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

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. Charts Code

```
<!DOCTYPE html>
<html>
<head>
    <title>Investment Analysis Dashboard</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/d3/7.8.5/d3.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f5f5f5;
        }

        .dashboard {
            display: grid;
            grid-template-columns: repeat(2, 1fr);
            gap: 20px;
            padding: 20px;
            max-width: 1200px;
            margin: 0 auto;
        }

        .chart-container {
            background: #fff;
            border-radius: 8px;
            padding: 15px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }

        .title {
            font-size: 18px;
```

```
            font-weight: bold;
            margin-bottom: 15px;
            text-align: center;
            color: #333;
        }

        .tooltip {
            position: absolute;
            padding: 8px;
            background: rgba(0,0,0,0.8);
            color: #fff;
            border-radius: 4px;
            font-size: 12px;
            pointer-events: none;
        }

        .file-input-container {
            text-align: center;
            margin: 20px 0;
        }

        .file-input-container input {
            padding: 10px;
            border: 2px solid #ddd;
            border-radius: 4px;
        }

        .loading {
            text-align: center;
            color: #666;
            margin: 20px 0;
            display: none;
        }

        svg {
            max-width: 100%;
            height: auto;
        }
    </style>
</head>
<body>
 =<h1 style="text-align: center; margin: 20px 0; color: #333;">Investment Analysis
Dashboard</h1>

 <div class="file-input-container">
     <input type="file" id="csvFile" accept=".csv" />
 </div>
```

```
<div id="loading" class="loading">Loading data...</div>

<div class="dashboard">
    <div class="chart-container">
        <div class="title">Gender Distribution</div>
        <div id="gender-chart"></div>
    </div>
    <div class="chart-container">
        <div class="title">Age Distribution</div>
        <div id="age-chart"></div>
    </div>
    <div class="chart-container">
        <div class="title">Investment Preferences</div>
        <div id="investment-preferences"></div>
    </div>
    <div class="chart-container">
        <div class="title">Expected Returns Distribution</div>
        <div id="expected-returns"></div>
    </div>
    <div class="chart-container">
        <div class="title">Investment Duration</div>
        <div id="investment-duration"></div>
    </div>
    <div class="chart-container">
        <div class="title">Investment Objectives</div>
        <div id="investment-objectives"></div>
    </div>
    <div class="chart-container">
        <div class="title">Investment Distribution (Box Plot)</div>
        <div id="box-plot"></div>
    </div>
    <div class="chart-container">
        <div class="title">Age vs Investment Preference (Violin Plot)</div>
        <div id="violin-plot"></div>
    </div>
    <div class="chart-container">
        <div class="title">Age vs Investment Preference (Regression)</div>
        <div id="regression-plot"></div>
    </div>
    <div class="chart-container">
        <div class="title">3D Investment Preference Analysis</div>
        <div id="scatter-3d"></div>
    </div>
</div>


<script>
```

```
// Set up the file input handler
document.getElementById('csvFile').addEventListener('change', handleFileSelect);

function handleFileSelect(event) {
    const file = event.target.files[0];
    const loading = document.getElementById('loading');

    if (file) {
        loading.style.display = 'block';
        const reader = new FileReader();

        reader.onload = function(e) {
            const text = e.target.result;
            const data = d3.csvParse(text, d3.autoType);
            clearCharts();
            createAllCharts(data);
            loading.style.display = 'none';
        };

        reader.readAsText(file);
    }
}

function clearCharts() {
    d3.selectAll("svg").remove();
}

function createGenderChart(data) {
    const width = 300;
    const height = 300;
    const radius = Math.min(width, height) / 2;

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

