Name	Rucha Kulkarni
Class	BE Computer Engineering (Batch F)
UID	2021300067
Exp No.	7

Aim: Creating Visualizations using D3.js on a Finance Dataset

Objectives:

- To explore and visualize a dataset related to Finance/ Banking/ Insurance/ Credit using D3.js.
- To create basic visualizations (Bar chart, Pie chart, Histogram, Timeline chart, Scatter plot, Bubble plot) to understand data distribution and trends.
- To create advanced visualizations (Word chart, Box and Whisker plot, Violin plot, Regression plot, 3D chart, Jitter) for deeper insights and complex relationships.
- To perform hypothesis testing using the Pearson correlation coefficient to evaluate relationships between numerical variables in the dataset.

Description:

Dataset used is Insurance Dataset available at

https://www.kaggle.com/datasets/ravalsmit/insurance-claims-and-policy-data

Customer ID:A unique identifier assigned to each customer. Useful for referencing individual records.

Age: The age of the customer. Important for understanding demographic trends and risk assessment.

Gender: The gender of the customer. May be relevant for analyzing risk profiles and insurance needs.

Marital Status: The marital status of the customer. Can influence risk and insurance product preferences.

Occupation: The profession of the customer. Helps in understanding income levels and risk factors associated with different jobs.

Income Level: The income level of the customer. Critical for assessing the ability to pay premiums and potential insurance needs.

Education Level: The highest level of education attained by the customer. May correlate with income and risk awareness.

Geographic Information: The region or area where the customer resides. Geographic location can impact risk profiles due to environmental factors.

Location: Specific location details (city, town, etc.). Similar significance as geographic information.

Behavioral Data: Data reflecting customer behavior or preferences. Useful for tailoring services and marketing strategies.

Purchase History: Records of previous purchases. Important for understanding customer loyalty and product preferences.

Policy Start Date: The date when the insurance policy was initiated. Useful for tracking policy duration and renewal patterns.

Policy Renewal Date: The date when the policy is due for renewal. Important for analyzing customer retention.

Claim History: Records of claims made by the customer. Essential for assessing risk and claim frequency.

Interactions with Customer Service: The number of times the customer has interacted with customer service. Can indicate customer satisfaction and engagement.

Insurance Products Owned: The types of insurance products the customer currently owns. Relevant for cross-selling and upselling strategies.

Coverage Amount: The total coverage amount of the insurance policy. Critical for understanding policy value and risk exposure.

Premium Amount: The amount the customer pays for their insurance policy. Important for revenue analysis and pricing strategies.

Deductible: The amount the insured must pay out of pocket before the insurance kicks in. Influences customer choice and risk behavior.

Policy Type: The type of insurance policy (e.g., life, health, auto). Useful for segmenting products and analyzing market trends.

Customer Preferences: Preferences related to services or products. Important for customer relationship management.

Preferred Communication Channel: The customer's preferred method of communication (e.g., phone, email, in-person). Useful for improving customer interactions.

Preferred Contact Time: The time of day the customer prefers to be contacted. Helps in scheduling interactions effectively.

Preferred Language: The language the customer prefers for communication. Important for personalized customer service.

Risk Profile:A classification of the customer based on risk factors. Essential for underwriting and risk assessment.

Previous Claims History:A record of past claims made by the customer. Influences risk evaluation and premium pricing.

Credit Score: The customer's credit score, reflecting their creditworthiness. Important for financial assessment and premium calculations.

Driving Record: The customer's driving history (e.g., clean, violations). Relevant for auto insurance risk assessment.

Life Events: Significant life events that may affect insurance needs (e.g., marriage, childbirth). Useful for targeted marketing.

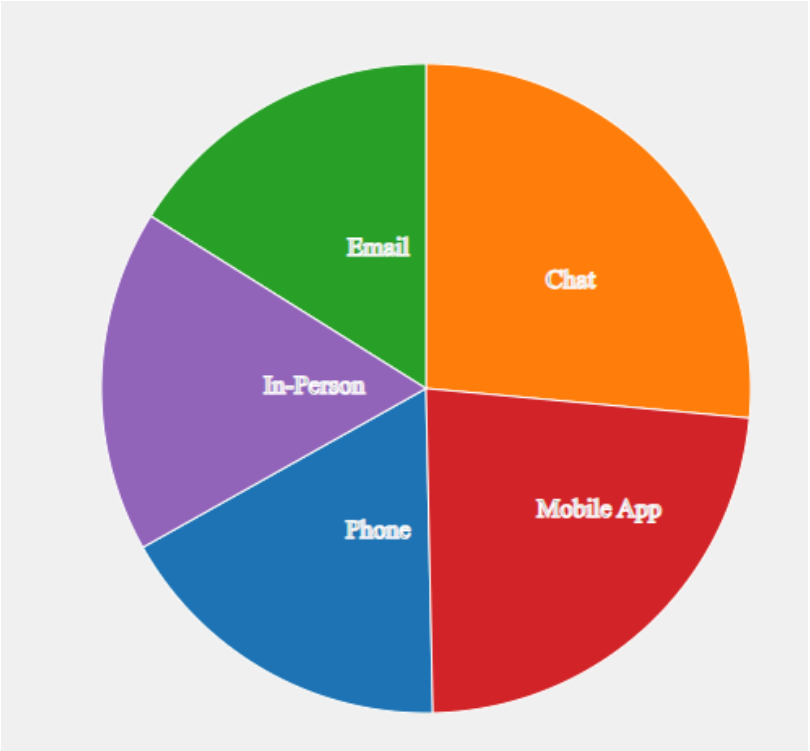
Segmentation Group:A grouping of customers based on shared characteristics. Helps in targeted marketing and analysis.

