to a single one, go to **Processing → Toolbox** and find **GDAL → Raster miscellaneous → Build virtual raster**. Double-click to launch the tool.

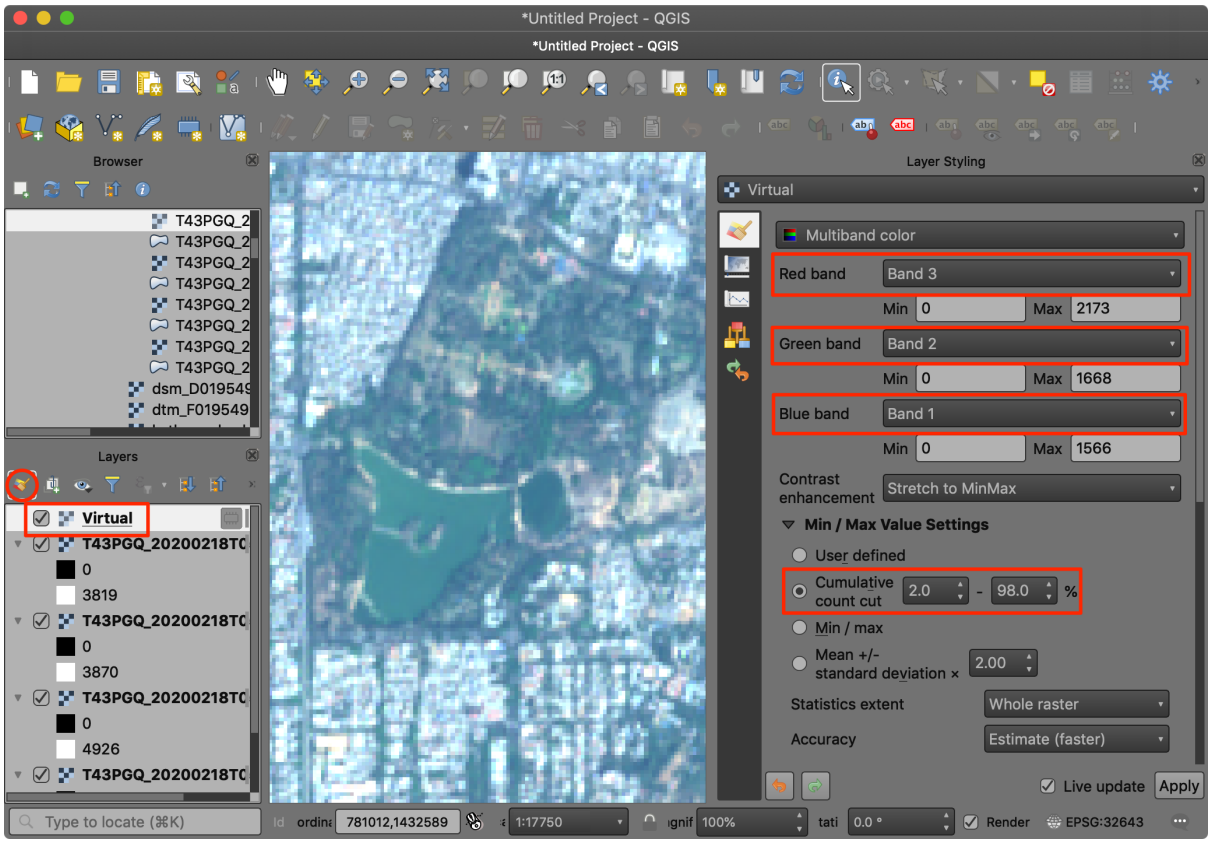
Select all 4 layers as *Input layers* and check *Place each input file into a separate band*. Next click *Run*.



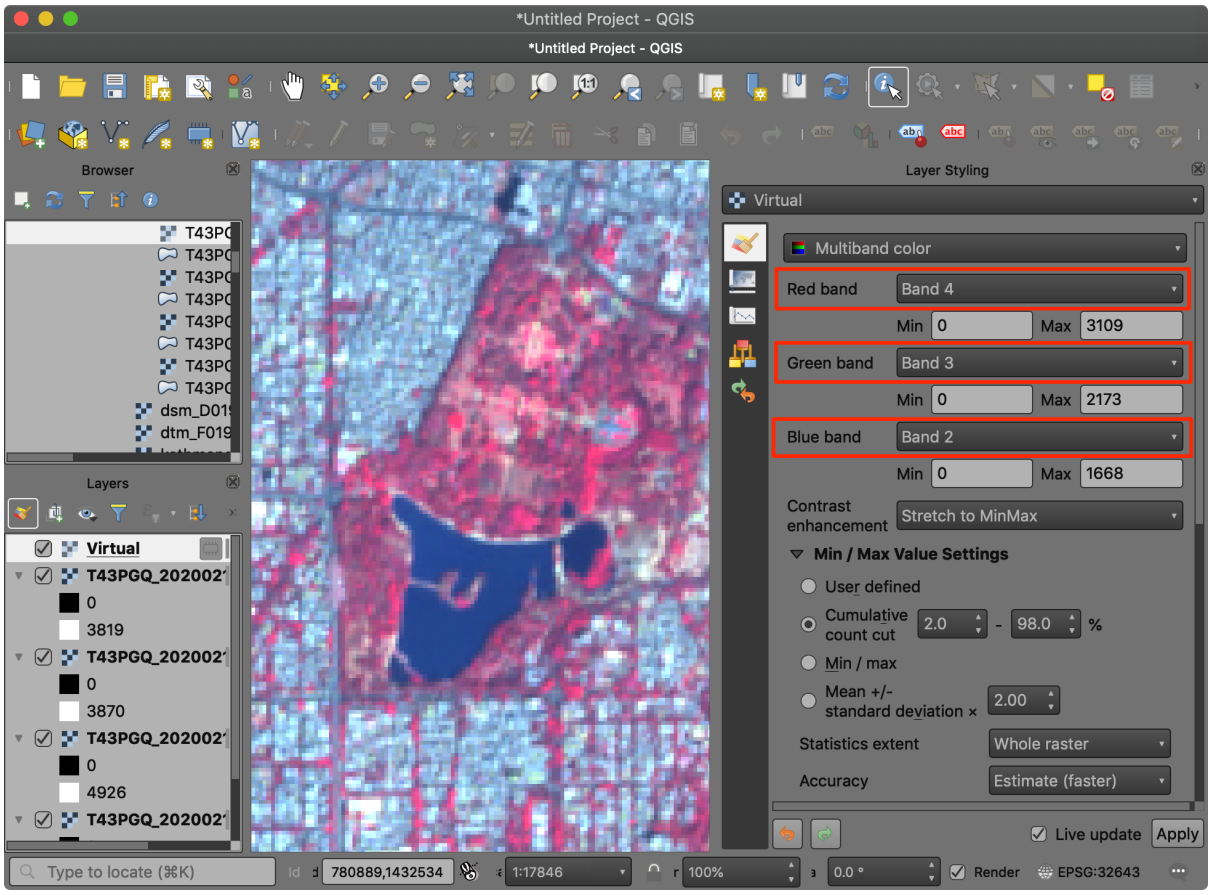

A new layer `Virtual` will be added to the *Layers* panel. This layer contains references to the 4 different images. Note that the order of the bands in alphabetical, so the mapping in the virtual raster is as follows:

| Virtual Raster Band | Image |
|---|---|
| Band 1 | B02 (Blue) |
| Band 2 | B03 (Green) |
| Band 3 | B04 (Red) |
| Band 4 | B08 (NIR) |

We will first visualize a *RGB Color Composite*. This is also referred as a *Natural Color Composite* since it is how the image would be perceived by the human eye. Place `Band 3`, `Band 2` and `Band 1` as *Red*, *Green* and *Blue* bands. Change the *Min/Max Value Settings* to `Cumulative count cut`. You will see the image now appears in natural colors.




Let's create a different type of visualization, known as *False Color Composite*. Here we want to create a *NRG Color Composite* where we put Near Infrared band in Red channel, Red band in green channel and Green band in blue channel. This visualization highlights the vegetation in red color and allows for easy visual interpretation of vegetation. Note how the water body and vegetation appeared almost the same color in the RGB composite, but are easy to separate in the NRG composite.




# Raster - Elevation Data

https://data.gov.uk/dataset/6a117171-5c59-4c7d-8e8b-8e7aefe8ee2e/lidar-composite-dtm-1m
# Analysis

## Exercise: Analyze Metro Rail Accesiblity

"operator" = 'BMRCL' AND "disused" != 'yes' AND "railway" = 'station' AND "disused:railway" != 'station'

## Exercise: Uber Movement Analysis

Uber provides anonymized data aggregated from over ten billion taxi trips to help urban planning around the world. The data is freely available in open-format for download. The dataset includes info such as travel times and speeds along each road segment.

We will take the Travel Times data for the city of Bangalore to analyze the traffic patterns and find out the areas that are accessible within 30 minutes of driving. This type of map is known as an Isochrone Map and is useful in urban planning.

The data comes in 2 parts. A GeoJSON file `bangalore_wards.json` depicting the boundaries of wards in the city and a CSV file `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate.c` that contains travel times aggregated at the ward level and hour of the day for the time period Q3 (July-Sep) of 2019. A whitepaper documents the methodology used for creating this dataset.

