

Ultra (Overpass Ultra) Tutorial

Felipe Valdez

Contents

1	What is Ultra?	1
2	What is Overpass?	2
3	Building an overpass query	2
4	Styling in Ultra	5
5	Adding interactivity to the map	9
5.1	Adding an initial center point and zoom level	9
5.2	Adding navigation controls to the map	10
5.3	Customizing the popup window	11
6	Complete query for an interactive map	12
7	Sharing you Ultra map	14
	Attribution	16

1 What is Ultra?



Ultra (née Overpass Ultra) is a web-application made to simplify making maps with [MapLibre GL JS](#) with data from various file/query types such as Overpass, GeoJSON, GPX, and more.

2 What is Overpass?



The [Overpass API](#) is a read-only service that lets users retrieve specific [OpenStreetMap](#) (OSM) data based on custom queries.

Unlike OSM's main API, which focuses on editing, Overpass is optimized for data retrieval, handling anything from a few elements to millions in minutes.

Users can filter data by location, object type, tags, and more. Overpass Ultra, a web-based tool, helps with query creation.

Refer to the [user manual](#) and [Overpass QL guide](#) for details.

Resulting map from this tutorial

[Open the resulting map in a new window](#)

3 Building an overpass query

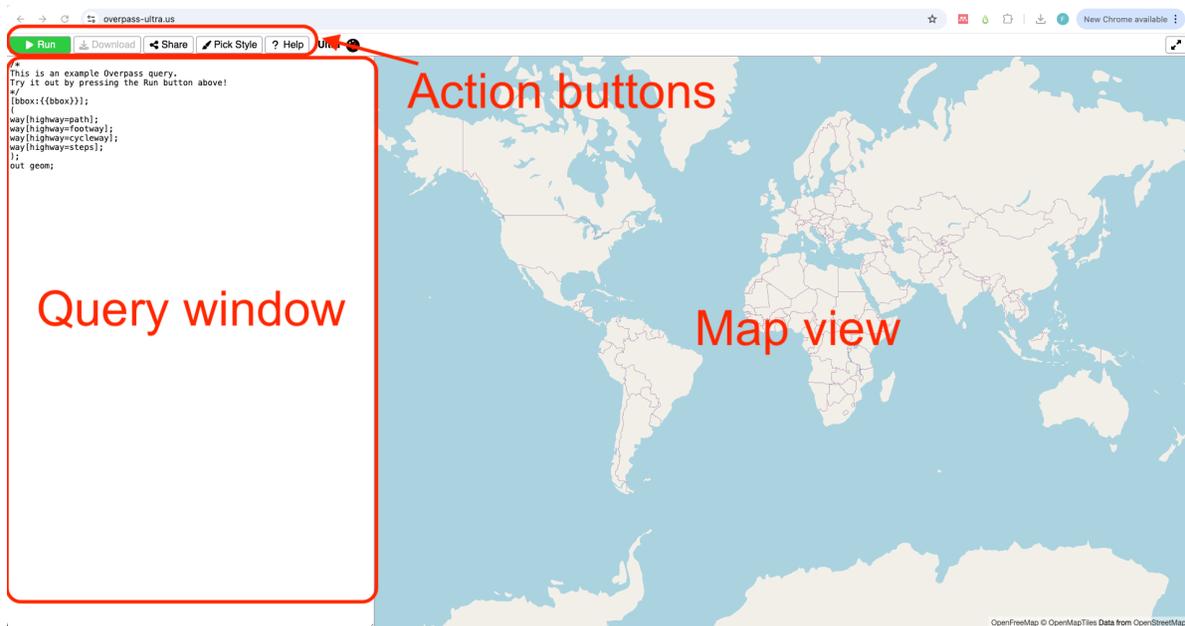
In this step-by-step tutorial we will learn how to create, style and share a map in Ultra.

We will be creating an interactive map of all recreation areas in Philadelphia.

01 *Open a new map in Ultra*

Go to <https://overpass-ultra.us/>.

The first time you open the web you will see a default map like this:



02 *Type your query*

A **query** is a way to filter and retrieve data from OpenStreetMap. It allows you to search for specific types of map features.

Queries in Ultra follow the structure of the [Overpass API](#).

If you are not familiar with OpenStreetMap data, see the info window below:

 **How data is organized in OSM?**

All data in OSM is represented by an **element**.

An **element** can be either a **node** , a **way**  or a **relation** .