Graphs and Observations:

Bar chart:

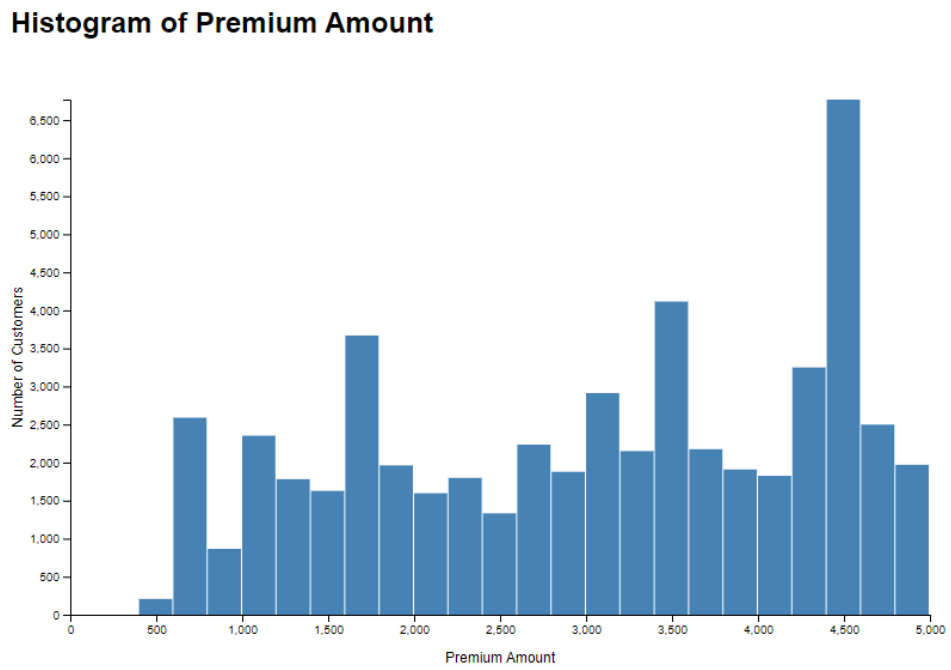
Observation: The bar graph shows that Group insurance policies have the most customers, indicating a strong preference for collective coverage. In contrast, Individual policies have the fewest customers, suggesting a need for better promotion of personalized insurance plans.

Pie Chart:



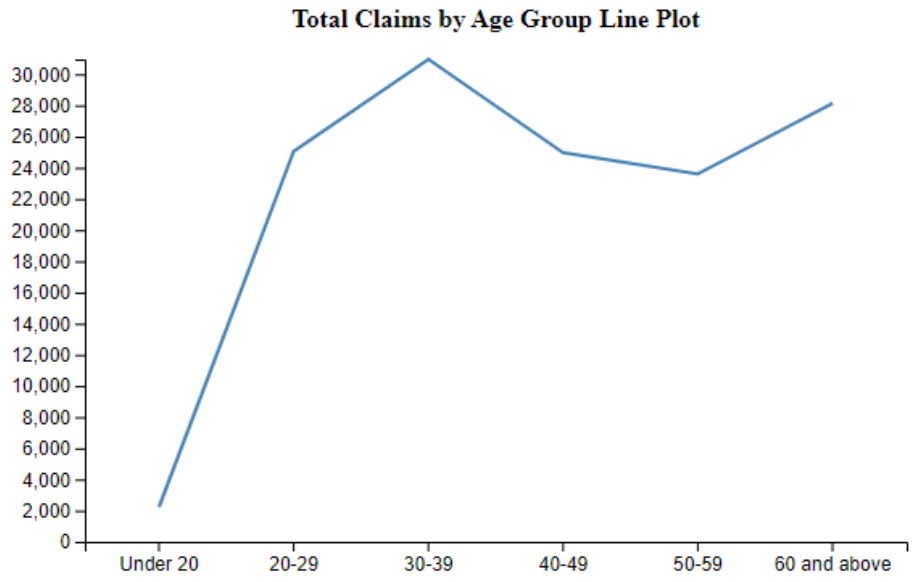
Observation: The pie chart indicates that customers prefer using mobile apps and chat for their interactions, reflecting a trend toward digital engagement. This suggests that insurance companies should prioritize enhancing their mobile and chat services to meet customer demands.