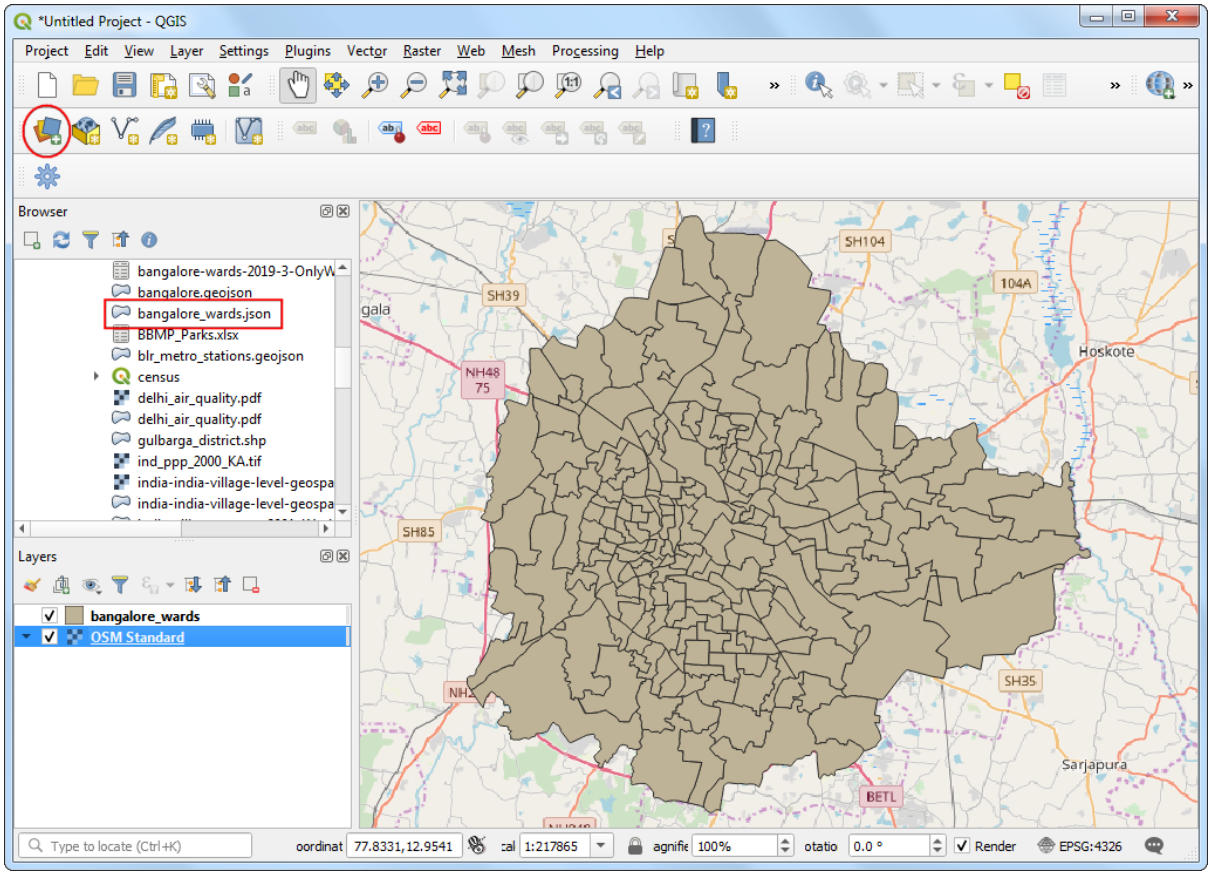
The structure of the CSV file is as follows

| sourceid | dstid | hod | mean_travel_time | ... |
|---|---|---|---|---|
| 55 | 111 | 8 | 2026.76 | ... |
| 22 | 25 | 13 | 770.07 | ... |

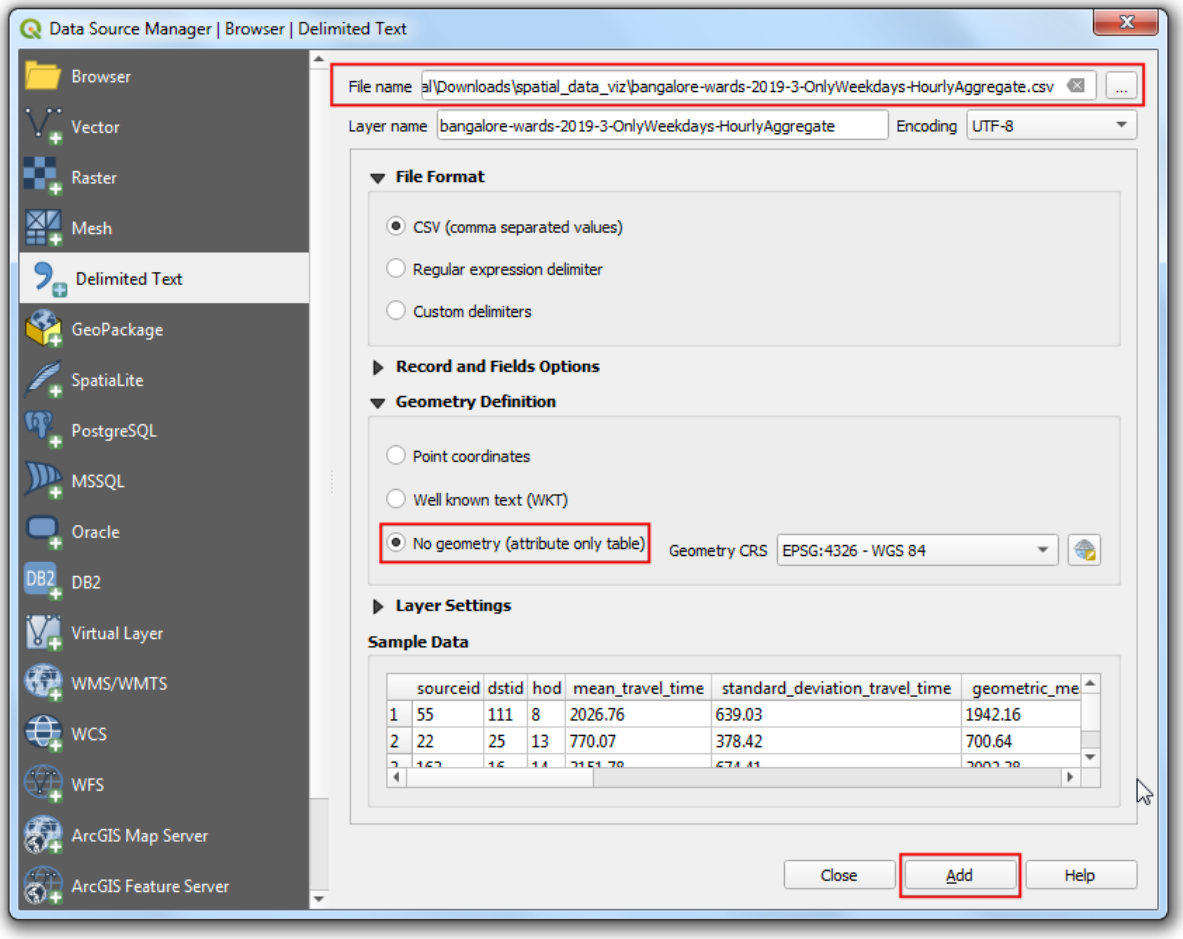
- **sourceid** and **dstid**: the pair of origin-destination ward ids
- **hod**: hour-of-day. Ranges from 0-23
- **mean_travel_time**: average travel time in seconds aggregated from all the trips

There are 198 wards in Bangalore. The file contains pair-wise travel time between each of them for all hours of the day. The total records of travel times between them would be 198 x 198 x 24 = 940896. But since not all wards have enough trips between them at certain hours, they are not provided. Thus our file contains a total of 818263 rows. This is still a lot of data and we need to do some Analysis to extract the right set from it.

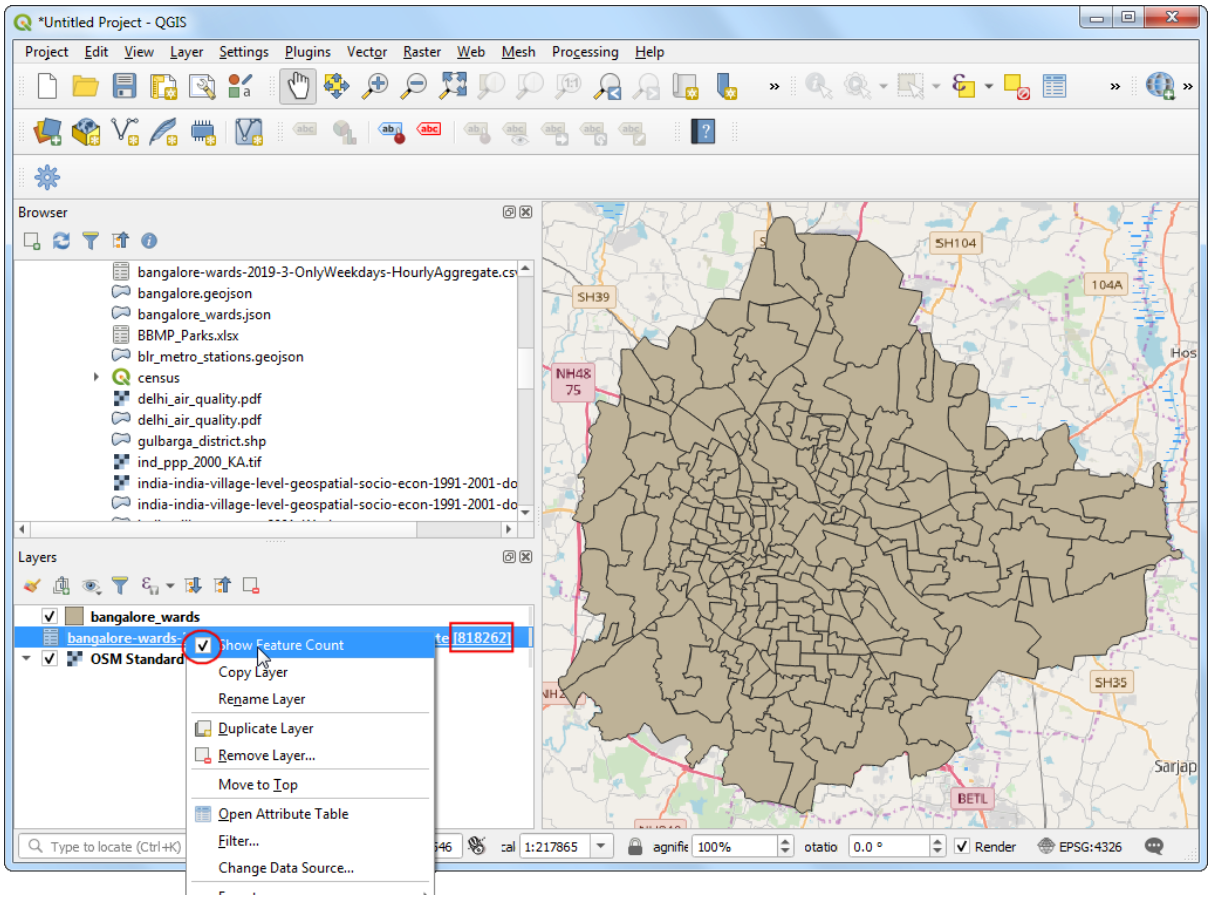
1. Let's first load a base layer from OpenStreetMap. Go to **Web → QuickMapServices → OSM → OSM Standard**. Next locate the `bangalore_wards.json` file in the *Browser* panel and drag it to the canvas. Once loaded, click *Open Data Source Manager* button.



