Introduction

Density Maps are a great way to visualize data quickly. In this exercise we will go over how to create three different kinds of density maps; HexBin, Grid, and Dot Grid. All three of these map styles combine similar points into a bin. We will create all three maps in one workspace using bookmarks to toggle each workflow on and off.

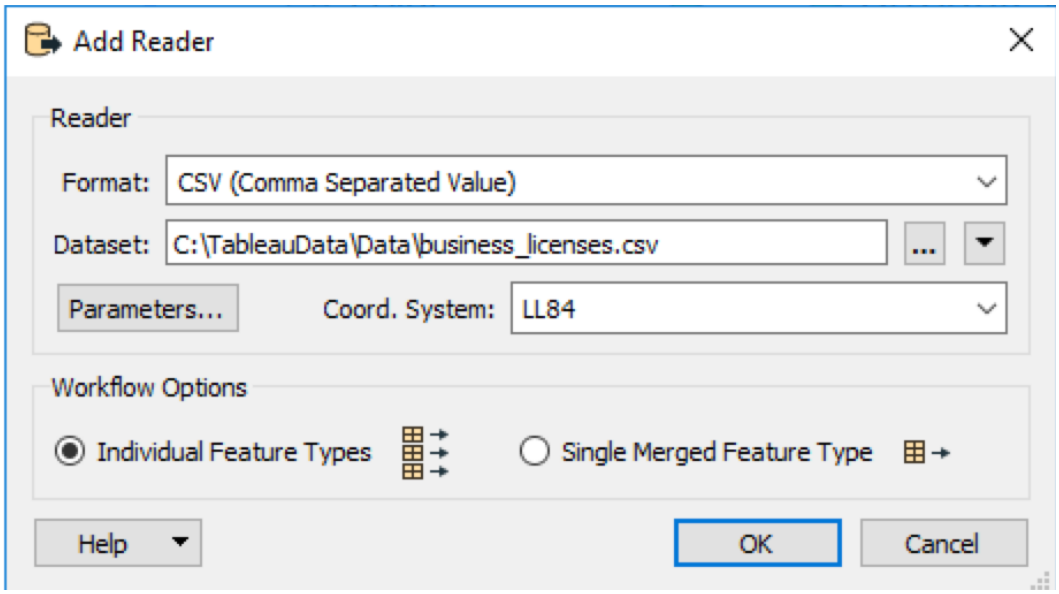
Step-by-Step Instructions

Prepare Data

1. Open a blank workspace and add a CSV reader

The first step is to read the CSV file. Drag the business license CSV file from the file explorer onto the blank canvas. Notice, that FME has already filled in the reader format and dataset. Alternatively, click the Add Reader on the toolbar.

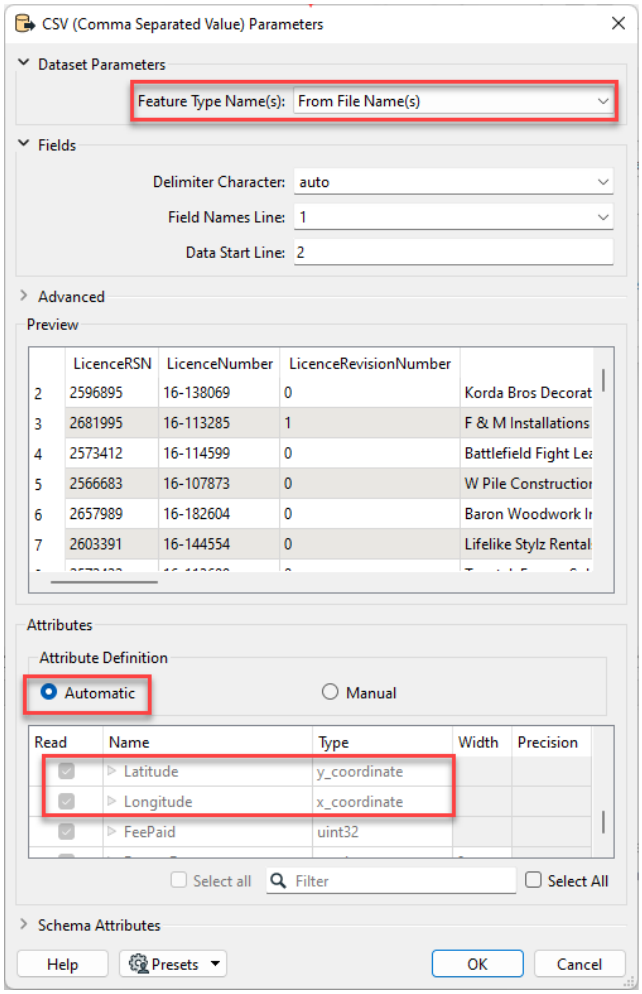
Reader Format:	CSV (Comma Separated Value)
Reader Dataset:	...\business_licenses.csv
Coord. System:	LL84



Add a CSV Reader, add the business_licenses.csv, set the coord system, then click on Parameters...

Click on the Parameters button. The Database Parameter allows us to choose different naming schemes for the layers or feature types that end up on the canvas. **Make sure it is set to "Feature Type from File Names".**

Confirm that the Attribute Definition has the Longitude and Latitude attributes set to x_coord and y_coord, respectively. By default, FME will map X/Y or Latitude/Longitude attributes to the x_coord and y_coord data types.



If necessary, change the Attribute Definition to Manual and update the Latitude and Longitude

Click Ok, and then click Ok again to add the Reader to the canvas. FME is able to create point geometry for each record with a latitude and longitude value.

2. Reproject to UTM83-10

Add a Reprojector transformer, and set the Destination Coordinate System: to UTM83-10. Connect it to the CSV Reader.

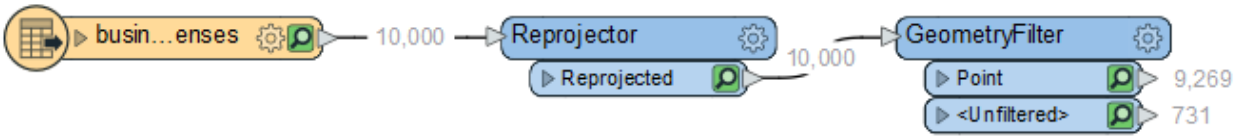
Why Reproject?

Since our source dataset created point geometry with latitude and longitude coordinates (using LL84) our units are degrees. Since we will be creating a variety of geometries for different analysis, it is best to use a coordinate system that utilizes a linear unit like meters or feet. You can always determine which units your coordinate system uses by checking the

coordinate system properties in any coordinate system dialog OR from the Tools > Browse Coordinate Systems tool.

3. Remove null geometry

Add the GeometryFilter to remove any points that do not have a latitude and longitude. Set the Output Types to Filter to Point. Connect this to the Reprojector.

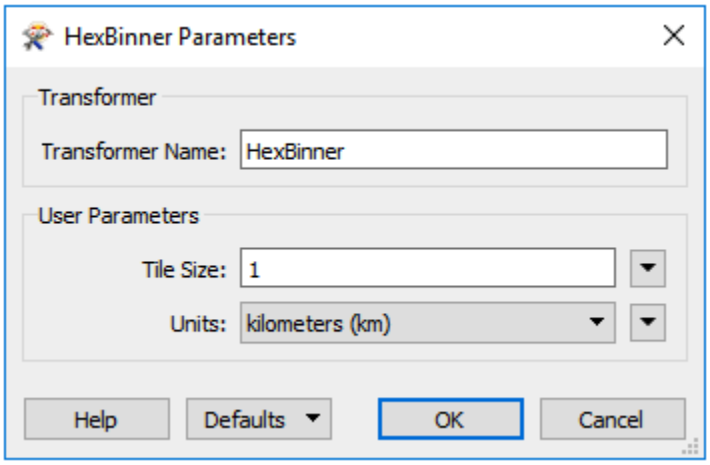


Part 1: Hex Bins

4. Add the HexBinner Transformer

The HexBinner transformer is a custom transformer available on the FME Hub. It creates hexagonal features that enclose point feature input. They aggregate data into a grid and can be used for further analysis.