    const genderCount = d3.group(data, d => d.gender);
    const pieData = Array.from(genderCount, ([key, value]) => ({
        gender: key,
        count: value.length
    }));

    const color = d3.scaleOrdinal()
        .domain(pieData.map(d => d.gender))
```

```
        .range(["#FF6B6B", "#4ECDC4"]);

    const pie = d3.pie()
        .value(d => d.count);

    const arc = d3.arc()
        .innerRadius(0)
        .outerRadius(radius - 40);

    const arcs = svg.selectAll("arc")
        .data(pie(pieData))
        .enter()
        .append("g");

    // Add tooltip
    const tooltip = d3.select("body").append("div")
        .attr("class", "tooltip")
        .style("opacity", 0);

    arcs.append("path")
        .attr("d", arc)
        .attr("fill", d => color(d.data.gender))
        .attr("stroke", "white")
        .style("stroke-width", "2px")
        .on("mouseover", function(event, d) {
            tooltip.transition()
                .duration(200)
                .style("opacity", .9);
            tooltip.html(`${d.data.gender}: ${d.data.count}`)
                .style("left", (event.pageX) + "px")
                .style("top", (event.pageY - 28) + "px");
        })
        .on("mouseout", function(d) {
            tooltip.transition()
                .duration(500)
                .style("opacity", 0);
        });

    // Add labels
    arcs.append("text")
        .attr("transform", d => `translate(${arc.centroid(d)})`)
        .attr("text-anchor", "middle")
        .text(d => `${d.data.gender}`);
}

function createAgeChart(data) {
    const margin = {top: 20, right: 20, bottom: 30, left: 40};
```

```
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#age-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})`);

    const x = d3.scaleLinear()
        .domain([d3.min(data, d => d.age) - 2, d3.max(data, d => d.age) + 2])
        .range([0, width]);

    const histogram = d3.histogram()
        .value(d => d.age)
        .domain(x.domain())
        .thresholds(x.ticks(10));

    const bins = histogram(data);

    const y = d3.scaleLinear()
        .domain([0, d3.max(bins, d => d.length)])
        .range([height, 0]);

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

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

    svg.selectAll("rect")
        .data(bins)
        .enter()
        .append("rect")
        .attr("x", d => x(d.x0) + 1)
        .attr("width", d => Math.max(0, x(d.x1) - x(d.x0) - 1))
        .attr("y", d => y(d.length))
        .attr("height", d => height - y(d.length))
        .style("fill", "#69b3a2")
        .style("opacity", 0.8);
}

function createInvestmentPreferencesChart(data) {
    const margin = {top: 20, right: 20, bottom: 70, left: 40};
    const width = 400 - margin.left - margin.right;
```

```
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#investment-preferences")
        .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})`);

    const investmentTypes = ['Mutual_Funds', 'Equity_Market', 'Debentures',
                             'Government_Bonds', 'Fixed_Deposits', 'PPF', 'Gold'];

    const averages = investmentTypes.map(type => ({
        type: type.replace('_', ' '),
        value: d3.mean(data, d => d[type])
    }));

    const x = d3.scaleBand()
        .range([0, width])
        .domain(averages.map(d => d.type))
        .padding(0.2);

    const y = d3.scaleLinear()
        .domain([0, d3.max(averages, d => d.value)])
        .range([height, 0]);

    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));

    svg.selectAll("bars")
        .data(averages)
        .enter()
        .append("rect")
        .attr("x", d => x(d.type))
        .attr("y", d => y(d.value))
        .attr("width", x.bandwidth())
        .attr("height", d => height - y(d.value))
        .attr("fill", "#4CAF50");
}
```

```
function createExpectedReturnsChart(data) {
    const width = 300;
    const height = 300;
    const radius = Math.min(width, height) / 2;

    const svg = d3.select("#expected-returns")
        .append("svg")
        .attr("width", width)
        .attr("height", height)
        .append("g")
        .attr("transform", `translate(${width/2},${height/2})`);

    const returnsCount = d3.group(data, d => d.Expect);
    const pieData = Array.from(returnsCount, ([key, value]) => ({
        returns: key,
        count: value.length
