Sardar Patel Institute of Technology

SEM VII:ADVANCE DATA VISUALIZATION.

Name	Smruti Sonekar
UID no.	2021700064
Branch	BE CSE DS (BATCH B)
Experiment no.	7

Topic:	Data Visualization and Hypothesis Testing on Financial Data Using D3.js
Aim:	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.
Theory:	Financial Data : https://www.kaggle.com/datasets/zaurbegiev/my-dataset 1. Basic Bar Chart: - Importance: This chart shows the distribution of loan status among the borrowers. It provides a straightforward visualization of the counts or frequencies of different loan statuses, allowing the user to quickly understand the overall composition of the loan portfolio. 2. Pie Chart: - Importance: The pie chart illustrates the distribution of home ownership types among the borrowers. This information can be useful for understanding the demographics of the loan applicants and potentially identifying any relationships between home ownership and other loan characteristics.

3. Stacked Bar Chart:
- Importance: The stacked bar chart provides a more detailed view of the relationship between home ownership and loan status. By stacking the loan status counts for each home ownership type, the chart allows the user to see the relative distribution of loan statuses within each home ownership category.

4. Scatter Plot:
- Importance: The scatter plot visualizes the relationship between two key loan characteristics: years of credit history and monthly debt. This can help identify any potential correlations or trends between these variables, which may be useful for understanding credit risk or other factors influencing loan decisions.

5. Box Plot:
- Importance: The box plot shows the distribution of credit scores across different loan terms. This can provide insights into how credit scores vary based on the loan term, which may be relevant for understanding lending policies or assessing credit risk.

6. Bubble Chart:
- Importance: The bubble chart visualizes the relationship between annual income, current credit balance, and loan purpose. The size of the bubbles represents the current loan amount, providing a multidimensional view of these loan characteristics.

Overall, these charts provide a comprehensive set of visualizations that can help the user analyze various aspects of the loan data, such as the distribution of loan statuses, home ownership types, credit scores, and the relationships between key loan characteristics. By understanding these insights, the user can gain valuable perspectives on the loan portfolio and potentially identify areas for further investigation or decision-making.

In Hypothesis Testing, In this analysis, we examined the correlation between Annual Income and Current Loan Amount to test the hypothesis about their potential relationship.
Null Hypothesis (H0): There is no correlation between customer income and loan approval amount.
Alternative Hypothesis (H1): There is a positive correlation between customer income and loan approval amount.

Pearson Correlation Coefficient: `0.0314` — This is a very low positive value, indicating a negligible positive correlation between annual income and loan amount.

P-value: `5.03e-21` — This value is extremely low, suggesting that even though the correlation is weak, it is statistically significant.
Since the **p-value** is much less than the significance level (alpha: **0.05**), we reject

	the null hypothesis.

Program:

script.js code:

```
d3.csv("/loan_df.csv").then(data => {
    console.log(data);  // Confirm data is loaded
    // Call functions to create charts with the loaded data

    drawScatterPlot(data);
    drawPieChart(data);
    basicBarChart(data);
    drawStackedBarChart(data);
    drawCreditScoreBoxPlot(data);
    drawBubbleChart(data);
});

function basicBarChart(data) {
    const width = 400, height = 300, margin = { top: 40, right: 20, bottom: 60, left: 50 };

    // Create the SVG container and set dimensions
    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})`);

    // Process data for Loan Status counts
    const counts = d3.rollup(data, v => v.length, d => d["Loan Status"]);

    // Create X and Y scales
    const x = d3.scaleBand()
        .domain([...counts.keys()])
        .range([0, width])
        .padding(0.1);
```

```
const y = d3.scaleLinear()
    .domain([0, d3.max(counts.values())])
    .nice() // Adds padding to y-axis
    .range([height, 0]);

// Create X and Y axes
svg.append("g")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x))
    .selectAll("text")
    .attr("dy", ".75em")
    .attr("dx", "-.75em")
    .attr("transform", "rotate(-30)")
    .style("text-anchor", "end");

svg.append("g")
    .call(d3.axisLeft(y).ticks(5));

// Add bars
svg.selectAll(".bar")
    .data([...counts.entries()])
    .enter().append("rect")
    .attr("class", "bar")
    .attr("x", d => x(d[0]))
    .attr("y", d => y(d[1]))
    .attr("width", x.bandwidth())
    .attr("height", d => height - y(d[1]))
    .attr("fill", "salmon");

// Add title
svg.append("text")
    .attr("x", width / 2)
    .attr("y", -margin.top / 2)
    .attr("text-anchor", "middle")
    .style("font-size", "16px")
    .style("font-weight", "bold")
    .text("Loan Status Distribution");

// Add X-axis label
```

```
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Loan Status");

    // Add Y-axis label
    svg.append("text")
        .attr("transform", "rotate(-90)")
        .attr("y", -margin.left + 15)
        .attr("x", -height / 2)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Number of Loans");
}


function drawPieChart(data) {
    const width = 450, height = 450, margin = 40;
    const radius = Math.min(width, height) / 2 - margin;

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

    // Prepare data by calculating counts of each
ownership type
    const ownershipCounts = d3.rollup(data, v => v.length,
d => d["Home Ownership"]);
    const formattedData =
Object.fromEntries(ownershipCounts);

    // Set up color scale
    const color = d3.scaleOrdinal()
        .domain(Object.keys(formattedData))
        .range(d3.schemeSet2);

    // Generate pie and arc data
    const pie = d3.pie()
        .value(d => d[1])
        .sort(null);

    const data_ready = pie(Object.entries(formattedData));

    const arc = d3.arc()
        .innerRadius(0)
        .outerRadius(radius);

    // Draw slices
    svg.selectAll("path")
        .data(data_ready)
        .enter()
        .append("path")
        .attr("d", arc)
        .attr("fill", d => color(d.data[0]))
        .attr("stroke", "white")
        .style("stroke-width", "2px")
        .style("opacity", 0.7);

    // Add labels
    svg.selectAll("text")
        .data(data_ready)
        .enter()
        .append("text")
        .text(d => `${d.data[0]}: ${d.data[1]}`)
        .attr("transform", d =>
`translate(${arc.centroid(d)})`)
        .style("text-anchor", "middle")
        .style("font-size", "10px")
        .style("fill", "black");

    // Title for chart
    svg.append("text")
        .attr("x", 0)
        .attr("y", -height / 2 + margin-12)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .style("font-weight", "bold")
        .text("Home Ownership Distribution");

    // Tooltip