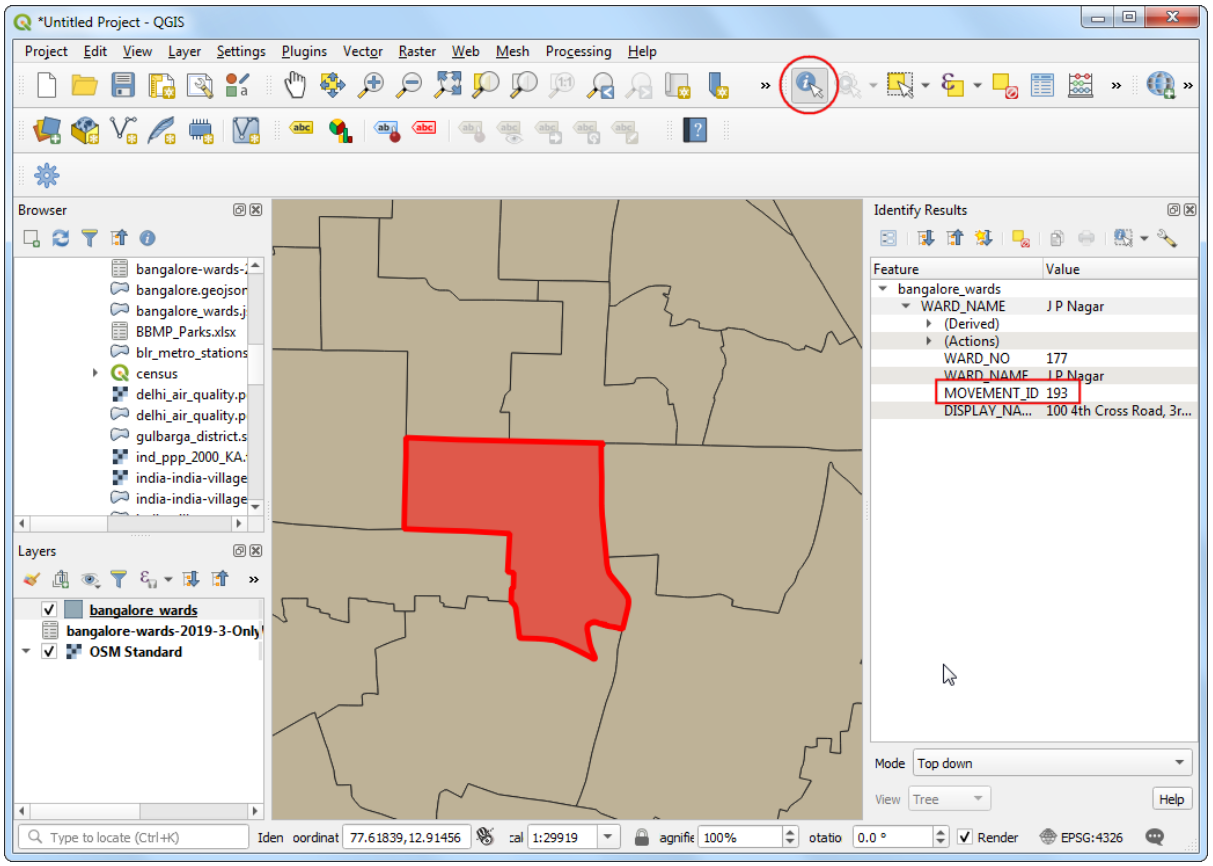

2. Switch to the *Delimited Text* tab. Browse to the `bangalore-wards-2019-3-OnlyWeekdays-Hourl` file and select it. Since this CSV file is just tabular data, select *No geometry (attribute only table)* option and click *Add*.



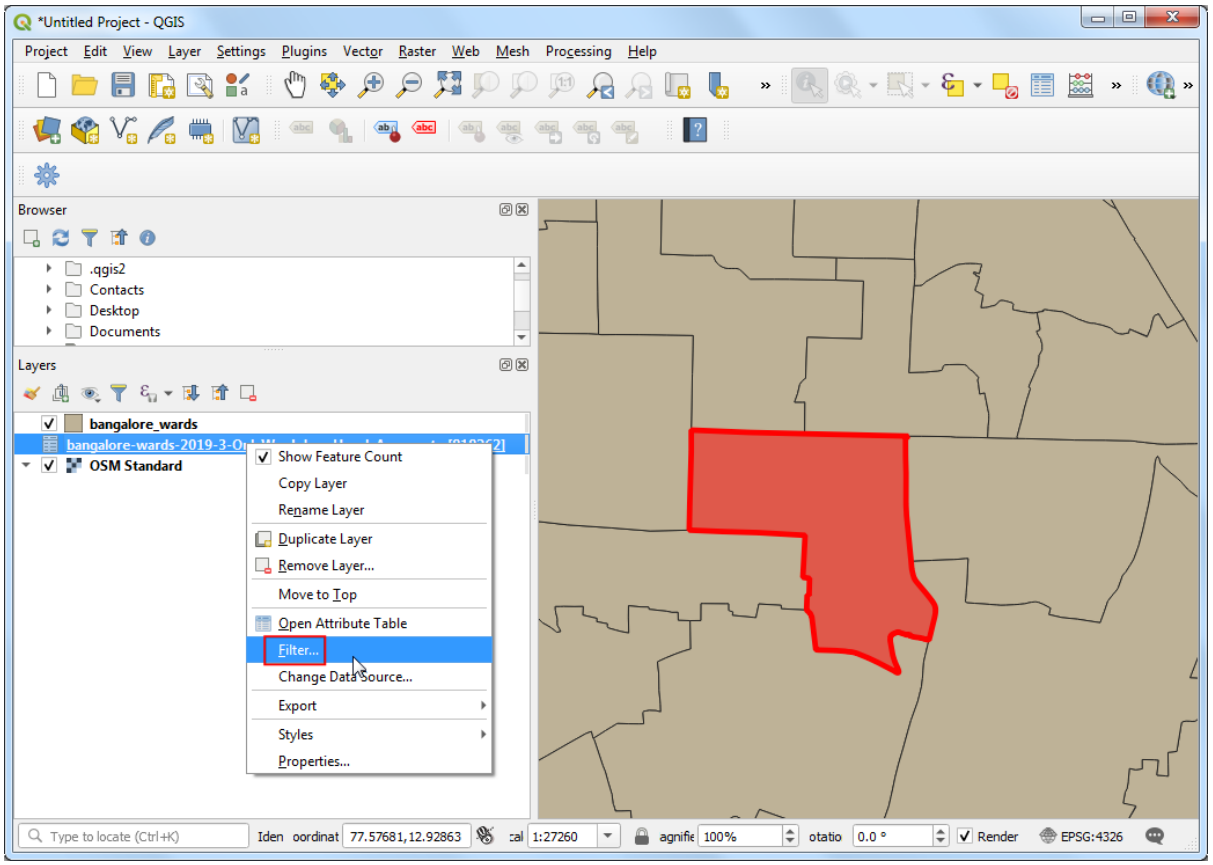

3. Once the `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate` layer is added to the *Layers* panel, right-click on it and select *Show Feature Count*. The total rows from the table will be displayed next to it.




4. For the purpose of this exercise, we will calculate all areas that are accessible within 30 minutes from Spatial Thoughts office. When you find the area on the basemap, you can select the *Identify* button and select `bangalore_wards` layer and click on it. The results will confirm that the office is located in the *JP Nagar* ward with the *MOVEMENT_ID* **193**.

