Problem Sheet 1

Here is the solution to 1.1:

```html
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>CSV Data Visualization</title>
<link rel="stylesheet" href="styles.css"> <!-- Link to the CSS file -->
<script src="https://d3js.org/d3.v7.min.js"></script> <!-- Include D3.js -->
</head>
<body>
<h1>CSV Data Visualization</h1>
<div id="content">
<!-- Content generated by JavaScript will go here -->
</div>

<!-- SVG Container -->
<svg id="svg-container" width="700" height="550"></svg>

<!-- Link to your JavaScript file -->
<script type="module" src="index.js"></script>
</body>
</html>
```

And here is the associating `javascript` file:

```javascript
let storedData = []; // Variable to store the parsed data


function processData(parsedData) {
        console.log("Parsed Data:", parsedData); // Log the parsed data to
inspect its structure

        // Store the parsed data for later use
        storedData = parsedData;

        // Filter to get only EU cities
        const euCities = parsedData.filter(row => row.eu === 'true');
```

```
  // Select the SVG container
        const svg = d3.select("#svg-container");

        // Define scales for positioning circles
        const xScale = d3.scaleLinear()
                .domain([0, d3.max(euCities, d => +d.population)]) // Use
lowercase 'population'
                .range([50, 650]); // Padding for aesthetics

        const yScale = d3.scaleBand()
                .domain(euCities.map(d => d.city)) // Use lowercase 'city'
                .range([50, 500])
                .padding(0.1);

        // Bind data and create circles
        svg.selectAll("circle")
                .data(euCities)
                .enter()
                .append("circle")
                .attr("cx", d => xScale(+d.population)) // Use lowercase
'population'
                .attr("cy", d => yScale(d.city) + yScale.bandwidth() / 2) //
Use lowercase 'city'
                .attr("r", d => +d.population > 1000000 ? 8 : 4) //
Conditional radius based on population
                .attr("fill", "steelblue");

// Add text labels
        svg.selectAll("text")
                .data(euCities)
                .enter()
                .append("text")
                .attr("x", d => xScale(+d.population))
                .attr("y", d => yScale(d.city) + yScale.bandwidth() / 2 -
10) // Position text above the circle
                .attr("text-anchor", "middle")
                .attr("font-size", "12px")
                .attr("font-family", "sans-serif") // .attr("padding",
"5px")
                .attr("fill", "black")
                .attr("opacity", d => +d.population > 1000000 ? 1 : 0) //
Show only for population > 1,000,000
                .text(d => d.city);

        // Log each city and its population
        euCities.forEach(row => {
                console.log(`City: ${row.city}, Population:
${row.population}`); // Use lowercase
        });

        // Calculate total population of EU cities
        const totalPopulation = euCities.reduce((sum, row) => sum +
Number(row.population), 0); // Use lowercase
        console.log(`Total Population of EU Cities: ${totalPopulation}`);

}

// Fetch the CSV file using D3's csv function
d3.csv('./data/cities_and_population.csv').then(parsedData => {
        processData(parsedData);
        }).catch(error => {
        console.error("Error loading the CSV file:", error);
});
```

Here are the solutions to 1.2:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bar Chart</title>
    <link rel="stylesheet" href="styles2.css">
    <script src="https://d3js.org/d3.v7.min.js"></script>
  </head>
  <body>
    <svg width="960" height="500"></svg>
    <script type="module" src="index_2.js"></script>
  </body>
</html>
```

and here is the `javascript` file for the associating webpage:

```
const svg = d3.select('svg');
const width  = +svg.attr('width');
const height = +svg.attr('height');
```

```
d3.csv('./data/countries.csv').then(data => {     // Data loading
  data.forEach(d => {
    d.population = +d.population * 1000;    // Data parsing
  });
  render(data);                             // Data rendering (calls
'render')
});

const render = data => {
  const margin = { top: 60, bottom: 50, left: 100, right: 40 }; // Adjusted
top margin for title
  const innerWidth = width - margin.left - margin.right;
  const innerHeight = height - margin.top - margin.bottom;

  const g = svg                             // append a new group, for inner
margins
    .append('g')
    .attr('transform', `translate(${margin.left},${margin.top})`);

  const xValue = d => d.population;         // data accessors
  const yValue = d => d.country;            // (making rest of code easier to
adapt)

  const xScale = d3.scaleLinear()           // scale to set bar widths
    .domain([0, d3.max(data, xValue)])
    .range([0, innerWidth]);

  const yScale = d3.scaleBand()             // scale to set bar heights
    .domain(data.map(yValue))
    .range([0, innerHeight])
    .paddingInner(0.1);

  // Add our chart elements (bars)
  g
    .selectAll('rect')                      // select all existing rectangles
(none)
    .data(data).enter()                     // create data join
    .append('rect')                         // append one rectangle per
element
      .attr('class','bar')
      .attr('width', d => xScale(xValue(d)))
      .attr('height', yScale.bandwidth())
      .attr('y', d => yScale(yValue(d)));

