Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

(Autonomous Institute Affiliated to University of Mumbai)
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

Name: Tathagat Sengupta

UID: 2021300110

Experiment Design for Creating Visualizations using D3.js on a Finance Dataset

1. 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.

2. Steps to Perform the Experiment

1. Choose the Dataset:

- ○ Select a dataset related to Finance/Banking/Insurance/Credit. Example datasets:
  - ■ Bank loan data
  - ■ Insurance claims data
  - ■ Credit card transaction data
  - ■ Stock market historical data

2. Set Up Development Environment:

- ○ Install necessary tools:
  - ■ **D3.js**: A JavaScript library for data visualizations.
  - ■ Text editor/IDE (e.g., VS Code).
  - ■ Web server (e.g., Live Server plugin for VS Code) to view D3.js projects.

3. Data Preprocessing:

- ○ Clean and preprocess the data for visualization. Handle missing values, format date-time fields, and categorize data where needed.

4. Create Basic Visualizations (Using D3.js):

- ○ **Bar Chart:** Show the distribution of a categorical variable like loan types, insurance claims by category, etc.
- ○ **Pie Chart:** Show proportions for categories like credit card transaction types. ○

Histogram: Display the distribution of numerical data like loan amounts. ○ **Timeline Chart:** Visualize trends over time (e.g., stock prices over a year). ○ **Scatter Plot:** Display relationships between two numerical variables like income and loan amount.

○ **Bubble Plot:** Visualize multi-variable relationships (e.g., customer age, loan amount, and loan duration).

5. Create Advanced Visualizations (Using D3.js):

○ **Word Chart:** Analyze the frequency of words in text data (e.g., claims descriptions).

○ **Box and Whisker Plot:** Visualize the spread of financial data (e.g., claim amounts).

○ **Violin Plot:** Show the distribution of a variable (e.g., distribution of interest rates).

○ **Regression Plot (Linear/Nonlinear):** Explore the relationship between two variables (e.g., loan amount and interest rate).

○ **3D Chart:** Visualize multivariate data in 3D (e.g., loan amount, interest rate, and credit score).

○ **Jitter Plot:** Display scatter plot points with jitter to avoid overlap, especially for categorical data.

6. Perform Hypothesis Testing (Using Pearson Correlation Coefficient): ○ **Step 1:** Formulate a hypothesis (e.g., "There is a positive correlation between customer income and loan approval amount").

○ **Step 2:** Calculate the Pearson correlation coefficient to test the hypothesis.

○ **Step 3:** Interpret the results (if the coefficient is close to 1, there is a strong positive correlation).

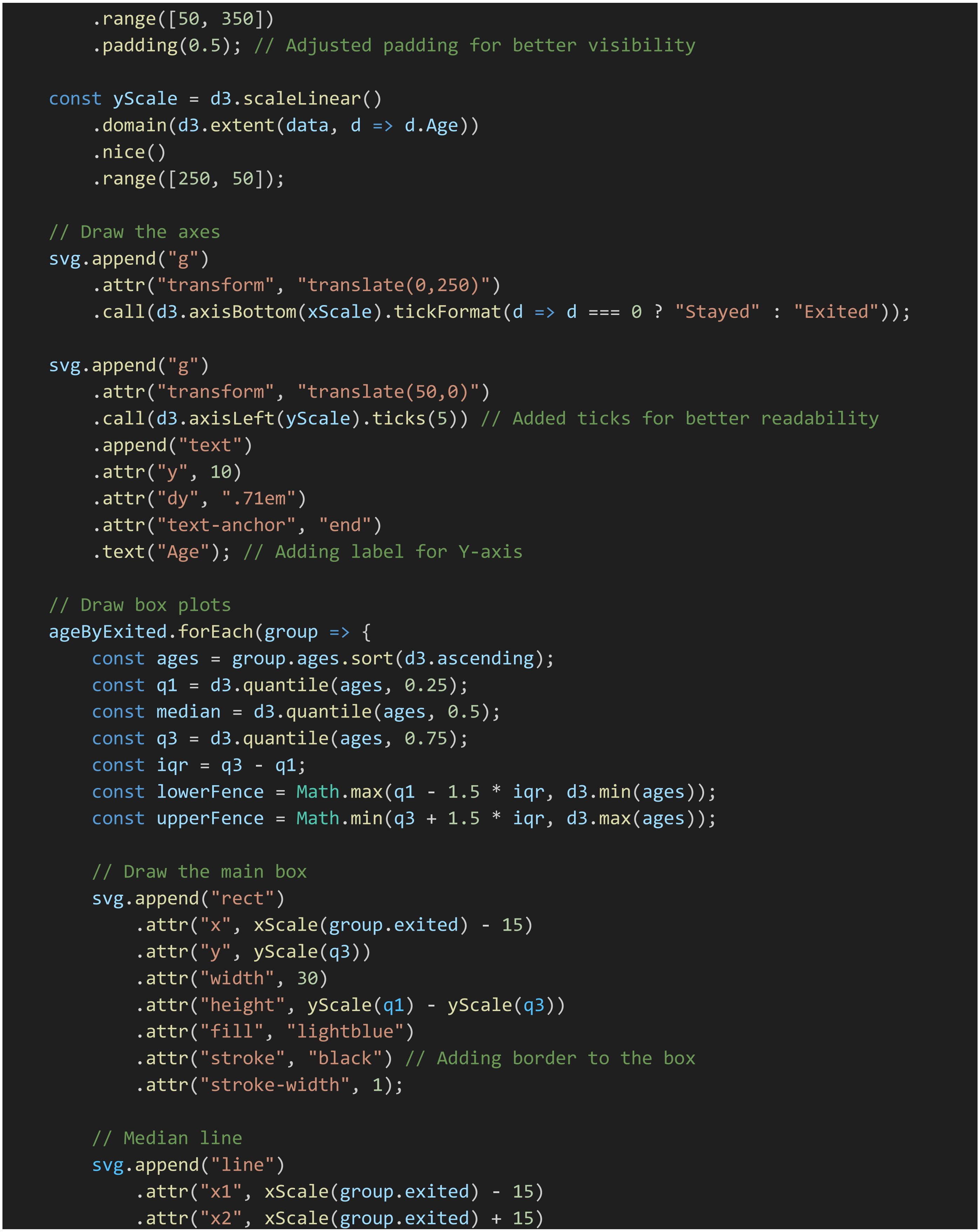
7. Write Observations:

○ Analyze each chart for trends, patterns, and outliers.

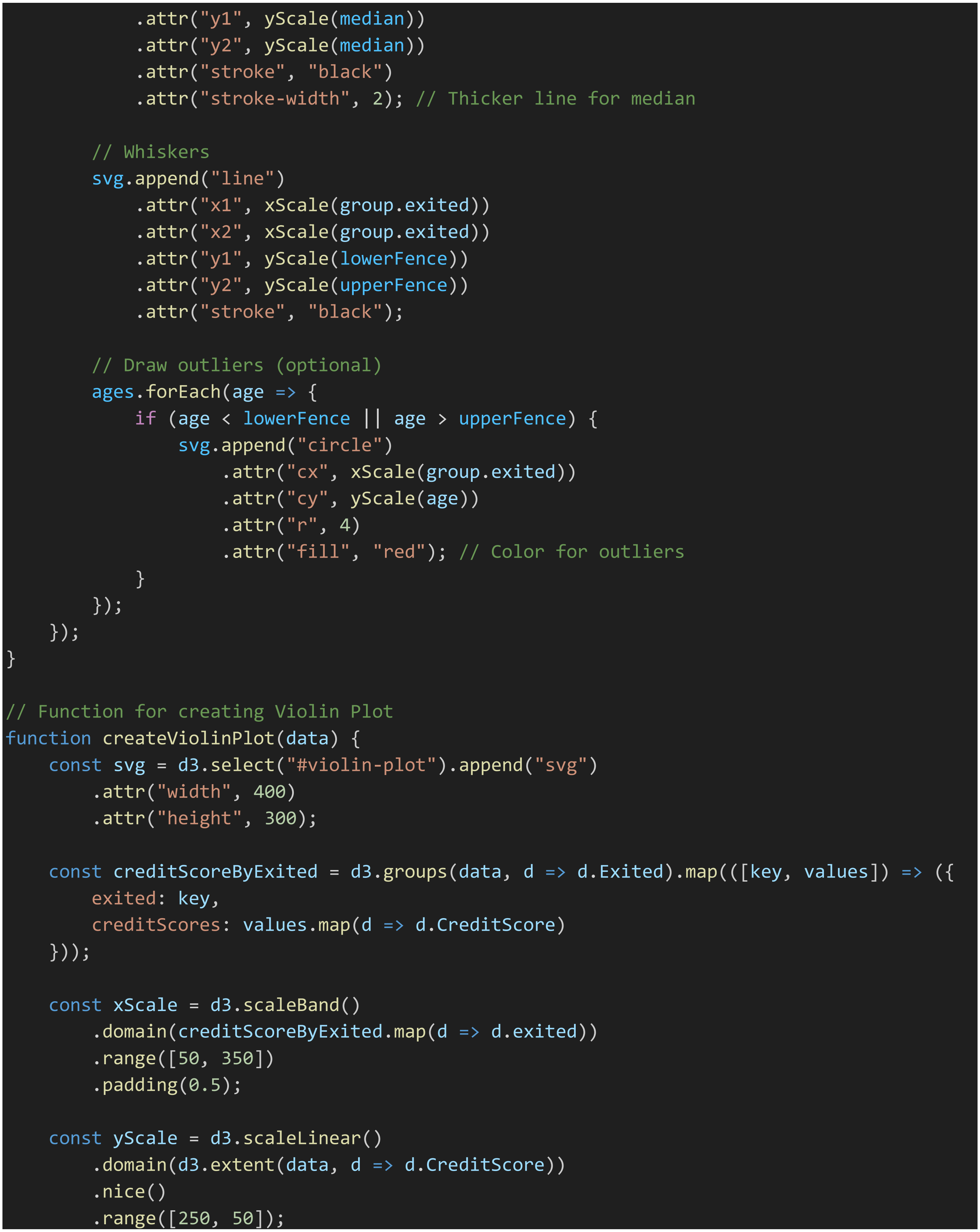
○ Identify insights that help in making business decisions (e.g., "Higher loan amounts are correlated with higher credit scores").