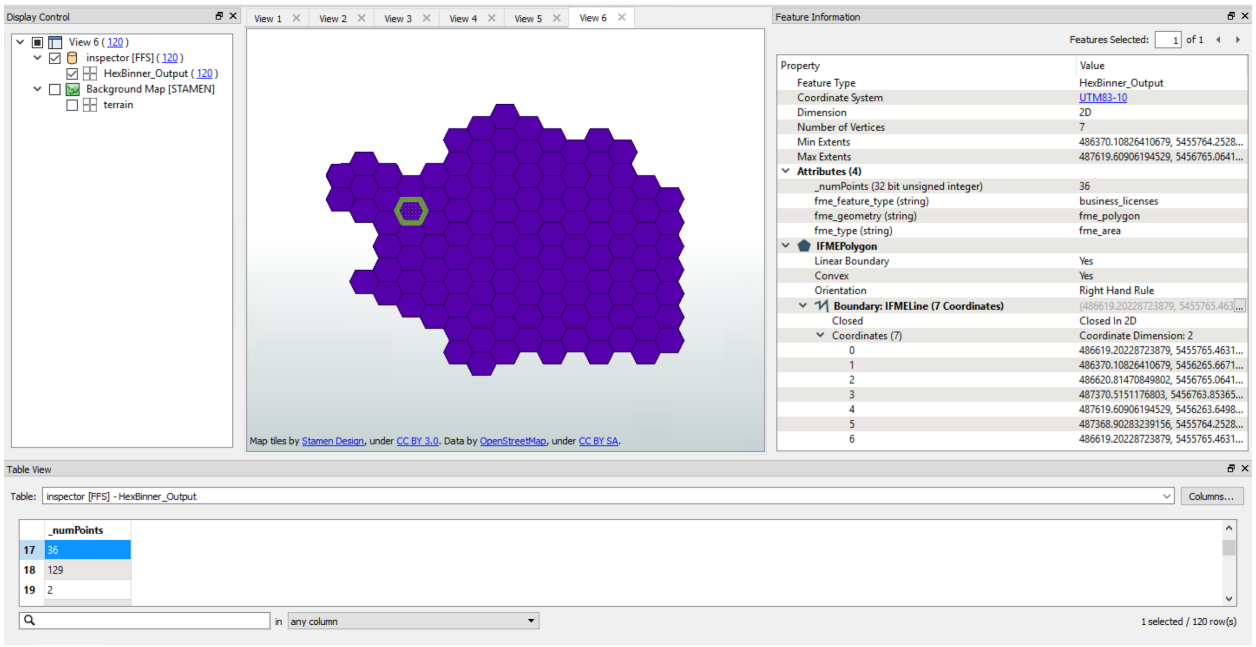
We are going to create a HexBin that is one kilometer in size. Add the HexBinner to the workspace, you will notice that it is green, while the other transformers we've seen before have been blue. When a transformer is green (embedded) or teal (linked to an .fmx file) it means it is a custom transformer. In the parameters for the HexBinner, change the Tile Size to 1, and ensure that the Units are set to Kilometers.Connect the Input port to the Point Output port on the GeometryFilter.



Set the Tile Size to 1 and ensure the units are set to Kilometers

5. Inspect the results

With feature caching enabled, run the translation up to the HexBinner by selecting the transformer and clicking the Run to This option. In the Visual Preview Window, you should see a grid of hexagons. If you click on a hexagon, you can see how many business licenses are in each hexbin in the _numPoints attribute that was created.



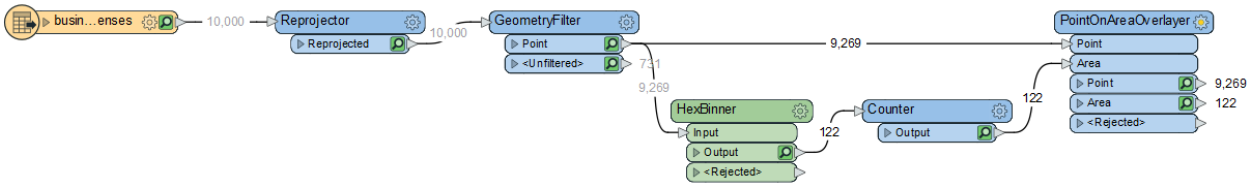
Inspect the HexBinner Transformer in Visual Preview. All the points appear grouped in hexagons

6. Determine which businesses are in each bin

We would like to determine which businesses are in each bin, not just a count. To do this we will add the Counter transformer. Open the Counter Parameters and set the Count Output Attribute, name it HexBinId, then accept the defaults for the rest of the parameters. Click ok. **Connect the Counter to the HexBinner Output port**.

Next, add the PointOnAreaOverlayer to add the HexBinId to each of the points. In the PointOnAreaOverlayer parameters, enable the Merge Attributes option under the Attribute Accumulation section. This will allow us to pass the HexBinID from the Area onto the points.

Connect the Counter Output port to the Area Input port on the PointOnAreaOverlayer. Then **Connect the GeometryFilter Point Output port to the PointOnAreaOverlayer Point Input port**.



7. Write to a Tableau .hyper Writer

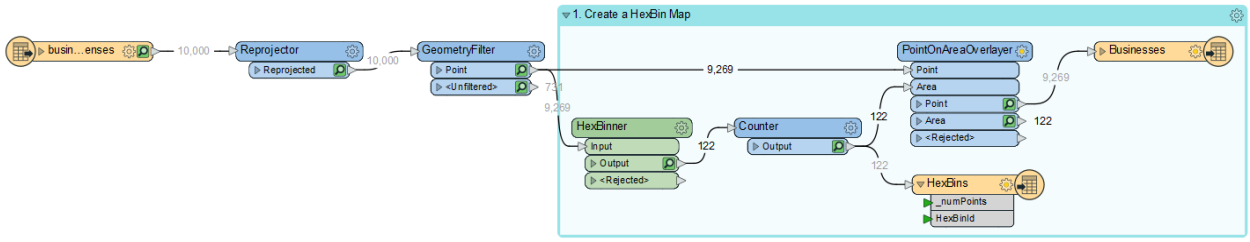
Add a Tableau Hyper writer and set it to your Outputs folder, also set the **Feature Type Definition to Automatic**...

Change the Feature Type properties, **name the table HexBins**, and the **Geometry to hyper_polygon**. Click ok.

Connect this writer to the Output port on the Counter Transformer.

Then **right click on the canvas click "Insert Writer Feature Type..."**. Add another writer for the point data. **Name this one Businesses** and change the **geometry to hyper_point**.

Connect the Businesses writer feature type to the PointOnAreaOverlayer Point port. Finally, to tidy up the workspace and allow us to enable/disable this translation, **add a bookmark from Hexbinner onwards**. To create a bookmark, select everything you would like to add to the bookmark and then right click on the workspace and click Insert Bookmark (or use the keyboard shortcut CTRL+B).



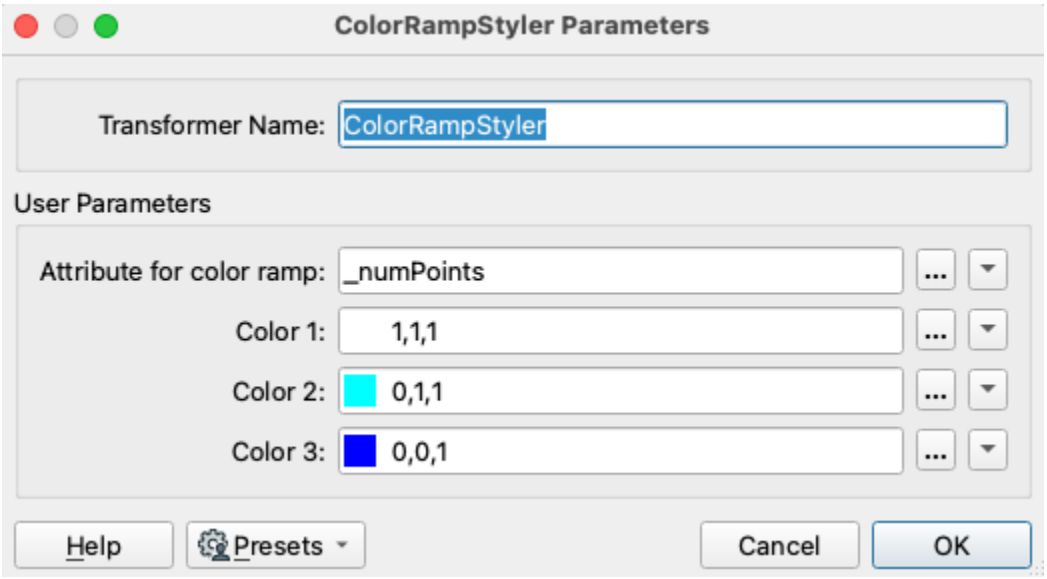
Final translation layout, ensure your writers are pointed to the correct output ports

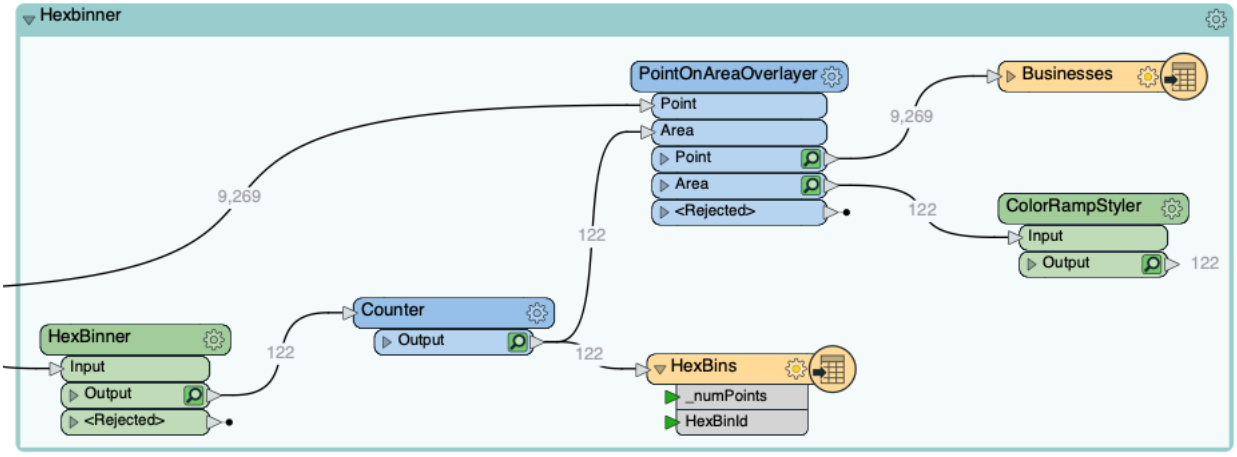
8. Run translation and view data in Visual Preview

While running the translation, if you encounter a Rejected Error, either connect a logger to each Rejection Output port, or in the Navigator > Workspace Parameters > Translation > Rejected Feature Handling, set it to Continue Translation.