Histogram:



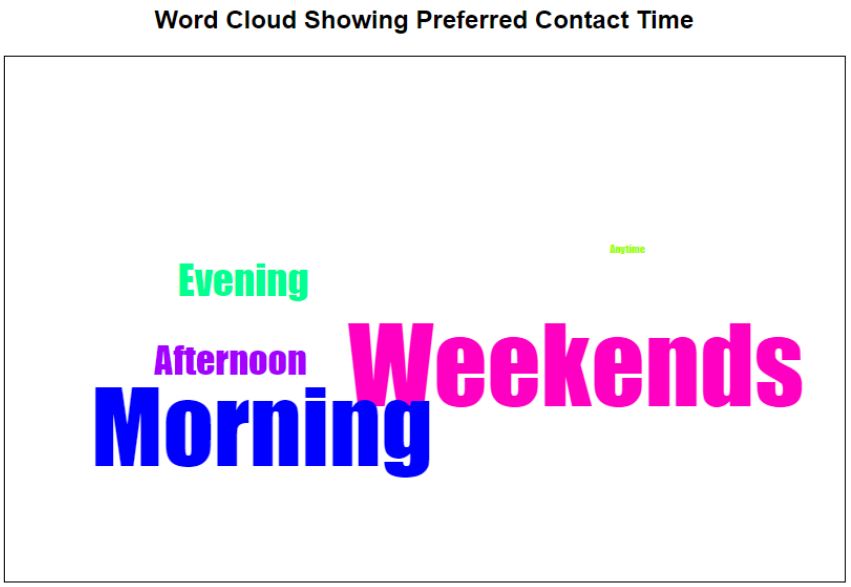
Observation: The histogram of premium amounts indicates that the highest concentration of customers is around a premium amount of 4500, suggesting it is the most common premium level. Following that, the premium amounts of 1600 and 3500 also show significant customer interest.

Line Chat:



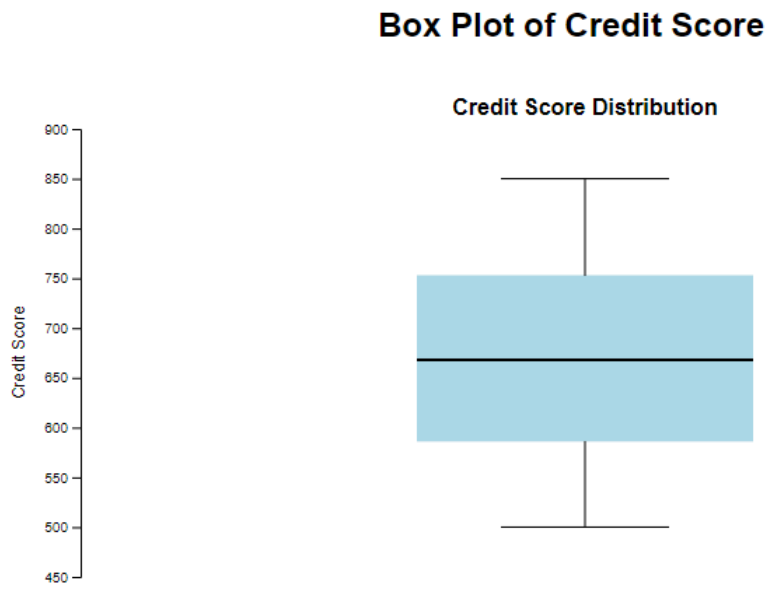
Observation: The line plot indicates how total claims vary with age.The line plot shows that the age groups 30-39 and 60+ have the highest claims, indicating increased risks during mid-life and later years.

Word Chart:



Observation: The word cloud indicates that Weekends and Morning are the most preferred contact times, suggesting that customers favor these periods for communication.

Box Plot:



Observation: The box plot of credit scores shows a median of 650, with a range from 500 to 850, indicating a consistent spread of scores without outliers. This suggests that the majority of customers have credit scores concentrated within this defined range.

Correlation Coefficient:

Pearson Correlation Matrix:

	Customer ID	Age	Income Level	Location
Customer ID	1.000000	-0.014102	0.010541	0.012178
Age	-0.014102	1.000000	-0.003447	0.000338
Income Level	0.010541	-0.003447	1.000000	-0.013807
Location	0.012178	0.000338	-0.013807	1.000000
Claim History	-0.017217	-0.001781	-0.012692	0.015215
Coverage Amount	0.003788	0.006746	-0.018024	0.014825
Premium Amount	-0.007172	0.003665	0.002025	-0.000665
Deductible	0.000798	0.015671	-0.011172	-0.012636
Risk Profile	-0.010624	-0.023871	-0.012492	-0.001497
Previous Claims History	0.006145	0.005837	-0.004761	-0.003182
Credit Score	0.018498	0.001589	-0.019717	0.006222