    }));

    const color = d3.scaleOrdinal()
        .domain(pieData.map(d => d.returns))
        .range(d3.schemeSet3);

    const pie = d3.pie()
        .value(d => d.count);

    const arc = d3.arc()
        .innerRadius(radius * 0.4)
        .outerRadius(radius - 40);

    const arcs = svg.selectAll("arc")
        .data(pie(pieData))
        .enter()
        .append("g");

    arcs.append("path")
        .attr("d", arc)
        .attr("fill", d => color(d.data.returns))
        .attr("stroke", "white")
        .style("stroke-width", "2px");

    arcs.append("text")
        .attr("transform", d => `translate(${arc.centroid(d)})`)
        .attr("text-anchor", "middle")
        .style("font-size", "12px")
        .text(d => d.data.returns);
}
```

```
function createInvestmentDurationChart(data) {
    const margin = {top: 20, right: 20, bottom: 70, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#investment-duration")
        .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})`);

    const durationCount = Array.from(d3.group(data, d => d.Duration),
        ([key, value]) => ({duration: key, count: value.length}));

    const x = d3.scaleBand()
        .range([0, width])
        .domain(durationCount.map(d => d.duration))
        .padding(0.2);

    const y = d3.scaleLinear()
        .domain([0, d3.max(durationCount, d => d.count)])
        .range([height, 0]);

    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));

    svg.selectAll("bars")
        .data(durationCount)
        .enter()
        .append("rect")
        .attr("x", d => x(d.duration))
        .attr("y", d => y(d.count))
        .attr("width", x.bandwidth())
        .attr("height", d => height - y(d.count))
        .attr("fill", "#FF9800");
}

function createInvestmentObjectivesChart(data) {
    const margin = {top: 20, right: 20, bottom: 70, left: 40};
```

```
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#investment-objectives")
        .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})`);

    const objectiveCount = Array.from(d3.group(data, d => d.Objective),
        ([key, value]) => ({objective: key, count: value.length}));

    const x = d3.scaleBand()
        .range([0, width])
        .domain(objectiveCount.map(d => d.objective))
        .padding(0.2);

    const y = d3.scaleLinear()
        .domain([0, d3.max(objectiveCount, d => d.count)])
        .range([height, 0]);

    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));

    svg.selectAll("bars")
        .data(objectiveCount)
        .enter()
        .append("rect")
        .attr("x", d => x(d.objective))
        .attr("y", d => y(d.count))
        .attr("width", x.bandwidth())
        .attr("height", d => height - y(d.count))
        .attr("fill", "#9C27B0");
}

// Function to create all charts with the loaded data
function createAllCharts(data) {
    createGenderChart(data);
    createAgeChart(data);
```

```
    createInvestmentPreferencesChart(data);
    createExpectedReturnsChart(data);
    createInvestmentDurationChart(data);
    createInvestmentObjectivesChart(data);
    // createWordCloud(data);
    createBoxPlot(data);
    createViolinPlot(data);
    createRegressionPlot(data);
    create3DScatterPlot(data);
}

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

    // Process text data from investment objectives and reasons
    const words = data.flatMap(d => [
        {text: d.Reason_Equity, size: 20},
        {text: d.Reason_Mutual, size: 20},
        {text: d.Reason_Bonds, size: 20},
        {text: d.Reason_FD, size: 20},
        {text: d.Objective, size: 25},
        {text: d.Purpose, size: 25}
    ]).filter(d => d.text); // Remove any undefined entries

    const svg = d3.select("#word-cloud")
        .append("svg")
        .attr("width", width)
        .attr("height", height)
        .append("g")
        .attr("transform", `translate(${width/2},${height/2})`);

    // Create word cloud layout
    const layout = d3.layout.cloud()
        .size([width - margin.left - margin.right, height - margin.top -
margin.bottom])
        .words(words)
        .padding(5)
        .rotate(() => ~~(Math.random() * 2) * 90)
        .fontSize(d => d.size)
        .on("end", draw);

    layout.start();

    function draw(words) {
        svg.selectAll("text")
```

```
            .data(words)