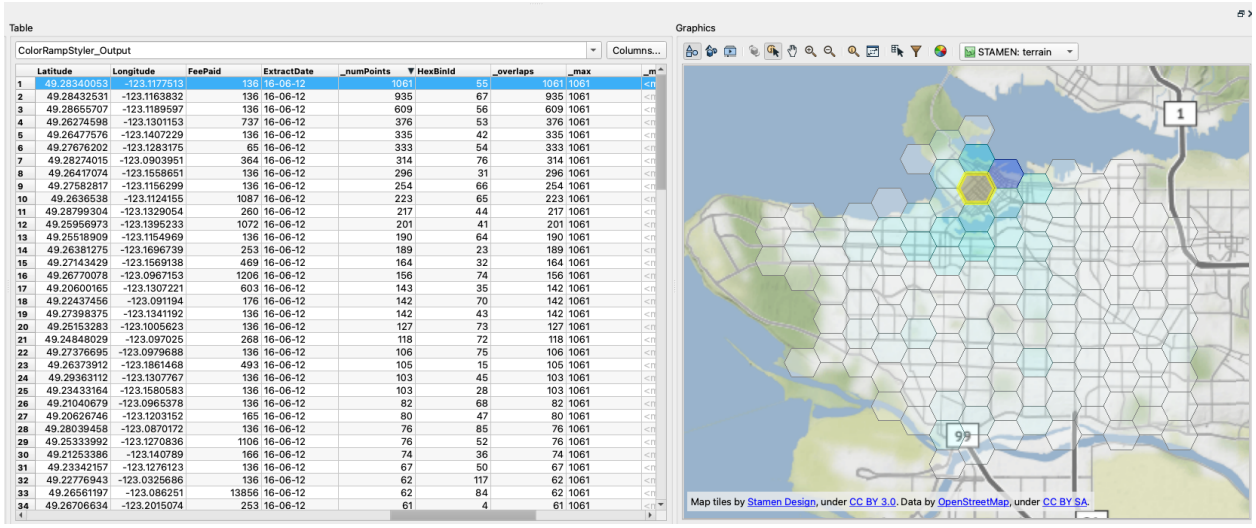
To emulate what this would look like in Tableau, add a ColorRampStyler to the canvas and connect it to the PointOnAreaOveralyers's Area port. In the ColorRampStyler parameters, set the **Attribute for color ramp to _numPoints** and use the following color settings:

- **Color 1: 1,1,1**
- **Color 2: 0,1,1**
- **Color 3: 0,0,1**

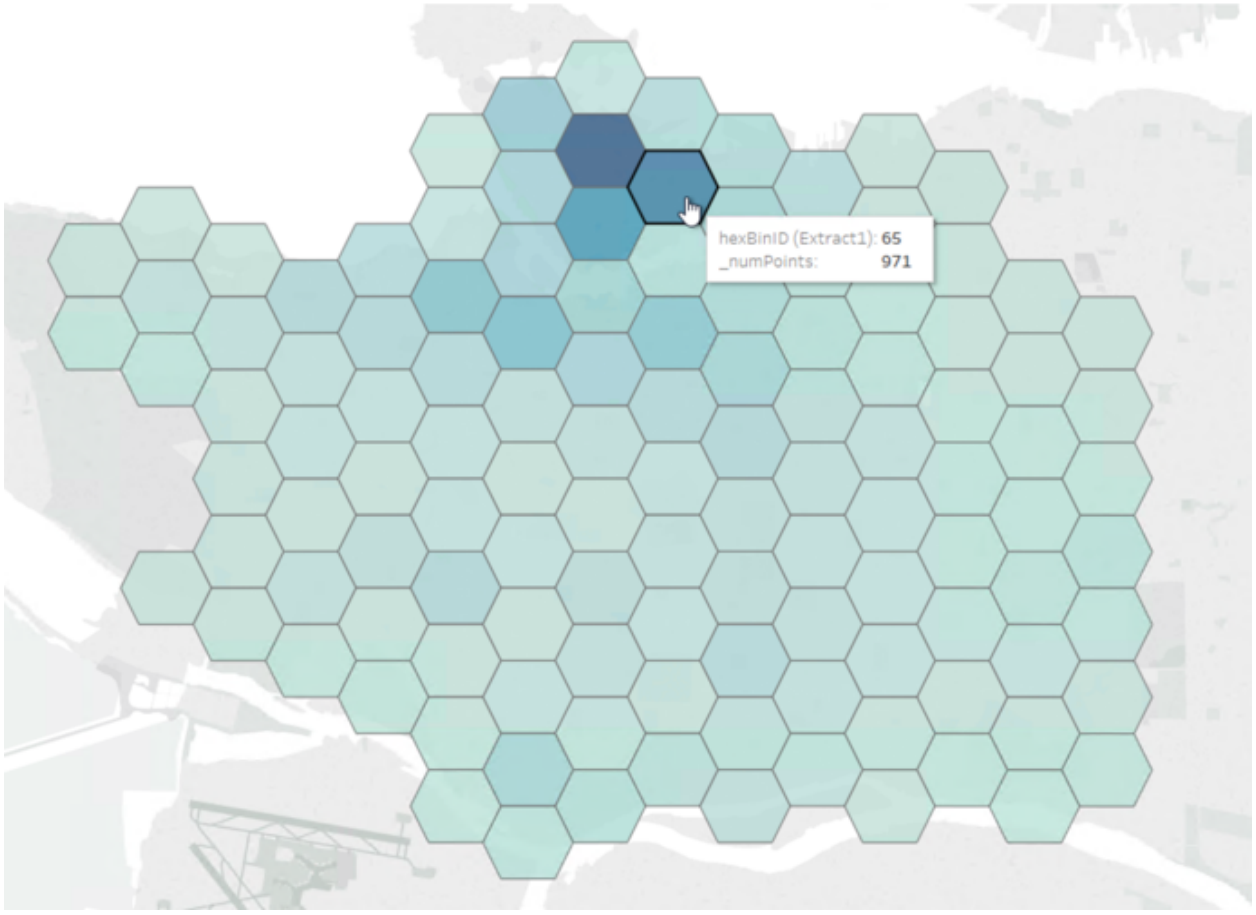




Once you have configured these parameters, run the workspace and inspect the feature cache on the ColorRampStyler:



In Tableau, the output of this workspace would look similar to the image below:



A hex bin map completed and styled in tableau.

Part 2: Grid Map

9. Disable the HexBinner Bookmark

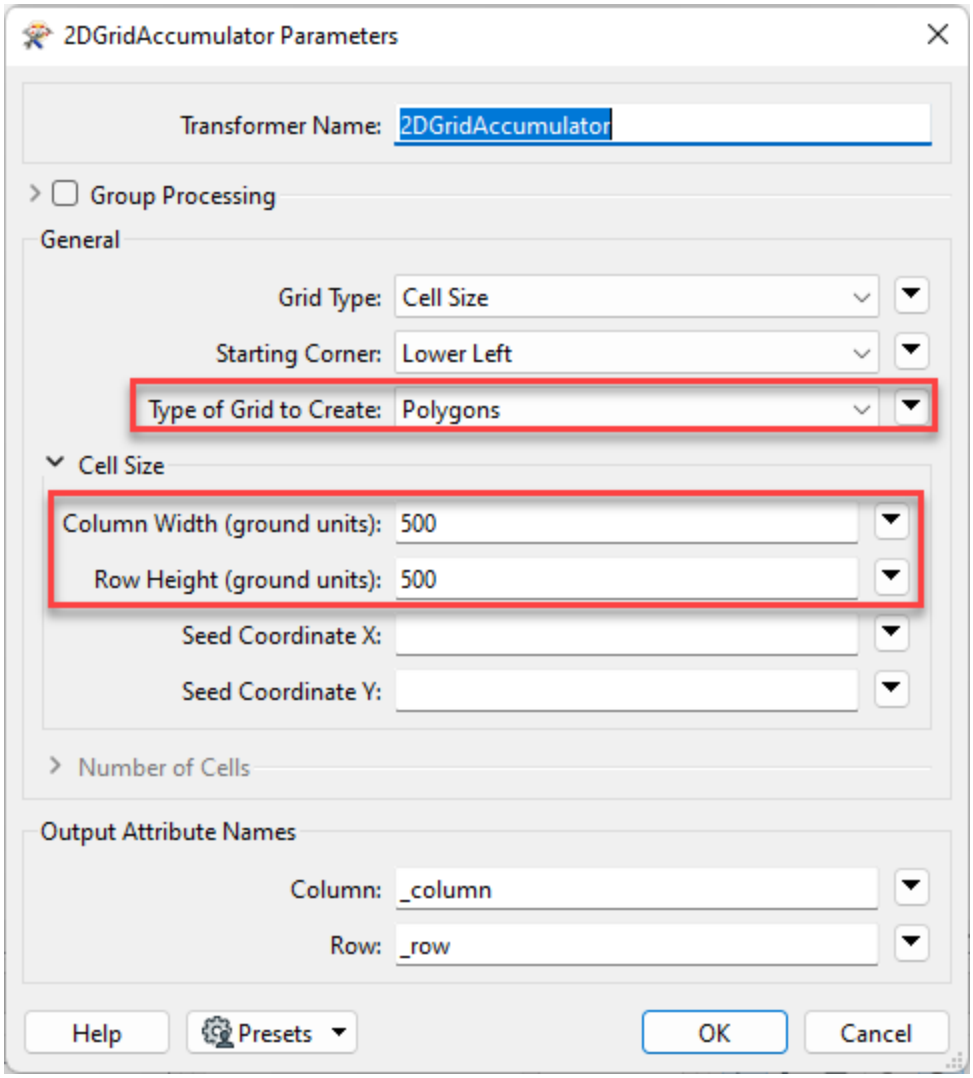
Right click on the Hexbinner bookmark and click disable (shortcut: CTRL+E). Now when we run our new translation, this translation branch won't run.

