lab-3.1\index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="description" content="D3 Scatter Plot Exercise">
    <meta name="keywords" content="HTML, D3, JavaScript, SVG, Scatter Plot">
    <meta name="author" content="Joe Bloggs">
    <title>Drawing with Data - Scatter Plot</title>
    <!-- Step 1: Include the D3.js library -->
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        /* Optional: Add some basic styling for the chart container */
        body {
            font-family: sans-serif;
            text-align: center;
        }

        .chart-container {
            margin: 20px auto;
            border: 1px solid #ccc;
            display: inline-block;
            /* To make the container fit the SVG */
        }

        h1,
        footer {
            color: #333;
        }
    </style>
</head>

<body>

    <h1>Drawing with Data - Scatter Plot</h1>

    <div class="chart-container">
        <!-- The SVG will be created here by D3 -->
    </div>

    <script>
        // --- Configuration Variables ---

        // Define the dimensions of the SVG canvas
        const w = 500;
        const h = 100;
        const padding = 30; // Add padding to prevent circles from being cut off

        // Step 2: Define the new dataset for the scatter plot
        // Each inner array: [x_coordinate, y_coordinate, radius_size (optional)]
        const dataset = [
            [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
            [410, 12], [475, 44], [25, 67], [85, 21], [220, 88],
            [600, 150] // Outlier
        ];

        // --- D3 Code ---

        //Create scale functions
        const xScale = d3.scaleLinear()
            .domain([0, d3.max(dataset, function (d) { return d[0]; })])
            .range([padding, w - padding]);

        const yScale = d3.scaleLinear()
            .domain([0, d3.max(dataset, function (d) { return d[1]; })])
            .range([h - padding, padding]); // Reversed range for y-axis

        // Create the SVG element
        const svg = d3.select(".chart-container")
            .append("svg")
            .attr("width", w)
            .attr("height", h);

        // Step 3: Create and position the circles
        svg.selectAll("circle")
            .data(dataset)
            .enter()
            .append("circle")
            .attr("cx", function (d) {
                // The first value of the inner array (d[0]) is the x-coordinate.
                return xScale(d[0]);
            })
            .attr("cy", function (d) {
                // The second value (d[1]) is the y-coordinate.
                return yScale(d[1]);
            })
            .attr("r", function (d) {
                // Optional: The third value (d[2]) is used for the radius.
                return 5;
            })
            .attr("fill", function (d) {
                // Style important data points in red (e.g., where y > 80).
                if (d[1] > 80) {
                    return "red";
                }
                return "slategrey"; // Default color
            });

        // Step 4: Add labels to the scatter plot
        svg.selectAll("text")
            .data(dataset)
            .enter()
            .append("text")
            .text(function (d) {
                // The label text shows the coordinates.
                return d[0] + "," + d[1];
            })
            .attr("x", function (d) {
                // Position the label slightly to the right of the circle.
                return xScale(d[0]) + 10; // Offset by radius + a little extra
            })
            .attr("y", function (d) {
                // Position the label vertically aligned with the circle's center.
                return yScale(d[1]);
            })
            .attr("font-family", "sans-serif")
            .attr("font-size", "11px")
            .attr("fill", "black");

    </script>

    <footer>
        <p style="color:grey; font-style: italic;">
            COS30045 Data Visualisation<br>
            Joe Bloggs
        </p>
    </footer>