	Claim History	Coverage Amount	Premium Amount
Customer ID	-0.017217	0.003788	-0.007172
Age	-0.001781	0.006746	0.003665
Income Level	-0.012692	-0.018024	0.002025
Location	0.015215	0.014825	-0.000665
Claim History	1.000000	-0.000336	-0.019950
Coverage Amount	-0.000336	1.000000	-0.001647
Premium Amount	-0.019950	-0.001647	1.000000
Deductible	0.000248	-0.007675	-0.001168
Risk Profile	-0.006573	0.008810	0.014742
Previous Claims History	-0.017262	-0.012060	0.023006
Credit Score	-0.002706	-0.000468	-0.012993

	Deductible	Risk Profile	Previous Claims History
Customer ID	0.000798	-0.010624	0.006145
Age	0.015671	-0.023871	0.005837
Income Level	-0.011172	-0.012492	-0.004761
Location	-0.012636	-0.001497	-0.003182
Claim History	0.000248	-0.006573	-0.017262
Coverage Amount	-0.007675	0.008810	-0.012060
Premium Amount	-0.001168	0.014742	0.023006
Deductible	1.000000	0.005847	0.021080
Risk Profile	0.005847	1.000000	0.014037
Previous Claims History	0.021080	0.014037	1.000000
Credit Score	0.003211	-0.013158	-0.002146

```
P-values Matrix:
                        Customer ID       Age Income Level  Location  \
Customer ID                     NaN  0.001107     0.014758  0.004849
Age                        0.001107       NaN     0.425332  0.937776
Income Level               0.014758  0.425332          NaN  0.001404
Location                   0.004849  0.937776     0.001404       NaN
Claim History              0.000068  0.680304     0.003327  0.000433
Coverage Amount            0.380881  0.118685     0.000031  0.000605
Premium Amount             0.097115  0.396542      0.63945   0.87772
Deductible                 0.853501  0.000289     0.009759  0.003469
Risk Profile               0.013993       0.0     0.003859  0.729175
Previous Claims History    0.155216  0.176958     0.270748  0.461708
Credit Score               0.000019  0.713173     0.000005  0.150104

                        Claim History Coverage Amount Premium Amount  \
Customer ID                  0.000068        0.380881       0.097115
Age                          0.680304        0.118685       0.396542
Income Level                 0.003327        0.000031        0.63945
Location                     0.000433        0.000605        0.87772
Claim History                     NaN         0.93804       0.000004
Coverage Amount               0.93804             NaN       0.703299
Premium Amount               0.000004        0.703299            NaN
Deductible                   0.954256        0.075863       0.786979
Risk Profile                 0.128416        0.041563       0.000649
Previous Claims History      0.000065        0.005277            0.0
Credit Score                 0.531375        0.913755       0.002651

                        Deductible Risk Profile Previous Claims History  \
Customer ID               0.853501     0.013993                 0.155216
Age                       0.000289          0.0                 0.176958
Income Level              0.009759     0.003859                 0.270748
Location                  0.003469     0.729175                 0.461708
Claim History             0.954256     0.128416                 0.000065
Coverage Amount           0.075863     0.041563                 0.005277
Premium Amount            0.786979     0.000649                      0.0
Deductible                     NaN     0.176248                 0.000001
Risk Profile              0.176248          NaN                 0.001166
Previous Claims History   0.000001     0.001166                      NaN
Credit Score              0.457653     0.002338                 0.619649
```

Claim History & Premium Amount: A weak negative correlation (-0.0199) suggests that as claim history increases, the premium amount may slightly decrease.

Age & Risk Profile: A very weak negative correlation (-0.0239) suggests that older individuals may have a slightly lower risk profile.

Code in D3.js:

Bar Plot:

```
<!DOCTYPE html>

<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Insurance Data Visualization</title>
  <script src="https://d3js.org/d3.v6.min.js"></script>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: flex-start;
      flex-direction: column;
      margin: 0;
      padding: 20px;
    }
    .bar {
      fill: #4CAF50;
    }
    .bar:hover {
      fill: orange;
    }
    .box {
      fill: #6BAED6;
    }
    .median {
      stroke: black;
      stroke-width: 2;
    }
    svg {
```

```
      margin: 20px;
    }
  </style>
</head>
<body>

  <!-- Policy Type Distribution Bar Chart -->
  <svg id="policyTypeChart" width="600" height="400"></svg>

  <!-- Claim History Pie Chart -->
  <svg id="claimHistoryChart" width="600" height="400"></svg>

  <!-- Income Level by Marital Status Box Plot -->
  <svg id="incomeBoxPlot" width="600" height="400"></svg>

  <script>
    // Load the data
    d3.csv("insurance_data.csv").then(function(data) {

      // --- Policy Type Distribution Bar Chart ---
      const policyCount = d3.rollup(data, v => v.length, d => d["Policy
Type"]);
      const policyData = Array.from(policyCount, ([key, value]) => ({
key, value }));

      const policySvg = d3.select("#policyTypeChart");
      const margin = { top: 40, right: 30, bottom: 60, left: 60 };
      const width = +policySvg.attr("width") - margin.left -
margin.right;
```

```
      const height = +policySvg.attr("height") - margin.top -
margin.bottom;


      const x = d3.scaleBand()
                  .domain(policyData.map(d => d.key))
                  .range([0, width])
                  .padding(0.1);


      const y = d3.scaleLinear()
                  .range([height, 0]);


      const policyG = policySvg.append("g")
                               .attr("transform",
`translate(${margin.left},${margin.top})`);


      y.domain([0, d3.max(policyData, d => d.value)]);


      policyG.append("g")
        .attr("class", "axis axis--x")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x).tickFormat(d => d));


      policyG.append("g")
        .attr("class", "axis axis--y")
        .call(d3.axisLeft(y));


      // Add title
      policySvg.append("text")
        .attr("x", (width / 2) + margin.left)
        .attr("y", margin.top / 2 + 20)
```

```
    .attr("text-anchor", "middle")
    .style("font-size", "16px")
    .text("Distribution of Policy Types");

// X Axis Label
policySvg.append("text")
    .attr("x", width / 2 + margin.left)
    .attr("y", height + margin.top + 40)
    .attr("text-anchor", "middle")
    .text("Policy Type");

// Y Axis Label
policySvg.append("text")
    .attr("transform", "rotate(-90)")
    .attr("y", margin.left - 60)
    .attr("x", -(height / 2 + margin.top))
    .attr("dy", "1em")
    .attr("text-anchor", "middle")
    .text("Number of Customers");

// Create bars
policyG.selectAll(".bar")
    .data(policyData)
    .enter().append("rect")
      .attr("class", "bar")
      .attr("x", d => x(d.key))
      .attr("y", d => y(d.value))
      .attr("width", x.bandwidth())
      .attr("height", d => height - y(d.value));
```

```
      // --- Claim History Pie Chart ---
      const claimData = d3.rollup(data, v => v.length, d => d["Claim History"]);
      const claimDataArray = Array.from(claimData, ([key, value]) => ({ key, value }));

      const claimSvg = d3.select("#claimHistoryChart");
      const claimWidth = +claimSvg.attr("width") / 2;
      const claimHeight = +claimSvg.attr("height") / 2;
      const radius = Math.min(claimWidth, claimHeight) / 2;

      const claimG = claimSvg.append("g")
                             .attr("transform", `translate(${claimWidth}, ${claimHeight})`);

      const color = d3.scaleOrdinal(d3.schemeCategory10);

      const pie = d3.pie().value(d => d.value);
      const arc = d3.arc().outerRadius(radius - 10).innerRadius(0);

      const arcs = pie(claimDataArray);

      claimG.selectAll(".arc")
        .data(arcs)
        .enter().append("g")
          .attr("class", "arc")
        .append("path")
          .attr("d", arc)
          .style("fill", (d) => color(d.data.key));
```

```
      // Add title
      claimSvg.append("text")
        .attr("x", claimWidth)
        .attr("y", margin.top / 2)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .text("Claim History Distribution");









    }).catch(error => {
      console.error('Error loading the CSV file:', error);
    });
  </script>
</body>
</html>
```

Box Plot:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Box Plot of Credit Score</title>
```

```
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
    #box-plot {
      margin: auto;
      width: 800px;
      height: 400px;
    }
  </style>
</head>
<body>

<h2>Box Plot of Credit Score</h2>
<div id="box-plot"></div>

<script>
  // Load the dataset from insurance_data.csv
  d3.csv("insurance_data.csv").then(function(data) {
    console.log("Data loaded:", data);  // Debug the data

    // Prepare the Credit Score data (convert strings to numbers)
    const creditScores = data.map(d => +d["Credit Score"]).filter(d =>
!isNaN(d)); // Filter non-numeric values

    // Calculate statistics for the box plot: min, max, median, and
quartiles
    const sortedScores = creditScores.sort(d3.ascending);
```

```
    const q1 = d3.quantile(sortedScores, 0.25);
    const median = d3.quantile(sortedScores, 0.5);
    const q3 = d3.quantile(sortedScores, 0.75);
    const iqr = q3 - q1;  // Interquartile range
    const min = Math.max(d3.min(sortedScores), q1 - 1.5 * iqr);  //
Lower bound (outliers excluded)
    const max = Math.min(d3.max(sortedScores), q3 + 1.5 * iqr);  //
Upper bound (outliers excluded)

    const boxPlotData = {min, q1, median, q3, max};
    console.log("Box plot data:", boxPlotData);  // Log box plot data

    // Set up SVG dimensions
    const width = 800;
    const height = 400;
    const margin = {top: 40, right: 30, bottom: 40, left: 50};

    // Create the SVG container
    const svg = d3.select("#box-plot")
      .append("svg")
        .attr("width", width)
        .attr("height", height)
      .append("g")
        .attr("transform", `translate(${margin.left},${margin.top})`);