3. Programs and Code

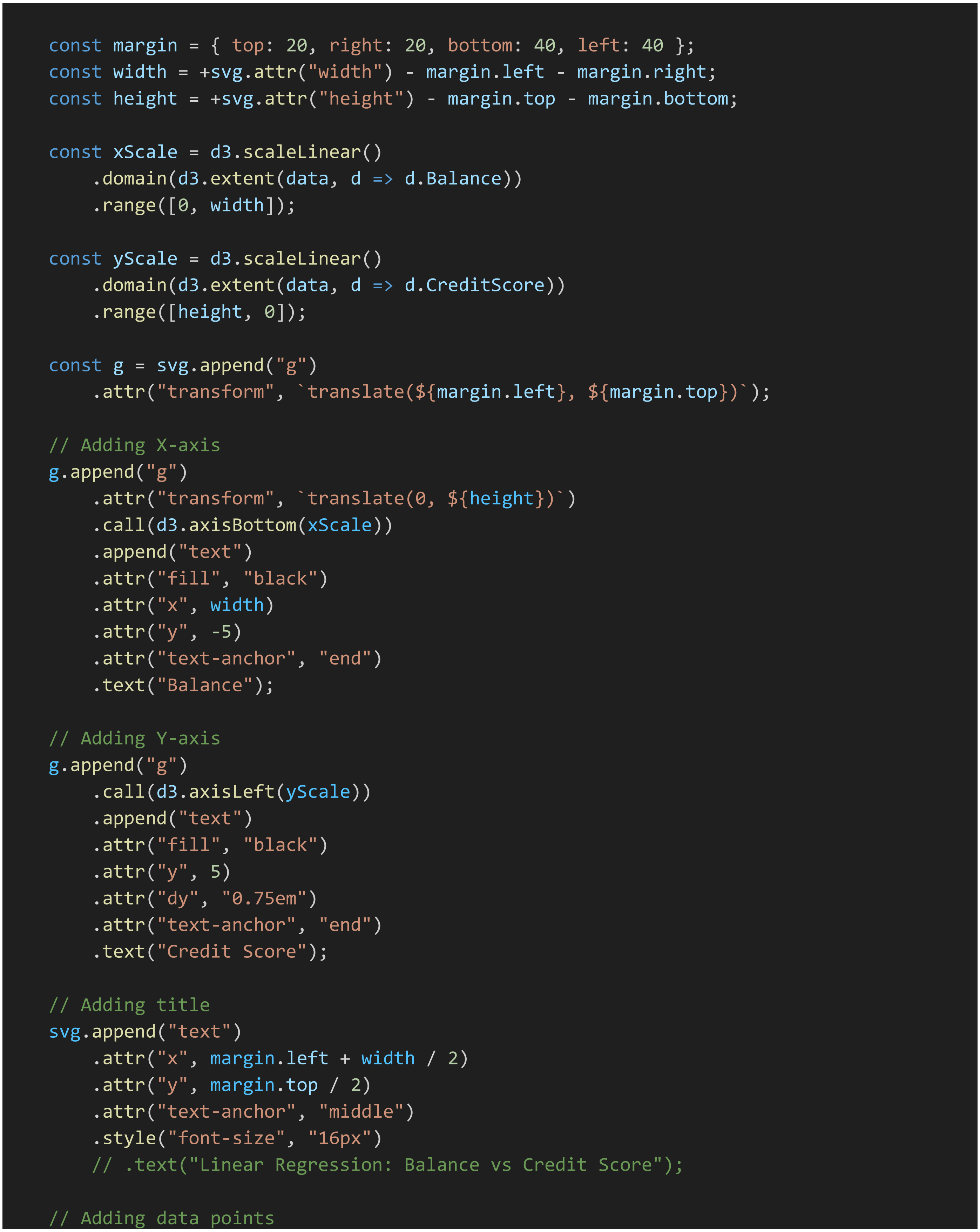
```
      // Load data from CSV
d3.csv("churn.csv").then(function(data) {

    // Parse numeric fields
    data.forEach(d => {
        d.CreditScore = +d.CreditScore;
        d.Age = +d.Age;
        d.Balance = +d.Balance;
        d.NumOfProducts = +d.NumOfProducts;
        d.EstimatedSalary = +d.EstimatedSalary;
        d.Exited = +d.Exited;
    });

    // Bar Chart - Geography vs. Exited
    createBarChart(data);
    // Pie Chart - Gender vs. Exited
    createPieChart(data);
    // Histogram - Age Distribution
    createHistogram(data);
    // Box Plot - Age vs. Exited
    createBoxPlot(data);
    // Violin Plot - CreditScore vs. Exited
    createViolinPlot(data);
    // Regression Plot - Balance vs. CreditScore
    createRegressionPlot(data);

    analyzeCorrelations(data);  // Call the function to analyze correlations
}).catch(error => {
    console.error("Error loading the CSV file:", error);
});
// Function for creating Bar Chart
function createBarChart(data) {
    const svg = d3.select("#bar-chart").append("svg")
        .attr("width", 400)
        .attr("height", 400);

    // Summarizing data based on geography and counting the total number of products
    const productsData = d3.rollup(data, v => d3.sum(v, d => d.NumOfProducts), d =>
d.Geography);

    const xScale = d3.scaleBand()
        .domain(Array.from(productsData.keys()))
        .range([0, 350])
        .padding(0.2);

    const yScale = d3.scaleLinear()
        .domain([0, d3.max(productsData.values())]) // Set the domain to the maximum sum
of products
        .range([250, 0]);

    // Drawing the bars
    svg.append("g")
        .selectAll("rect")
        .data(Array.from(productsData))
        .enter()
        .append("rect")
        .attr("x", d => xScale(d[0]))
        .attr("y", d => yScale(d[1]))
        .attr("width", xScale.bandwidth())
        .attr("height", d => 250 - yScale(d[1])) // Ensure proper height
        .attr("fill", "steelblue");

    // Adding the X-axis
    svg.append("g")
        .attr("transform", "translate(0,250)")
        .call(d3.axisBottom(xScale))
        .selectAll("text")
        .style("font-size", "12px"); // Styling X-axis labels

    // Adding the Y-axis
    const yAxis = svg.append("g")
        .attr("transform", "translate(0,0)")
        .call(d3.axisLeft(yScale))
        .selectAll("text")
        .attr("fill", "black") // Set text color
        .style("font-size", "12px"); // Set text size

    // Adding values on top of each bar
    svg.selectAll("rect")
        .data(Array.from(productsData))
        .enter()
        .append("text")
        .attr("x", d => xScale(d[0]) + xScale.bandwidth() / 2) // Centering text on bars
        .attr("y", d => yScale(d[1]) - 5) // Positioning text above the bars
        .attr("text-anchor", "middle")
        .text(d => d[1]) // Displaying the sum of products
        .style("fill", "black")
        .style("font-size", "10px");

    // Adding a legend for the Y-axis
    svg.append("text")
        .attr("x", 180)
        .attr("y", 20)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Total Number of Products");