            .enter().append("text")
            .style("font-size", d => `${d.size}px`)
            .style("fill", () => d3.schemeCategory10[~~(Math.random() * 10)])
            .attr("text-anchor", "middle")
            .attr("transform", d => `translate(${d.x},${d.y})rotate(${d.rotate})`)
            .text(d => d.text);
    }
}

function createBoxPlot(data) {
    const margin = {top: 20, right: 20, bottom: 70, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#box-plot")
        .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 investment preference data
    const investmentTypes = ['Mutual_Funds', 'Equity_Market', 'Debentures',
                             'Government_Bonds', 'Fixed_Deposits', 'PPF', 'Gold'];

    const boxPlotData = investmentTypes.map(type => {
        const values = data.map(d => +d[type]).sort(d3.ascending);
        return {
            type: type.replace('_', ' '),
            q1: d3.quantile(values, 0.25),
            median: d3.quantile(values, 0.5),
            q3: d3.quantile(values, 0.75),
            iqr: d3.quantile(values, 0.75) - d3.quantile(values, 0.25),
            min: d3.min(values),
            max: d3.max(values)
        };
    });

    const x = d3.scaleBand()
        .range([0, width])
        .domain(boxPlotData.map(d => d.type))
        .padding(0.2);

    const y = d3.scaleLinear()
        .domain([0, d3.max(boxPlotData, d => d.max)])
        .range([height, 0]);
```

```
// Add axes
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 boxes
const boxWidth = x.bandwidth();

const boxes = svg.selectAll("g.box")
    .data(boxPlotData)
    .enter()
    .append("g")
    .attr("class", "box")
    .attr("transform", d => `translate(${x(d.type)},0)`);

// Draw boxes
boxes.append("rect")
    .attr("x", 0)
    .attr("y", d => y(d.q3))
    .attr("width", boxWidth)
    .attr("height", d => y(d.q1) - y(d.q3))
    .attr("fill", "#69b3a2")
    .attr("stroke", "black");

// Draw median lines
boxes.append("line")
    .attr("x1", 0)
    .attr("x2", boxWidth)
    .attr("y1", d => y(d.median))
    .attr("y2", d => y(d.median))
    .attr("stroke", "black")
    .attr("stroke-width", 2);

// Draw whiskers
boxes.append("line")
    .attr("x1", boxWidth/2)
    .attr("x2", boxWidth/2)
    .attr("y1", d => y(d.min))
    .attr("y2", d => y(d.q1))
    .attr("stroke", "black");
```

```
    boxes.append("line")
        .attr("x1", boxWidth/2)
        .attr("x2", boxWidth/2)
        .attr("y1", d => y(d.max))
        .attr("y2", d => y(d.q3))
        .attr("stroke", "black");
}

function createViolinPlot(data) {
    const margin = {top: 20, right: 20, bottom: 70, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#violin-plot")
        .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 age data by investment preference
    const investmentTypes = ['Mutual_Funds', 'Equity_Market'];
    const violinData = investmentTypes.map(type => {
        const values = data.map(d => ({
            type: type.replace('_', ' '),
            value: d.age,
            preference: +d[type]
        })).filter(d => d.preference > 3); // Filter for high preference

        return {
            type: type.replace('_', ' '),
            values: values
        };
    });

    const x = d3.scaleBand()
        .range([0, width])
        .domain(investmentTypes.map(d => d.replace('_', ' ')))
        .padding(0.2);

    const y = d3.scaleLinear()
        .domain([d3.min(data, d => d.age), d3.max(data, d => d.age)])
        .range([height, 0]);

    // Add axes
    svg.append("g")
```

```
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x));

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

    // Compute kernel density estimation for each violin
    const kde = kernelDensityEstimator(kernelEpanechnikov(7), y.ticks(50));

    violinData.forEach(vData => {
        const density = kde(vData.values.map(d => d.value));
        const xScale = d3.scaleLinear()
            .range([0, x.bandwidth()/2])
            .domain([0, d3.max(density, d => d[1])]);

        // Draw the violin shape
        const area = d3.area()
            .x0(x.bandwidth()/2)
            .x1(d => x.bandwidth()/2 + xScale(d[1]))
            .y(d => y(d[0]))
            .curve(d3.curveCatmullRom);

        svg.append("path")
            .datum(density)
            .attr("transform", `translate(${x(vData.type)},0)`)