    // Set up scales
    const xScale = d3.scaleBand()
      .domain(["Credit Score"])
      .range([0, width - margin.left - margin.right])
      .padding(0.5);
```

```
const yScale = d3.scaleLinear()
  .domain([d3.min(creditScores) - 50, d3.max(creditScores) + 50])
  .range([height - margin.top - margin.bottom, 0]);

// Add y-axis
svg.append("g")
  .call(d3.axisLeft(yScale));

// Draw the box (from q1 to q3)
svg.append("rect")
  .attr("x", xScale("Credit Score"))
  .attr("y", yScale(q3))
  .attr("width", xScale.bandwidth())
  .attr("height", yScale(q1) - yScale(q3))
  .attr("fill", "lightblue");

// Draw the median line
svg.append("line")
  .attr("x1", xScale("Credit Score"))
  .attr("x2", xScale("Credit Score") + xScale.bandwidth())
  .attr("y1", yScale(median))
  .attr("y2", yScale(median))
  .attr("stroke", "black")
  .attr("stroke-width", 2);

// Draw min and max whiskers
svg.append("line")
  .attr("x1", xScale("Credit Score") + xScale.bandwidth() / 2)
```

```
    .attr("x2", xScale("Credit Score") + xScale.bandwidth() / 2)
    .attr("y1", yScale(min))
    .attr("y2", yScale(q1))
    .attr("stroke", "black");

  svg.append("line")
    .attr("x1", xScale("Credit Score") + xScale.bandwidth() / 2)
    .attr("x2", xScale("Credit Score") + xScale.bandwidth() / 2)
    .attr("y1", yScale(q3))
    .attr("y2", yScale(max))
    .attr("stroke", "black");

  // Draw the min and max horizontal whiskers
  svg.append("line")
    .attr("x1", xScale("Credit Score") + xScale.bandwidth() / 4)
    .attr("x2", xScale("Credit Score") + 3 * xScale.bandwidth() / 4)
    .attr("y1", yScale(min))
    .attr("y2", yScale(min))
    .attr("stroke", "black");

  svg.append("line")
    .attr("x1", xScale("Credit Score") + xScale.bandwidth() / 4)
    .attr("x2", xScale("Credit Score") + 3 * xScale.bandwidth() / 4)
    .attr("y1", yScale(max))
    .attr("y2", yScale(max))
    .attr("stroke", "black");

  // Add title and axis labels
  svg.append("text")
```

```
      .attr("x", (width - margin.left - margin.right) / 2)
      .attr("y", -10)
      .attr("text-anchor", "middle")
      .attr("font-size", "16px")
      .attr("font-weight", "bold")
      .text("Credit Score Distribution");

    svg.append("text")
      .attr("x", -height / 2 + margin.top)
      .attr("y", -margin.left + 10)
      .attr("transform", "rotate(-90)")
      .attr("text-anchor", "middle")
      .attr("font-size", "12px")
      .text("Credit Score");

  }).catch(function(error) {
    console.error("Error loading the CSV file:", error);  // Error handling
  });
</script>

</body>
</html>
```

Histogram:

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

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Histogram of Premium Amount</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
    .bar {
      fill: steelblue;
    }
    .bar:hover {
      fill: orange;
    }
    .axis-label {
      font-size: 12px;
    }
  </style>
</head>
<body>

<h2>Histogram of Premium Amount</h2>
<div id="chart"></div>

<script>
  // Set the dimensions and margins of the graph
  const margin = {top: 30, right: 30, bottom: 40, left: 50},
        width = 800 - margin.left - margin.right,
```

```
        height = 500 - margin.top - margin.bottom;

  // Append the SVG object to the body of the page
  const svg = d3.select("#chart")
    .append("svg")
      .attr("width", width + margin.left + margin.right)
      .attr("height", height + margin.top + margin.bottom)
    .append("g")
      .attr("transform", `translate(${margin.left},${margin.top})`);

  // Load the dataset
  d3.csv("insurance_data.csv").then(data => {

    // Convert Premium Amount to a numerical value
    data.forEach(d => {
      d["Premium Amount"] = +d["Premium Amount"];
    });

    // Set the x scale (Premium Amount)
    const x = d3.scaleLinear()
      .domain([0, d3.max(data, d => d["Premium Amount"])]) // Input
data range
      .range([0, width]); // Output range on the graph

    // Create the histogram bins
    const histogram = d3.histogram()
      .value(d => d["Premium Amount"]) // Accessor to the Premium
Amount field
      .domain(x.domain())  // Set the domain for the x scale
      .thresholds(x.ticks(20)); // Number of bins
```

```
    // Group the data into bins
    const bins = histogram(data);

    // Set the y scale (Number of customers per bin)
    const y = d3.scaleLinear()
      .domain([0, d3.max(bins, d => d.length)]) // Max count in the
bins
      .range([height, 0]);

    // Append the bars to the graph
    svg.selectAll("rect")
      .data(bins)
      .enter()
      .append("rect")
        .attr("x", d => x(d.x0)) // x0 is the lower bound of the bin
        .attr("y", d => y(d.length)) // Height of the bin
        .attr("width", d => x(d.x1) - x(d.x0) - 1) // Bin width
        .attr("height", d => height - y(d.length)) // Bar height
        .attr("class", "bar");

    // Add the x-axis
    svg.append("g")
      .attr("transform", `translate(0,${height})`)
      .call(d3.axisBottom(x));

    // Add the y-axis
    svg.append("g")
      .call(d3.axisLeft(y));
```

```
    // X-axis label
    svg.append("text")
      .attr("class", "axis-label")
      .attr("x", width / 2)
      .attr("y", height + margin.bottom)
      .style("text-anchor", "middle")
      .text("Premium Amount");

    // Y-axis label
    svg.append("text")
      .attr("class", "axis-label")
      .attr("transform", "rotate(-90)")
      .attr("y", -margin.left + 10)
      .attr("x", -height / 2)
      .style("text-anchor", "middle")
      .text("Number of Customers");

  }).catch(error => {
    console.error("Error loading the CSV file:", error);
  });
</script>

</body>
</html>
```