</body>

</html>
```

lab-3.2\index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="description" content="D3 Scatter Plot with Axes Exercise">
    <meta name="keywords" content="HTML, D3, JavaScript, SVG, Scatter Plot, Axes">
    <meta name="author" content="Joe Bloggs">
    <title>Scaled Scatter Plot with Axis</title>
    <!-- Step 1: Include the D3.js library -->
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        /* Optional: Add some basic styling for the chart container */
        body {
            font-family: sans-serif;
            text-align: center;
        }

        .chart-container {
            margin: 20px auto;
            border: 1px solid #ccc;
            display: inline-block;
            /* To make the container fit the SVG */
        }

        h1,
        footer {
            color: #333;
        }
    </style>
</head>

<body>

    <h1>Scaled Scatter Plot with Axis</h1>

    <div class="chart-container">
        <!-- The SVG will be created here by D3 -->
    </div>

    <!-- Include the separate JavaScript file -->
    <script src="script.js"></script>

    <footer>
        <p style="color:grey; font-style: italic;">
            COS30045 Data Visualisation<br>
            Joe Bloggs
        </p>
    </footer>

</body>

</html>
```

lab-3.2\script.js

```javascript
// --- Configuration Variables ---

// Define the dimensions of the SVG canvas
const w = 500;
const h = 300; // Increased height to accommodate axes
const padding = 60; // Increased padding for axes

// Step 2: Define the new dataset for the scatter plot
// Each inner array: [x_coordinate, y_coordinate]
const dataset = [
    [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
    [410, 12], [475, 44], [25, 67], [85, 21], [220, 88],
    [600, 150] // Outlier
];

// --- D3 Code ---

//Create scale functions
const xScale = d3.scaleLinear()
    .domain([0, d3.max(dataset, function (d) { return d[0]; })])
    .range([padding, w - padding]);

const yScale = d3.scaleLinear()
    .domain([0, d3.max(dataset, function (d) { return d[1]; })])
    .range([h - padding, padding]); // Reversed range for y-axis

//Create axis functions
const xAxis = d3.axisBottom()
    .ticks(5)
    .scale(xScale);

const yAxis = d3.axisLeft()
    .ticks(5)
    .scale(yScale);

// Create the SVG element
const svg = d3.select(".chart-container")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

// Step 3: Create and position the circles
svg.selectAll("circle")
    .data(dataset)
    .enter()
    .append("circle")
    .attr("cx", function (d) {
        // The first value of the inner array (d[0]) is the x-coordinate.
        return xScale(d[0]);
    })
    .attr("cy", function (d) {
        // The second value (d[1]) is the y-coordinate.
        return yScale(d[1]);
    })
    .attr("r", function (d) {
        // Circle radius
        return 5;
    })
    .attr("fill", function (d) {
        // Style important data points in red (e.g., where y > 80).
        if (d[1] > 80) {
            return "red";
        }
        return "slategrey"; // Default color
    });

// Step 4: Add labels to the scatter plot
svg.selectAll("text")
    .data(dataset)
    .enter()
    .append("text")
    .text(function (d) {
        // The label text shows the coordinates.
        return d[0] + "," + d[1];
    })
    .attr("x", function (d) {
        // Position the label slightly to the right of the circle.
        return xScale(d[0]) + 10; // Offset by radius + a little extra
    })
    .attr("y", function (d) {
        // Position the label vertically aligned with the circle's center.
        return yScale(d[1]);
    })
    .attr("font-family", "sans-serif")
    .attr("font-size", "11px")
    .attr("fill", "green");

// Step 5: Add the x-axis at the bottom of the chart
svg.append("g")
    .attr("transform", "translate(0, " + (h - padding) + ")")
    .call(xAxis);

// Step 6: Add the y-axis at the left of the chart
svg.append("g")
    .attr("transform", "translate(" + padding + ", 0)")
    .call(yAxis);
```

School of Science, Computing and Engineering Technologies

COS30045

LAB 4.1 Design Studio

Overview

In this lab you will be given a sample data set and asked to identify the different data and attribute types. You will also think about some questions about this data set that might be answered by a visualisation.

ardd_fatalities_Jan2020_0.xlsx (download from Canvas)

Download and review this data set before attempting this exercise.

1 Interpreting the data set

Complete the LAB 4.1 Quiz.

2 Visualisation Design

Think of three questions you would like to answer with that require a data visualistion.