Each element is described using **tags** which are the combination of a **key** and a **value**

.

For example, a coffee shop is represented by an element type **node** with tags **amenity=cafe**.

Learn more about elements and tags [here](#).

The ‘anatomy’ of a query: *Hover over the numbers on the right in the code to reveal what each line on the query does*

The following is the default query you see when you open Ultra for the first time:

```

1  /*
2
3
4  */ # <1>
5  [bbox:{{bbox}}]; # <2>
6  (
7  way[highway=path]; # <3>
8  way[highway=footway];
9  way[highway=cycleway];
10 way[highway=steps];
11 ); # <4>
12 out geom; # <5>

```

- ① This is a comment. Everything inside `/* */` will not be considered in the query.
- ② This line defines a `bbox` which limits the query to what you are viewing on the map.
- ③ This line retrieves an element type `way` that has the key `highway` and value `path`.
- ④ All single queries within the `()` are grouped together.
- ⑤ The output format for your query. In this case `geom` returns the actual shape of the features.

For this example we will use the following query.

As there are different types of recreational areas, we will be using a group of three queries with different combinations of value for the key `leisure`. We will be querying elements tagged: `"leisure":"park"`, `"leisure":"playground"` and `"leisure":"garden"`. There are more tags that can describe these areas. Explore your own case study in [OSM](#).

i Note

Remember to zoom in to your interest area. Keep your query area small to retrieve data faster.

```

1  [bbox:{{bbox}}]; # <6>
2  ( # <7>
3  nwr[leisure="park"]; # <8>
4  nwr[leisure="playground"]; # <9>
5  nwr[leisure="garden"]; # <10>
6  );
7  out geom; # <11>

```

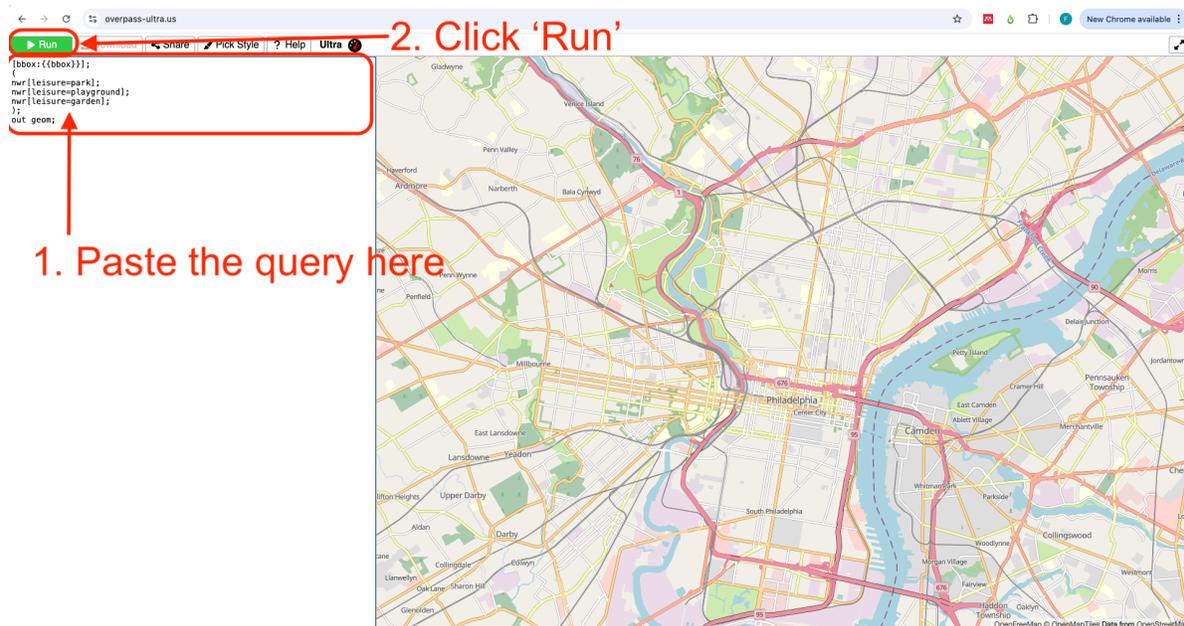
- ⑥ We will keep the `[bbox:{{bbox}}];` line to filter the query to the map view.
- ⑦ Here we start our grouped query. As there are different types of areas, we will be using three values.

