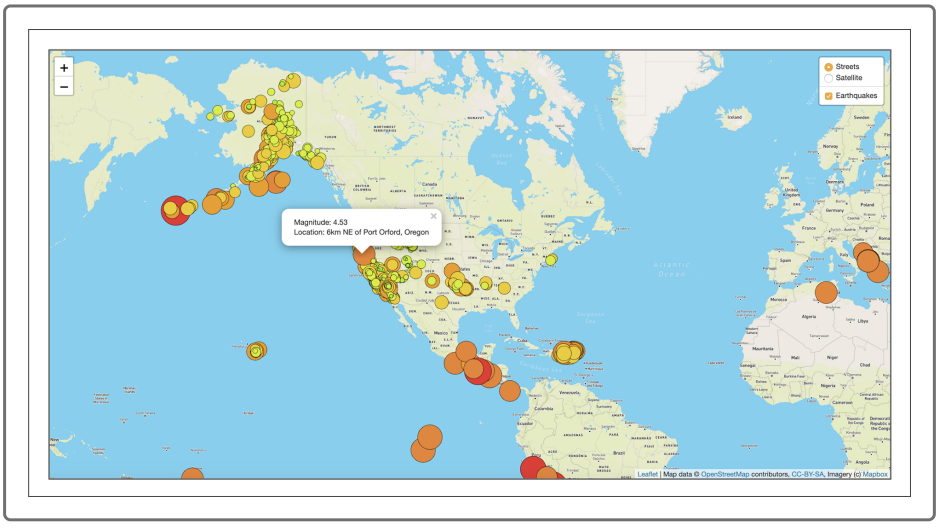
# 13.6.4 Add Earthquake Data as an Overlay

**The** earthquake map is looking great. Sadhana thinks that having the earthquake data as an overlay on both the Streets and Satellite tile layers, would be a nice added feature so users can turn the data on and off.

After adding an overlay for the earthquakes, our map should look similar to the following map, allowing the viewer to toggle off and on the earthquake data. The default setting will always show the data:




Before we write the code to create this map, make a copy of the `logicStep3.js` file and name it `logicStep4.js`. Now, let's edit the file.

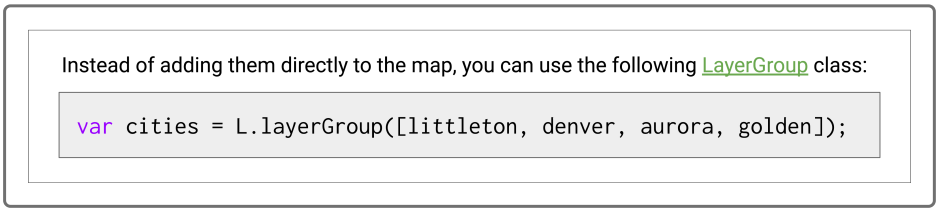
Refer to **Layer Groups and Layers Control (https://leafletjs.com/examples/layers-control/)** for guidance on how to add

data as an overlay to the map.

NOTE

> The base layers or tile layers, the Streets and Satellite, are mutually exclusive, and only one can be visible at a time on our map. Whereas, overlays are anything that you want to add to the map, which are "laid over" all the base layers and are visible all the time.

In the example below, from the **Layer Groups and Layers Control (https://leafletjs.com/examples/layers-control/)** page, we can add data to a LayerGroup class. In the example given, the `cities` variable is assigned to the `layerGroup()`. For our purposes, we'll use the earthquake data:




Let's create an overlay layer for our earthquake data. Add the following code to your `logicStep4.js` file below the code for the base layer that holds the two different map styles:

```
// Create the earthquake layer for our map.
let earthquakes = new L.layerGroup();
```

Next, define the overlay object to add it to the map. Add the following code below the earthquake layer group:

```
// We define an object that contains the overlays.
// This overlay will be visible all the time.
let overlays = {
  Earthquakes: earthquakes
};
```

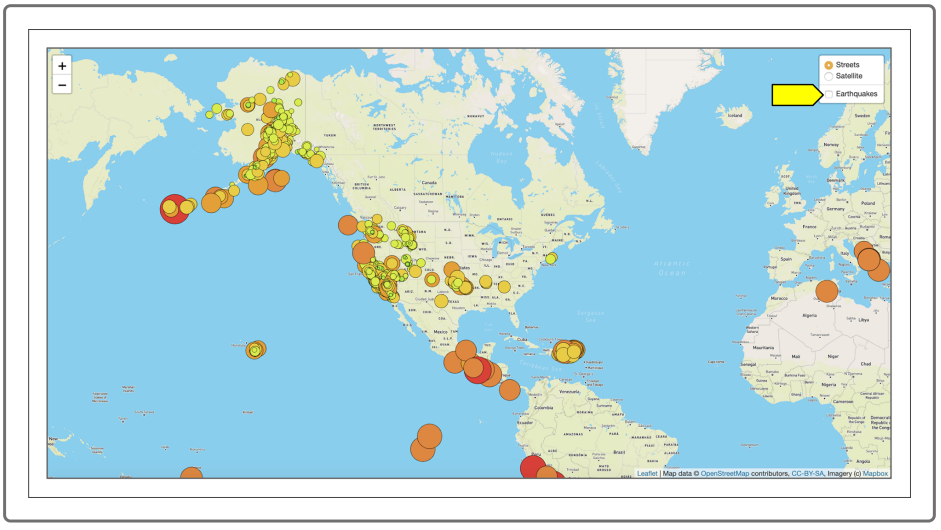
To add the overlay to the map, add the variable `overlays` to the Layers Control object. Edit the Layers Control object so that the `overlays` object will show up on the tile layers control:

```
// Then we add a control to the map that will allow the user to change
// which layers are visible.
L.control.layers(baseMaps, overlays).addTo(map);
```