    const tooltip = d3.select("body").append("div")
        .attr("class", "tooltip")
        .style("position", "absolute")
        .style("padding", "8px")
        .style("background", "rgba(0, 0, 0, 0.7)")
        .style("color", "white")
        .style("border-radius", "4px")
        .style("display", "none");

    svg.selectAll("path")
        .on("mouseover", (event, d) => {
            tooltip.style("display", "block")
                .html(`${d.data[0]}: ${d.data[1]}`);
        })
        .on("mousemove", (event) => {
            tooltip.style("left", (event.pageX + 10) +
"px")
                .style("top", (event.pageY - 20) + "px");
        })
        .on("mouseout", () => tooltip.style("display",
"none"));
}


function drawStackedBarChart(data) {
    const margin = { top: 40, right: 30, bottom: 50, left:
60 };
    const width = 700 - margin.left - margin.right;
    const height = 400 - margin.top - margin.bottom;
    // Set up SVG container
    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})`);

    // Process data for stacked layout
    const groupedData = d3.groups(data, d => d["Home
Ownership"], d => d["Loan Status"])
        .map(d => ({
            "Home Ownership": d[0],
            ...Object.fromEntries(d[1].map(subgroup =>
[subgroup[0], subgroup[1].length]))
        }));

    const subgroups = Array.from(new Set(data.map(d =>
d["Loan Status"])));
    const ownershipTypes = Array.from(new Set(data.map(d
=> d["Home Ownership"])));

    // Create scales
    const x = d3.scaleBand()
        .domain(ownershipTypes)
        .range([0, width])
        .padding(0.3);

    const y = d3.scaleLinear()
        .domain([0, d3.max(groupedData, d =>
d3.sum(subgroups, key => d[key] || 0))])
        .nice()
        .range([height, 0]);

    const color = d3.scaleOrdinal()
        .domain(subgroups)
        .range(d3.schemeSet2);
```

```
// Stack data
const stackedData = d3.stack()
    .keys(subgroups)
    (groupedData);

// Draw stacked bars
svg.append("g")
    .selectAll("g")
    .data(stackedData)
    .enter().append("g")
    .attr("fill", d => color(d.key))
    .selectAll("rect")
    .data(d => d)
    .enter().append("rect")
    .attr("x", d => x(d.data["Home Ownership"]))
    .attr("y", d => y(d[1]))
    .attr("height", d => y(d[0]) - y(d[1]))
    .attr("width", x.bandwidth());

// Add axes
svg.append("g")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x))
    .selectAll("text")
    .style("text-anchor", "middle");

svg.append("g")
    .call(d3.axisLeft(y));

// Add title and labels
svg.append("text")
    .attr("x", width / 2)
    .attr("y", -10)
    .attr("text-anchor", "middle")
    .style("font-size", "16px")
    .text("Home Ownership by Loan Status");

svg.append("text")
```

```
        .attr("x", -height / 2)
        .attr("y", -margin.left + 15)
        .attr("transform", "rotate(-90)")
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Count of Loans");

    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Home Ownership");

    // Tooltip
    const tooltip = d3.select("body").append("div")
        .style("position", "absolute")
        .style("padding", "8px")
        .style("background", "rgba(0, 0, 0, 0.6)")
        .style("color", "white")
        .style("border-radius", "4px")
        .style("display", "none");

    svg.selectAll("rect")
        .on("mouseover", (event, d) => {
            tooltip.style("display", "block")
                .html(`Home Ownership: ${d.data["Home
Ownership"]}<br>Loan Status:
${d3.select(event.target.parentNode).datum().key}<br>Count
: ${d[1] - d[0]}`);
        })
        .on("mousemove", (event) => {
            tooltip.style("left", (event.pageX + 10) +
"px")
                .style("top", (event.pageY - 20) + "px");
        })
        .on("mouseout", () => tooltip.style("display",
"none"));
}
```

