**MAPPING EARTHQUAKES**

**Introduction:**

We enjoy how the seismic overlay and two separate maps we used to generate the earthquake map turned out. Now, they wanted to view all of the earthquakes with a magnitude greater than 4.5 on the map and the data on a third map, while we wanted to see the earthquake data in relation to the positions of the tectonic plates on the world.

**Objectives:**

We have the following objectives or goals to cover in this assignment which are as follow.

* Adding a Tectonic plate data.
* Adding a major earthquake data.
* Adding an Additional map.

**Codes:**

Writing a following code in order to creating a workspace for our deliverables procedure.

|  |
| --- |
| <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <meta http-equiv="X-UA-Compatible" content="IE=edge"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <title>Leaflet-Basic-Map</title> |
|  |  |
|  | <!-- Leaflet CSS --> |
|  | <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" |
|  | integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A==" |
|  | crossorigin=""/> |
|  |  |
|  | <!-- d3 JavaScript --> |
|  | <script src="https://d3js.org/d3.v5.min.js"></script> |
|  |  |
|  | <!-- Our CSS --> |
|  | <link rel="stylesheet" type="text/css" href="static/css/style.css"> |
|  |  |
|  | </head> |
|  | <body> |
|  | <!-- The div that holds our map --> |
|  | <div id="mapid"></div> |
|  |  |
|  | <!-- Leaflet JavaScript --> |
|  | <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js" |
|  | integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA==" |
|  | crossorigin=""></script> |
|  | <!-- API key --> |
|  | <script type="text/javascript" src="static/js/config.js"></script> |
|  | <!-- Our JavaScript --> |
|  | <script type="text/javascript" src="static/js/cities.js"></script> |
|  | <script type="text/javascript" src="static/js/challenge\_logic.js"></script> |
|  | </body> |
|  | </html> |

**Code for Style:**

|  |
| --- |
| html, |
|  | body, |
|  | #mapid { |
|  | width: 100%; |
|  | height: 100%; |
|  | padding: 0; |
|  | margin: 0; |
|  | } |
|  | .legend { |
|  | padding: 10px; |
|  | line-height: 18px; |
|  | color: #555; |
|  | background-color: #fff; |
|  | border-radius: 5px; |
|  | } |
|  | .legend i { |
|  | width: 18px; |
|  | height: 18px; |
|  | float: left; |
|  | margin-right: 8px; |
|  | opacity: 0.7; |
|  | } |