10. Add 2DGridAccumulator

Add the 2DGridAccumulator to the Point Output port on the GeometryFilter.

In the parameters set both the Column Width (ground units) and Row Height (ground units) to 500. This will be the size of our grid. *If you have time, try setting it to 250 or 750 just to see what the results look like.*

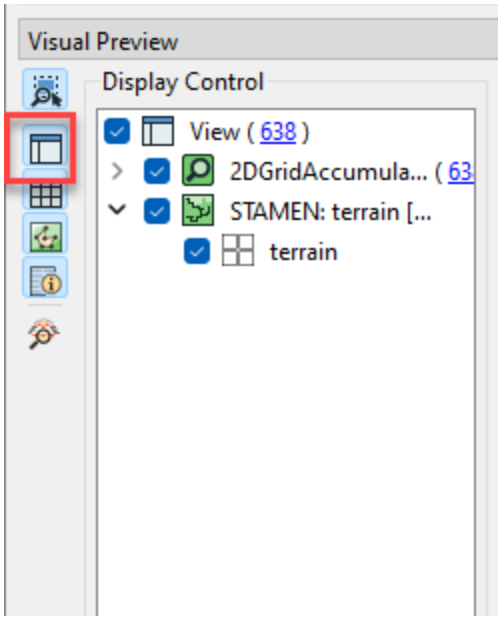
Then set the Type of Grid to Create to Polygons. This will create a square grid with lines. Again if you have time, experiment with Points (Corners) and Points (Centers). Once you have set your values, accept the defaults for everything else, and click ok.



Set Column Width and Row Height to 500, and set the Type of Grid to Created to Polygons

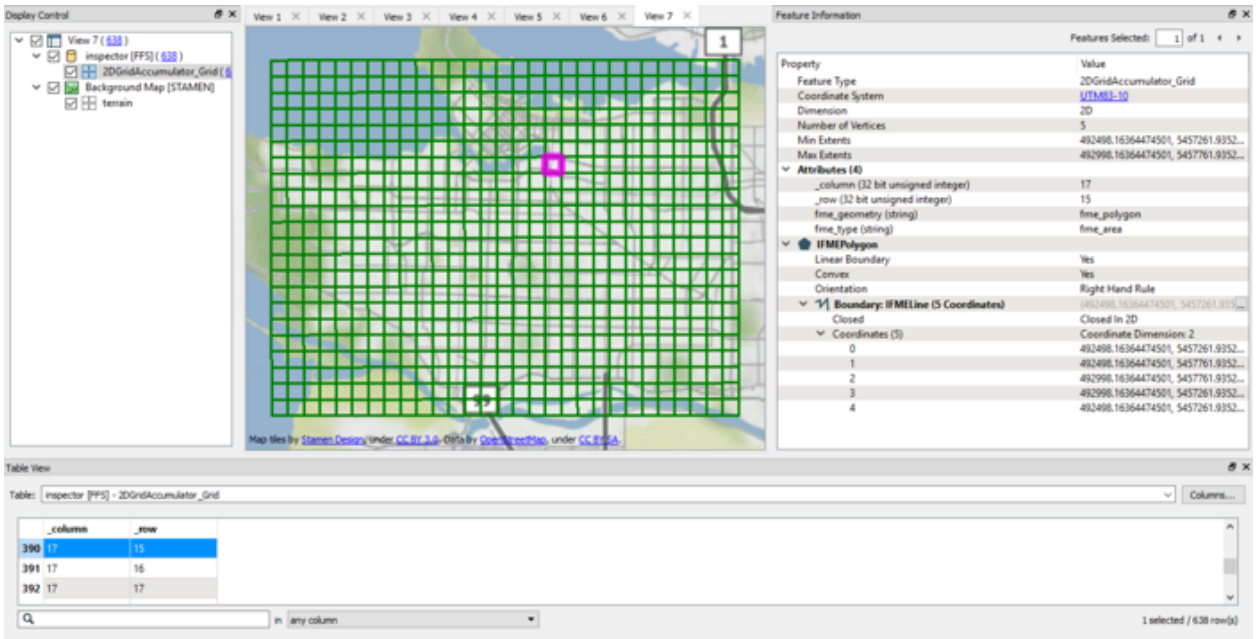
11. Inspect the results

Run the workspace up to the 2DGridAccumulator and inspect the Grid Output port to view the results. In the Visual Preview Window, under the Display Control, click the properties for the 2DGridAccumulator_Grid. If the Display Control window is not visible, you can enable it in the Visual Preview window by clicking the following button: 

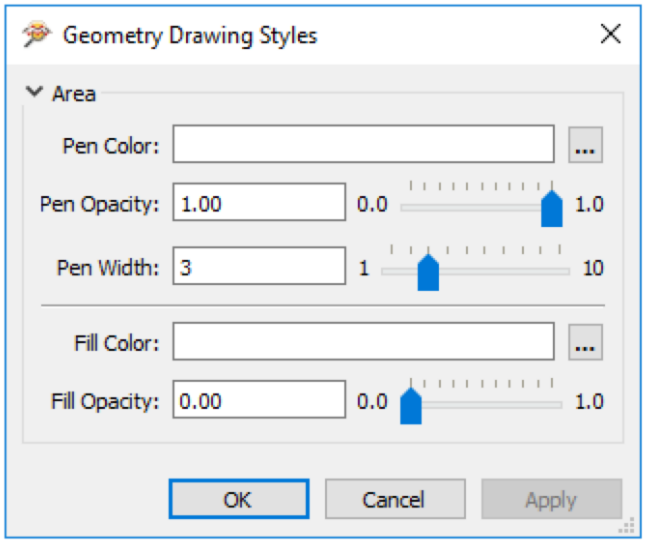


Then change the Fill Opacity to 0, and increase the Pen Width to 3. Click ok. If you would like to add a background map, click on Tools > FME Options... Then under Background Map, change the Background Format to Stamen Maps. This is the free background map service that comes with FME. Then click on Parameters... to set the Map List. The example is using Terrain, but you can use the one you like the best. Click ok.

Zoom in to see the background map inside each of the individual squares of the grid.



Inspect the 2DGridAccumulator in the Data Inspector. Change the drawing styles to view the background map