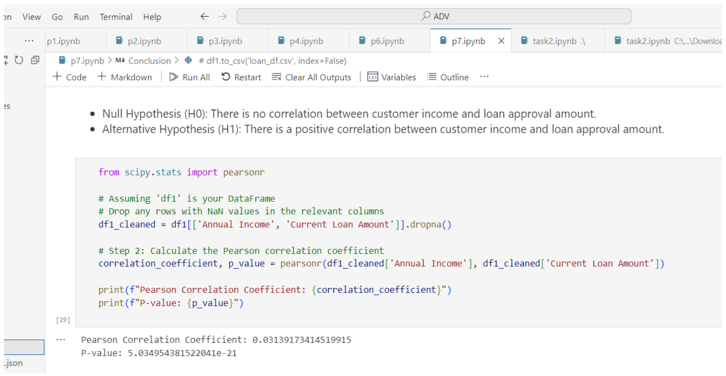
```
function drawScatterPlot(data) {
    const margin = { top: 30, right: 30, bottom: 60, left:
60 };
    const width = 700 - margin.left - margin.right;
    const height = 400 - margin.top - margin.bottom;

    // Set up SVG container
    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})`);

    // Define scales
    const x = d3.scaleLinear()
        .domain([0, d3.max(data, d => +d["Years of Credit
History"])])
        .range([0, width])
        .nice();

    const y = d3.scaleLinear()
        .domain([0, d3.max(data, d => +d["Monthly
Debt"])])
        .range([height, 0])
        .nice();

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

    svg.append("g")
        .call(d3.axisLeft(y));
    // Add axis labels
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Years of Credit History");

    svg.append("text")
        .attr("x", -height / 2)
        .attr("y", -margin.left + 15)
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .style("font-size", "12px")
        .text("Monthly Debt");

    // Title
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", -10)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .text("Scatter Plot of Monthly Debt vs. Years of
Credit History");

    // Add scatter plot points
    svg.append("g")
        .selectAll("dot")
        .data(data)
        .enter().append("circle")
        .attr("cx", d => x(d["Years of Credit History"]))
        .attr("cy", d => y(d["Monthly Debt"]))
        .attr("r", 5)
        .style("fill", "orange");

    // Tooltip
    const tooltip = d3.select("body").append("div")
        .style("position", "absolute")
        .style("padding", "8px")
        .style("background", "rgba(0, 0, 0, 0.7)")
        .style("color", "white")
        .style("border-radius", "4px")
        .style("display", "none");

    svg.selectAll("circle")
        .on("mouseover", (event, d) => {
            tooltip.style("display", "block")
                .html(`Years of Credit History: ${d["Years of Credit History"]}<br>Monthly Debt: $${d["Monthly Debt"]}`);
        })
        .on("mousemove", (event) => {
            tooltip.style("left", (event.pageX + 10) + "px")
                .style("top", (event.pageY - 20) + "px");
        })
        .on("mouseout", () => tooltip.style("display", "none"));
}

function drawCreditScoreBoxPlot(data) {
    const margin = { top: 30, right: 30, bottom: 60, left: 80 };
    const width = 700 - margin.left - margin.right;
    const height = 400 - margin.top - margin.bottom;

    // Create SVG container
    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})`);

    // Group data by "Term" and calculate quartiles
    const groupedData = d3.group(data, d => d["Term"]);
    const summaryData = Array.from(groupedData, ([term,
values]) => {
        const creditScores = values.map(d => +d["Credit
Score"]).sort(d3.ascending);
        const q1 = d3.quantile(creditScores, 0.25);
        const median = d3.quantile(creditScores, 0.5);
        const q3 = d3.quantile(creditScores, 0.75);
        const iqr = q3 - q1;
        const min = d3.min(creditScores);
        const max = d3.max(creditScores);
        return { term, q1, median, q3, iqr, min, max };
    });

    // X-axis scale for "Term"
    const x = d3.scaleBand()
        .domain(summaryData.map(d => d.term))
        .range([0, width])
        .padding(0.2);

    // Y-axis scale for "Credit Score"
    const y = d3.scaleLinear()
        .domain([0, d3.max(summaryData, d => d.max) *
1.1])
        .range([height, 0]);

    // Add X and Y axis
    svg.append("g")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x))
        .selectAll("text")
        .attr("transform", "rotate(-45)")
        .style("text-anchor", "end");

    svg.append("g").call(d3.axisLeft(y));

    // Add axis labels
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Loan Term");

    svg.append("text")
        .attr("x", -height / 2)
        .attr("y", -margin.left + 15)
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .style("font-size", "12px")
        .text("Credit Score");

    // Title
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", -10)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .text("Box Plot of Credit Score by Loan Term");

    // Draw box plots
    svg.selectAll(".box")
        .data(summaryData)
        .enter().append("rect")
        .attr("x", d => x(d.term))
        .attr("y", d => y(d.q3))
        .attr("height", d => y(d.q1) - y(d.q3))
        .attr("width", x.bandwidth())
        .attr("fill", "#4682b4");

    // Add median lines
    svg.selectAll(".medianLine")
        .data(summaryData)