            .attr("d", area)
            .style("fill", "#69b3a2")
            .style("opacity", 0.6);

        // Mirror the violin shape
        const areaLeft = d3.area()
            .x0(x.bandwidth()/2)
            .x1(d => x.bandwidth()/2 - xScale(d[1]))
            .y(d => y(d[0]))
            .curve(d3.curveCatmullRom);

        svg.append("path")
            .datum(density)
            .attr("transform", `translate(${x(vData.type)},0)`)
            .attr("d", areaLeft)
            .style("fill", "#69b3a2")
            .style("opacity", 0.6);
    });
}

// Helper functions for violin plot
function kernelDensityEstimator(kernel, X) {
```

```
    return function(V) {
        return X.map(x => [x, d3.mean(V, v => kernel(x - v))]);
    };
}

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

function createRegressionPlot(data) {
    const margin = {top: 20, right: 20, bottom: 30, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#regression-plot")
        .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})`);

    // Create data points for age vs investment preference
    const points = data.map(d => ({
        x: d.age,
        y: +d.Mutual_Funds // Using Mutual Funds preference as y-axis
    }));

    const x = d3.scaleLinear()
        .domain([d3.min(points, d => d.x), d3.max(points, d => d.x)])
        .range([0, width]);

    const y = d3.scaleLinear()
        .domain([0, d3.max(points, d => d.y)])
        .range([height, 0]);

    // Add axes
    svg.append("g")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x))
        .append("text")
        .attr("x", width)
        .attr("y", -6)
        .text("Age");

    svg.append("g")
```

```
        .call(d3.axisLeft(y))
        .append("text")
        .attr("y", 6)
        .attr("dy", ".71em")
        .text("Investment Preference");

    // Add scatter points
    svg.selectAll(".point")
        .data(points)
        .enter().append("circle")
        .attr("class", "point")
        .attr("cx", d => x(d.x))
        .attr("cy", d => y(d.y))
        .attr("r", 4)
        .style("fill", "#4CAF50")
        .style("opacity", 0.5);

    // Calculate regression line
    const regression = linearRegression(points);

    // Add regression line
    svg.append("line")
        .attr("x1", x(d3.min(points, d => d.x)))
        .attr("y1", y(regression.intercept + regression.slope * d3.min(points, d
=> d.x)))
        .attr("x2", x(d3.max(points, d => d.x)))
        .attr("y2", y(regression.intercept + regression.slope * d3.max(points, d
=> d.x)))
        .style("stroke", "red")
        .style("stroke-width", 2);
}

// Helper function for regression calculation
function linearRegression(data) {
    const n = data.length;
    const sumX = data.reduce((sum, point) => sum + point.x, 0);
    const sumY = data.reduce((sum, point) => sum + point.y, 0);
    const sumXY = data.reduce((sum, point) => sum + (point.x * point.y), 0);
    const sumXX = data.reduce((sum, point) => sum + (point.x * point.x), 0);

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

    return { slope, intercept };
}

function create3DScatterPlot(data) {
```

```
    const margin = {top: 20, right: 20, bottom: 30, left: 40};
    const width = 400 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const svg = d3.select("#scatter-3d")
        .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})`);

    // Project 3D points onto 2D space using simple projection
    const points = data.map(d => ({
        x: +d.Mutual_Funds,
        y: +d.Equity_Market,
        z: +d.Fixed_Deposits,
        projected: project3DTo2D(+d.Mutual_Funds, +d.Equity_Market,
+d.Fixed_Deposits)
    }));

    const x = d3.scaleLinear()
        .domain([d3.min(points, d => d.projected.x), d3.max(points, d =>
d.projected.x)])
        .range([0, width]);

    const y = d3.scaleLinear()
        .domain([d3.min(points, d => d.projected.y), d3.max(points, d =>
d.projected.y)])
        .range([height, 0]);

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

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

    // Add points
    svg.selectAll(".point")
        .data(points)
        .enter().append("circle")
        .attr("class", "point")
        .attr("cx", d => x(d.projected.x))
        .attr("cy", d => y(d.projected.y))
        .attr("r", 5)
```