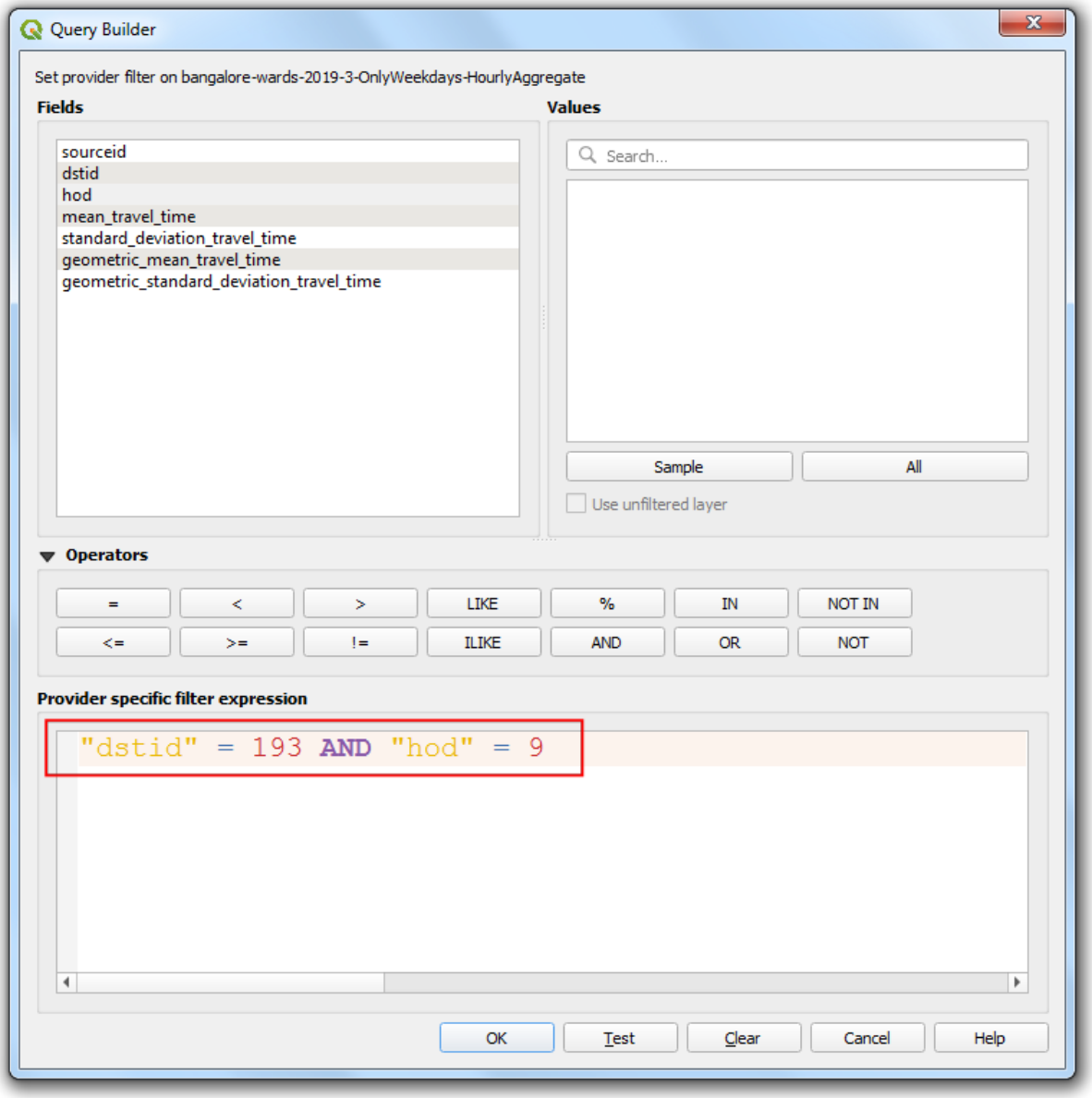



5. We can filter the travel time records to just those which have this ward as the destination. We can also restrict our analysis to the peak morning commute hours of 9am-10am. Right-click the `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate` layer and select *Filter*.