        .enter().append("line")
        .attr("x1", d => x(d.term))
        .attr("x2", d => x(d.term) + x.bandwidth())
        .attr("y1", d => y(d.median))
        .attr("y2", d => y(d.median))
        .attr("stroke", "black")
```

```
        .attr("stroke-width", 2);

    // Add whiskers (min and max lines)
    svg.selectAll(".whisker")
        .data(summaryData)
        .enter().append("line")
        .attr("x1", d => x(d.term) + x.bandwidth() / 2)
        .attr("x2", d => x(d.term) + x.bandwidth() / 2)
        .attr("y1", d => y(d.min))
        .attr("y2", d => y(d.q1))
        .attr("stroke", "black");

    svg.selectAll(".whisker")
        .data(summaryData)
        .enter().append("line")
        .attr("x1", d => x(d.term) + x.bandwidth() / 2)
        .attr("x2", d => x(d.term) + x.bandwidth() / 2)
        .attr("y1", d => y(d.q3))
        .attr("y2", d => y(d.max))
        .attr("stroke", "black");

    // Outliers as individual circles
    svg.selectAll(".outlier")
        .data(data.filter(d => +d["Credit Score"] <
d3.min(summaryData, s => s.q1) || +d["Credit Score"] >
d3.max(summaryData, s => s.q3)))
        .enter().append("circle")
        .attr("cx", d => x(d["Term"]) + x.bandwidth() / 2)
        .attr("cy", d => y(d["Credit Score"]))
        .attr("r", 3)
        .attr("fill", "red");
}


function drawBubbleChart(data) {
    const margin = { top: 40, right: 40, bottom: 60, left:
60 };
    const width = 600 - margin.left - margin.right;
    const height = 500 - margin.top - margin.bottom;
    // Create SVG container
    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})`);

    // Scales for x (Annual Income) and y (Current Credit
Balance)
    const x = d3.scaleLog()
        .domain([80000, 165557393.0])  // Adjusted min and
max values
        .range([0, width]);

    const y = d3.scaleLinear()
        .domain([0, 32878968.0])  // Adjusted max for
Current Credit Balance
        .range([height, 0]);

    // Custom color scale for specified "Purpose"
categories
    const purposeColors = {
        "Debt Consolidation": "#1f77b4",
        "Other": "#ff7f0e",
        "Home Improvement": "#2ca02c",
        "Business": "#d62728",
        "Vehicle Purchase": "#9467bd",
        "Medical Expenses": "#8c564b",
        "Home Purchase": "#e377c2",
        "Vacation": "#7f7f7f",
        "Major Purchase": "#bcbd22"
    };
    const color = d => purposeColors[d["Purpose"]] ||
"#17becf"; // Default color if not listed
```

```
    // Bubble size scale based on "Current Loan Amount"
    const radius = d3.scaleSqrt()
        .domain([0, d3.max(data, d => +d["Current Loan
Amount"])])
        .range([2, 10]);

    // X-axis
    svg.append("g")
        .attr("transform", `translate(0, ${height})`)
        .call(d3.axisBottom(x).ticks(10, "~s"));

    // Y-axis
    svg.append("g")
        .call(d3.axisLeft(y).ticks(10).tickFormat(d =>
`$${d / 1000}k`));

    // Add X-axis label
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", height + margin.bottom - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text("Annual Income ($)");

    // Add Y-axis label
    svg.append("text")
        .attr("x", -height / 2)
        .attr("y", -margin.left + 15)
        .attr("text-anchor", "middle")
        .attr("transform", "rotate(-90)")
        .style("font-size", "12px")
        .text("Current Credit Balance ($)");

    // Title
    svg.append("text")
        .attr("x", width / 2)
        .attr("y", -10)
        .attr("text-anchor", "middle")
        .style("font-size", "16px")
        .text("Annual Income vs. Current Credit Balance by
Purpose");

    // Create bubbles with slight jitter to reduce overlap
    svg.selectAll("circle")
        .data(data)
        .enter().append("circle")
        .attr("cx", d => x(d["Annual Income"]))
        .attr("cy", d => y(d["Current Credit Balance"]) +
(Math.random() - 0.5) * 10)
        .attr("r", d => radius(d["Current Loan Amount"]))
        .attr("fill", d => color(d))
        .attr("opacity", 0.7)
        .attr("stroke", "black");

    // Legend for Purpose categories
    const legend = svg.selectAll(".legend")
        .data(Object.keys(purposeColors))
        .enter().append("g")
        .attr("class", "legend")
        .attr("transform", (d, i) => `translate(10,${i *
20})`);

    legend.append("rect")
        .attr("x", width - 20)
        .attr("width", 12)
        .attr("height", 12)
        .style("fill", d => purposeColors[d]);

    legend.append("text")
        .attr("x", width - 25)
        .attr("y", 9)
        .attr("dy", ".35em")
        .style("text-anchor", "end")
        .style("font-size", "10px")
        .text(d => d);
}
```