    svg.append("text")
        .attr("x", 180)
        .attr("y", 35)
        .attr("text-anchor", "middle")
        .style("font-size", "10px")
        .text("by Country");
}



// Function for creating Pie Chart
function createPieChart(data) {
    const svg = d3.select("#pie-chart").append("svg").attr("width", 400).attr("height", 300);
    const radius = 150;
    const g = svg.append("g").attr("transform", "translate(200,150)"); // Center the pie
chart

    // Summarizing data by gender
    const pieData = d3.rollup(data, v => d3.sum(v, d => d.Exited), d => d.Gender);
    const colorScale = d3.scaleOrdinal().domain(pieData.keys()).range(["#ff9999",
"#66b3ff"]);

    // Creating pie chart data
    const pie = d3.pie().value(d => d[1])(Array.from(pieData));

    // Drawing the pie chart
    const arc = d3.arc().innerRadius(0).outerRadius(radius);

    g.selectAll("path")
        .data(pie)
        .enter()
        .append("path")
        .attr("d", arc)
        .attr("fill", d => colorScale(d.data[0]))
        .attr("stroke", "white")
        .attr("stroke-width", "2px")
        .on("mouseover", function(event, d) {
            const percentage = ((d.data[1] / d3.sum(Array.from(pieData.values()))) *
100).toFixed(2);
            d3.select(this)
                .transition()
                .duration(200)
                .attr("opacity", 0.7);
            svg.append("text")
                .attr("class", "tooltip")
                .attr("x", 200)
                .attr("y", 20)
                .attr("text-anchor", "middle")
                .style("font-size", "12px")
                .text(`${d.data[0]}: ${percentage}% (${d.data[1]})`);
        })
        .on("mouseout", function() {
            d3.select(this)
                .transition()
                .duration(200)
                .attr("opacity", 1);
            svg.select(".tooltip").remove(); // Remove the tooltip
        });
    // Adding labels for the pie slices
    g.selectAll("text")
        .data(pie)
        .enter()
        .append("text")
        .attr("transform", d => `translate(${arc.centroid(d)})`) // Positioning the text at the center of the slice
        .attr("dy", ".35em")
        .style("text-anchor", "middle")
        .text(d => `${d.data[0]}`); // Label by Gender

    // Adding title to the pie chart
    svg.append("text")
        .attr("x", 200)
        .attr("y", 320)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .style("font-weight", "bold")
        .text("Customer Exits by Gender");
}


// Function for creating Histogram
function createHistogram(data) {
    const svg = d3.select("#histogram").append("svg").attr("width", 400).attr("height", 300);

    // Set up x-scale for Age
    const xScale = d3.scaleLinear()
        .domain(d3.extent(data, d => d.Age))
        .range([50, 350]);  // Padding on left for y-axis

    // Generate histogram bins
    const histogram = d3.histogram()
        .domain(xScale.domain())
        .thresholds(xScale.ticks(10))
        .value(d => d.Age);
    const bins = histogram(data);

    // Set up y-scale for frequency
    const yScale = d3.scaleLinear()
        .domain([0, d3.max(bins, d => d.length)])