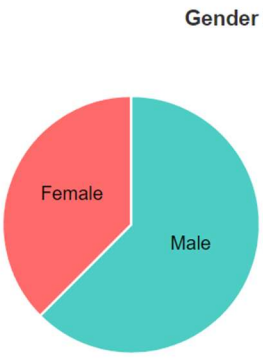
```
        .style("fill", d => d3.interpolateViridis(d.z / 5)) // Color based on z-
value
        .style("opacity", 0.7);

    // Add axes labels
    svg.append("text")
        .attr("transform", `translate(${width/2},${height + margin.bottom})`)
        .style("text-anchor", "middle")
        .text("Investment Preference (Projected)");

    svg.append("text")
        .attr("transform", "rotate(-90)")
        .attr("y", 0 - margin.left)
        .attr("x", 0 - (height / 2))
        .attr("dy", "1em")
        .style("text-anchor", "middle")
        .text("Risk vs Return (Projected)");

    // Add legend for z-axis (depth)
    const legendHeight = 100;
    const legendWidth = 20;

    const legendScale = d3.scaleLinear()
        .domain([0, 5])
        .range([legendHeight, 0]);

    const legend = svg.append("g")
        .attr("transform", `translate(${width + margin.right - legendWidth},
${height/2 - legendHeight/2})`);

    // Create gradient for legend
    const defs = svg.append("defs");
    const gradient = defs.append("linearGradient")
        .attr("id", "legend-gradient")
        .attr("x1", "0%")
        .attr("x2", "0%")
        .attr("y1", "0%")
        .attr("y2", "100%");

    gradient.selectAll("stop")
        .data(d3.range(0, 1.1, 0.1))
        .enter().append("stop")
        .attr("offset", d => d * 100 + "%")
        .attr("stop-color", d => d3.interpolateViridis(d));

    // Add gradient rectangle
    legend.append("rect")
```

```
        .attr("width", legendWidth)
        .attr("height", legendHeight)
        .style("fill", "url(#legend-gradient)");

    // Add legend axis
    const legendAxis = d3.axisRight(legendScale)
        .ticks(5);

    legend.append("g")
        .attr("transform", `translate(${legendWidth},0)`)
        .call(legendAxis);

    // Add legend title
    legend.append("text")
        .attr("transform", `translate(${legendWidth/2},${-10})`)
        .style("text-anchor", "middle")
        .text("Fixed Deposits");
}

// Helper function to project 3D coordinates to 2D
function project3DTo2D(x, y, z) {
    // Simple isometric projection
    const angle = Math.PI / 6; // 30 degrees
    const projectedX = (x - z) * Math.cos(angle);
    const projectedY = y + (x + z) * Math.sin(angle);

    return {
        x: projectedX,
        y: projectedY
    };
}

// Initialize with sample data
const sampleData = [
    {
        gender: "Female", age: 34, Mutual_Funds: 1, Equity_Market: 2, Debentures:
5,
        Government_Bonds: 3, Fixed_Deposits: 7, PPF: 6, Gold: 4,
        Expect: "20%-30%", Duration: "1-3 years", Objective: "Capital
Appreciation"
    },
    {
        gender: "Male", age: 30, Mutual_Funds: 3, Equity_Market: 6, Debentures: 4,
        Government_Bonds: 2, Fixed_Deposits: 5, PPF: 1, Gold: 7,
        Expect: "20%-30%", Duration: "3-5 years", Objective: "Capital
Appreciation"
    }
```