For each data question you will need to consider the following:

Which data attributes (columns) do you need to answer this question?

Do you need to transform any of the data?

Does the data type change when you transform the data? If so how.

Make a sketch of how you think your visualisation might look and add to this document.

Your Question 1: Are there seasonal patterns in road fatalities that correlate with holiday periods and weather conditions?

Data attributes needed:

- `Date` (to extract month and day)
- `State` (different states have different seasonal patterns)
- `Crash_Type` (weather-related vs other factors)
- Potentially `Speed_Limit` and `National_Road_Type` (highway travel during holidays)

Data transformations:

- Convert `Date` to month/season categories (Summer: Dec-Feb, Autumn: Mar-May, Winter: Jun-Aug, Spring: Sep-Nov)

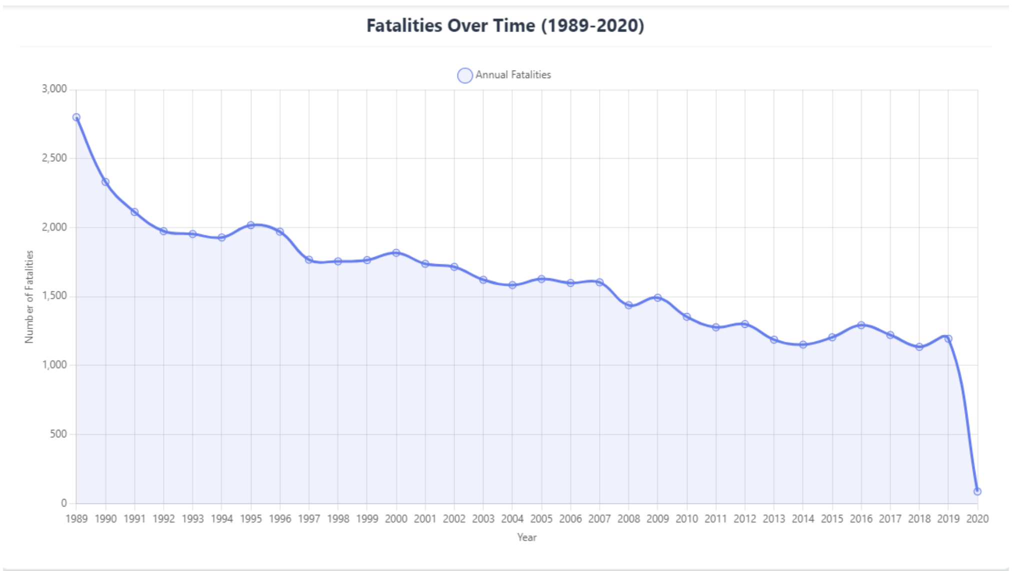
- Create binary flags for major holiday periods (Christmas/New Year: Dec 20-Jan 10, Easter: varies, School holidays)
- Group crashes by seasonal weather patterns
- Calculate fatality rates per capita for each season to account for population changes

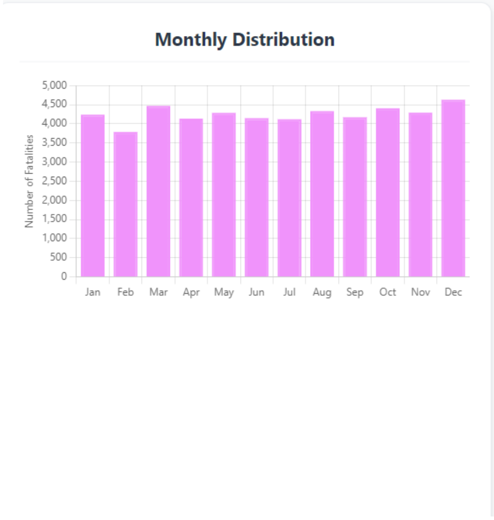
Data type changes:

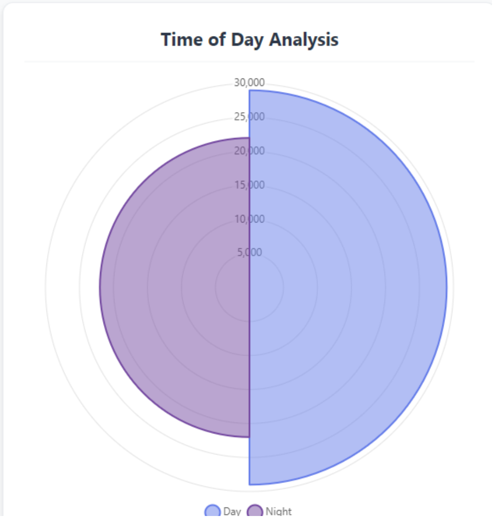
- Date (datetime) → Categorical (season names)
- Date (datetime) → Boolean (is_holiday_period)
- Numeric counts → Rates per 100,000 population

Visualization sketch:A multi-panel dashboard with:

- Line chart showing monthly fatalities over multiple years with holiday periods highlighted
- Heatmap calendar showing daily fatality patterns
- Grouped bar chart comparing seasonal fatality rates by state
- Small multiples showing crash type distribution by season







Your Question 2: How do demographic risk factors (age and gender) interact with road user types to identify the most vulnerable populations?

Data attributes needed:

- `Age`
- `Gender`
- `Road_User` (Driver, Passenger, Pedestrian, Motorcyclist, Cyclist, etc.)
- `Crash_Type`
- `Time` (to see if risk patterns vary by time of day)

Data transformations:

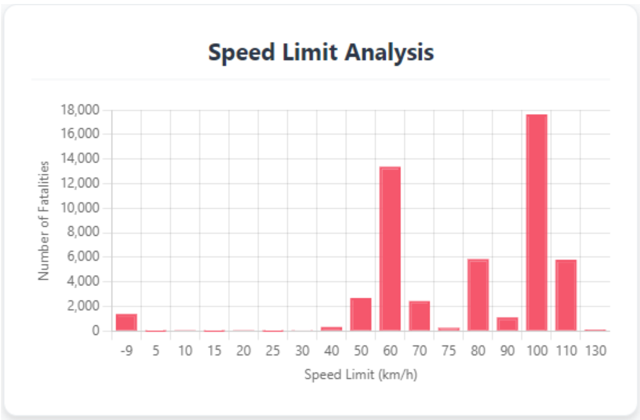
- Group `Age` into meaningful cohorts (17-25, 26-39, 40-59, 60+)
- Create interaction variables combining age groups, gender, and road user type
- Calculate risk ratios compared to baseline population demographics
- Transform time into peak/off-peak categories

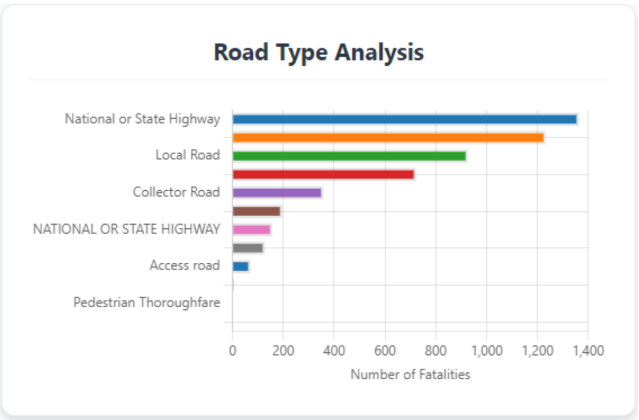
Data type changes:

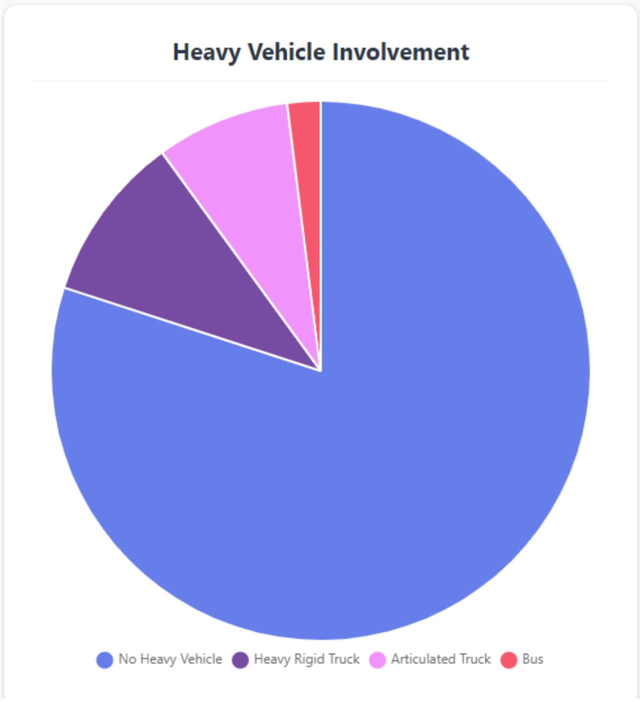
- Age (numeric) → Categorical age groups
- Time (datetime/numeric) → Categorical (Morning Peak, Day, Evening Peak, Night)
- Raw counts → Risk ratios and rates per demographic group

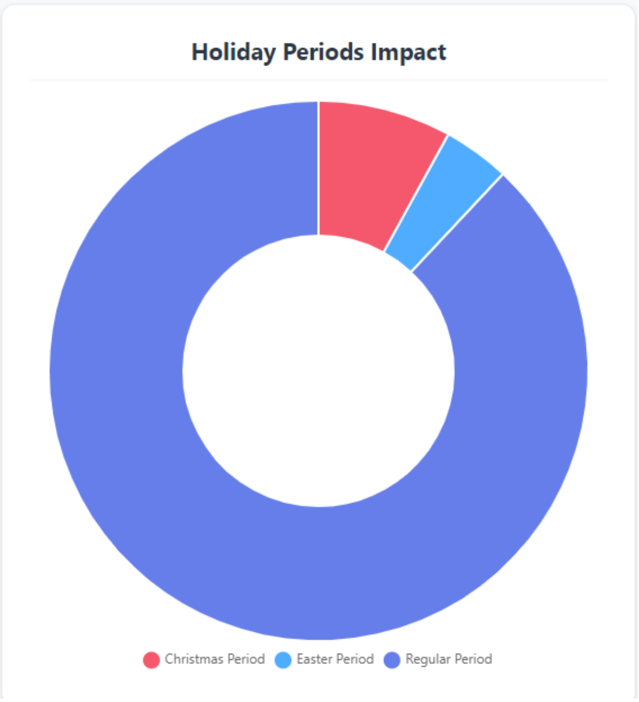
Visualization sketch:

- Bubble chart with age groups on x-axis, road user types on y-axis, bubble size = fatalities, color = gender
- Stacked bar chart showing road user type distribution within each age-gender combination
- Risk matrix heatmap showing fatality rates per 100,000 for each demographic-road user combination
- Small multiples radar charts showing risk profiles by time of day









Your Question 3: What is the relationship between road infrastructure (speed limits, road types) and crash severity across different geographic regions?