- ⑧ One way to tag recreational areas is `leisure:"park"`. Note that we use `nwr` to get any node, way or relation.
- ⑨ Our second query is `leisure:"playground"`.
- ⑩ The third type we are querying is `leisure:"garden"`.
- ⑪ We close our query retrieving the geometry of the elements.

03 Run your query



Copy and paste this query in Ultra's query window, and then click 'Run'



After some seconds you will see the results display on the map. All results are shown in yellow. In the next steps we will learn how to style the results.

Learn more on how to build a query [here](#)

4 Styling in Ultra

In Ultra, you can style the map elements by adding a `style:` key on a YAML front-matter. This YAML front-matter is a way to add metadata to the query file. All these will be read by Maplibre and Ultra when rendering your resulting map.

01 Add a title and description

The YAML front-matter has to be framed inside `---`, just like in the example below.

```

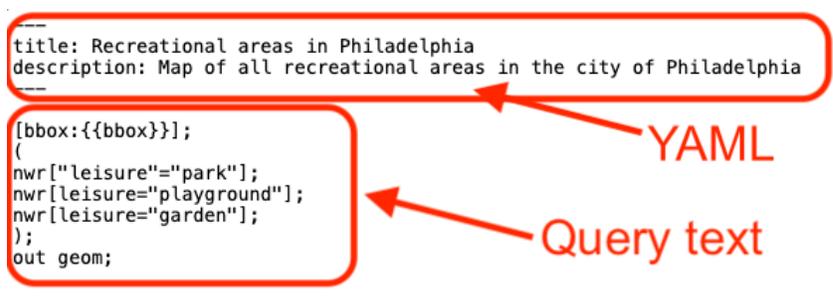
1 --- # <1>
2 title: Recreational areas in Philadelphia # <2>
3 description: Map of all recreational areas in the city of Philadelphia # <3>
4 --- # <4>

```

- ① Opening the YAML front-matter text.
- ② Title property to be included.
- ③ Description text to be included.
- ④ Closing line of the YAML front-matter.

Copy and paste (or type) the YAML front-matter with the title and description properties into the Ultra query we created in the previous section.

Paste the YAML front-matter **before** the query text.



02 Add a custom style

Styling is possible in Ultra within the YAML front-matter. A **style** is a series of rules on how to visually render a map, what elements are drawn and which colors, icons, sizes, and more to use.

Ultra uses the [Maplibre Style Spec](#). This style is specified by **properties**.

In this section we describe some of the basic **properties** needed to create a custom style.

The first step is to add **style:** to the YAML front-matter. You can type it right after the **description:** property.

We will assign colors to the polygons we obtained from the query. For that we will add a layer of type fill (polygon) with **type: fill**. There are other types of layers you can add: 'background', 'circle', 'heatmap', 'fill-extrusion', 'line', 'symbol'. Each serves different purposes and works for specific elements, if it is a point, for example, you might use **type:symbol**.

Then we specify the property of our layer of type 'fill' we want to modify.

We add **fill-color:** to change the color of our polygons. Each layer type has its own properties that can be modified. See [Layers](#) on the Maplibre Style Spec documentation.

03 Add conditionals to assign colors based on the feature properties

Since we want to color each type of recreational area with a different color, we are using `- case` (line 9) to introduce conditions. We type a conditional statement followed by a specific color for each type of recreational area.

On line 10 of the example code we add `[get, leisure]` to select the tag `leisure` from the properties. Then we ask for those elements that have `"park"` as the value for that tag. The `==` symbol states this.

On line 11, we specify the color we want to assign to the polygons with tag `leisure=park`. The colors can be specified using the 'rgb' (red, green, blue) format. You can use tools like [Google Color Picker](#) to know the 'rgb' code of a color.

Lines 12 to 15 on the sample code contain the same conditional + color for the other two types of recreational areas: `playground` and `garden`.

On line 16 we introduce a fallback color. This color will be assigned to polygons that do not satisfy the conditions in our `case`. You always need to assign a fallback color in Maplibre Style Spec.