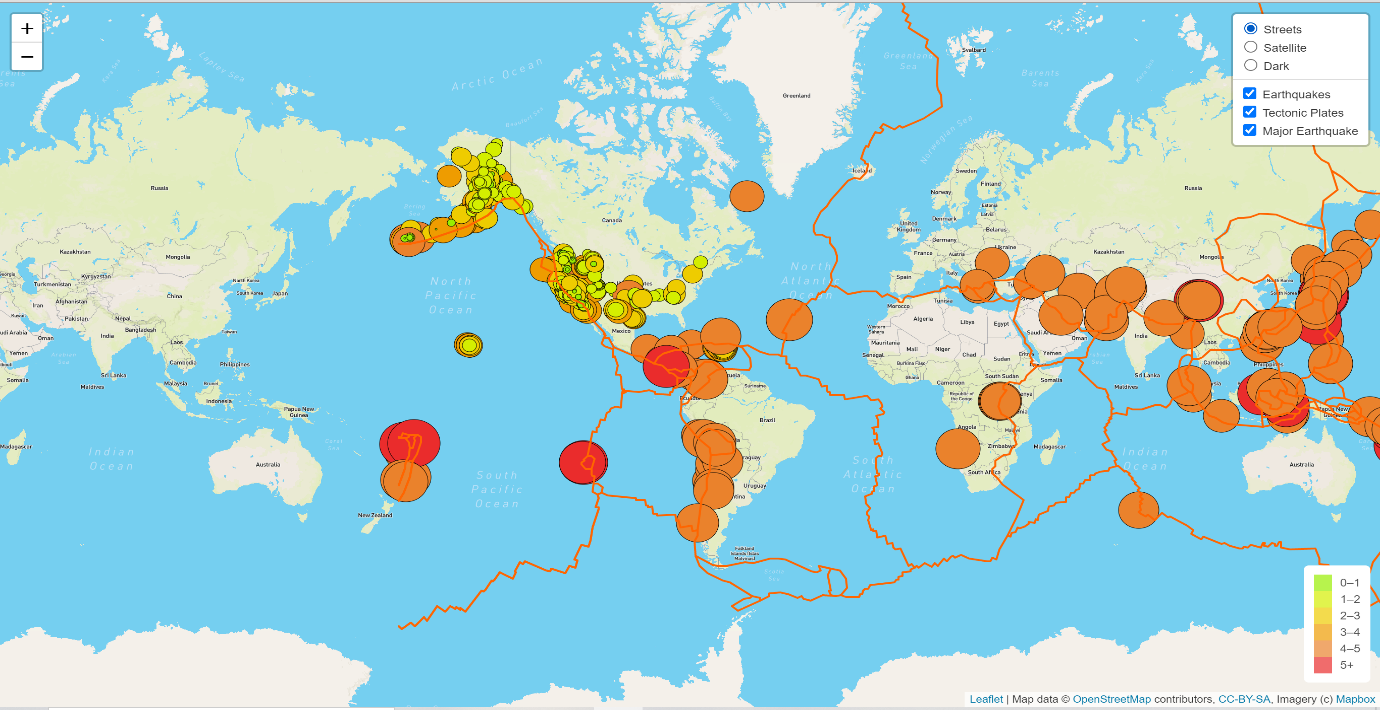
**Code for Creating a Logic:**

|  |
| --- |
| / Add console.log to check to see if our code is working. |
|  | console.log("working"); |
|  |  |
|  | // We create the tile layer that will be the background of our map. |
|  | let streets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/{z}/{x}/{y}?access\_token={accessToken}', { |
|  | attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery (c) <a href="https://www.mapbox.com/">Mapbox</a>', |
|  | maxZoom: 18, |
|  | accessToken: API\_KEY |
|  | }); |
|  |  |
|  | // We create the second tile layer that will be the background of our map. |
|  | let satelliteStreets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/satellite-streets-v11/tiles/{z}/{x}/{y}?access\_token={accessToken}', { |
|  | attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery (c) <a href="https://www.mapbox.com/">Mapbox</a>', |
|  | maxZoom: 18, |
|  | accessToken: API\_KEY |
|  | }); |
|  |  |
|  | // We create the second tile layer that will be the background of our map. |
|  | let dark = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/dark-v10/tiles/{z}/{x}/{y}?access\_token={accessToken}', { |
|  | attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery (c) <a href="https://www.mapbox.com/">Mapbox</a>', |
|  | maxZoom: 18, |
|  | accessToken: API\_KEY |
|  | }); |
|  |  |
|  |  |
|  | // Create the map object with center, zoom level and default layer. |
|  | let map = L.map('mapid', { |
|  | center: [40.7, -94.5], |
|  | zoom: 4, |
|  | layers: [streets] |
|  | }); |
|  |  |
|  | // Create a base layer that holds all three maps. |
|  | let baseMaps = { |
|  | "Streets": streets, |
|  | "Satellite": satelliteStreets, |
|  | "Dark": dark |
|  | }; |
|  |  |
|  | // 1. Add a 3rd layer group for the major earthquake data. |
|  | let allEarthquakes = new L.LayerGroup(); |
|  | let tectonicPlates = new L.LayerGroup(); |
|  | let majorEarthquake = new L.LayerGroup(); |
|  |  |
|  |  |
|  | // 2. Add a reference to the major earthquake group to the overlays object. |
|  | let overlays = { |
|  | "Earthquakes": allEarthquakes, |
|  | "Tectonic Plates": tectonicPlates, |
|  | "Major Earthquake": majorEarthquake |
|  | }; |
|  |  |
|  | // 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); |
|  |  |
|  | // Retrieve the earthquake GeoJSON data. |
|  | d3.json("https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all\_week.geojson").then(function(data) { |
|  |  |
|  | // This function returns the style data for each of the earthquakes we plot on |
|  | // the map. We pass the magnitude of the earthquake into two separate functions |
|  | // to calculate the color and radius. |
|  | function styleInfo(feature) { |
|  | return { |
|  | opacity: 1, |
|  | fillOpacity: 1, |
|  | fillColor: getColor(feature.properties.mag), |
|  | color: "#000000", |
|  | radius: getRadius(feature.properties.mag), |
|  | stroke: true, |
|  | weight: 0.5 |
|  | }; |
|  | } |
|  |  |
|  | // This function determines the color of the marker based on the magnitude of the earthquake. |
|  | function getColor(magnitude) { |
|  | if (magnitude > 5) { |
|  | return "#ea2c2c"; |
|  | } |
|  | if (magnitude > 4) { |
|  | return "#ea822c"; |
|  | } |
|  | if (magnitude > 3) { |
|  | return "#ee9c00"; |
|  | } |
|  | if (magnitude > 2) { |
|  | return "#eecc00"; |
|  | } |
|  | if (magnitude > 1) { |
|  | return "#d4ee00"; |
|  | } |
|  | return "#98ee00"; |
|  | } |
|  |  |
|  | // This function determines the radius of the earthquake marker based on its magnitude. |
|  | // Earthquakes with a magnitude of 0 were being plotted with the wrong radius. |
|  | function getRadius(magnitude) { |
|  | if (magnitude === 0) { |
|  | return 1; |
|  | } |
|  | return magnitude \* 4; |
|  | } |
|  |  |
|  | // 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(allEarthquakes); |
|  |  |
|  | // Then we add the earthquake layer to our map. |
|  | allEarthquakes.addTo(map); |
|  |  |
|  | // 3. Retrieve the major earthquake GeoJSON data >4.5 mag for the week. |
|  | d3.json("https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5\_week.geojson").then(function(data) { |
|  |  |
|  | // 4. Use the same style as the earthquake data. |
|  | function styleInfo(feature) { |
|  | return { |
|  | opacity: 1, |
|  | fillOpacity: 1, |
|  | fillColor: getColor(feature.properties.mag), |
|  | color: "#000000", |
|  | radius: getRadius(feature.properties.mag), |
|  | stroke: true, |
|  | weight: 0.5 |
|  | }; |
|  | } |
|  | // 5. Change the color function to use three colors for the major earthquakes based on the magnitude of the earthquake. |
|  | function getColor(magnitude) { |
|  | if (magnitude > 5) { |
|  | return "#ea2c2c"; |
|  | } |
|  | if (magnitude > 4) { |
|  | return "#ea822c"; |
|  | } |
|  | return "#98ee00"; |
|  | } |
|  |  |
|  | // 6. Use the function that determines the radius of the earthquake marker based on its magnitude. |
|  | function getRadius(magnitude) { |
|  | if (magnitude === 0) { |
|  | return 1; |
|  | } |
|  | return magnitude \* 4; |
|  | } |
|  |  |
|  | // 7. Creating a GeoJSON layer with the retrieved data that adds a circle to the map |
|  | // sets the style of the circle, and displays the magnitude and location of the earthquake |
|  | // after the marker has been created and styled. |
|  | L.geoJson(data, { |
|  | pointToLayer: function(feature, latlng) { |
|  | return L.circleMarker(latlng); |
|  | }, |
|  | style: styleInfo, |
|  | onEachFeature: function(feature, layer) { |
|  | layer.bindPopup("Magnitude: " + feature.properties.mag + "<br>Location: " + feature.properties.place); |
|  | } |
|  | }).addTo(majorEarthquake); |
|  | // 8. Add the major earthquakes layer to the map. |
|  | majorEarthquake.addTo(map); |
|  | // 9. Close the braces and parentheses for the major earthquake data. |
|  | }); |
|  |  |
|  | // Here we create a legend control object. |
|  | let legend = L.control({ |
|  | position: "bottomright" |
|  |  |
|  | }); |
|  |  |
|  | // Then add all the details for the legend |
|  | legend.onAdd = function() { |
|  | let div = L.DomUtil.create("div", "info legend"); |
|  |  |
|  | const magnitudes = [0, 1, 2, 3, 4, 5]; |
|  | const colors = [ |
|  | "#98ee00", |
|  | "#d4ee00", |
|  | "#eecc00", |
|  | "#ee9c00", |
|  | "#ea822c", |
|  | "#ea2c2c" |
|  | ]; |
|  |  |
|  | // Looping through our intervals to generate a label with a colored square for each interval. |
|  | for (var i = 0; i < magnitudes.length; i++) { |
|  | console.log(colors[i]); |
|  | div.innerHTML += |
|  | "<i style='background: " + colors[i] + "'></i> " + |
|  | magnitudes[i] + (magnitudes[i + 1] ? "&ndash;" + magnitudes[i + 1] + "<br>" : "+"); |
|  | } |
|  | return div; |
|  | }; |
|  |  |
|  | // Finally, we our legend to the map. |
|  | legend.addTo(map); |
|  |  |
|  |  |
|  | // Use d3.json to make a call to get our Tectonic Plate geoJSON data. |
|  | d3.json("https://raw.githubusercontent.com/fraxen/tectonicplates/master/GeoJSON/PB2002\_boundaries.json").then((data) => { |
|  | console.log(data); |
|  | L.geoJson(data, { |
|  | style: {color: "#ff6500", weight: 2}, |
|  | }).addTo(tectonicPlates) |
|  | }); |
|  | tectonicPlates.addTo(map); |
|  | }); |