Result:

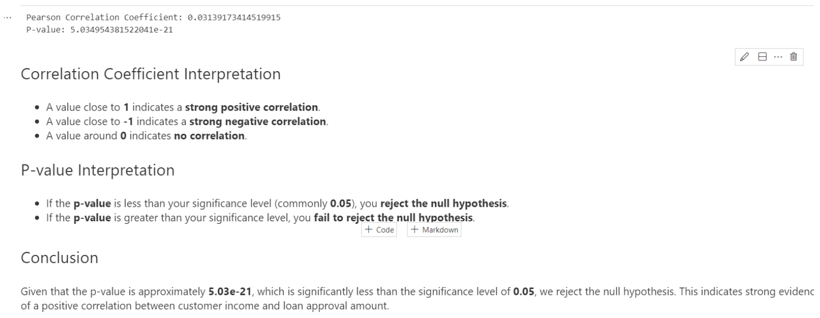
Hypothesis Testing:



- Null Hypothesis (H0): There is no correlation between customer income and loan approval amount.
- Alternative Hypothesis (H1): There is a positive correlation between customer income and loan approval amount.

```python
from scipy.stats import pearsonr

# Assuming 'df1' is your DataFrame
# Drop any rows with NaN values in the relevant columns
df1_cleaned = df1[['Annual Income', 'Current Loan Amount']].dropna()

# Step 2: Calculate the Pearson correlation coefficient
correlation_coefficient, p_value = pearsonr(df1_cleaned['Annual Income'], df1_cleaned['Current Loan Amount'])

print(f"Pearson Correlation Coefficient: {correlation_coefficient}")
print(f"P-value: {p_value}")
```

```
Pearson Correlation Coefficient: 0.031391734145199 15
P-value: 5.034954381522041e-21
```



```
Pearson Correlation Coefficient: 0.031391734145199 15
P-value: 5.034954381522041e-21
```

Correlation Coefficient Interpretation

- A value close to **1** indicates a **strong positive correlation**.
- A value close to **-1** indicates a **strong negative correlation**.
- A value around **0** indicates **no correlation**.