04 Run the query again

Copy and paste the sample code below on your Ultra map and run the query again.

```
1  ---
2  title: Recreational areas in Philadelphia
3  description: Map of all recreational areas in the city of Philadelphia
4  style: # <1>
5    layers: # <2>
6      - type: fill # <3>
7        paint: # <4>
8          fill-color: # <5>
9            - case # <6>
10             - [ ==, [ get, leisure ], "park" ] # <7>
11               - rgb(159, 247, 7) # green # <8>
12             - [ ==, [ get, leisure ], "playground" ] # <9>
13               - rgb(80, 163, 91) # dark green # <9>
14             - [ ==, [ get, leisure ], "garden" ] # <9>
15               - rgb(181, 159, 16) # brown # <9>
16             - rgb(0, 0, 0) # <10>
17  ---
18  [bbox:{{bbox}}];
19  (
20    nwr[leisure=park];
21    nwr[leisure=playground];
22    nwr[leisure=garden];
```

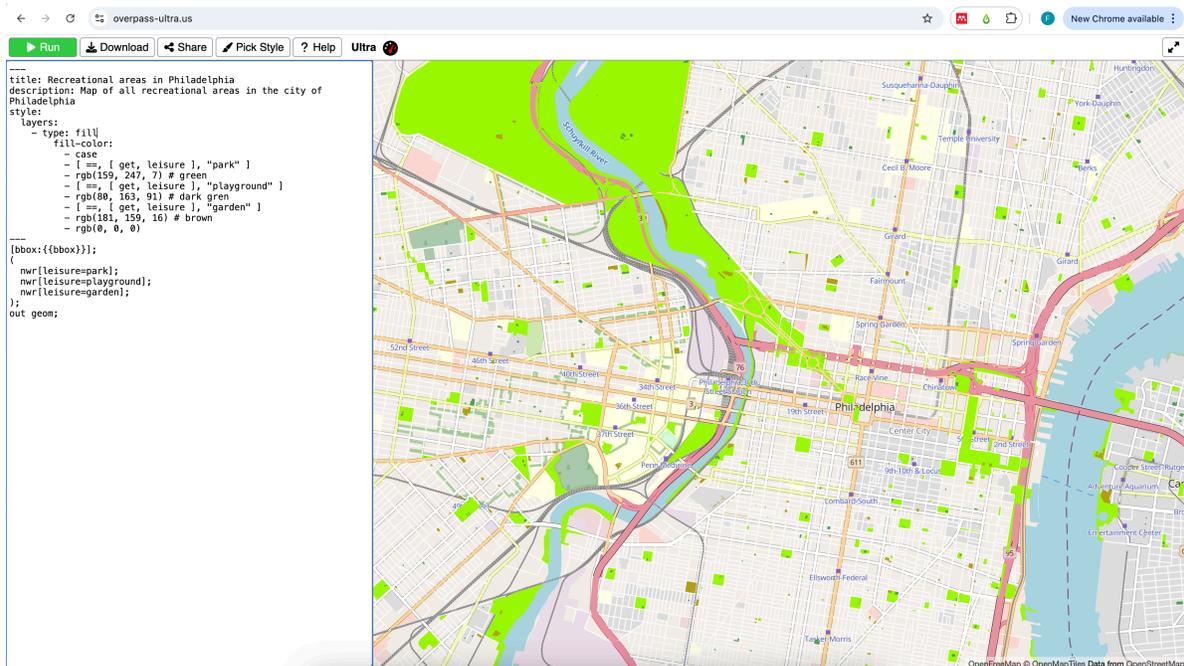
23

```
);
```

24

```
out geom;
```

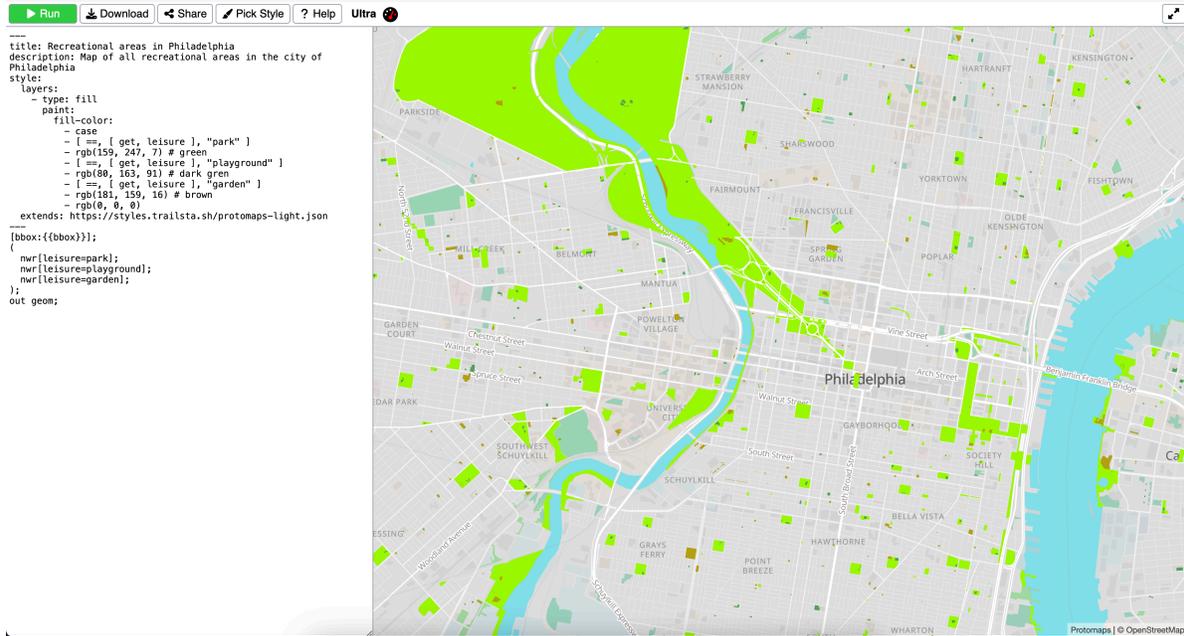
- ① We start the style section with this line.
- ② The most important property is `layers` which adds something to the map.
- ③ We define our layer type as `fill` as we want the polygons to be painted in the map.
- ④ There are two main types of properties `paint:` and `layout`.
- ⑤ We define the `fill-color` to be the property we want to modify.
- ⑥ The `case` expression will assign a value to an output of a condition (lines 10 to 15).
- ⑦ We search for features that have `park` as the value of `leisure`.