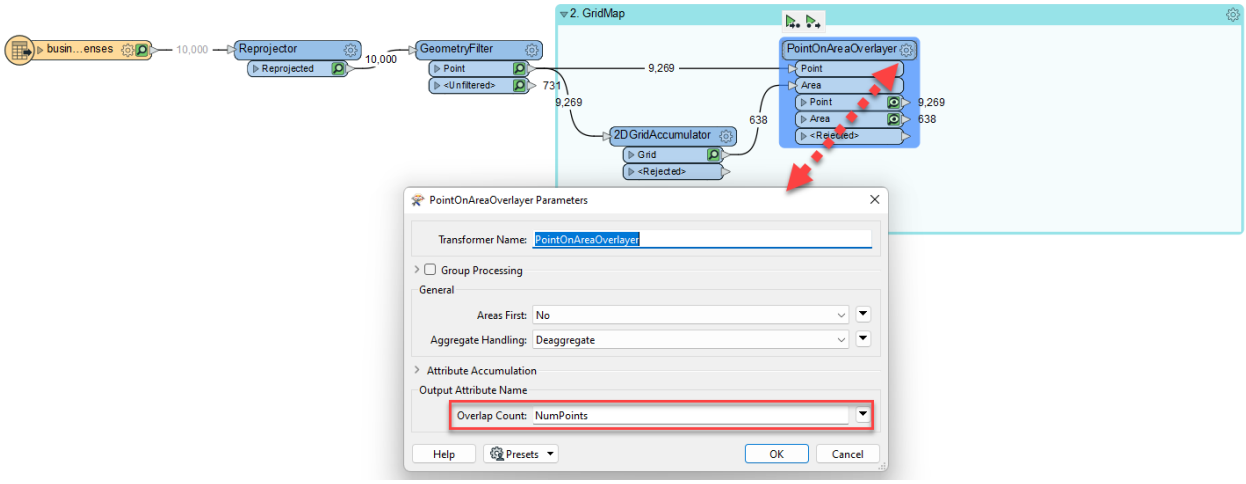


Change the Pen Width to 3, and the Fill Opacity to 0

12. Determine how many points fall within each grid square

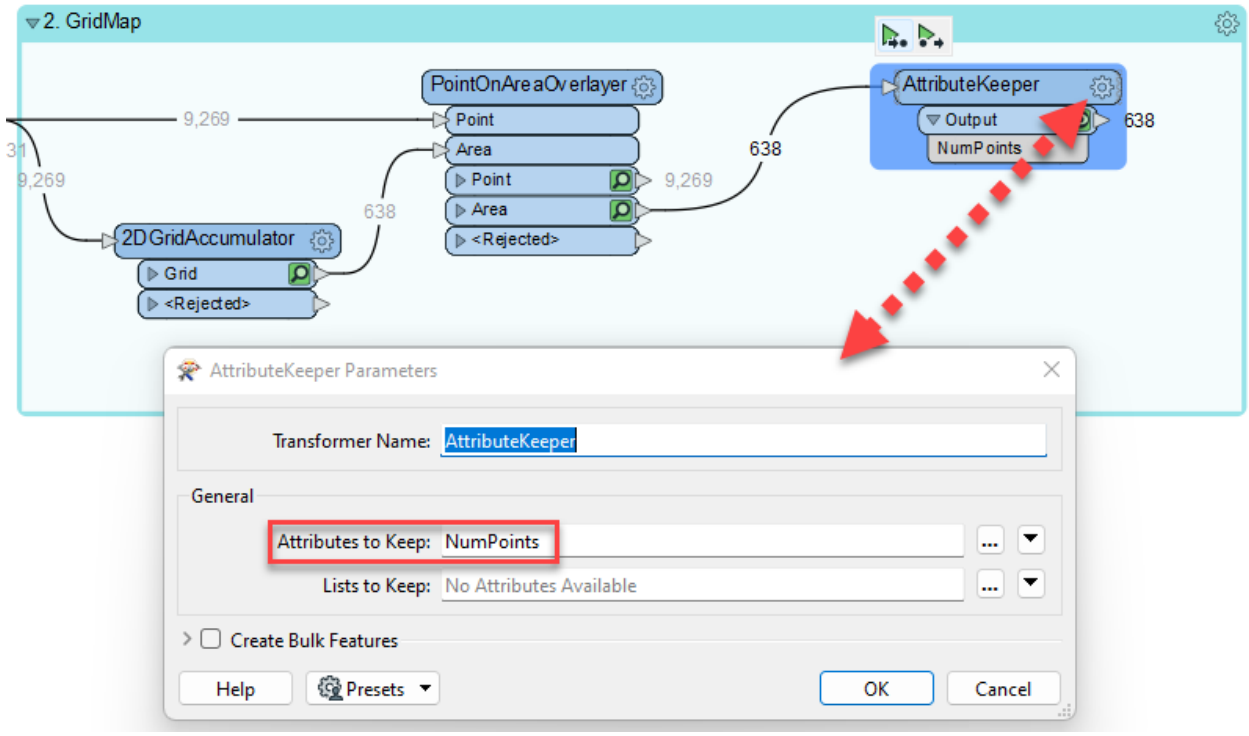
Now we need to determine how many of our business license points fall within each grid square. To do this we will add the PointOnAreaOverlayer.

Open up the parameters, let's change the name of Overlap Count Attribute to NumPoints. Click ok. Connect the Area Input port to the Grid Output port, and the Point Input port to the Point Output port on the GeometryFilter.



13. Keep only the attribute NumPoints

We are only interested in the attribute NumPoints, so we will **add the AttributeKeeper transformer to the Area Output port on the second PointOnAreaOverlayer.** The AttributeKeeper works just like the AttributeManager, but it is more efficient if you only want to keep certain attributes. Open up the AttributeKeeper parameters, then under Attributes to Keep, select NumPoints.



Only keep the NumPoints attribute

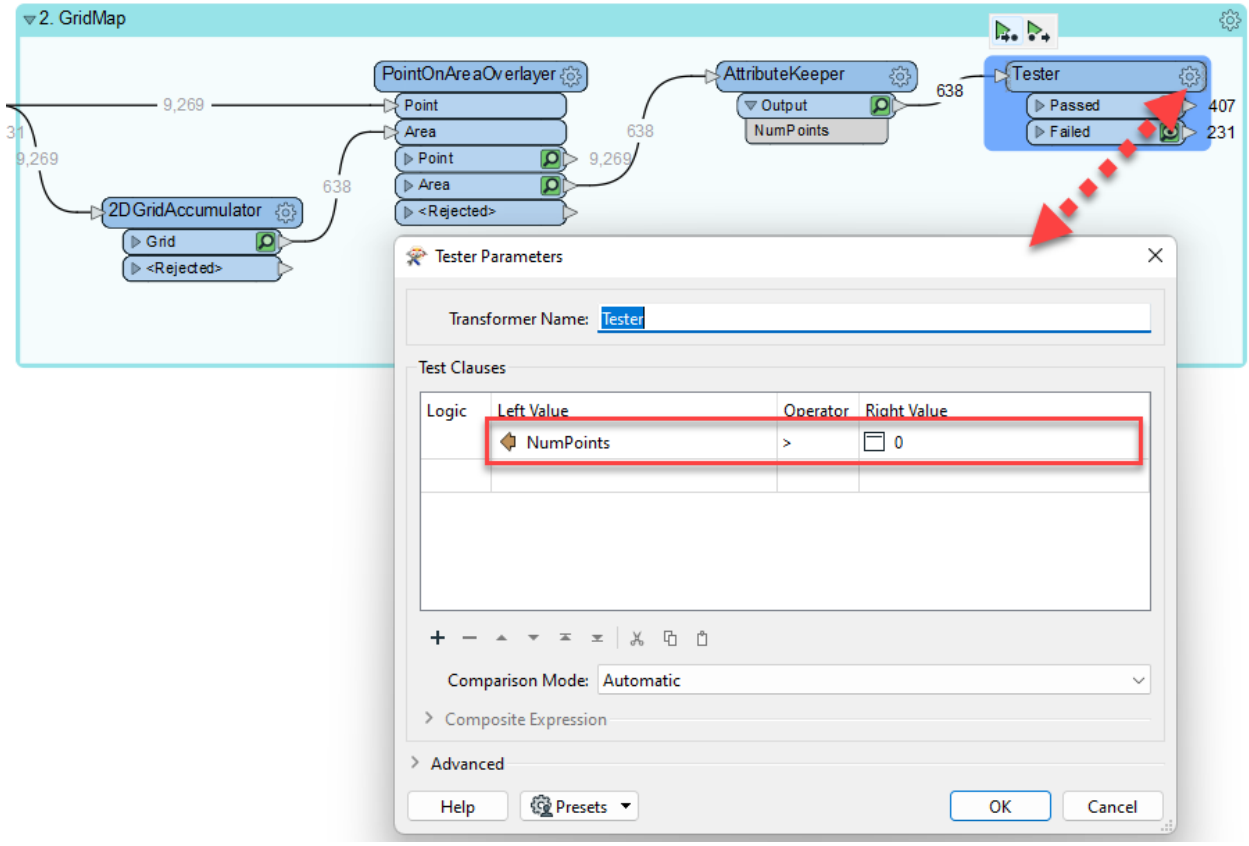
14. Inspect the results

It is a good idea to double-check your results after adding or subtracting attributes to ensure that the data you want is present and you've removed the data you don't want. Run the workspace up to the AttributeKeeper and inspect the feature cache with Visual Preview.

Confirm in the Table View, that the only attribute is NumPoint and that it has values. You might have to scroll to see values other than 0 since our grid covers ocean, there will be a lot of zeros.

15. Remove grid squares with zeros

As you can see, there are a lot of zeros. We have no need to display those, so let's remove them to tidy up our grid. Add a Tester transformer to the Output port of the AttributeKeeper. Then set it to Numpoints > 0. Now our grid will only display squares with values other than 0.



16. Write to Tableau

Finally, we need to write our grid to Tableau. Right-click on the workspace canvas and click Insert Writer Feature Type, then change the Table Name to GridMap, and set the Geometry to hyper_polygon. Finally, connect it to the Passed Output port on the Tester transformer and run the translation.