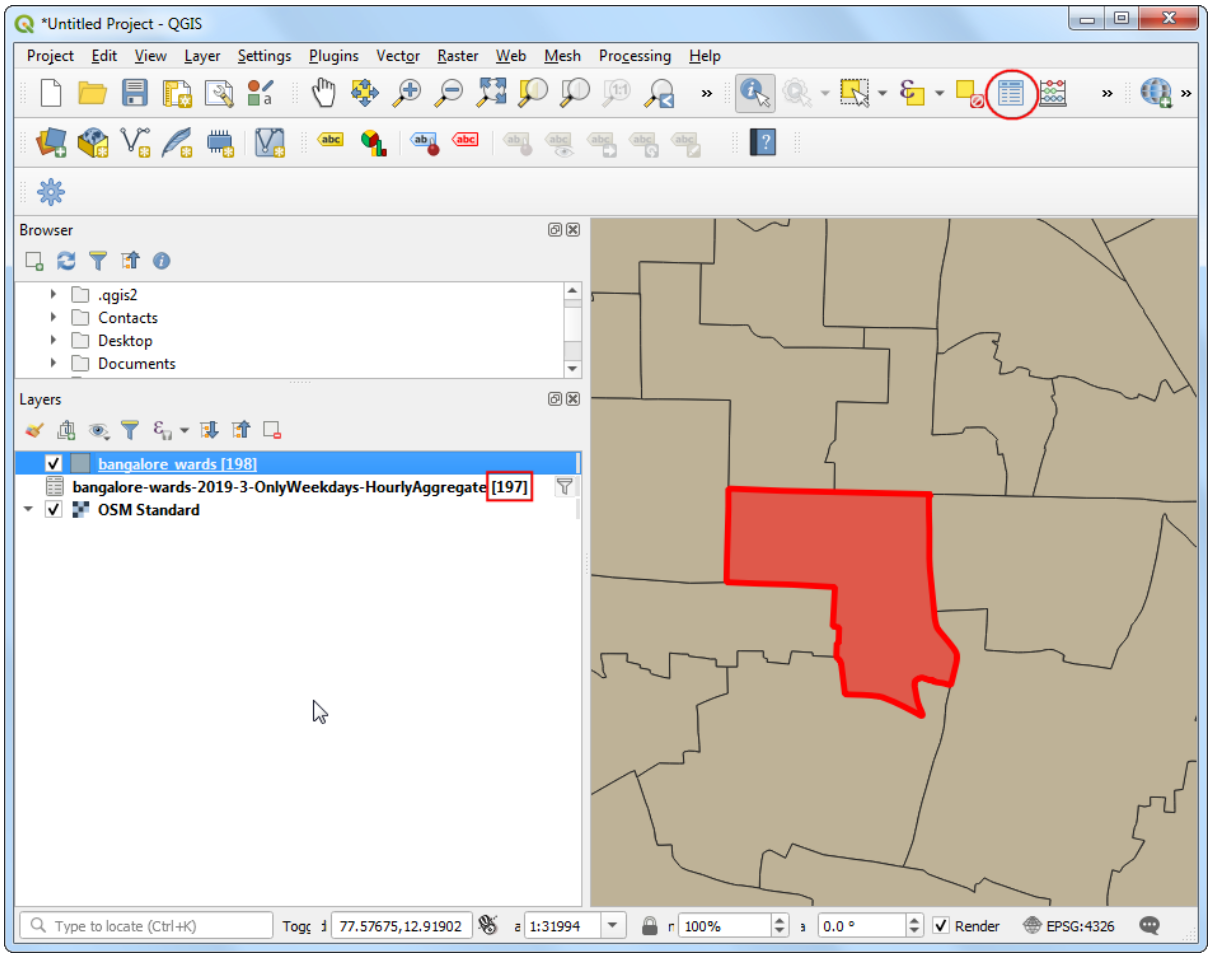


6. Enter the following filter expression and click *OK*.

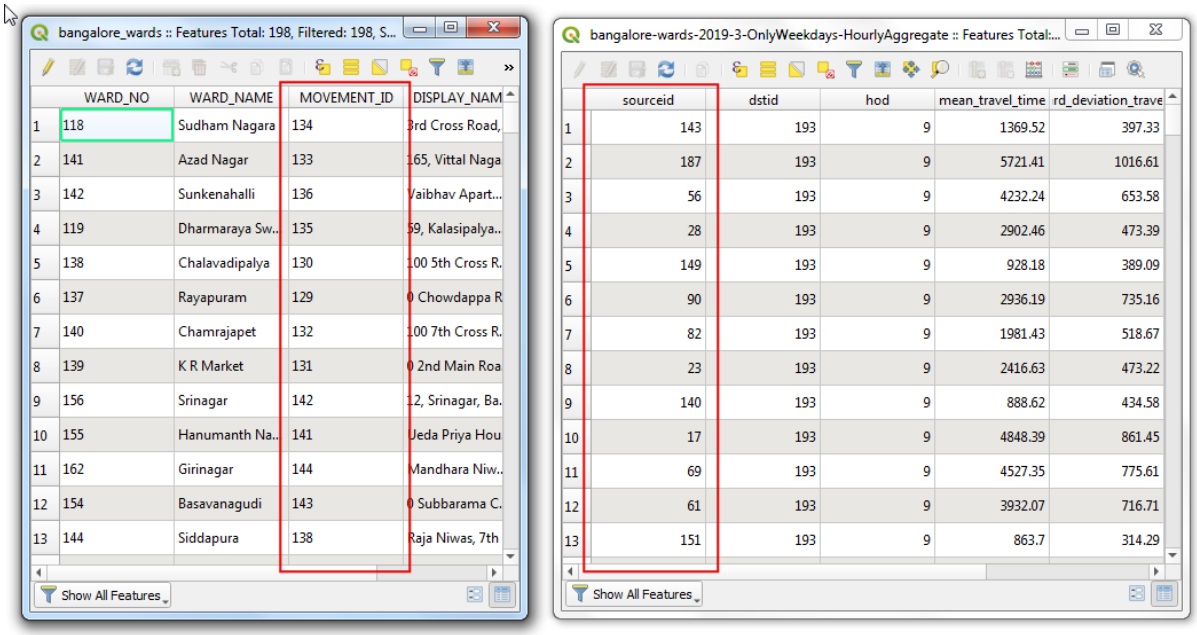
```
"dstid" = 193 AND "hod" = 9
```



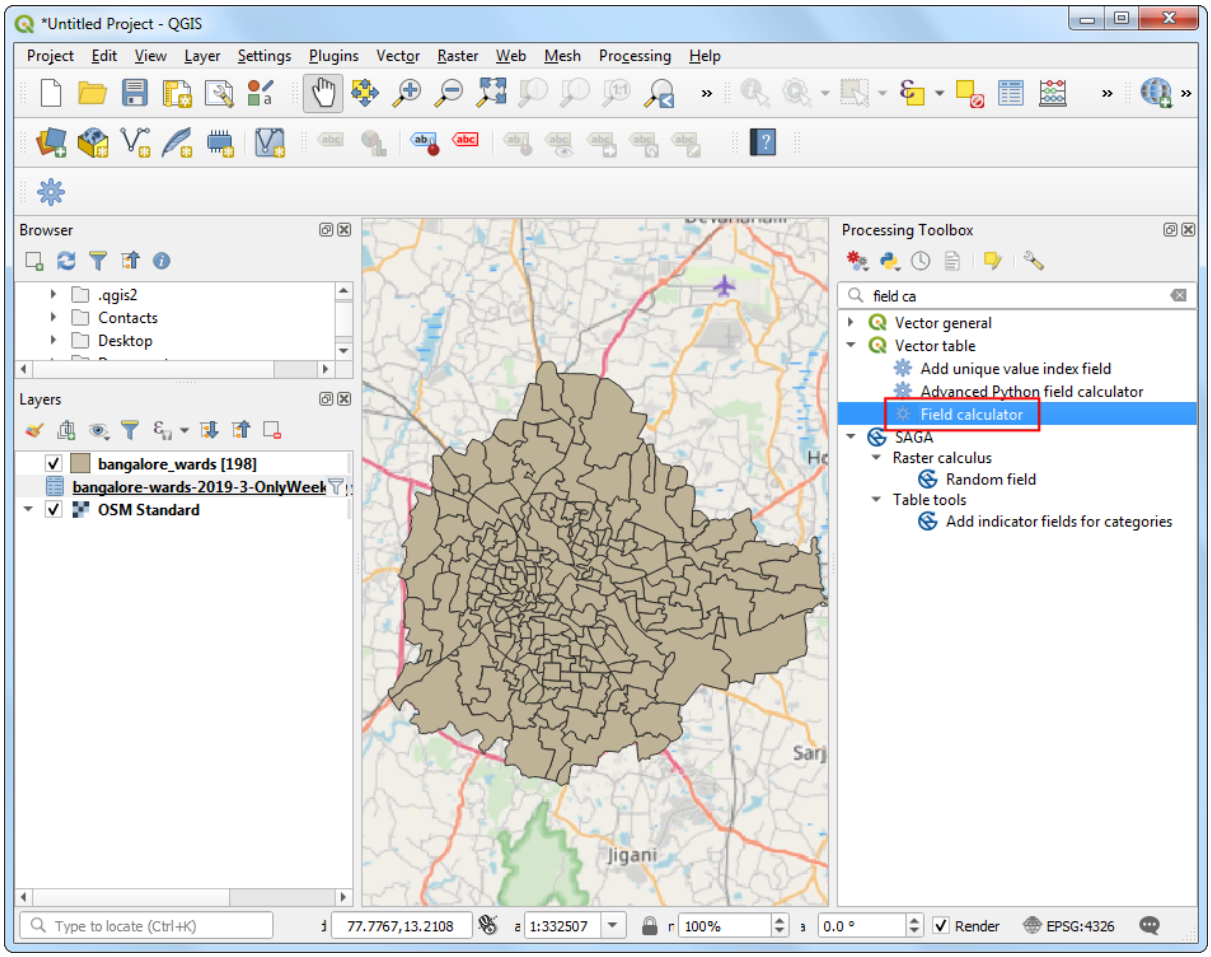

7. Back in the main QGIS window, you will see that the number of records in the filtered table are now down to just *197*. Since there are a total of 198 wards in the city, we have records of travel times between the 1 destination ward and 197 source wards. Open the attribute table of both the layers using the *Open Attribute Table* button in the *Attributes* toolbar.



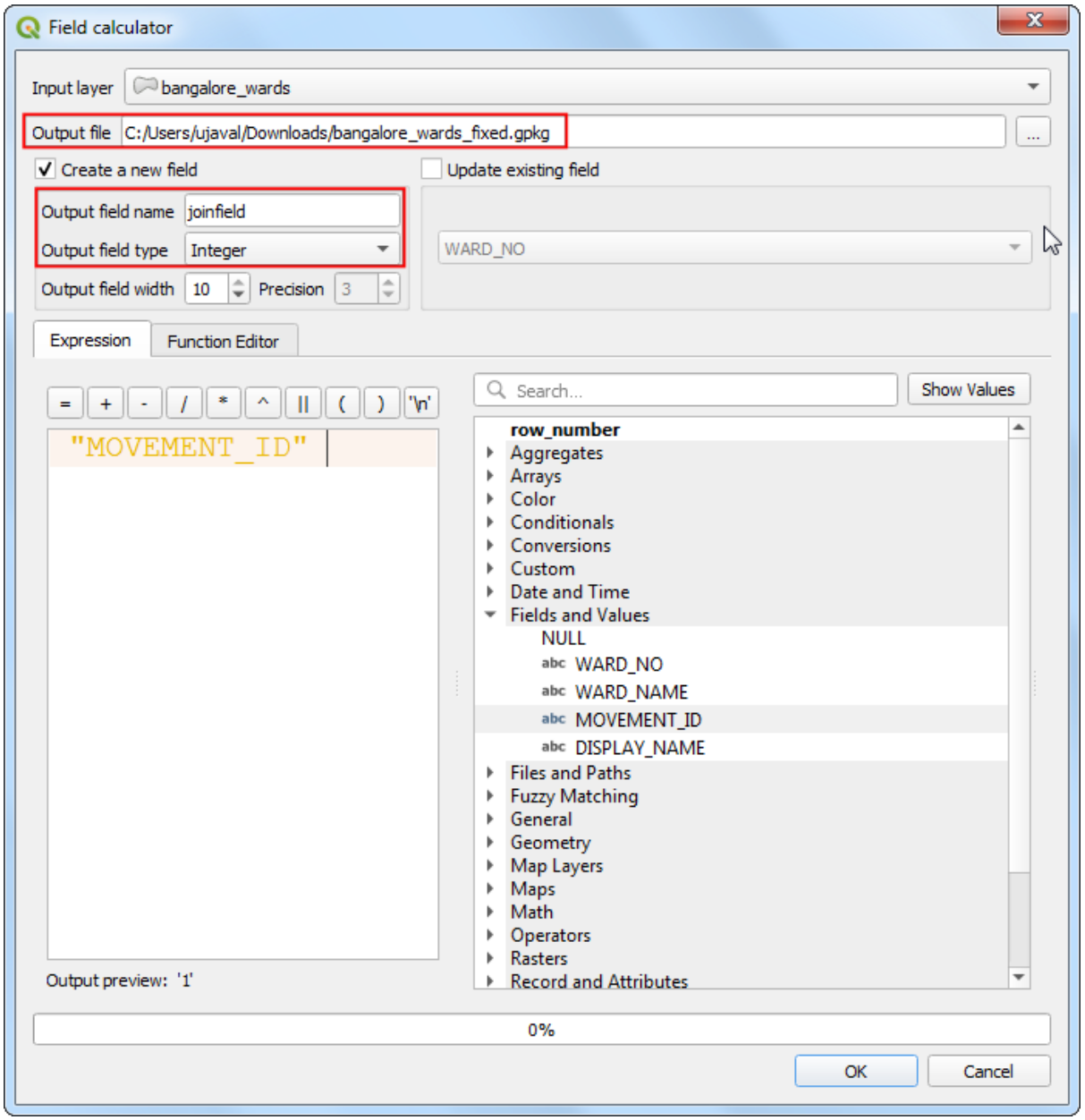

8. Now we have the shapes of the ward in the layer `bangalore_wards` and tabular information in the layer `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate`. We can join the attribute information to the shapes using a common attribute. Here the **MOVEMENT_ID** column from the `bangalore_wards` layer and **sourceid** column from the `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate` are unique ward identifiers that can be joined. This operation is called a *Table Join*.



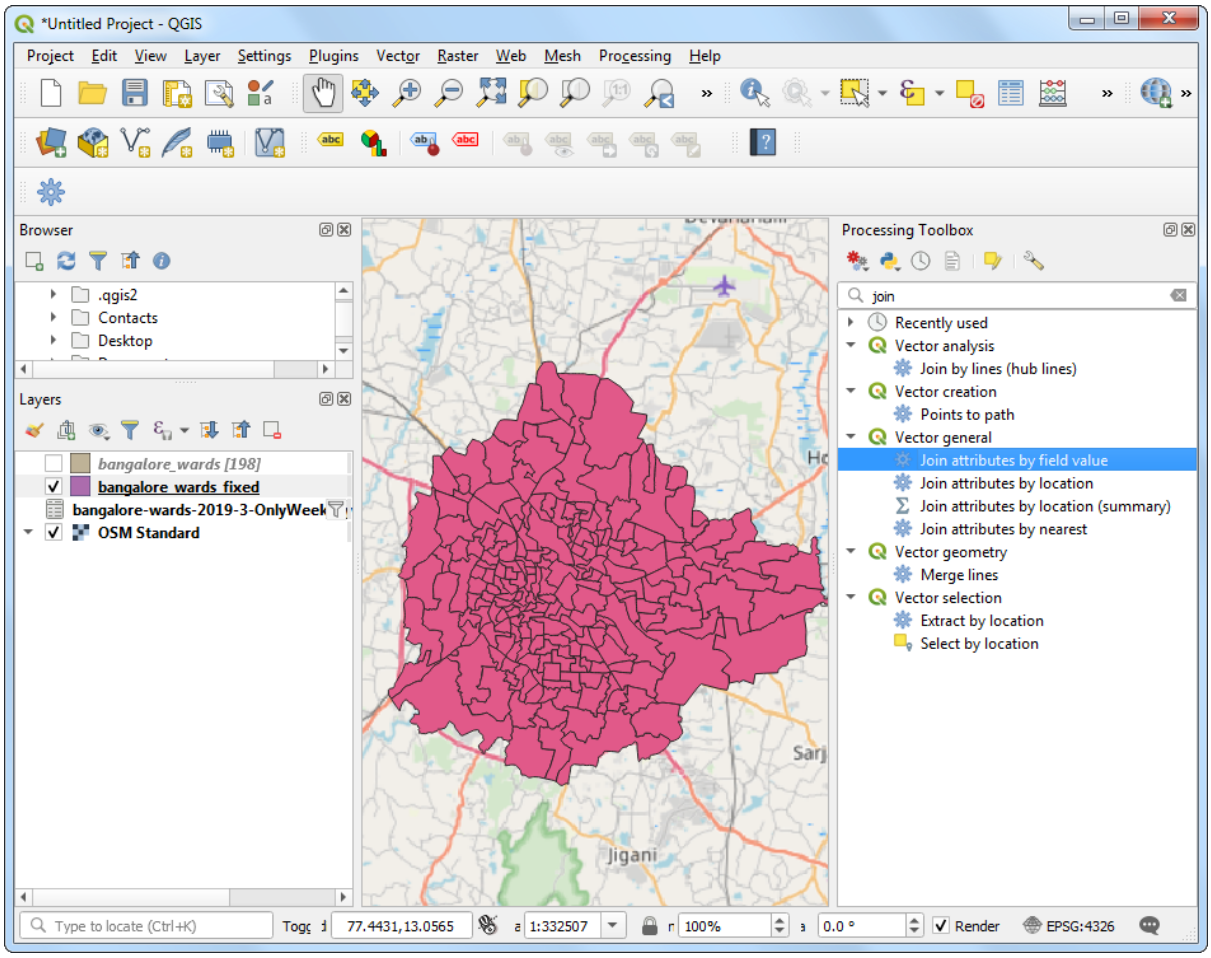
9. Before we can join these two layers, we must ensure that the values in both columns match exactly. Thought they appear the same, they are of different type. Since GeoJSON format has no way of specifying property types, all values are assumed to be of the type *String* - i.e. Text. But when we import a CSV to QGIS, QGIS intelligently determines the types of the columns based on the values and hence has assigned the type *Integer* to the column **sourceid**. So we need to convert the column from the GeoJSON to an integer type as well. Go to **Processing → Toolbox**. Search for the *Field Calculator* algorithm. Double-click to launch it.



