





I would like to write a code in svelte that does the following: 1) Sum up all the purchases for a specific CustomerKey in a file called sales CSV 2) Then match that CustomerKey to the CustomerCity and determine the total sales in each city through the CustomersCSV 3) Then add the longitude and latitude of each CustomerCity using the extractedCityLocationsCSV 4) Finally visualise the data on a map using the latitude and longitude of each city and a SVG circle radius to represent the amount of sales in that city Attached is the pictures of each CSV example. Please explain how to construct this in svelte, step by step

Temp Code:

<script>

    import { onMount } from 'svelte';

    import { csv } from 'd3-fetch';

    let totalSalesByCustomer = [];

    let totalSalesByCity = [];

    let citySalesWithLocation = [];

        // Define SVG dimensions

        const width = 800; // Adjust as needed

        const height = 500; // Adjust as needed

    // Modify these scaling functions based on your actual data range and desired appearance

    function scaleLongitude(longitude) {

        return ((longitude + 180) % 360) \* (width / 360); // Wraps the longitude correctly within the SVG width

    }

    function scaleLatitude(latitude) {

        const scaledLatitude = 90 - latitude; // Convert latitude to positive up as in Cartesian coordinates

        return scaledLatitude \* (height / 180); // Scale latitude to fit within the SVG height

    }

    function scaleSalesToRadius(sales) {

        const maxRadius = 50; // Maximum radius of a circle

        const scaledRadius = Math.sqrt(sales) \* 0.1;

        return Math.min(scaledRadius, maxRadius);

    }

    function getColorBySales(sales) {

        if (sales > 100000) return '#003366'; // Dark Blue

        if (sales > 50000) return '#0066CC'; // Medium Blue

        if (sales > 10000) return '#6699FF'; // Light Blue

        return '#CCDDFF'; // Very Light Blue

    }

    async function loadData() {

        const salesData = await csv('data/Sales.csv');

        const customersData = await csv('data/Customers.csv');

        const cityLocationsData = await csv('data/extractedCityLocations.csv');

        // Assuming country data is also in the cityLocationsData or customersData

        let cityInfo = cityLocationsData.reduce((acc, loc) => {

            acc[loc.CustomerCity] = { latitude: loc.Latitude, longitude: loc.Longitude, country: loc.CustomerCountry };

            return acc;

        }, {});

        citySalesWithLocation = customersData.map(customer => {

            let city = customer.CustomerCity;

            let sales = cityInfo[city] ? cityInfo[city].sales : 0;

            let country = cityInfo[city] ? cityInfo[city].country : 'Unknown';

            return {

                city: city,

                sales: sales,

                latitude: cityInfo[city] ? parseFloat(cityInfo[city].latitude) : 0,

                longitude: cityInfo[city] ? parseFloat(cityInfo[city].longitude) : 0,

                country: country

            };

        });

        console.log('City Sales with Location:', citySalesWithLocation);

    }

    onMount(() => {

        loadData();

    });

</script>

<style>

    svg {

        width: 100%;

        height: 500px;

        border: 1px solid #ccc;

    }

</style>

<svg width={width} height={height}>

    {#each citySalesWithLocation as { city, sales, latitude, longitude, country }}

        <circle cx={scaleLongitude(longitude)} cy={scaleLatitude(latitude)}

                r={scaleSalesToRadius(sales)} fill={getColorBySales(sales)}

                stroke="black" stroke-width="1">

            <title>{city}, {country}: {sales}</title>

        </circle>

    {/each}

</svg>

tempWorking:

<h1>Relative Sales for each city</h1>

<script>

import { onMount } from 'svelte';

import { csv } from 'd3-fetch';

let citySalesWithLocation = [];

async function loadData() {

const salesData = await csv('data/Sales.csv');

const customersData = await csv('data/Customers.csv');

const cityLocationsData = await csv('data/extractedCityLocations.csv');

// Prepare sales data by customer key

const totalSalesByCustomer = salesData.reduce((acc, sale) => {

let key = sale.CustomerKey;

let quantity = parseFloat(sale.OrderQuantity);

acc[key] = (acc[key] || 0) + quantity;

return acc;

}, {});

// Prepare city information with sales data

const cityInfo = customersData.reduce((acc, customer) => {

let key = customer.CustomerKey;

let city = customer.CustomerCity;

let country = customer.CustomerCountry; // Assuming country is in Customers.csv

if (!acc[city]) acc[city] = { sales: 0, country: country, customers: [] };

if (totalSalesByCustomer[key]) {

acc[city].sales += totalSalesByCustomer[key];

}

if (!acc[city].customers.includes(key)) {

acc[city].customers.push(key);

}

return acc;

}, {});

// Combine city sales data with location

citySalesWithLocation = cityLocationsData.map(location => {

let city = location.CustomerCity;

return {

city: city,

sales: cityInfo[city] ? cityInfo[city].sales : 0,

latitude: parseFloat(location.Latitude),

longitude: parseFloat(location.Longitude),

country: cityInfo[city] ? cityInfo[city].country : 'Unknown'

};

});

console.log('City Sales with Location:', citySalesWithLocation);

}

onMount(() => {

loadData();

});

// Dimensions of the SVG

const width = 10000;

const height = 2000;

// Helper functions for scaling and coloring

function scaleLongitude(longitude) {

return ((longitude + 5) % 360) \* (width / 360);

}

function scaleLatitude(latitude) {

const scaledLatitude = 80 - latitude;

return scaledLatitude \* (height / 180);

}

function scaleSalesToRadius(sales) {

const maxRadius = 50;

const scaledRadius = Math.sqrt(sales) \* 0.08;

return Math.min(scaledRadius, maxRadius);

}

function getColorBySales(sales) {

if (sales > 100000) return '#003366';

if (sales > 50000) return '#0066CC';

if (sales > 10000) return '#6699FF';

return '#CCDDFF';

}

</script>

<style>

svg {

width: 100%;

height: 500px;

border: 1px solid #ccc;

}

</style>

<svg width={width} height={height}>

{#each citySalesWithLocation as { city, sales, latitude, longitude, country }}

<circle cx={scaleLongitude(longitude)} cy={scaleLatitude(latitude)}

r={scaleSalesToRadius(sales)} fill={getColorBySales(sales)}

stroke="black" stroke-width="1">

<title>{city}, {country}: {sales}</title>

</circle>

{/each}

</svg>