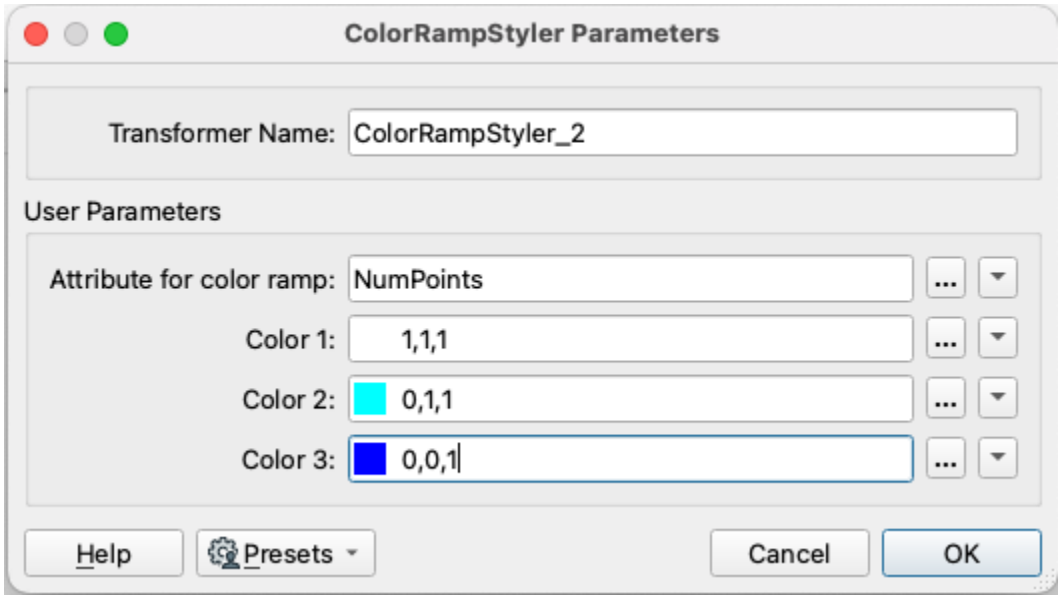
17. Add Bookmark

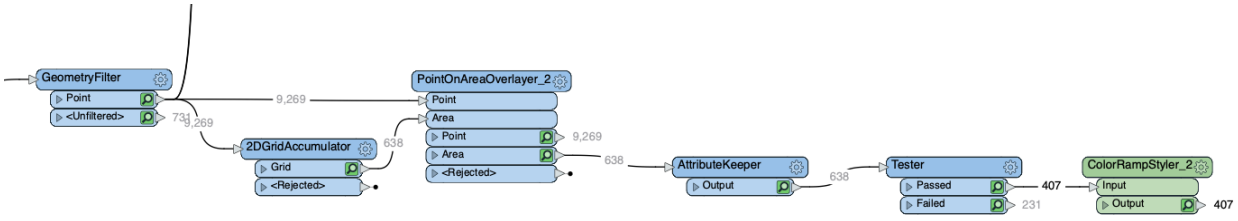
For this branch of the translation, we will only bookmark our writer called GridMap, this is because we will reuse all of the other transformers for the Dot Grid Map translation. Select the writer and either right click on the canvas to add the bookmark or use the keyboard shortcut ctrl-B .

18. Run translation and view data in Visual Preview

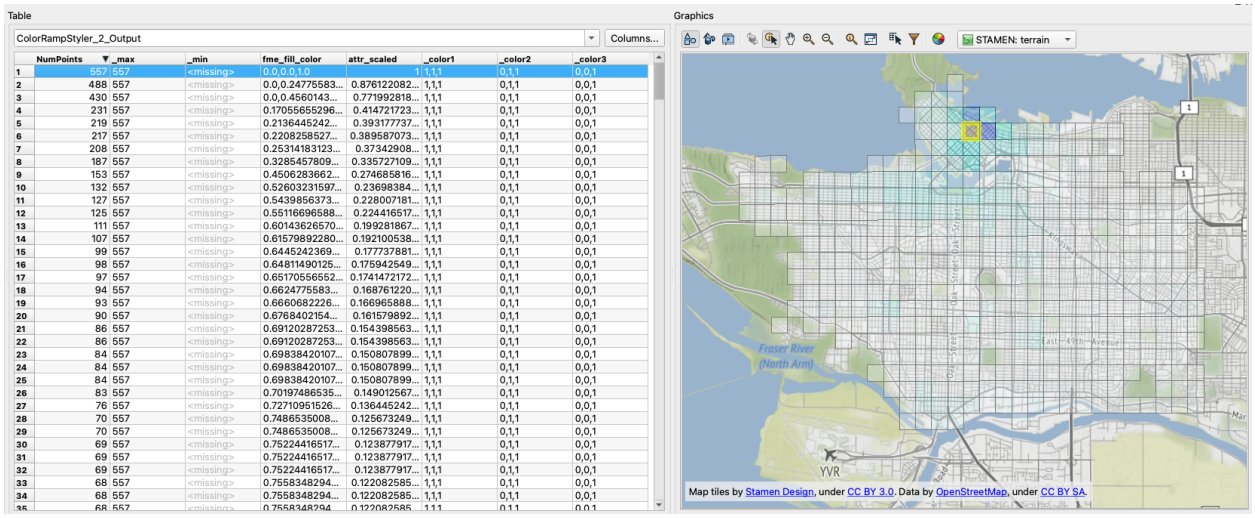
To emulate what this would look like in Tableau, use another ColorRampStyler and connect it to the Tester's Passed port. In the ColorRampStyler parameters, set the **Attribute for color ramp to NumPoints** and use the following color settings:

- **Color 1: 1,1,1**
- **Color 2: 0,1,1**
- **Color 3: 0,0,1**

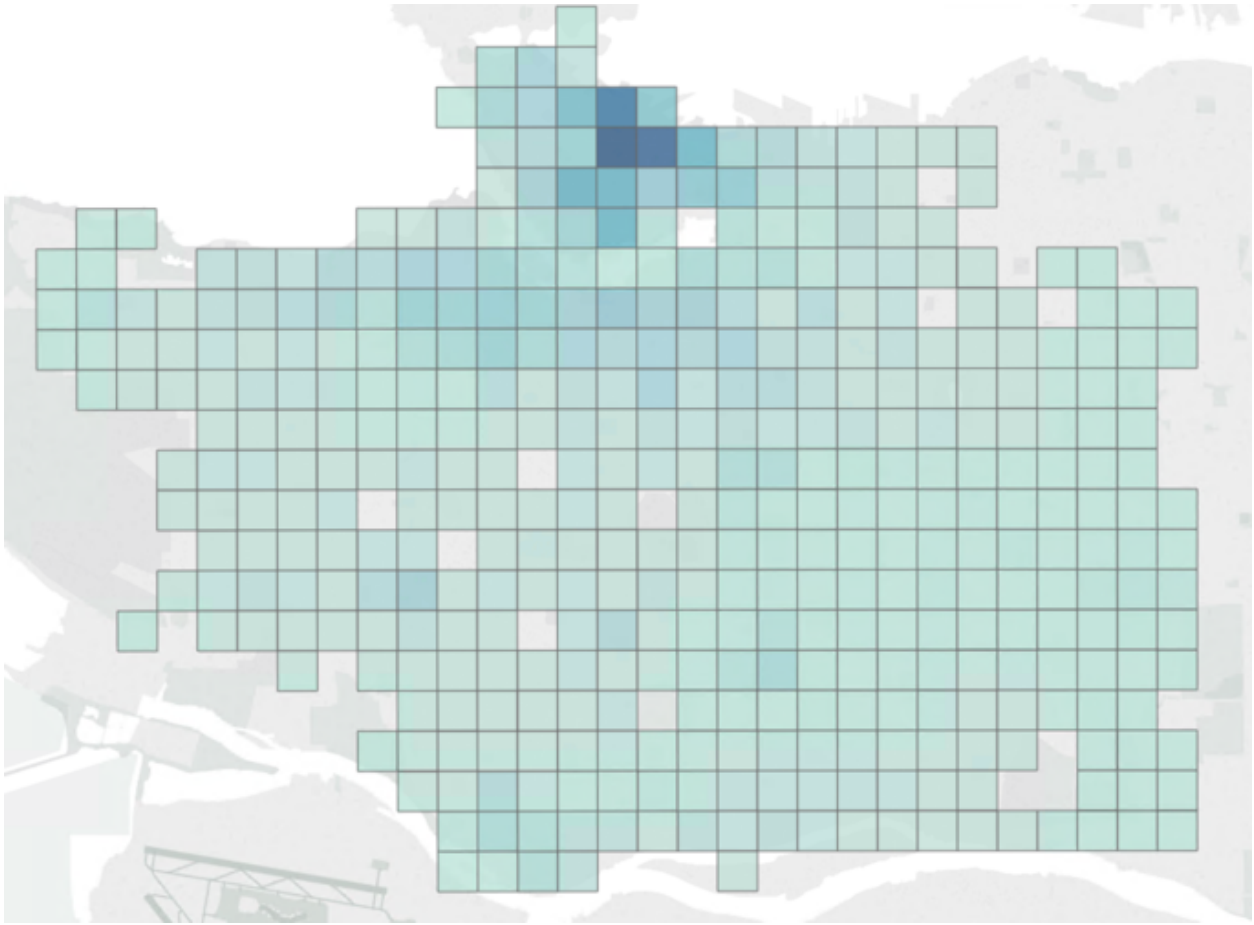




Once you have configured these parameters, run the workspace and inspect the feature cache on the ColorRampStyler:



In Tableau, the output would look similar to the image below:



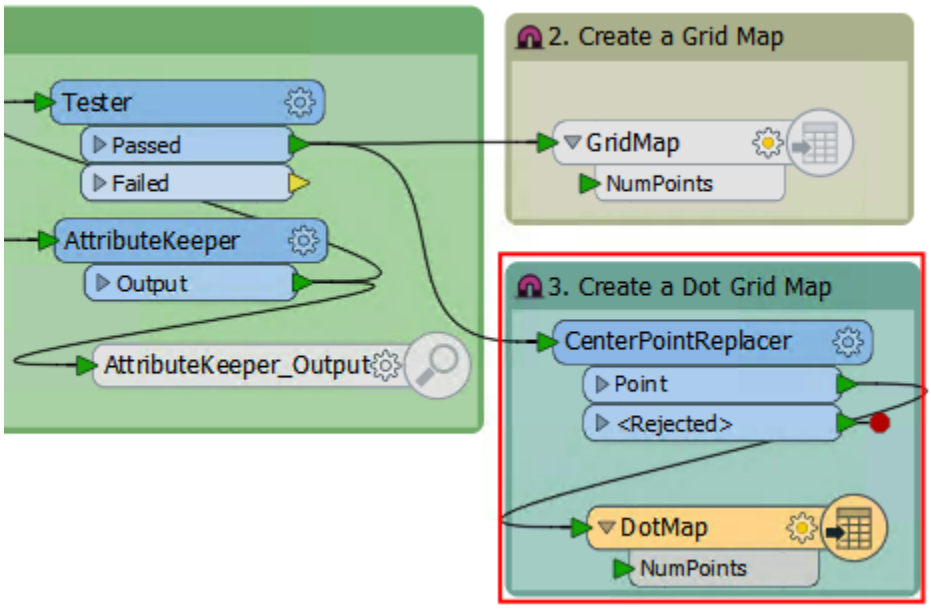
Grid Map styled in Tableau

Part 3: Dot Grid Map

19. Add a CenterPointReplacer

For the final map, we will be building on the previous example. Ensure all the other bookmarks are disabled.

To create a Dot Grid Map, we only need to add a couple of things to our previous translation. Add a CenterPointReplacer transformer to the Passed Output port on the Tester. This will find the center point of the polygon square we created with the 2DGridAccumulator and turn it into a point.

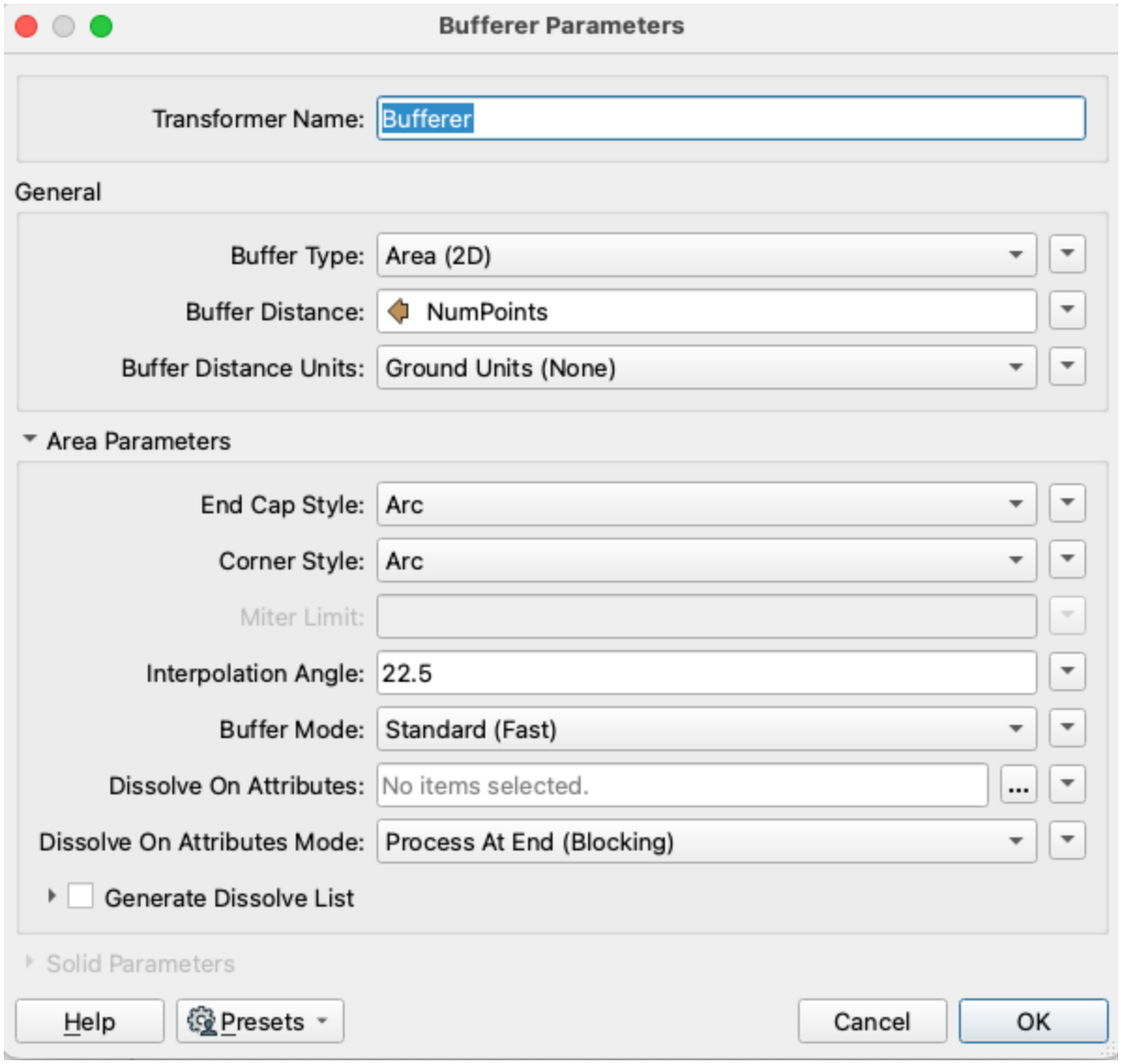


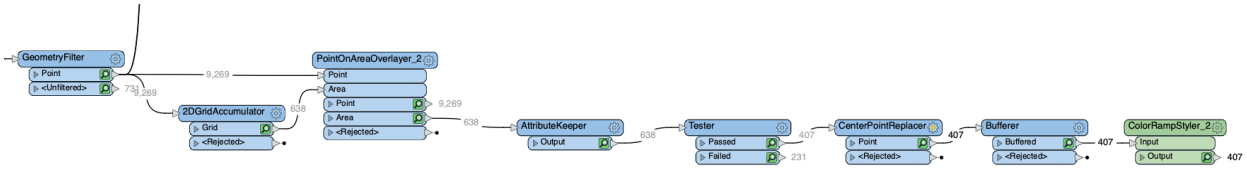
Complete translation for the Dot Grid Map. Add the CenterPointReplacer after the Tester

20. Add a Bufferer

If you were writing to Tableau, you would be able to skip this step as Tableau is capable of representing dot sizes based on attribute values. However, since we are keeping this exercise in the FME environment, we can replicate it by using a Bufferer transformer and inspecting it in the Visual Preview window.

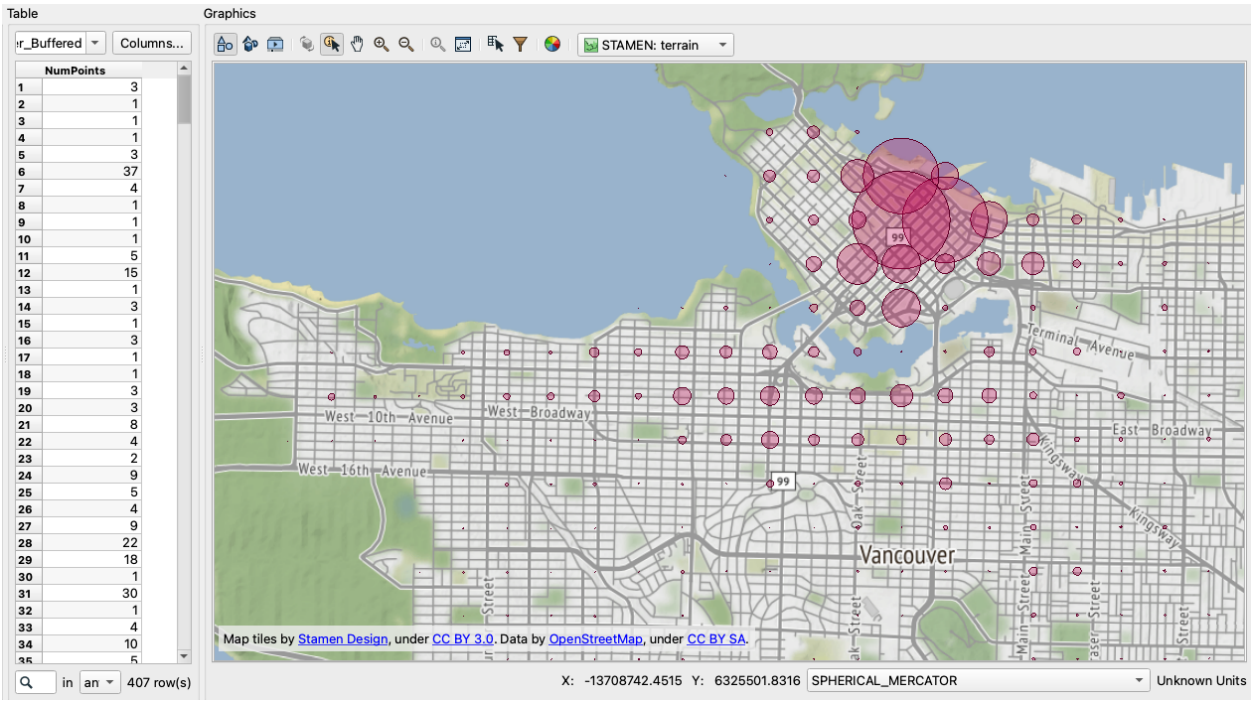
Add a Bufferer after the CenterPointReplacer and in the parameters, set the Buffer distance to the NumPoint attribute.





