Deployed Project:

https://ajw2-collab.github.io/Assignment-3

Rationale:

Visual encodings:

X-axis & other cx(s): The data given from have the time value that was interpreted as UTCtime value instead of regular ones, thus I discovered that when zoomed in each data (formed as a circle) couldn't line up with the tick for dates on the X axis. Thus, I used utc format across everything including using scaleUtc() for getting cx(s), except the timestamp data from the csv, as it is formatted already.

Y-axis & other cy(s): Essentially the vertical positions based on the usAQI data from the csv file. However, same as the x-axis and many cx(s) values, they have to be mapped into the pixel locations, using scaleLinear(). All of these above require to be mapped within svg's set height and width, with margins. (some can go beyond margins but were hidden by cover <rect> elements)

Area band + Mean line (P10s, P90 & mean): First by normalizing dates of all the data into 15th during the mapping action, which was later sorted so that it could be used in a quantile function that only returns the number for percentiles (10th and 90th) and a mean function that returns the mean for each month. (Because they are grouped by the 15th of that month). Then it was mapped into an array with an object that contains date, mean, p10 and p90. For area and mean, the date was mapped by scaleUtc as the x in both line and area function. Then, mean mapped by scaleLinear() became y for line; p10 and p90 mapped by scaleLinear() as well became y0 and y1, respectively, for area.

Background AQI bands: Created in a for loop that takes every level from AQI_LEVELS, then make a <rect> starting at translated x = 0 and y = scaled max, and height becomes the the min - (max + 1), except the lowest one, because I don't want the band to go below the axis.

Color for tool tips of circles selected: Created in page then loaded it, a preset color based on how the original dataset's array is arranged.

Interactions:

Zoom + Pan: There exists a scalable factor for scaleExtent factor, gained by multiplying by boundaries getTimt() values difference by .000000001, controlling how much it could scale. With updated x locations called zx using zoomTransform.rescaleX(), we could update all the previous functions that change x location elements in <svg> into newly mapped zx values.

Hover probe (both cards and line): When hovering in the drawing are within the margin, it monitors PointerEvent, as the cursor moves, it detects the closest data points in x locations (mapped by zoom) and decides whether to snap and show the lines and tooltip depending whether that location is within a certain threshold of pixels. Meanwhile location of the line is dictated by the transformed (or not) x value shared by the ones in tooltip. While the tooltip is on either the left side or the right side of the line, depending on whether the width of the <svg> is going to fit them or not, it grabs and shows the AQI PM2.5 and location of each data.

Contextual pie chart: Grabs data that its data are within the 2 ends of transformed (or not) domain(), and generates a pie chart categorized based on their levels of AQI, determined by whether it is within the min and max of different levels.

Alternatives considered:

Brushing + Zooming that includes changing displayed y axis: I was thinking about a more zoomed in version of things, so data could be easily selected. However, I realize that the y axis is kind of irrelevant for us to zoom in and check. The reason for users to zoom in is often due to the high density of data, however, in this case, the data isn't dense across the y-axis, but in the scale of time. Therefore, this alternative was not considered. Thus, zooming and selecting the zoomed extent for further analysis (pie chart) is more effective and clear.

Multiple mean lines and areas when examining all station data: Most of the data are collected within the Pittsburgh area, I believe comparing trends is kind of unnecessary and confusing as multiple lines are merged together if plotted. Thus I believe a tool tip that compares and ranks each station is sufficient.

Goal of the Second Visualization:

The first question that this visualization answers is: "Within the currently visible date range, how often did each AQI category occur?"

The goal here is to let users quickly assess the distribution of air quality conditions for any time window, without manually counting.

The second question that this visualization answers is: "Within the selected date, which station records the worst air quality condition?"

The goal here is to let users quickly rank each station's recording without checking each data individually.

Development Process:

It took me roughly 40 hours to complete everything, while the hover line is the most time-consuming part (roughly 6 hours). However I would say the most challenging part is the learning process and referencing different sources of tutorials and examples. (links are in the comments of the svelte files)

I have VSCode copilot as well as ChatGPT to guide me through most TypeScript issues as well as asking what the examples from D3 tutorials mean.

Prompt Examples:

"How do I correctly type d3.pie() and d3.arc()?"
"Explain how the .attr("transform", `translate(0,${innerHeight})`) would make in <svg>"