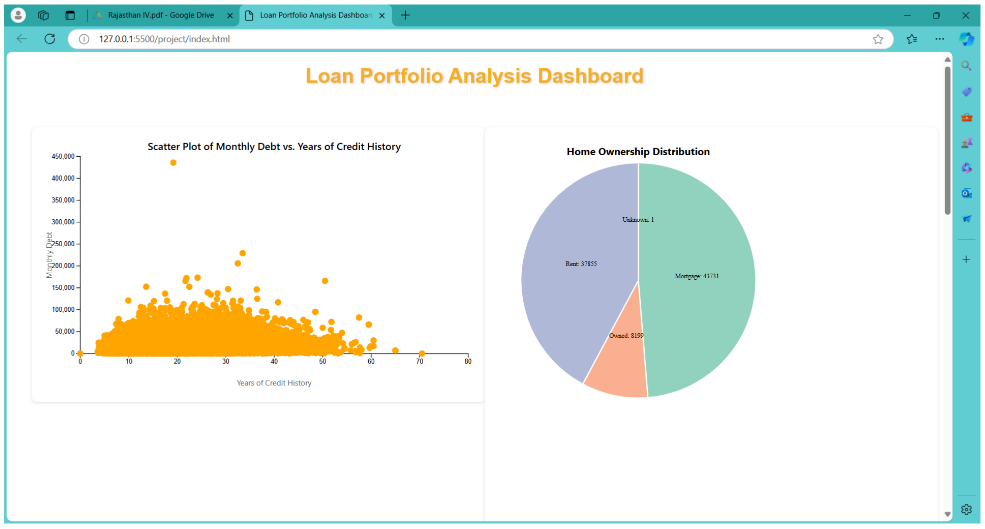
P-value Interpretation

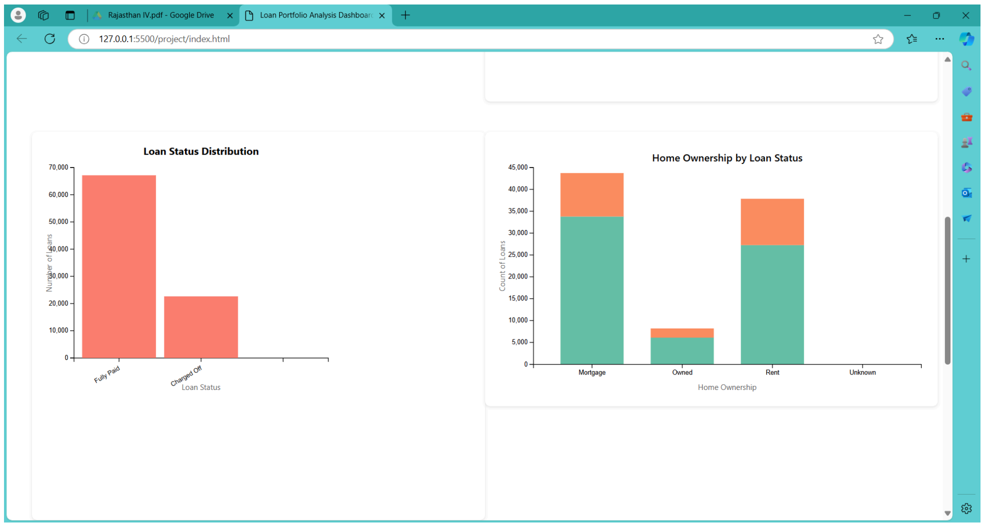
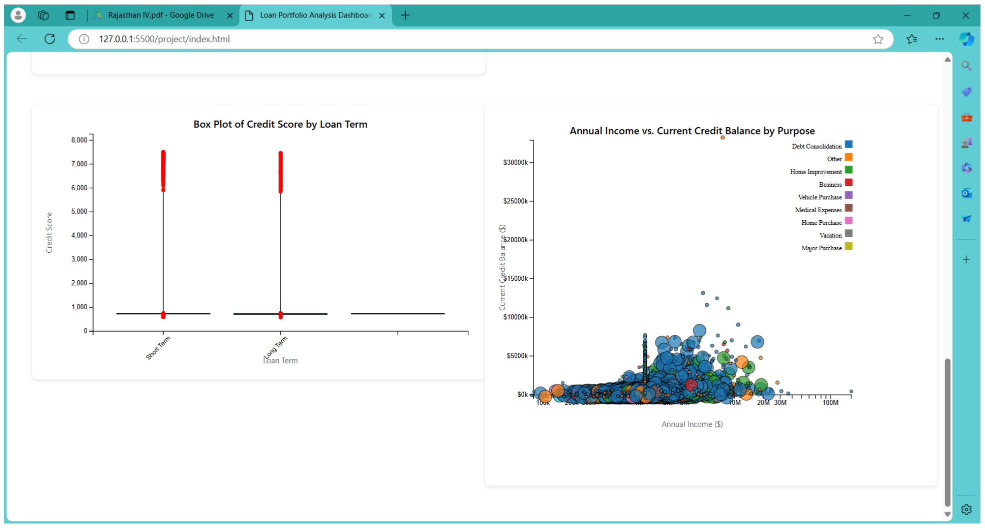
- If the **p-value** is less than your significance level (commonly **0.05**), you **reject the null hypothesis**.
- If the **p-value** is greater than your significance level, you **fail to reject the null hypothesis**.

Conclusion

Given that the p-value is approximately **5.03e-21**, which is significantly less than the significance level of **0.05**, we reject the null hypothesis. This indicates strong evidenc of a positive correlation between customer income and loan approval amount.

DASHBOARD


Rajasthan IV.pdf - Google Drive
Loan Portfolio Analysis Dashboard
127.0.0.1:5500/project/index.html
Loan Portfolio Analysis Dashboard
Scatter Plot of Monthly Debt vs. Years of Credit History
Monthly Debt
450,000
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0
0
10
20
30
40
50
60
70
80
Years of Credit History
Home Ownership Distribution
Unknown: 1
Rent: 37855
Mortgage: 43731
Owned: 8199

	 
Conclusion:	Learnt about D3.js in details and implemented in this experiment and also studied about hypothesis testing and its importance.