```
// Width and height
const w = 500;
const h = 500;

// Define map projection
const projection = d3.geoMercator()
  .center([134, -28])
  .translate([w / 2, h / 2])
  .scale(500);

// Define path generator
const path = d3.geoPath().projection(projection);

const color = d3.scaleOrdinal()
  .range([
    "#d73027", "#fdae61", "#fee08b", "#3288bd", "#4d4d4d",
    "#8073ac", "#b3de69", "#fccde5", "#d9d9d9"
  ]);

// Create SVG
const svg = d3.select(".placeholder-charts-svgAnchor")
  .append("svg")
  .attr("width", w)
  .attr("height", h);

let australia; // For overall element
let active = null;

// Load GeoJSON data asynchronously
async function drawMap() {
  const json = await d3.json("preproccessed_data/aust.json");
  const stateData = await d3.csv("preproccessed_data/location_state_FCA.csv");

  const states = svg.selectAll("path")
    .data(json.features)
    .enter().append("path")
    .attr("d", path)
    .attr("class", d => d.properties.STATE_NAME.replace(/\s/g, ""))
    .attr("stroke", "white")
    .attr("opacity", 1)
    .attr("fill", (d, i) => color(i))
    .on("click", drawState)
    .style("cursor", "pointer");

    d3.select(".state-info").html(`
      <p>Select a state for statistics</p>
      `);


  function drawState(event, d) {
    if (active && active.node() === this) return reset();
    if (active) active.classed("active", false);
    active = d3.select(this).classed("active", true);

    const bounds = path.bounds(d);
    const dx = bounds[1][0] - bounds[0][0];
    const dy = bounds[1][1] - bounds[0][1];
    const x = (bounds[0][0] + bounds[1][0]) / 2;
    const y = (bounds[0][1] + bounds[1][1]) / 2;
    const scale = 2;
    const translate = [w / 2 - scale * x, h / 2 - scale * y];

    states.transition().duration(1000)
      .attr("transform", `translate(${translate})scale(${scale})`)
      .attr("stroke-width", `${1 / scale}px`)
      .attr("opacity", d2 => d2 === d ? 1 : 0)
      .attr("pointer-events", d2 => d2 === d ? "all" : "none");

    if (australia) {
      australia.transition().duration(1000).attr("opacity", 0);
    }

    const matched = stateData.find(row => row.JURISDICTION === d.properties.STATE_NAME);

    const infoDiv = d3.select(".state-info");
    const props = d.properties;

    infoDiv.html(`
      <h2>${props.STATE_NAME}</h2>
      <p><strong>Sum of Fines:</strong> ${matched ? abbreviateNumber(matched["Sum of All Fines/Charges/Arrest"]) : 'N/A'}</p>
      <p>Population: ${matched ? abbreviateNumber(matched["Population"]) : 'N/A'}</p>
      <p>Area: ${matched ? abbreviateNumber(matched["Area (km²)"]) : 'N/A'} km²</p>
    `);
  }

  function reset() {
    if (active) active.classed("active", false);
    active = null;

    states.transition().duration(1000)
      .attr("stroke-width", "1px")
      .attr("opacity", 1)
      .attr("pointer-events", "all")
      .attr("transform", "");

    if (australia) {
      australia.transition().duration(1000).attr("opacity", 0.1);
    }

    d3.select(".state-info").html(`
      <p>Select a state for statistics</p>
      `);
  }
```

```
  // Optional: Add a background for Australia
  australia = svg.append("rect")
    .attr("width", w)
    .attr("height", h)
    .attr("fill", "#f0f0f0")
    .lower()  // Put it behind everything
    .attr("opacity", 0.1);
}

function abbreviateNumber(value) {
  if (value === null || value === undefined || isNaN(value)) return 'N/A';
  const suffixes = ["", "K", "M", "B", "T"];
  const suffixNum = Math.floor( (""+value).length / 3 );
  let shortValue = parseFloat((value / Math.pow(1000, suffixNum)).toFixed(2));
  if (shortValue % 1 === 0) shortValue = shortValue.toFixed(0);
  return shortValue + suffixes[suffixNum];
}

drawMap();
```