        .nice()  // Ensure the y-axis rounds to a nice number
        .range([250, 50]); // Inverted for proper orientation

    // Draw the bars
    svg.append("g")
        .selectAll("rect")
        .data(bins)
        .enter()
        .append("rect")
        .attr("x", d => xScale(d.x0))
        .attr("y", d => yScale(d.length))
        .attr("width", d => xScale(d.x1) - xScale(d.x0) - 1)
        .attr("height", d => 250 - yScale(d.length))
        .attr("fill", "orange");

    // Add x-axis
    svg.append("g")
        .attr("transform", "translate(0,250)")
        .call(d3.axisBottom(xScale).ticks(10));

    // Add y-axis
    svg.append("g")
        .attr("transform", "translate(50,0)") // Align y-axis with x-axis padding
        .call(d3.axisLeft(yScale).ticks(5))
        .append("text")
        .attr("fill", "black")
        .attr("x", -30)
        .attr("y", 15)
        .attr("dy", "-1em")
        .style("text-anchor", "end")
        // .text("Frequency");
}


// Similarly, define createBoxPlot, createViolinPlot, createRegressionPlot functions with
D3 syntax for advanced visualizations
// Function for creating Box Plot
function createBoxPlot(data) {
    const svg = d3.select("#box-plot").append("svg")
        .attr("width", 400)
        .attr("height", 300);

    // Group data by 'Exited' status
    const ageByExited = d3.groups(data, d => d.Exited).map(([key, values]) => ({
        exited: key,
        ages: values.map(d => d.Age)
    }));

    const xScale = d3.scaleBand()
        .domain(ageByExited.map(d => d.exited))
```



```
        .range([50, 350])
        .padding(0.5); // Adjusted padding for better visibility

    const yScale = d3.scaleLinear()
        .domain(d3.extent(data, d => d.Age))
        .nice()
        .range([250, 50]);

    // Draw the axes
    svg.append("g")
        .attr("transform", "translate(0,250)")
        .call(d3.axisBottom(xScale).tickFormat(d => d === 0 ? "Stayed" : "Exited"));

    svg.append("g")
        .attr("transform", "translate(50,0)")
        .call(d3.axisLeft(yScale).ticks(5)) // Added ticks for better readability
        .append("text")
        .attr("y", 10)
        .attr("dy", ".71em")
        .attr("text-anchor", "end")
        .text("Age"); // Adding label for Y-axis

    // Draw box plots
    ageByExited.forEach(group => {
        const ages = group.ages.sort(d3.ascending);
        const q1 = d3.quantile(ages, 0.25);
        const median = d3.quantile(ages, 0.5);
        const q3 = d3.quantile(ages, 0.75);
        const iqr = q3 - q1;
        const lowerFence = Math.max(q1 - 1.5 * iqr, d3.min(ages));
        const upperFence = Math.min(q3 + 1.5 * iqr, d3.max(ages));

        // Draw the main box
        svg.append("rect")
            .attr("x", xScale(group.exited) - 15)
            .attr("y", yScale(q3))
            .attr("width", 30)
            .attr("height", yScale(q1) - yScale(q3))
            .attr("fill", "lightblue")
            .attr("stroke", "black") // Adding border to the box
            .attr("stroke-width", 1);

        // Median line
        svg.append("line")
            .attr("x1", xScale(group.exited) - 15)
            .attr("x2", xScale(group.exited) + 15)
```
]