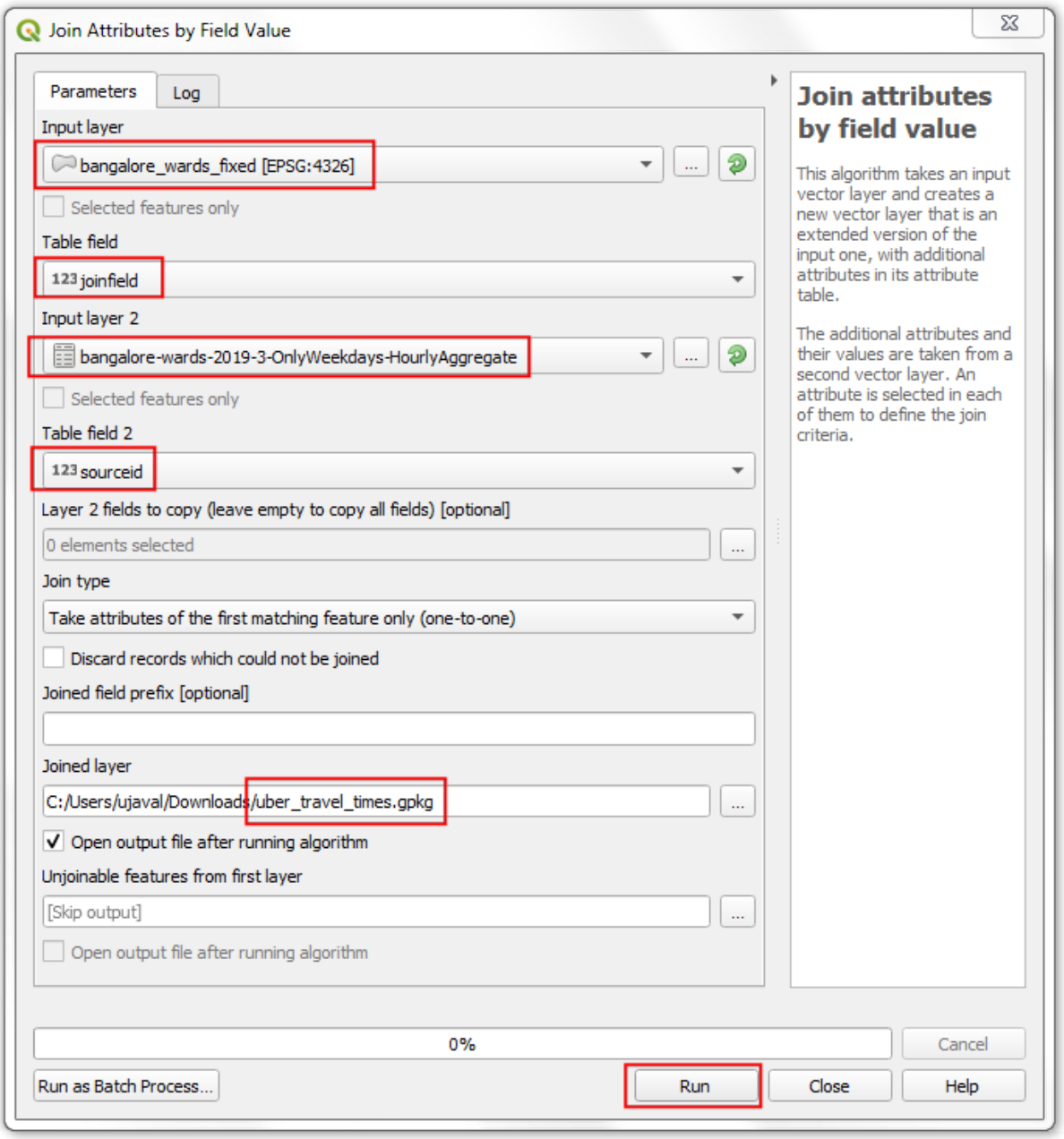

10. Choose `bangalore_wards` as the *Input Layer*. Click the … button next to the *Output file* and name the output as `bangalore_wards_fixed.gpkg`. Name the new field as `joinfield` and select the *Output field type* as `Integer`. Enter `"MOVEMENT_ID"` as the expression and click *OK*. Close the field calculator.



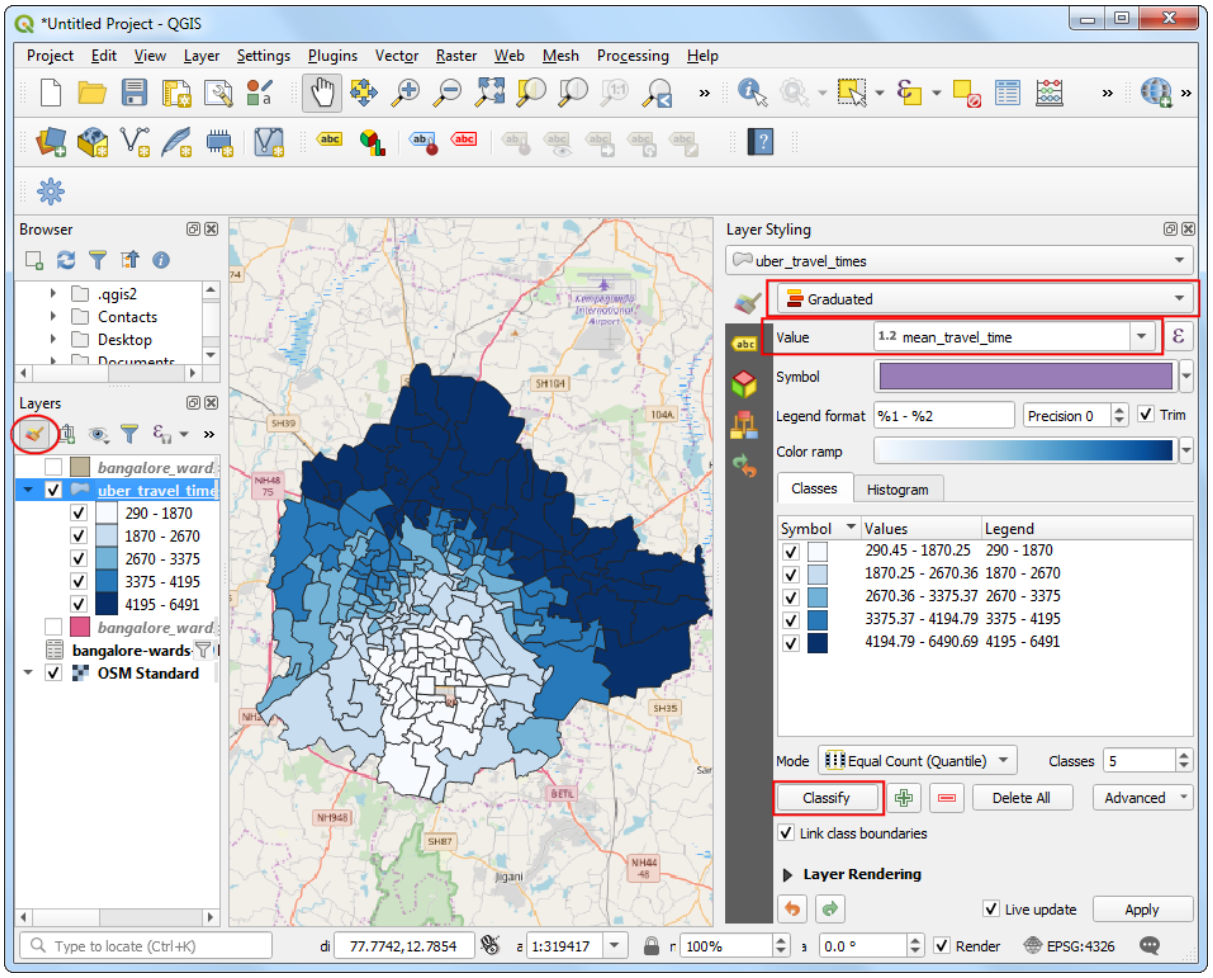

11. A new layer `bangalore_wards_fixed` will be added to the *Layers* panel. Now we are ready to perform the join. Search for and locate the *Join attributes by field value* algorithm from the Processing Toolbox.