- ⑧ This color is assigned to those features that satisfy the condition established in the previous line.
- ⑨ We do the same evaluation and assignment for those features with `playground` (line 13) and `garden` (line 15).
- ⑩ This is the fallback color. This will be assigned if there are features that do not satisfy any of the previous conditions.



05 Change the background style

Ultra comes with a variety of background styles you can choose. To change the style click on the [Pick Style](#) button on the top menu bar and select any style from the list.

In this example we chose 'Protomaps Light' because it uses shadows of grey for most of the elements, highlighting the vibrant greens we chose for the parks.



Notice that an extra line is added to the YAML front-matter with the URL of the background style used. You can choose from any of the styles in Ultra or use your own from a URL pointing to a JSON file.

5 Adding interactivity to the map

In this section we explore some additional options to customize your resulting map. From an initial center point and zoom, to adding navigation controls and popup windows.

5.1 Adding an initial center point and zoom level

Ultra allows adding some additional options for your map, equivalent to Maplibre Style Spec [MapOptions](#).

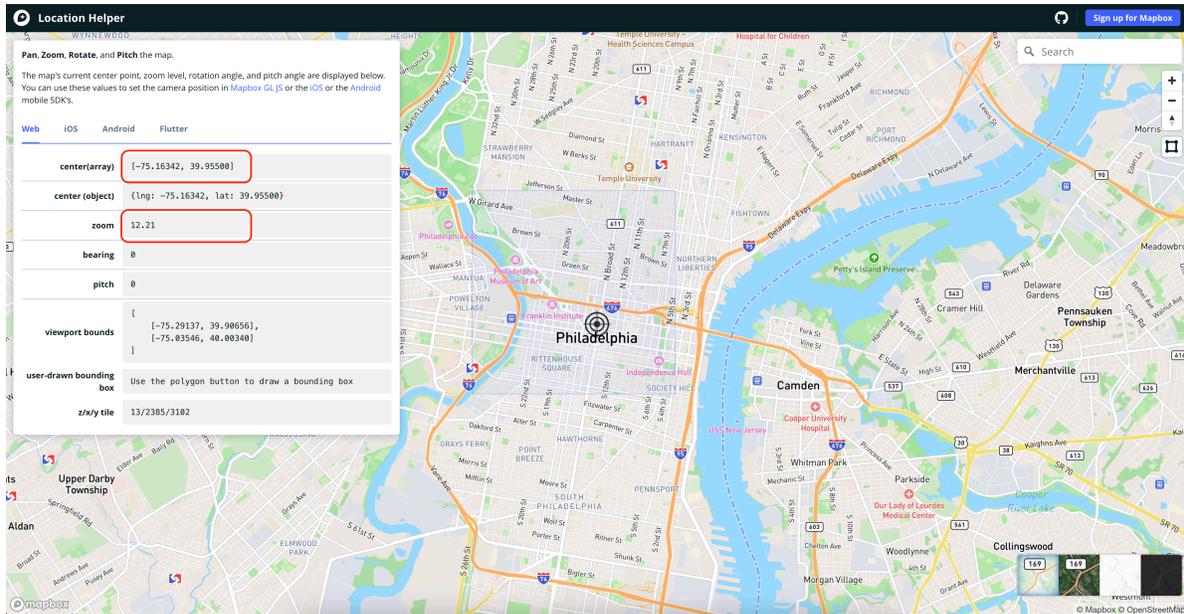
Adding an initial center point and zoom is helpful to ensure the map starts in the desired location for the viewer.