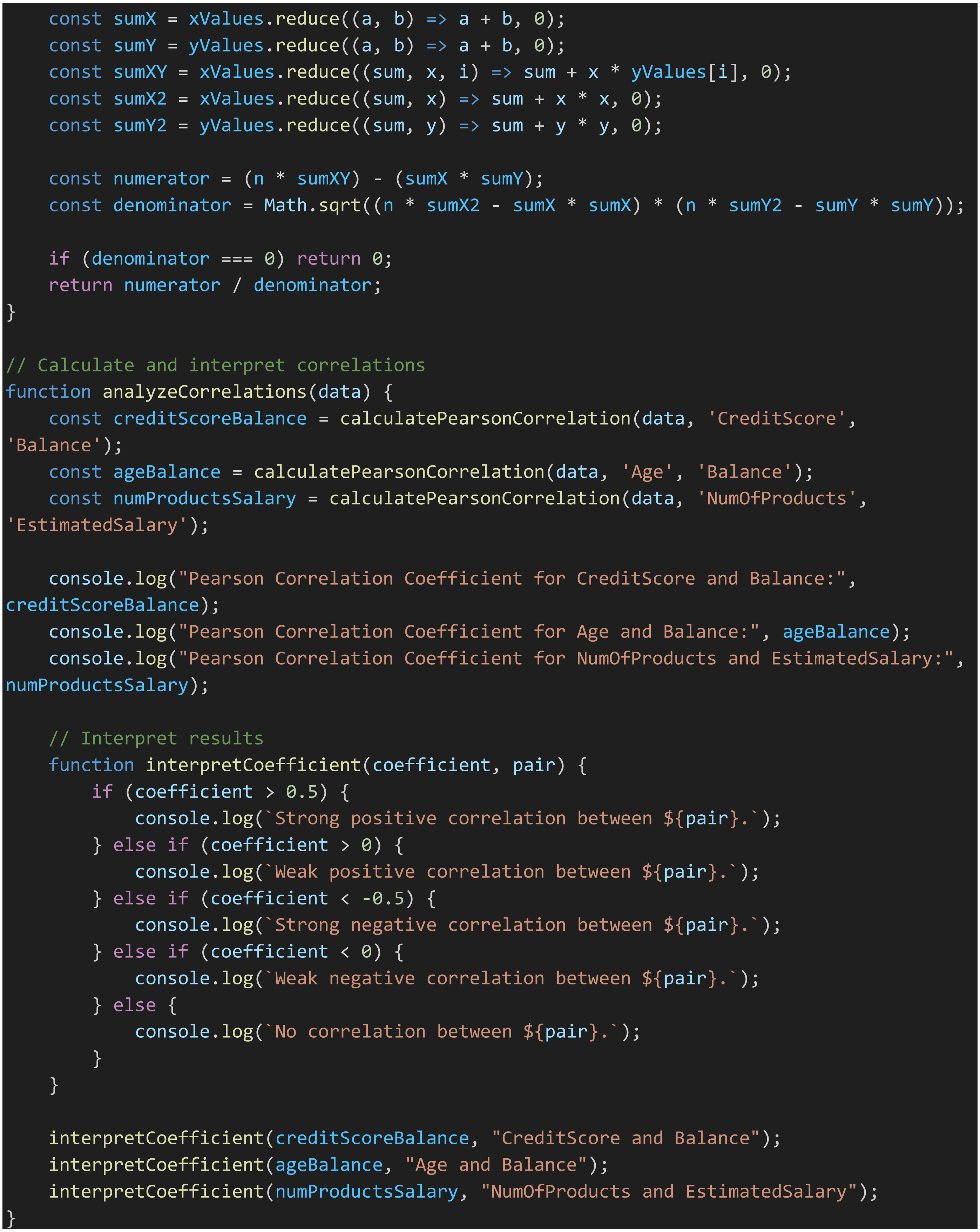


```
            .attr("y1", yScale(median))
            .attr("y2", yScale(median))
            .attr("stroke", "black")
            .attr("stroke-width", 2); // Thicker line for median

        // Whiskers
        svg.append("line")
            .attr("x1", xScale(group.exited))
            .attr("x2", xScale(group.exited))
            .attr("y1", yScale(lowerFence))
            .attr("y2", yScale(upperFence))
            .attr("stroke", "black");

        // Draw outliers (optional)
        ages.forEach(age => {
            if (age < lowerFence || age > upperFence) {
                svg.append("circle")
                    .attr("cx", xScale(group.exited))
                    .attr("cy", yScale(age))
                    .attr("r", 4)
                    .attr("fill", "red"); // Color for outliers
            }
        });
    });
}

// Function for creating Violin Plot
function createViolinPlot(data) {
    const svg = d3.select("#violin-plot").append("svg")
        .attr("width", 400)
        .attr("height", 300);

    const creditScoreByExited = d3.groups(data, d => d.Exited).map(([key, values]) => ({
        exited: key,
        creditScores: values.map(d => d.CreditScore)
    }));

    const xScale = d3.scaleBand()
        .domain(creditScoreByExited.map(d => d.exited))
        .range([50, 350])
        .padding(0.5);

    const yScale = d3.scaleLinear()
        .domain(d3.extent(data, d => d.CreditScore))
        .nice()
        .range([250, 50]);
    // Adding X-axis
    svg.append("g")
        .attr("transform", "translate(0,250)")
        .call(d3.axisBottom(xScale).tickFormat(d => d === 0 ? "Stayed" : "Exited"));

    // Adding Y-axis
    svg.append("g")
        .attr("transform", "translate(50,0)")
        .call(d3.axisLeft(yScale));

    // Adding Y-axis label
    svg.append("text")
        .attr("x", -40)
        .attr("y", 20)
        .attr("text-anchor", "end")
        .text("Credit Score");

    // Adding violins
    creditScoreByExited.forEach(group => {
        const density = kernelDensityEstimator(
            kernelEpanechnikov(7),
            yScale.ticks(30)
        )(group.creditScores);

        const maxDensity = d3.max(density, d => d[1]);

        const xViolin = d3.scaleLinear()
            .domain([-maxDensity, maxDensity])
            .range([xScale(group.exited) - 15, xScale(group.exited) + 15]);

        svg.append("path")
            .datum(density)
            .attr("fill", group.exited === 0 ? "lightblue" : "lightgreen") // Different colors for categories
            .attr("opacity", 0.7)
            .attr("d", d3.line()
                .curve(d3.curveBasis)
                .x(d => xViolin(d[1]))
                .y(d => yScale(d[0]))
            )
            .attr("stroke", "black") // Adding stroke for visibility
            .attr("stroke-width", 0.5);
    });

    // Legend
    const legend = svg.append("g")
        .attr("transform", "translate(300,20)");

    legend.append("rect")
        .attr("width", 10)
        .attr("height", 10)
        .attr("fill", "lightblue");

    legend.append("text")
        .attr("x", 15)
        .attr("y", 10)
        .text("Exited");

    legend.append("rect")
        .attr("y", 20)
        .attr("width", 10)
        .attr("height", 10)
        .attr("fill", "lightgreen");

    legend.append("text")
        .attr("x", 15)
        .attr("y", 30)
        .text("Stayed");

    // Kernel Density Estimator function
    function kernelDensityEstimator(kernel, X) {
        return function(V) {
            return X.map(x => [x, d3.mean(V, v => kernel(x - v))]);
        };
    }

    // Epanechnikov Kernel function
    function kernelEpanechnikov(k) {
        return function(v) {
            return Math.abs(v /= k) <= 1 ? 0.75 * (1 - v * v) / k : 0;
        };
    }
}



// Function for creating Regression Plot
function createRegressionPlot(data) {
    const svg = d3.select("#regression-plot").append("svg")
        .attr("width", 400)
        .attr("height", 300);
```