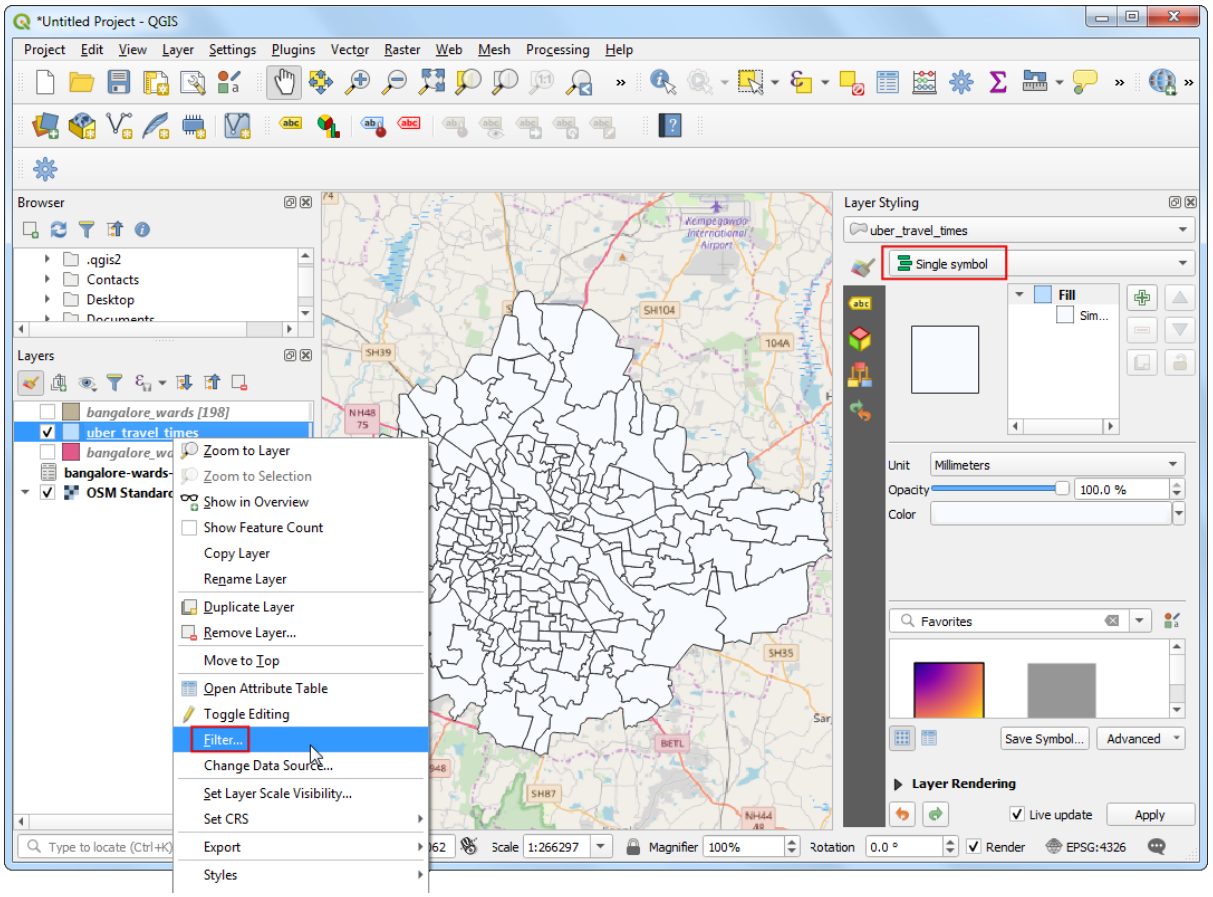

12. Select `bangalore_wards_fixed` as the *Input layer* and `joinfield` as the *Table field*. Select `bangalore-wards-2019-3-OnlyWeekdays-HourlyAggregate` as the *Input layer 2* and `sourceid` as the *Table field 2*. Name the *Joined layer* as `uber_travel_times.gpkg` and click *Run*.




13. A new layer `uber_travel_times` will be added to the *Layers* panel. Let's style it to visualize the result of the join. Click *Open the Layer Styling Panel*. Select the `Graduated` renderer and `mean_travel_time` as the *Value*. Select a color ramp and click *Classify*. You will see the map showing increasing travel times further you go from the destination.

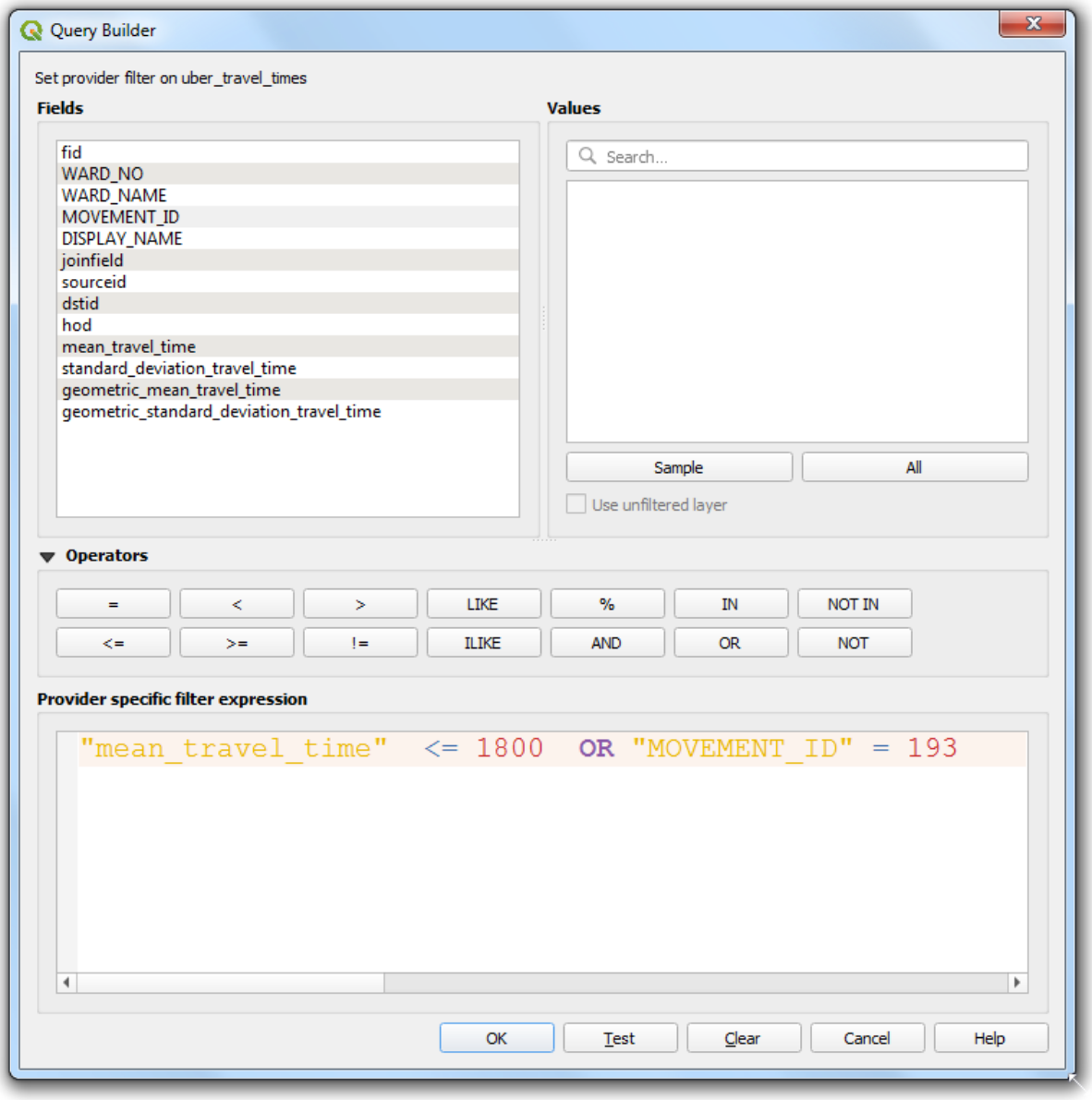



14. But we are looking to analyze and extract areas that are within 30 minutes of travel time, so we need to do some more processing. Switch the styling back to the *Single symbol* renderer. Right-click the `uber_travel_times` layer and select *Filter*.

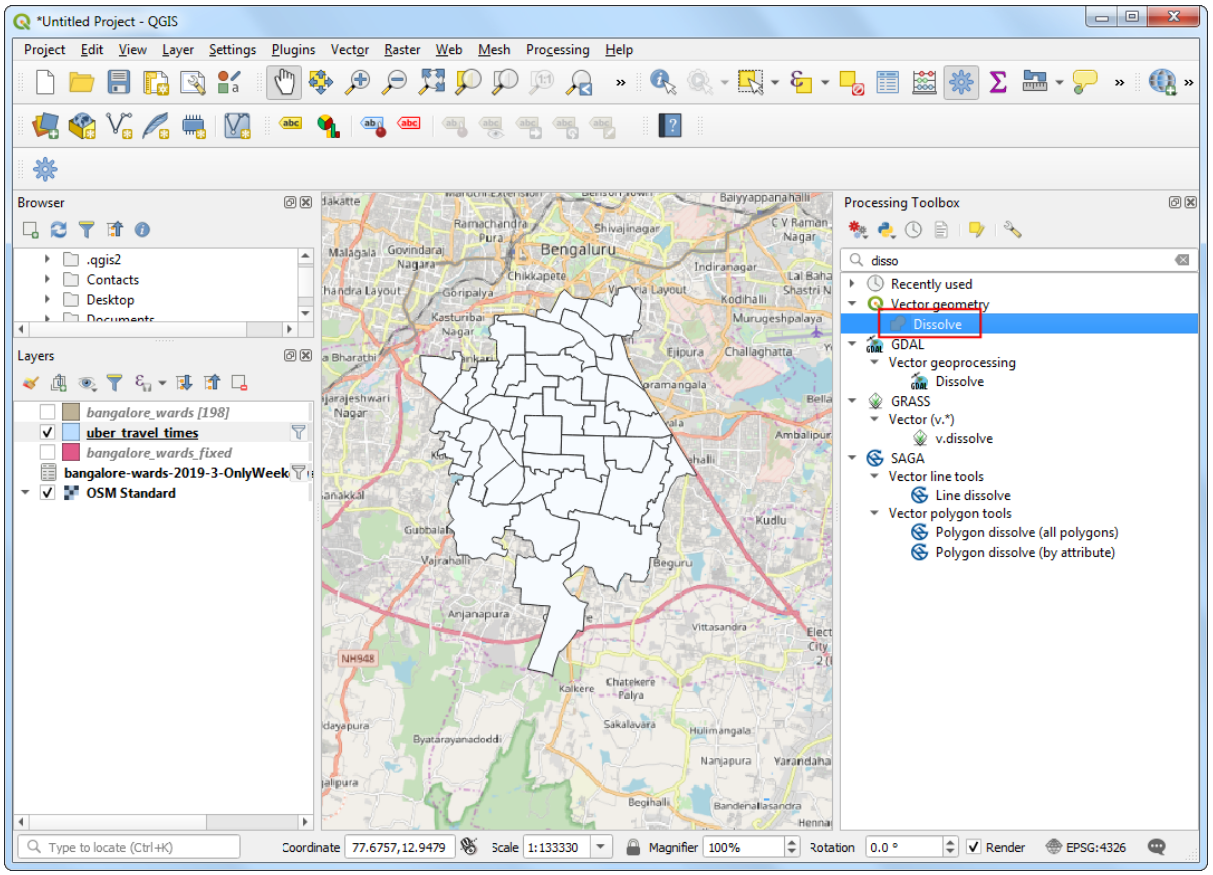



15. Enter the following expression to select all wards within 1800 seconds (30 minutes) of mean travel time. We also need to include our destination ward which will have 0 travel time.