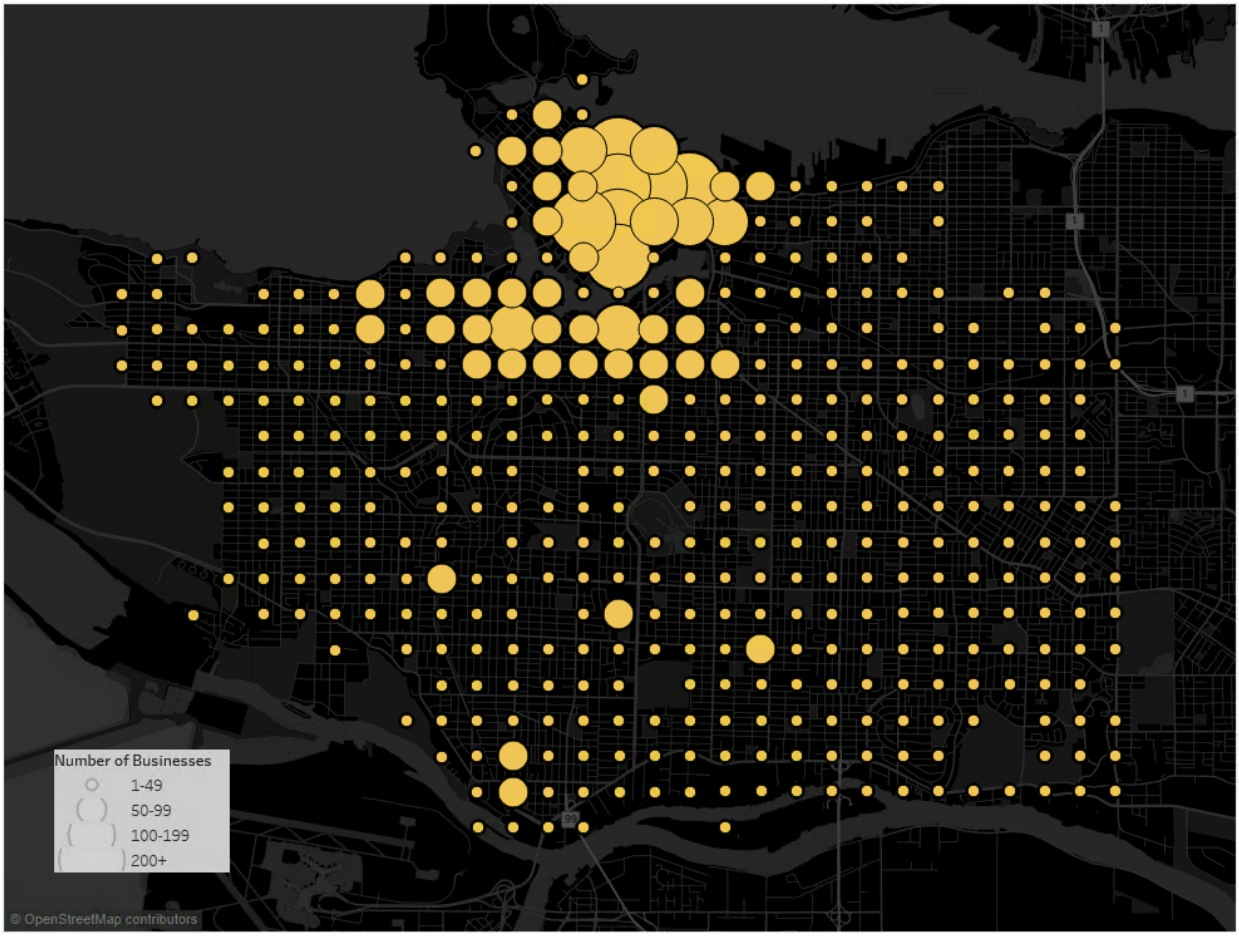
21. View the Result

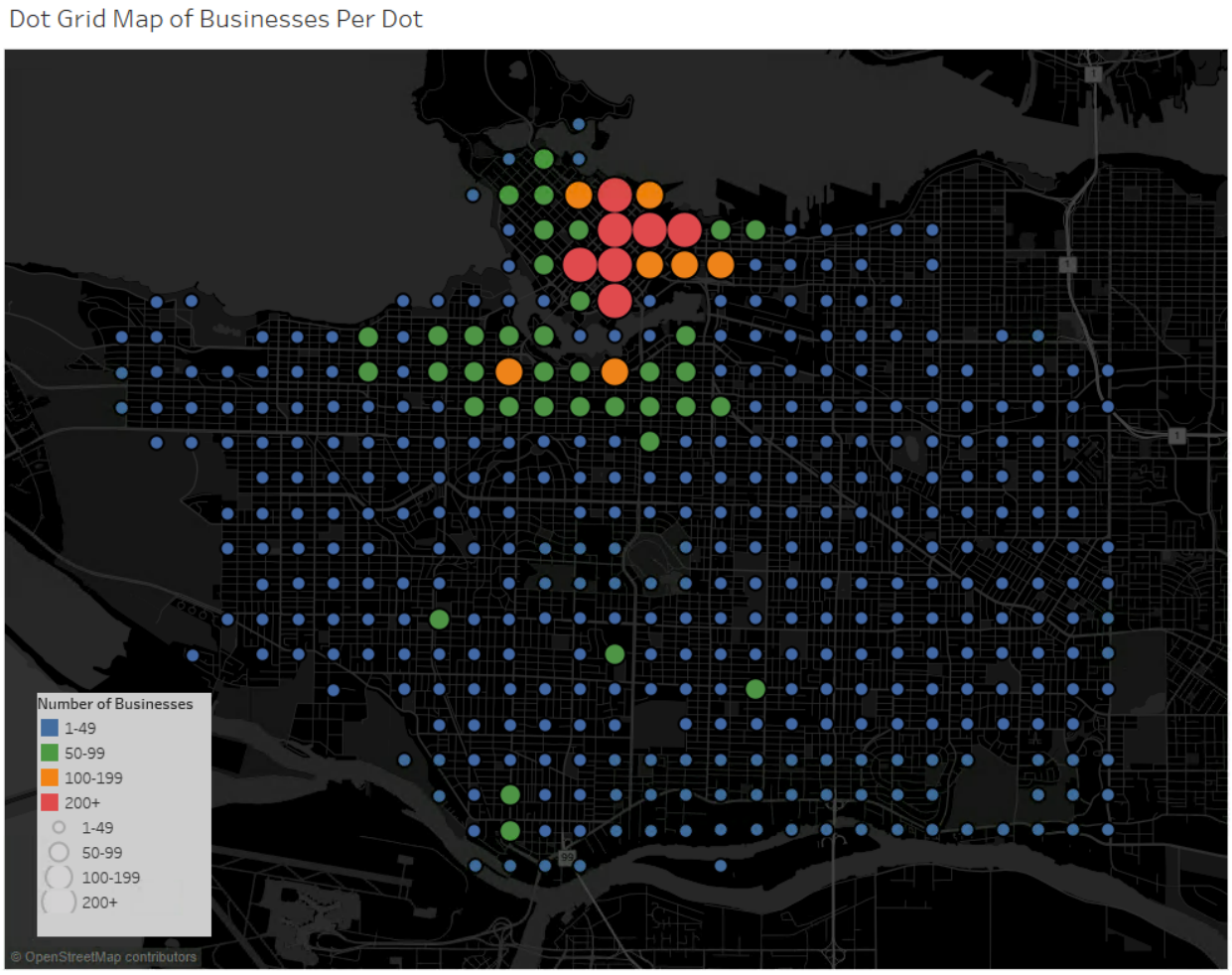
Run the workspace and inspect the feature cache on the Bufferer. *Again, if you wanted to colorize your features, you could use the ColorRampStyler after the Bufferer.*



In Tableau, the output would look similar to the image below:

Dot Grid Map of Businesses Per Dot





Dot Grid map styled in Tableau. Map shows both the color and size scale for the number of businesses

In order to achieve this look in Tableau, open a new sheet, double-click on the geometry to create a map. Add NumPoints to Size. On the sidebar, a legend with the sizes appears. Double-click it to open the size properties, increase the minimum and maximum dot size to increase the exaggeration. To change the appearance of the Dot Map, create a color grouping for NumPoints then add it to Colors.

In this exercise, you learned how to create a variety of maps for Tableau using Hexbins, Grids, and Dots in addition to how you can create similar outputs with FME which can be helpful when creating HTML, PNG, or PDF reports with FME.