**Creating an Array for cities location with respect to population**

|  |
| --- |
|  |
|  | let cities = [{ |
|  | location: [40.7128, -74.0059], |
|  | city: "New York City", |
|  | state: "NY", |
|  | population: 8398748 |
|  | }, |
|  | { |
|  | location: [41.8781, -87.6298], |
|  | city: "Chicago", |
|  | state: "IL", |
|  | population: 2705994 |
|  | }, |
|  | { |
|  | location: [29.7604, -95.3698], |
|  | city: "Houston", |
|  | state: "TX", |
|  | population: 2325502 |
|  | }, |
|  | { |
|  | location: [34.0522, -118.2437], |
|  | city: "Los Angeles", |
|  | state: "CA", |
|  | population: 3990456 |
|  | }, |
|  | { |
|  | location: [33.4484, -112.0740], |
|  | city: "Phoenix", |
|  | state: "AZ", |
|  | population: 1660272 |
|  | } |
|  | ]; |

**Results:**

****

**Street Map View**

**Map

Description automatically generated**

**Satellite View**

**Map

Description automatically generated**

**Dark Map View**

**Conclusions:**

In this Assignment we are required to perform three deliverables with their code. We have successfully performed all those tasks and now we are fully aware of this assignment and concerns and problems related to this assignment are now addressed in Documentation as well as coding files.