Your `logicStep4.js` file should look like the following with the added code:

```
// Create a base layer that holds both maps.
let baseMaps = {
  "Streets": streets,
  "Satellite": satelliteStreets
};

// Create the earthquake layer for our map.
let earthquakes = new L.LayerGroup();

// We define an object that contains the overlays.
// This overlay will be visible all the time.
let overlays = {
  Earthquakes: earthquakes
};

// Then we add a control to the map that will allow the user to change which
// layers are visible.
L.control.layers(baseMaps, overlays).addTo(map);

```

If we open the `index.html` file in our browser, we see that the earthquake data has loaded, but the earthquake overlay button is not on:




Our `L.geoJSON()` layer code looks like this at this point:

```
// Creating a GeoJSON layer with the retrieved data.
  L.geoJson(data, {
      // We turn each feature into a circleMarker on the map.
      pointToLayer: function(feature, latlng) {
          console.log(data);
          return L.circleMarker(latlng);
        },
      // We set the style for each circleMarker using our styleInfo function.
    style: styleInfo,
      // We create a popup for each circleMarker to display the magnitude and location of the earthquake
      //  after the marker has been created and styled.
      onEachFeature: function(feature, layer) {
        layer.bindPopup("Magnitude: " + feature.properties.mag + "<br>Location: " + feature.properties.place);
      }
  }).addTo(map);

});
```

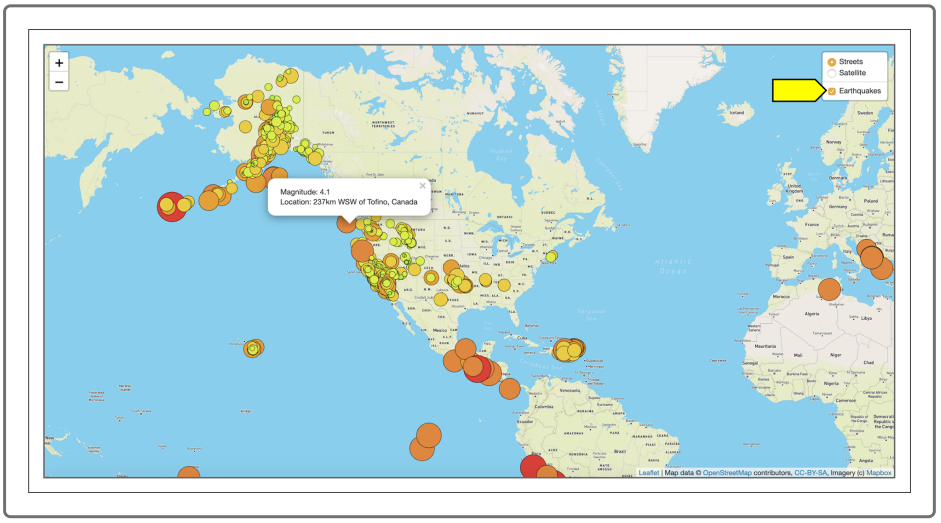
To have the Earthquakes overlay button "on," we need to:

- Replace the `map` variable in the `addTo(map)` function with `earthquakes`.
- Before the closing bracket and parenthesis of the `d3.json()` method we add the earthquake layer to the map, with `earthquakes.addTo(map);`.

Edit your `addTo(map)` function at the end of your `L.geoJSON()` layer code, as shown in the image to look like the following:

```
// Creating a GeoJSON layer with the retrieved data.
  L.geoJson(data, {
      // We turn each feature into a circleMarker on the map.
      pointToLayer: function(feature, latlng) {
          console.log(data);
          return L.circleMarker(latlng);
        },
      // We set the style for each circleMarker using our styleInfo function.
    style: styleInfo,
      // We create a popup for each circleMarker to display the magnitude and location of the earthquake
      //  after the marker has been created and styled.
      onEachFeature: function(feature, layer) {
        layer.bindPopup("Magnitude: " + feature.properties.mag + "<br>Location: " + feature.properties.place);
      }
  }).addTo(earthquakes);

    // Then we add the earthquake layer to our map.
    earthquakes.addTo(map);
});
```

Now, when we open the `index.html` file in our browser, we can see that the earthquake data has loaded *and* the earthquake overlay button is "on":




Nice job adding the earthquake data as an overlay to the map!

### ADD/COMMIT/PUSH

Add, commit, and push your changes to your Earthquakes_past7days branch.

Sadhana loves the map, but she thinks having a legend to indicate what magnitude is represented by each color would be helpful when viewing the map.