Data attributes needed:

- `State`
- `Speed_Limit`
- `National_Road_Type` (Local Road, State Highway, National Highway, etc.)
- `Remoteness` (Major Cities, Inner Regional, Outer Regional, Remote, Very Remote)
- Population density data (external data for per capita calculations)
- `Crash_Type` (to understand severity patterns)

Data transformations:

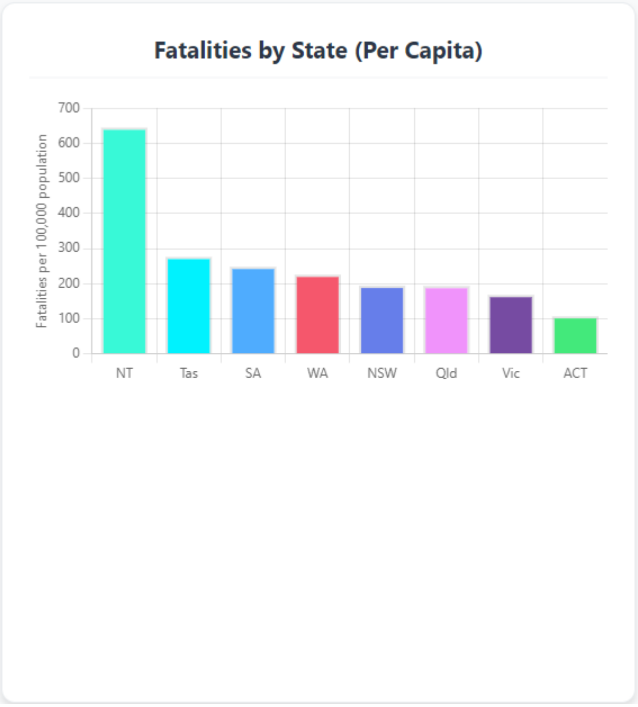
- Create speed limit categories (≤60km/h urban, 61-80km/h suburban, 81-100km/h rural, 100+km/h highway)
- Combine road type and speed limit into infrastructure risk categories
- Calculate fatality rates per kilometer of each road type (requires external road length data)
- Create remoteness-adjusted risk scores

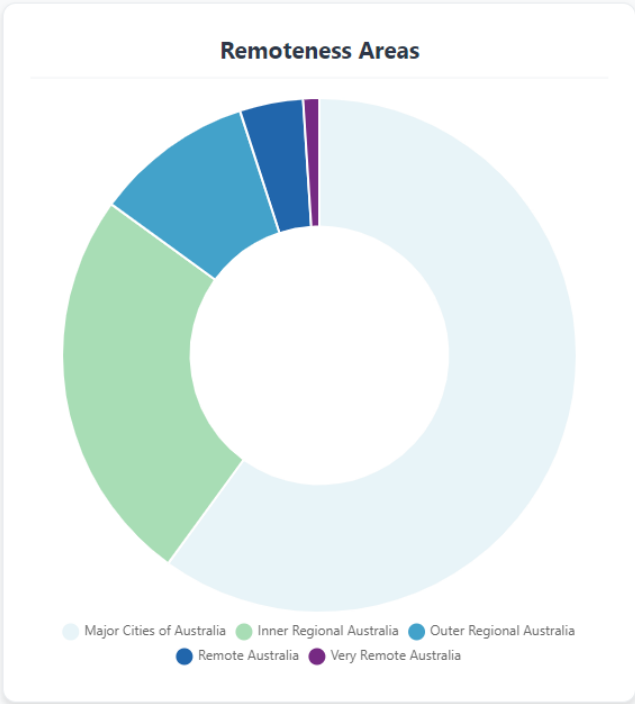
Data type changes:

- Speed_Limit (numeric) → Categorical speed zones
- Geographic coordinates → Remoteness categories
- Raw fatality counts → Fatalities per km of road infrastructure
- State names → Geographic regions (Metro vs Regional vs Remote)

Visualization sketch:

- Choropleth map of Australia showing fatality rates by remoteness areas
- Scatter plot with speed limit on x-axis, fatalities per km on y-axis, colored by road type, sized by traffic volume
- Parallel coordinates plot showing the relationship between state, remoteness, road type, speed limit, and fatality rate
- Box plots comparing fatality distributions across infrastructure categories





Include this file as evidence for your Demonstration 2