To add a center point, simply add the following to your query window, after the `description:` property and before the `style:`.

```
options:
  center: [-75.16342, 39.95500]
  zoom: 13
```

The values in this example are centered in the city of Philadelphia, PA.

To know the coordinates and zoom level for a different area you can use the [Mapbox Location Helper](#). Zoom to the area you want and copy/paste the values on `center(array)` and `zoom`.



5.2 Adding navigation controls to the map

Similar as the MapOptions, Ultra allows adding controls to the resulting map.



In this example we are adding a `NavigationControl`



a `GeolocateControl` and an `HTMLControl`.

To add these three controls, just copy and paste the following code lines after the `options:` property and before the `style:` in your query.

```
controls: # <1>
- type: NavigationControl # <2>
  position: top-right # <3>
- type: GeolocateControl # <4>
  options: # <4>
```

```

positionOptions: # <4>
  enableHighAccuracy: true # <4>
  trackUserLocation: true # <5>
- type: HTMLControl # <6>
  options: # <6>
  html: > # <7>
    <h1><center>Green areas in Philadelphia</center></h1> # <8>
  css: > # <9>
    h1 { # <9>
      position: fixed; # <9>
      top: 0; # <9>
      left: 0; # <9>
      right: 0; # <9>
    } # <9>

```

- ① Adding the `controls:` property.
- ② First we add a type `NavigationControl`.
- ③ Set the position of the control in the map window.
- ④ Second we add a `GeolocateControl`.
- ⑤ We enable the option to track the user location.
- ⑥ Third we add the `HTMLControl` to include a title for the map.
- ⑦ Open an html section.
- ⑧ Include the text for the title between the `<center>` `</center>`.
- ⑨ Style the title using css.

i Note

These options will be visible only when you share the query as an interactive map. See section 6 of this tutorial.

5.3 Customizing the popup window

Finally, another useful option to add is a customized popup window. This window will appear when the viewer clicks or hover the mouse over one of the elements from the query.

Here we add a customized popup window that shows the name and leisure type and will appear when one hovers over the polygon.

Paste the code below under the `description:` property on your query.

```

popupOnHover: true # <1>
popupTemplate: "{{tags.name}} - {{tags.leisure}}" # <2>

```

- ① This activates the popup on hover. The default is when click.
- ② We only add the feature name and leisure type to the popup.

6 Complete query for an interactive map

You can copy and paste the code below on your query window. Remember to zoom in to Philadelphia to have the correct bbox for the map configured here. If you are working on a different area of interest, zoom in to that area and change the `center:` and `zoom:` properties on the YAML front-matter before running the query.

```
1 ---
2 title: Recreational areas in Philadelphia
3 description: Map of all recreational areas in the city of Philadelphia
4 popupOnHover: true
5 popupTemplate: "{{tags.name}} - {{tags.leisure}}"
6 options:
7   center: [-75.17474, 39.95880]
8   zoom: 13.83
9 controls:
10  - type: NavigationControl
11    position: top-right
12  - type: GeolocateControl
13    options:
14      positionOptions:
15        enableHighAccuracy: true
16        trackUserLocation: true
17  - type: HTMLControl
18    options:
19      html: >
20        <h1><center>Green areas in Philadelphia</center></h1>
21      css: >
22        h1 {
23          position: fixed;
24          top: 0;
25          left: 0;
26          right: 0;
27        }
28 style:
29   layers:
30     - type: fill
31     paint:
```

```

32 fill-color:
33   - case
34   - [ ==, [ get, leisure ], "park" ]
35   - rgb(159, 247, 7) # green
36   - [ ==, [ get, leisure ], "playground" ]
37   - rgb(80, 163, 91) # dark green
38   - [ ==, [ get, leisure ], "garden" ]
39   - rgb(181, 159, 16) # brown
40   - rgb(0, 0, 0)
41 extends: https://styles.trailsta.sh/protomaps-light.json
42 ---
43 [bbox:{{bbox}}];
44 (
45   nwr[leisure=park];
46   nwr[leisure=playground];
47   nwr[leisure=garden];
48 );
49 out geom;