Word Cloud:

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Word Cloud for Preferred Communication Channel</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/d3-cloud/1.2.5/d3.layout.cl
oud.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
    #word-cloud {
      margin: auto;
      width: 100%;
      height: 500px;
    }
    text {
      font-family: Impact;
    }
  </style>
</head>
<body>

<h2>Word Cloud Showing Preferred Contact Time</h2>
<div id="word-cloud"></div>
```

```
<script>
  // Load the dataset
  d3.csv("insurance_data.csv").then(function(data) {
    console.log("Data loaded:", data);  // Check if the data is loaded correctly

    // Prepare data: Count the frequency of each communication channel, trimming extra spaces and filtering empty or invalid values
    const communicationCount = d3.rollup(
      data.filter(d => d["Preferred Contact Time"] && d["Preferred Contact Time"].trim() !== ""), // Filter out empty or invalid entries
      v => v.length,
      d => d["Preferred Contact Time"].trim()  // Trim spaces around values
    );

    console.log("Preferred Contact Time Count:", communicationCount); // Log the count for debugging

    // Convert the Map into an array of objects for word cloud input
    const wordData = Array.from(communicationCount, ([key, value]) => ({text: key, size: value}));

    // Determine minimum and maximum values for scaling
    const maxSize = d3.max(wordData, d => d.size);
    const minSize = d3.min(wordData, d => d.size);

    // Create a scale for the font size based on word frequency
    const fontSizeScale = d3.scaleLinear()
                            .domain([minSize, maxSize])
```

```
                            .range([10, 100]);  // Adjust font size
between 10 and 100


    // Set dimensions for the word cloud
    const width = 800;
    const height = 500;


    // Create the word cloud layout
    d3.layout.cloud()
      .size([width, height])
      .words(wordData)
      .padding(5)
      .rotate(() => ~~(Math.random() * 2) * 90)
      .fontSize(d => fontSizeScale(d.size))  // Use scaled font size
      .on("end", draw)
      .start();


    // Draw the word cloud
    function draw(words) {
      d3.select("#word-cloud")
        .append("svg")
          .attr("width", width)
          .attr("height", height)
          .style("border", "1px solid black")  // Debug the SVG
visibility
        .append("g")
          .attr("transform", `translate(${width / 2},${height / 2})`)
        .selectAll("text")
          .data(words)
        .enter().append("text")
```

```
            .style("font-size", d => d.size + "px")
            .style("fill", () => "hsl(" + Math.random() * 360 +
",100%,50%)")
            .attr("text-anchor", "middle")
            .attr("transform", d => `translate(${[d.x,
d.y]})rotate(${d.rotate})`)
            .text(d => d.text);
    }
  }).catch(function(error) {
    console.error("Error loading the CSV file:", error);  // Log an
error if the data doesn't load
  });
</script>