  // Create y-axis
  const yAxis = g.append('g').call(d3.axisLeft(yScale));
  // Remove unnecessary tick lines from the y-axis
  yAxis.selectAll('.domain, .tick line').remove();

  // Custom x-axis tick format
  const xAxisTickFormat = number =>
    d3.format('.3s')(number)
      .replace('G','B')
      .replace('0.00','0');

  // Define x-axis with custom tick format and tick size
  const xAxis = d3.axisBottom(xScale)
    .tickFormat(xAxisTickFormat)
    .tickSize(-innerHeight);

  // Create x-axis group
  const xAxisG = g.append('g')
    .call(xAxis)
    .attr('transform', `translate(0,${innerHeight})`);

  // Remove the domain line from the x-axis
  xAxisG.select('.domain').remove();

  // Append a label to the x-axis
  xAxisG.append('text')
    .attr('class', 'axis-label')
    .attr('y', 40) // Position below the axis
    .attr('x', innerWidth / 2)
    .attr('fill', 'black')
    .text('Population')
    .style('text-anchor', 'middle');

  // Append a title to the chart
  svg.append('text')
    .attr('class', 'title')
    .attr('x', width / 2)
    .attr('y', 30) // Position at the top of the SVG
    .attr('text-anchor', 'middle')
    .attr('fill', 'black')
    .text('Top 10 Most Populous Countries');
};
```

1.3: Here is the html page for the following code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cars Horsepower vs Weight</title> <!-- Replace with your desired
title -->
    <link rel="stylesheet" href="styles3.css"> <!-- Link to your CSS file --
>
    <script src="https://d3js.org/d3.v7.min.js"></script> <!-- Include D3.js
-->
</head>
<body>
    <div id="content">
        <!-- Content generated by JavaScript will go here -->
    </div>
    <svg id="svg-container" width="960" height="580"></svg> <!-- SVG
container for D3 visualizations -->
    <script type="module" src="index_3.js"></script> <!-- Link to your
JavaScript file -->
</body>
</html>
```

```javascript
// Select the SVG container
const svg = d3.select('#svg-container');
const width = +svg.attr('width');
const height = +svg.attr('height');

// Load the CSV data
d3.csv('./data/auto-mpg.csv').then(data => {
    // Parse the data
    data.forEach(d => {
        d.horsepower = +d.horsepower; // Convert 'horsepower' to a number
        d.weight = +d.weight; // Convert 'weight' to a number
    });

    // Call the render function with the parsed data
    render(data);
}).catch(error => {
    console.error("Error loading the CSV file:", error);
});

// Function to render the scatter plot
const render = data => {
    // Set margins and dimensions
    const margin = { top: 20, right: 20, bottom: 50, left: 70 }; //
Increased bottom and left margins
    const innerWidth = width - margin.left - margin.right;
    const innerHeight = height - margin.top - margin.bottom;

    // Create scales
    const xScale = d3.scaleLinear()
        .domain([40, d3.max(data, d => d.horsepower)]) // X domain from 40
to max horsepower
        .range([0, innerWidth]);

    const yScale = d3.scaleLinear()
        .domain([d3.max(data, d => d.weight), 1500]) // Inverted Y domain
from max weight to 1500
        .range([innerHeight, 0]); // Y scale goes from top (0) to bottom
(innerHeight)

    // Create a group for the chart
    const g = svg.append('g')
        .attr('transform', `translate(${margin.left},${margin.top})`);

    // Create circles for the scatter plot
    g.selectAll('circle')
        .data(data)
        .enter()
        .append('circle')
        .attr('cx', d => xScale(d.horsepower))
        .attr('cy', d => yScale(d.weight))
        .attr('r', 5) // Radius of the circles
        .attr('fill', 'steelblue')
        .attr('opacity', 0.6);

    // Create X axis with grid lines
    const xAxis = d3.axisBottom(xScale)
        .tickSize(-innerHeight); // Extend tick lines across the height of
the chart

    const xAxisGroup = g.append('g')
        .attr('transform', `translate(0,${innerHeight})`)
        .call(xAxis)
        .call(g => g.selectAll('.domain').remove()); // Remove the baseline

    // Create Y axis with grid lines
    const yAxis = d3.axisLeft(yScale)
        .tickSize(-innerWidth); // Extend tick lines across the width of the
chart
```

```
    const yAxisGroup = g.append('g')
        .call(yAxis)
        .call(g => g.selectAll('.domain').remove()); // Remove the baseline

    // Set opacity for grid lines
    xAxisGroup.selectAll('line').attr('opacity', 0.8);
    yAxisGroup.selectAll('line').attr('opacity', 0.8);

    // Append labels to axes with increased font size and sans-serif font
    g.append('text')
        .attr('class', 'axis-label')
        .attr('x', innerWidth / 2)
        .attr('y', innerHeight + 40) // Increased y position for better
visibility
        .attr('text-anchor', 'middle')
        .style('font-size', '14px') // Increase font size
        .style('font-family', 'sans-serif') // Set font to sans-serif
        .attr('opacity', 0.8) // Set opacity for the label
        .text('Horsepower');

    g.append('text')
        .attr('class', 'axis-label')
        .attr('transform', 'rotate(-90)')
        .attr('y', -50) // Increased y position for better visibility
        .attr('x', -innerHeight / 2)
        .attr('text-anchor', 'middle')
        .style('font-size', '14px') // Increase font size
        .style('font-family', 'sans-serif') // Set font to sans-serif
        .attr('opacity', 0.8) // Set opacity for the label
        .text('Weight');

    // Add a semi-transparent background rectangle for better visibility
    g.append('rect')
        .attr('width', innerWidth)
        .attr('height', innerHeight)
        .attr('fill', 'white')
        .attr('opacity', 0.5) // Set opacity for the background rectangle
        .attr('transform', `translate(0,0)`);
};
```