```

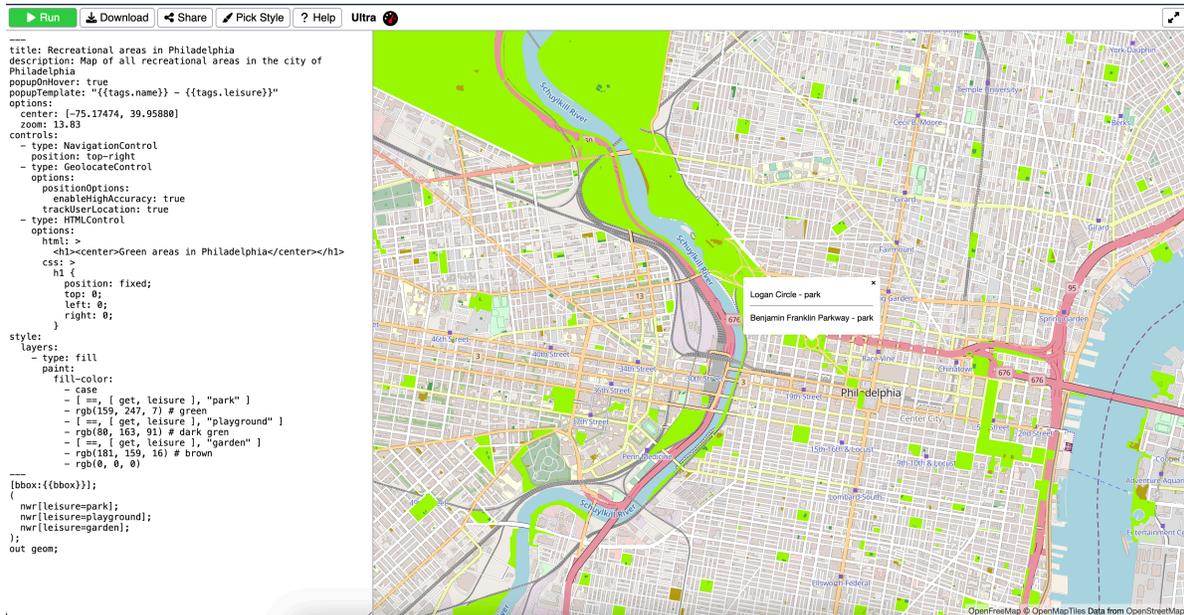


Figure 1: Resulting Query Map

7 Sharing you Ultra map

The last step is to share your resulting map.

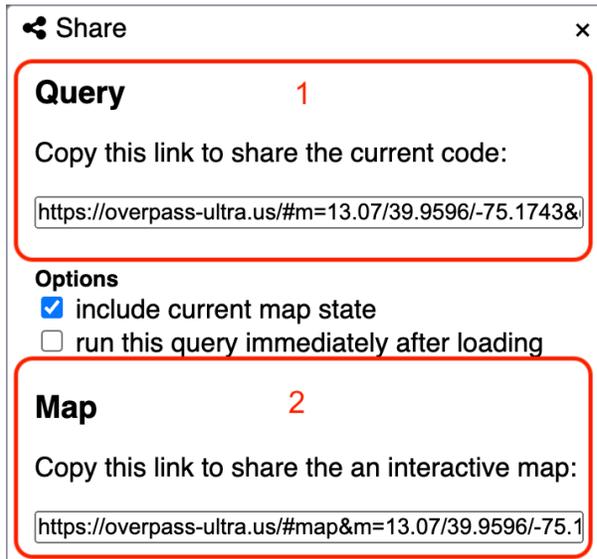
There are mutiple ways you can share it online, as a query or an interactive map.

01 *Click on Share*

Click on the Share 

02 *Select how you want to share*

In the new window, select how you want to share the map.



1. If you select to share as query, the link will open the map and query with all actions.

Attribution

Open Geospatial Data by Felipe Valdez is licensed under CC BY-NC-SA 4.0