</body>
</html>
```

Pie Chart:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Pie Chart: Customer Service Interactions</title>
  <script src="https://d3js.org/d3.v6.min.js"></script>


<br>
  <style>
```

```
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f4;
    }
    .arc {
      stroke: #fff;
    }
    .title {
      font-size: 16px;
      text-anchor: middle;
      font-weight: bold;
    }
  </style>
</head>
<body>

<svg id="pieChart" width="400" height="400"></svg>

<script>
  // Load the dataset from insurance_data.csv
  d3.csv("insurance_data.csv").then(function(data) {
    // Count occurrences of each type of interaction
    const counts = d3.rollup(data, v => v.length, d => d['Interactions with Customer Service']);
    const formattedData = Array.from(counts, ([interaction, count]) => ({ interaction, count }));
```

```
    // Set the dimensions and radius
    const width = 400;
    const height = 400;
    const radius = Math.min(width, height) / 2;

    // Create an SVG container
    const svg = d3.select("#pieChart")
                  .append("g")
                  .attr("transform", `translate(${width / 2}, ${height
/ 2})`);

    // Create a color scale
    const color = d3.scaleOrdinal(d3.schemeCategory10);

    // Create the pie layout
    const pie = d3.pie().value(d => d.count);

    // Create the arc generator
    const arc = d3.arc()
                  .innerRadius(0)
                  .outerRadius(radius);

    // Draw the arcs
    svg.selectAll(".arc")
      .data(pie(formattedData))
      .enter().append("g")
      .attr("class", "arc")
      .append("path")
      .attr("d", arc)
```

```
      .style("fill", d => color(d.data.interaction));


    // Add labels to the arcs
    svg.selectAll(".arc")
      .append("text")
      .attr("transform", d => `translate(${arc.centroid(d)})`)
      .attr("dy", ".35em")
      .text(d => d.data.interaction);


    // Add title to the chart
    svg.append("text")
      .attr("x", 0)
      .attr("y", -radius - 10)
      .attr("class", "title")
      .text("Customer Service Interactions Distribution");
  }).catch(error => {
    console.error('Error loading the CSV file:', error);
  });
</script>


</body>
</html>
```

Line Plot:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Total Claims Line Plot by Age Group</title>
  <script src="https://d3js.org/d3.v6.min.js"></script>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: flex-start;
      flex-direction: column;
      margin: 0;
      padding: 20px;
    }
    .line {
      fill: none;
      stroke: steelblue;
      stroke-width: 2;
    }
    .axis {
      font-size: 12px;
    }
    .title {
      font-size: 16px;
      text-anchor: middle;
      font-weight: bold;
    }
  </style>
</head>
<body>
```

```
<svg id="linePlot" width="600" height="400"></svg>

<script>
  // Load the data
  d3.csv("insurance_data.csv").then(function(data) {
    // Define age groups and calculate total claims per age group
    const ageGroups = d3.groups(data, d => {
      const age = +d["Age"];
      if (age < 20) return "Under 20";
      else if (age < 30) return "20-29";
      else if (age < 40) return "30-39";
      else if (age < 50) return "40-49";
      else if (age < 60) return "50-59";
      else return "60 and above";
    });

    const totalClaimsByAgeGroup = ageGroups.map(([ageGroup, values])
=> {
      const totalClaims = d3.sum(values, d => +d["Claim History"]);
      return { ageGroup, totalClaims };
    });

    // Define the age groups in ascending order
    const orderedAgeGroups = [
      "Under 20",
      "20-29",
      "30-39",
      "40-49",
```

```
        "50-59",
        "60 and above"
      ];

      // Create a mapping of the age group to total claims
      const orderedData = orderedAgeGroups.map(ageGroup => {
        const dataPoint = totalClaimsByAgeGroup.find(d => d.ageGroup
=== ageGroup);
        return { ageGroup, totalClaims: dataPoint ?
dataPoint.totalClaims : 0 }; // Use 0 if no claims
      });

      console.log(orderedData); // Debugging line

      // Create line plot
      const svg = d3.select("#linePlot");
      const margin = { top: 40, right: 30, bottom: 60, left: 60 };
      const width = +svg.attr("width") - margin.left - margin.right;
      const height = +svg.attr("height") - margin.top - margin.bottom;

      const x = d3.scaleBand()
                  .domain(orderedAgeGroups) // Correctly order the age
groups
                  .range([0, width])
                  .padding(0.1); // Adjust padding as needed

      const y = d3.scaleLinear()
                  .domain([0, d3.max(orderedData, d => d.totalClaims)
|| 1]) // Ensure at least 1
                  .range([height, 0]);
```

```
      const line = d3.line()
                     .x(d => x(d.ageGroup) + x.bandwidth() / 2) //
Center the line on the band
                     .y(d => y(d.totalClaims));

      const lineGroup = svg.append("g")
                           .attr("transform",
`translate(${margin.left},${margin.top})`);

      // Append the line path
      lineGroup.append("path")
        .datum(orderedData) // Use the ordered data for the line
        .attr("class", "line")
        .attr("d", line);

      // Add axes
      lineGroup.append("g")
        .attr("class", "axis")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x));

      lineGroup.append("g")
        .attr("class", "axis")
        .call(d3.axisLeft(y));

      // Add title
      svg.append("text")
        .attr("x", (width / 2) + margin.left)
        .attr("y", margin.top / 2)
```

```
        .attr("class", "title")
        .text("Total Claims by Age Group Line Plot");


    }).catch(error => {
      console.error('Error loading the CSV file:', error);
    });
  </script>
</body>
</html>
```

Conclusion:

Through this experiment, we gained valuable insights into D3.js and its powerful capabilities for data visualization. We explored how to effectively plot various types of graphs, including bar charts, line plots, histograms, and more.