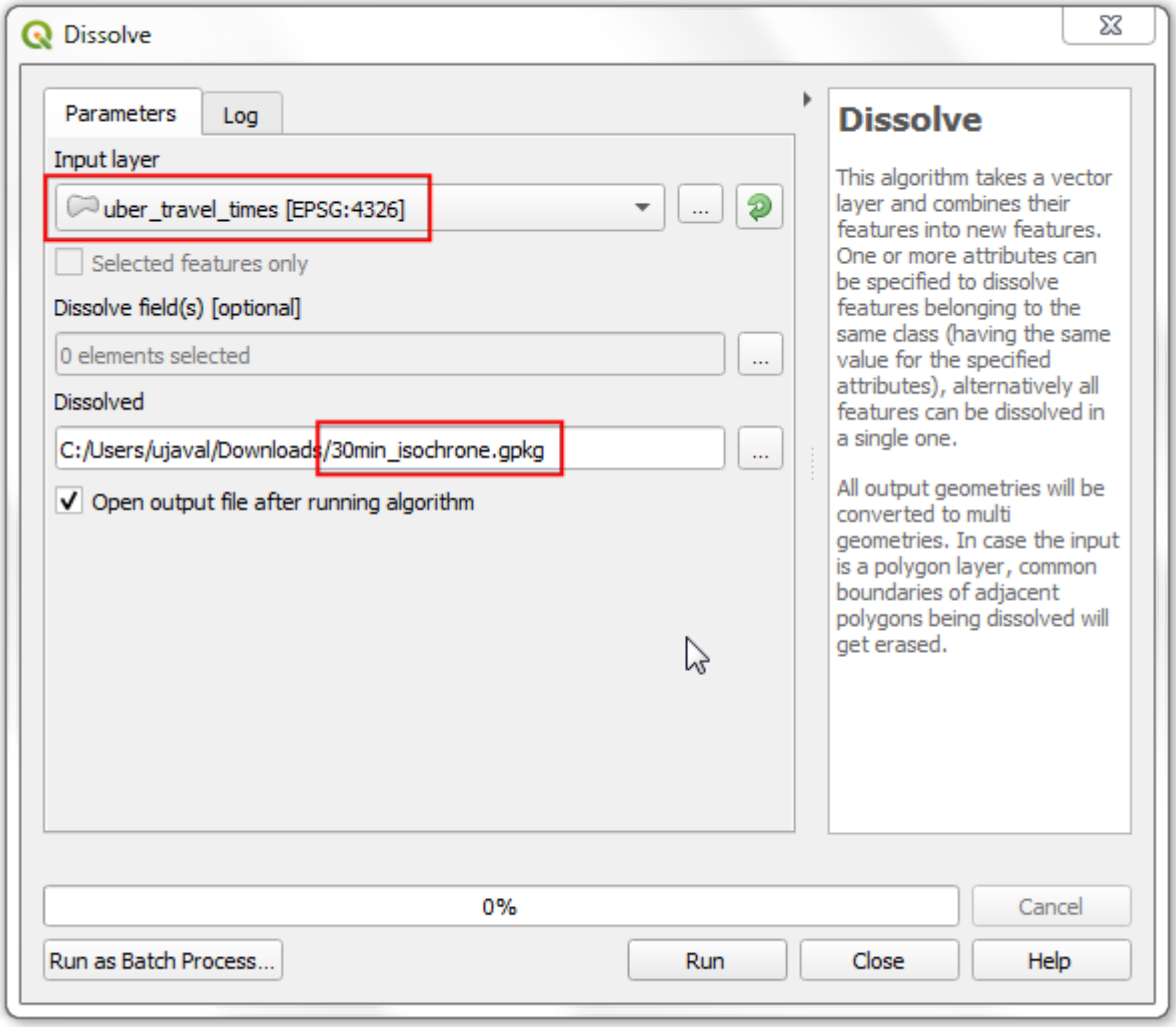
```
"mean_travel_time" <= 1800 OR "MOVEMENT_ID" = 193
```



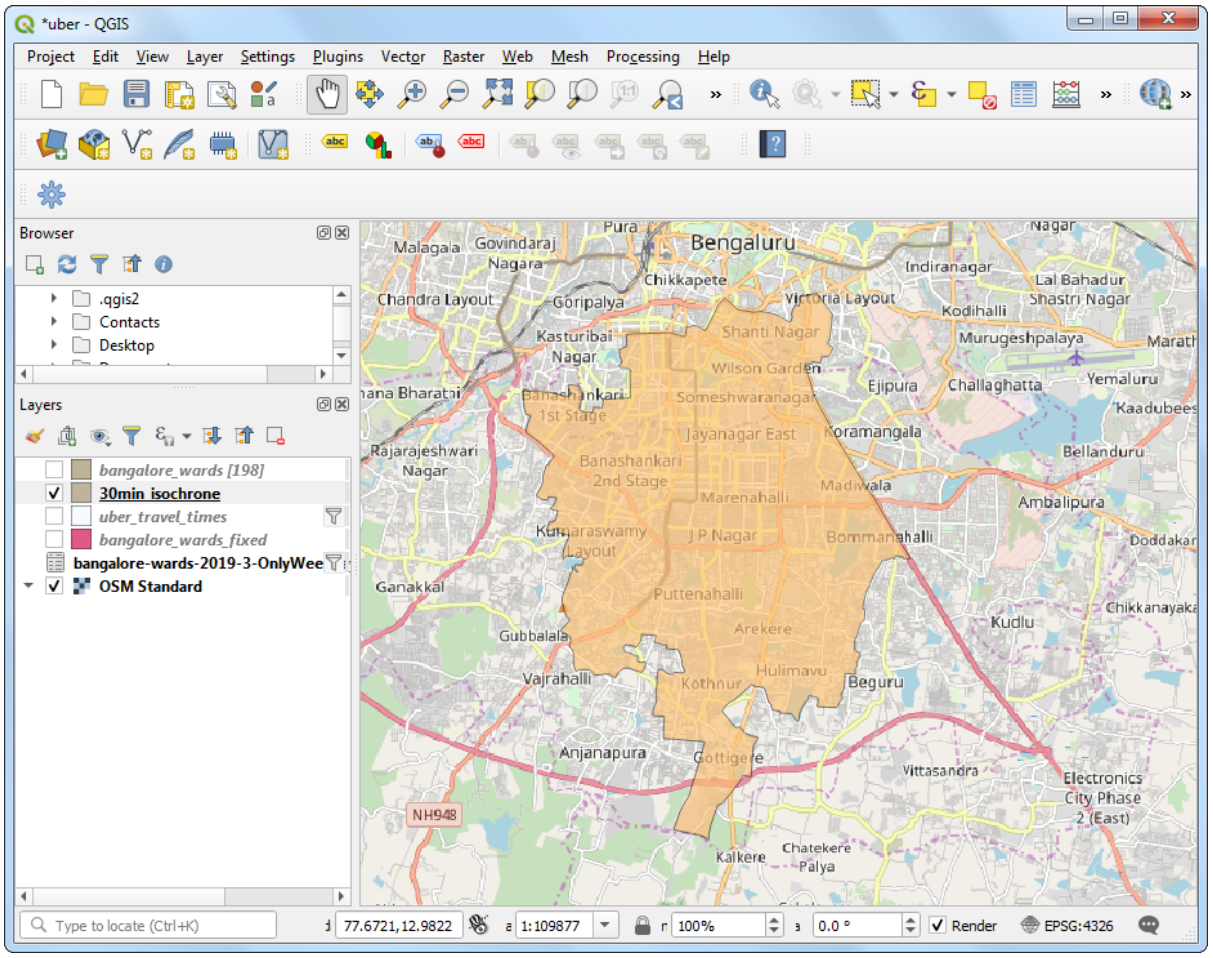

16. The layer will now show the cluster of polygons that constitute the area of interest. We will merge all of them to a single polygon now. Search and locate the *Dissolve* algorithm from the Processing Toolbox.




17. Select `uber_travel_times` as the *Input layer*. Name the *Dissolved* layer as `30min_isochrone.gpkg`. Click *Run*.




18. A new layer `30min_isochrone` will be added to the *Layers* panel showing the result of our analysis.




# Data Credits

- Delhi PM2.5 concentrations. Downloaded from OpenAQ data download service. Source data from Central Pollution Control Board (CPCB) India and EPAAirNow DOC.
- OSM Tile Layer: (c) OpenStreetMap contributors
- India Village-Level Geospatial Socio-Economic Data Set. Meiyappan, P., P. S. Roy, A. Soliman, T. Li, P. Mondal, S. Wang, and A. K. Jain. 2018. India Village-Level Geospatial Socio-Economic Data Set: 1991, 2001. Palisades, NY: NASA Socioeconomic Data and Applications Center (SEDAC). https://doi.org/10.7927/H4CN71ZJ. Accessed 15 February 2020.
- Census 2001 Districts: Downloaded from Datameet Spatial Data repository.
- Oxford Point Cloud. DTM, and DSM. Downloaded from Defra Data Services Platform. Crown Copyright 2019
- Kathmandu University Ground Drone Imagery. Downloaded from OpenAerialMap. Captured by WeRobotics
- Bangalore Sentinel-2 Imagery. Downloaded from Copernicus Open Access Hub. Copyright European Space Agency - ESA.
- Travel Times for Bangalore. Data retrieved from Uber Movement, (c) 2020 Uber Technologies, Inc., https://movement.uber.com.

# License