1.4: Here is the code for the fourth exercise:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>S and P 500 Chart</title>
    <link rel="stylesheet" href="styles.css"> <!-- Link to the CSS file -->
    <script src="https://d3js.org/d3.v7.min.js"></script> <!-- Include D3.js
-->
</head>
<body>
    <div id="content">
        <!-- Content generated by JavaScript will go here -->
    </div>

    <!-- SVG Container -->
    <svg id="svg-container" width="700" height="550"></svg>

    <!-- Link to your JavaScript file -->
    <script type="module" src="index.js"></script>
</body>
</html>
```

and here is the `javascript` file for the following code:

```javascript
// Select the SVG container
const svg = d3.select('#svg-container');
const width = +svg.attr('width');
const height = +svg.attr('height');

// Load the CSV data
d3.csv('./data/sp-500-index.csv').then(data => {
    // Parse the data
    data.forEach(d => {
        d.date = new Date(d.date); // Convert 'date' to a Date object
        d.close = +d.close; // Convert 'close' to a number
    });

    // Call the render function with the parsed data
    render(data);
}).catch(error => {
    console.error("Error loading the CSV file:", error);
});

// Function to render the line chart
```

```
const render = data => {
    // Set margins and dimensions
    const margin = { top: 20, right: 20, bottom: 50, left: 70 };
    const innerWidth = width - margin.left - margin.right;
    const innerHeight = height - margin.top - margin.bottom;

    // Create scales
    const xScale = d3.scaleTime()
        .domain(d3.extent(data, d => d.date)) // X domain from min to max
date
        .range([0, innerWidth]);

    const yScale = d3.scaleLinear()
        .domain([0, d3.max(data, d => d.close)]) // Y domain from 0 to max
close value
        .range([innerHeight, 0]); // Y scale goes from top (0) to bottom
(innerHeight)

    // Create a group for the chart
    const g = svg.append('g')
        .attr('transform', `translate(${margin.left},${margin.top})`);

    // Add title
    svg.append('text')
        .attr('x', width / 2)
        .attr('y', margin.top)
        .attr('text-anchor', 'middle')
        .style('font-size', '20px') // Set font size for the title
        .style('font-family', 'sans-serif') // Set font to sans-serif
        .text('S&P 500 Index');

    // Create the line generator
    const line = d3.line()
        .x(d => xScale(d.date))
        .y(d => yScale(d.close));

    // Append the area under the line
    g.append('path')
        .datum(data)
        .attr('fill', 'lightblue') // Background color for the area
        .attr('opacity', 0.3) // Set opacity for the area
        .attr('d', d3.area()
            .x(d => xScale(d.date))
            .y0(innerHeight) // Start from the bottom of the chart
            .y1(d => yScale(d.close))
        );
    // Append the line path to the SVG
    g.append('path')
        .datum(data)
        .attr('fill', 'none')
        .attr('stroke', '#2c3e50') // Darker stroke color
        .attr('stroke-width', 1.5)
        .attr('d', line);

    // Create X axis
    const xAxis = d3.axisBottom(xScale);
    g.append('g')
        .attr('transform', `translate(0,${innerHeight})`)
        .call(xAxis);

    // Create Y axis
    const yAxis = d3.axisLeft(yScale);
    g.append('g')
        .call(yAxis);

    // Append labels to axes
    g.append('text')
        .attr('class', 'axis-label')
        .attr('x', innerWidth / 2)
        .attr('y', innerHeight + 40)
        .attr('text-anchor', 'middle')
        .text('Date');

    g.append('text')
        .attr('class', 'axis-label')
        .attr('transform', 'rotate(-90)')
        .attr('y', -50)
        .attr('x', -innerHeight / 2)
        .attr('text-anchor', 'middle')
        .text('S&P 500 Close Price');
};
```