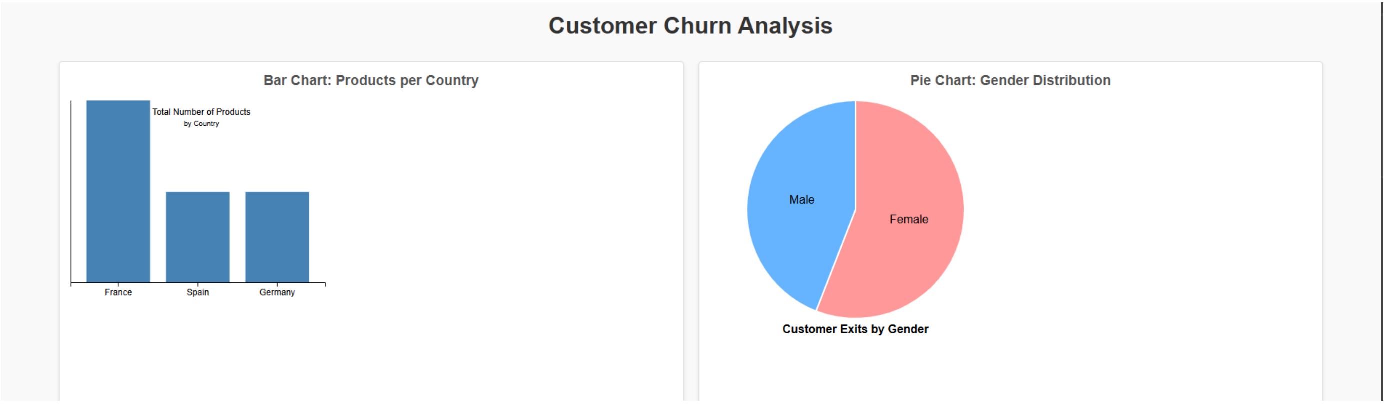
```
const margin = { top: 20, right: 20, bottom: 40, left: 40 };
const width = +svg.attr("width") - margin.left - margin.right;
const height = +svg.attr("height") - margin.top - margin.bottom;

const xScale = d3.scaleLinear()
    .domain(d3.extent(data, d => d.Balance))
    .range([0, width]);

const yScale = d3.scaleLinear()
    .domain(d3.extent(data, d => d.CreditScore))
    .range([height, 0]);

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

// Adding X-axis
g.append("g")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(xScale))
    .append("text")
    .attr("fill", "black")
    .attr("x", width)
    .attr("y", -5)
    .attr("text-anchor", "end")
    .text("Balance");

// Adding Y-axis
g.append("g")
    .call(d3.axisLeft(yScale))
    .append("text")
    .attr("fill", "black")
    .attr("y", 5)
    .attr("dy", "0.75em")
    .attr("text-anchor", "end")
    .text("Credit Score");

// Adding title
svg.append("text")
    .attr("x", margin.left + width / 2)
    .attr("y", margin.top / 2)
    .attr("text-anchor", "middle")
    .style("font-size", "16px")
    // .text("Linear Regression: Balance vs Credit Score");

// Adding data points
```

```
    g.selectAll("circle")
        .data(data)
        .enter()
        .append("circle")
        .attr("cx", d => xScale(d.Balance))
        .attr("cy", d => yScale(d.CreditScore))
        .attr("r", 3)
        .attr("fill", "blue")
        .attr("opacity", 0.5);

    // Calculate regression line
    const { slope, intercept } = linearRegression(data, d => d.Balance, d => d.CreditScore);

    // Draw regression line
    g.append("line")
        .attr("x1", xScale(d3.min(data, d => d.Balance)))
        .attr("y1", yScale(slope * d3.min(data, d => d.Balance) + intercept))
        .attr("x2", xScale(d3.max(data, d => d.Balance)))
        .attr("y2", yScale(slope * d3.max(data, d => d.Balance) + intercept))
        .attr("stroke", "red")
        .attr("stroke-width", 2);

    // Linear regression function
    function linearRegression(data, xAccessor, yAccessor) {
        const n = data.length;
        if (n === 0) return { slope: 0, intercept: 0 }; // Handle empty dataset

        const sumX = d3.sum(data, xAccessor);
        const sumY = d3.sum(data, yAccessor);
        const sumXY = d3.sum(data, d => xAccessor(d) * yAccessor(d));
        const sumX2 = d3.sum(data, d => xAccessor(d) ** 2);

        const slope = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX ** 2);
        const intercept = (sumY - slope * sumX) / n;

        return { slope, intercept };
    }
}

// Function to calculate Pearson correlation coefficient
function calculatePearsonCorrelation(data, xKey, yKey) {
    const xValues = data.map(d => d[xKey]);
    const yValues = data.map(d => d[yKey]);
    const n = xValues.length;
```



```
    const sumX = xValues.reduce((a, b) => a + b, 0);
    const sumY = yValues.reduce((a, b) => a + b, 0);
    const sumXY = xValues.reduce((sum, x, i) => sum + x * yValues[i], 0);
    const sumX2 = xValues.reduce((sum, x) => sum + x * x, 0);
    const sumY2 = yValues.reduce((sum, y) => sum + y * y, 0);

    const numerator = (n * sumXY) - (sumX * sumY);
    const denominator = Math.sqrt((n * sumX2 - sumX * sumX) * (n * sumY2 - sumY * sumY));

    if (denominator === 0) return 0;
    return numerator / denominator;
}

// Calculate and interpret correlations
function analyzeCorrelations(data) {
    const creditScoreBalance = calculatePearsonCorrelation(data, 'CreditScore',
'Balance');
    const ageBalance = calculatePearsonCorrelation(data, 'Age', 'Balance');
    const numProductsSalary = calculatePearsonCorrelation(data, 'NumOfProducts',
'EstimatedSalary');

    console.log("Pearson Correlation Coefficient for CreditScore and Balance:",
creditScoreBalance);
    console.log("Pearson Correlation Coefficient for Age and Balance:", ageBalance);
    console.log("Pearson Correlation Coefficient for NumOfProducts and EstimatedSalary:",
numProductsSalary);

    // Interpret results
    function interpretCoefficient(coefficient, pair) {
        if (coefficient > 0.5) {
            console.log(`Strong positive correlation between ${pair}.`);
        } else if (coefficient > 0) {
            console.log(`Weak positive correlation between ${pair}.`);
        } else if (coefficient < -0.5) {
            console.log(`Strong negative correlation between ${pair}.`);
        } else if (coefficient < 0) {
            console.log(`Weak negative correlation between ${pair}.`);
        } else {
            console.log(`No correlation between ${pair}.`);
        }
    }

    interpretCoefficient(creditScoreBalance, "CreditScore and Balance");
    interpretCoefficient(ageBalance, "Age and Balance");
    interpretCoefficient(numProductsSalary, "NumOfProducts and EstimatedSalary");
}
```

Output:

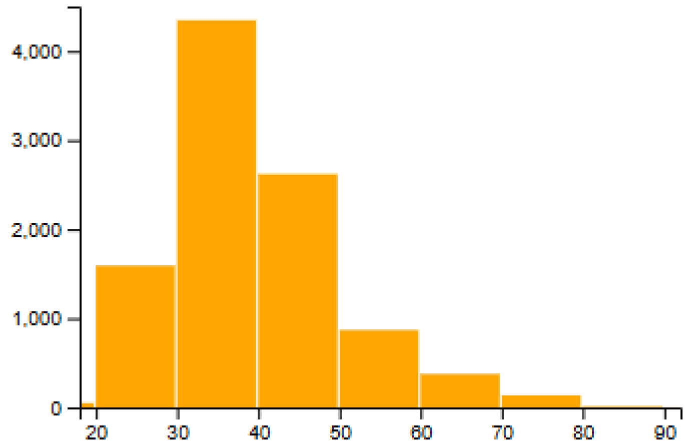


Customer Churn Analysis

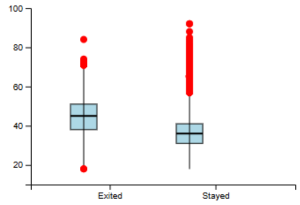
Bar Chart: Products per Country

Pie Chart: Gender Distribution

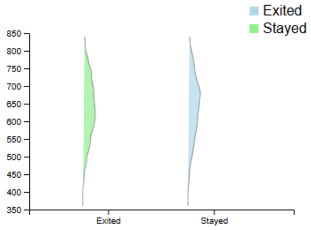
Histogram: Age Distribution



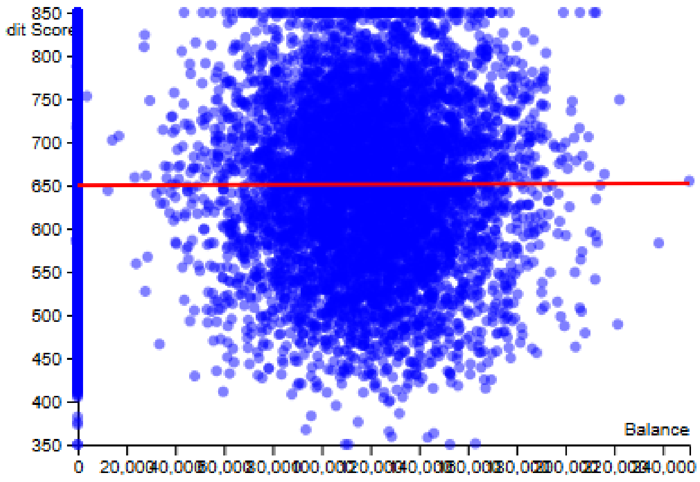
Box Plot: Balance by Tenure



Violin Plot: Credit scores by Customer exit status



Regression Plot: Credit Score vs Estimated Salary



Conclusion

The analysis of various visualizations reveals key insights into customer behavior and demographic trends:

1. Product Distribution by Country: France leads with the highest number of products, followed by Spain, while Germany has the lowest. This may suggest a correlation between the number of products and customer churn, as France also shows a significant number of customer exits.

2. Customer Gender Distribution: Approximately 60% of customers who exited are male, indicating a potential demographic trend worth exploring further.

3. Age Distribution: The age histogram indicates a right-skewed distribution, with a peak frequency in the 30-40 age group, suggesting a relatively young customer base. This demographic insight could inform targeted marketing and service planning.

4. Credit Score vs. Estimated Salary: The regression plot indicates a weak positive correlation between credit score and estimated salary, suggesting that higher salaries are associated with higher credit scores, although the relationship is influenced by various other factors.

5.Balance and Tenure: The box and violin plots reveal that customers who exited the bank tend to have higher balances but lower credit scores. This indicates a potential risk of losing customers who may be financially vulnerable or dissatisfied.

Recommendations

- Conduct further analyses on product types linked to higher churn rates and assess customer satisfaction to identify areas for improvement.
- Explore demographic factors influencing customer exits and consider simplifying product offerings for those with multiple products.
- Implement targeted surveys to gain deeper insights into the reasons behind customer churn, particularly among segments identified as high-risk.

These preliminary observations underscore the importance of understanding customer behavior and demographics for strategic decision-making, emphasizing the need for more detailed analyses to drive effective business strategies.