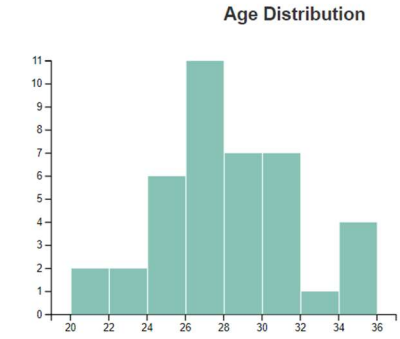
```
    // Add more sample data as needed
];

// Create initial charts with sample data
createAllCharts(sampleData);

</script>

</body>

</html>
```

3. Basic Charts Output

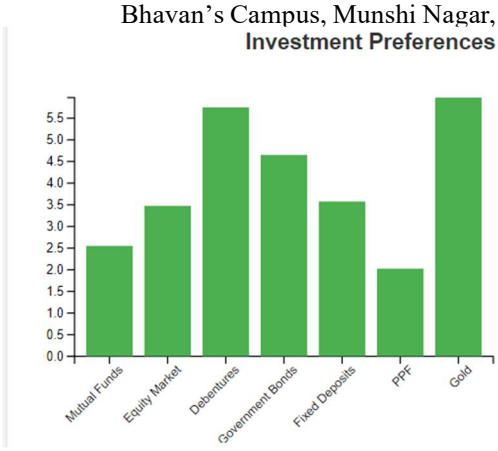
a. Pie Chart

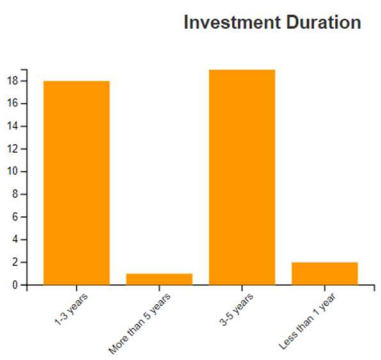


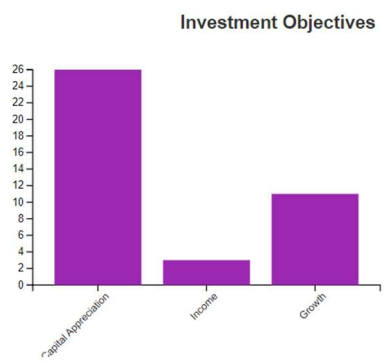
b. Histogram



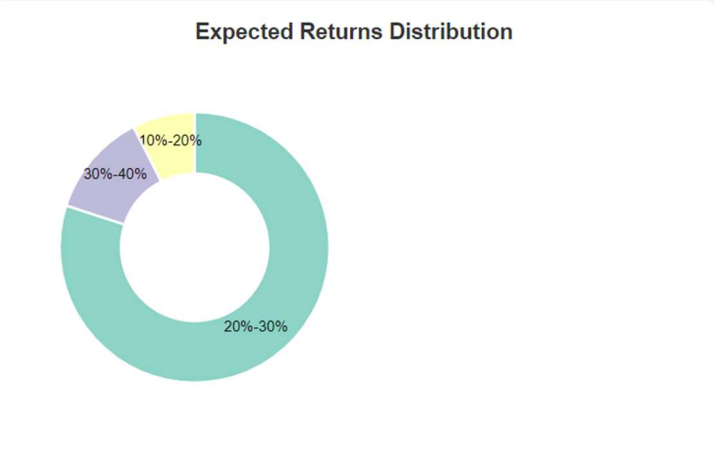
c. Bar charts



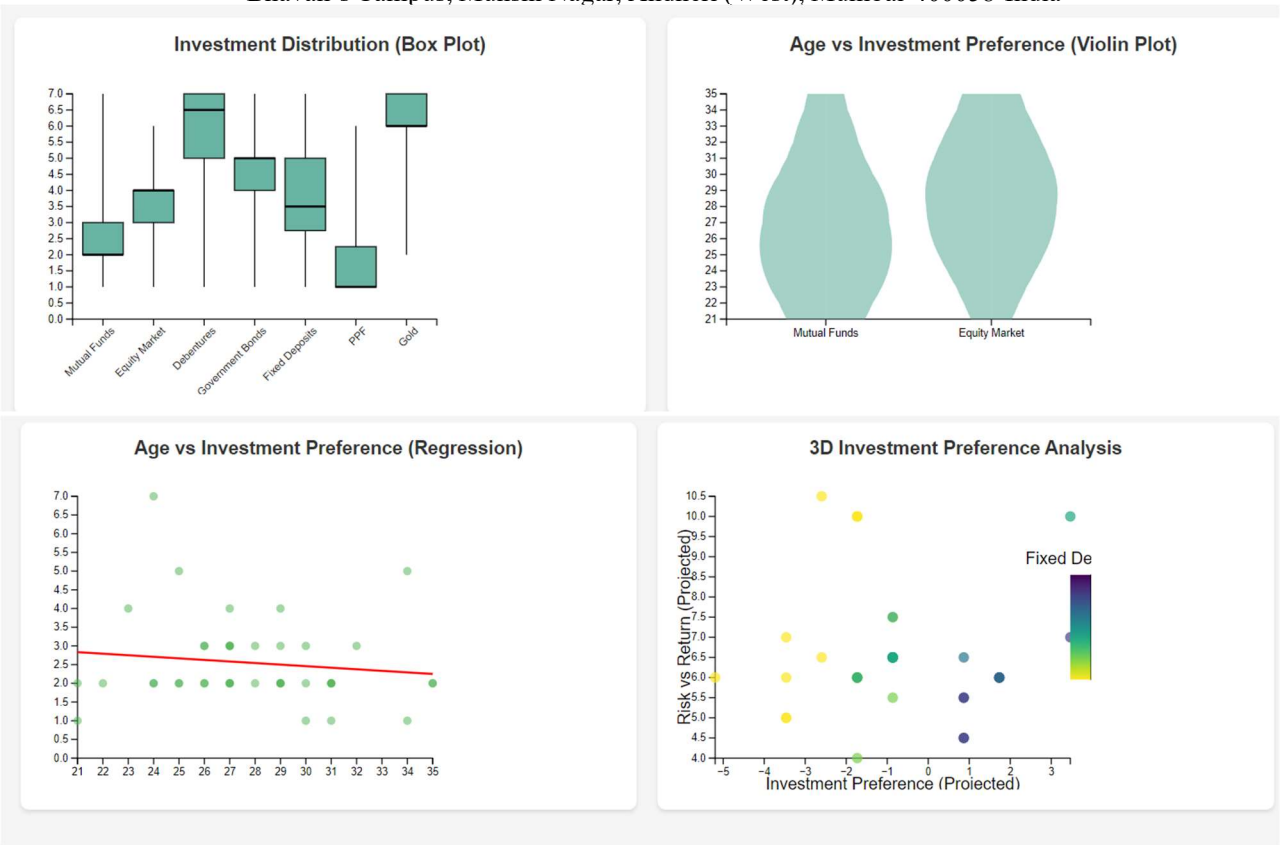




d. Donut Chart



4. Advanced Plots



5. Hypothesis Testing

Null Hypothesis (H0): There is no significant difference in the mean expected returns between male and female investors.

Alternative Hypothesis (H1): There is a significant difference in the mean expected returns between male and female investors.

Code:

```
      # %%
import pandas as pd
from statsmodels.stats.proportion import proportions_ztest
import numpy as np
from scipy.stats import norm

# %%
data = pd.read_csv('Finance_data.csv')

# %%
data.head()

# %%
# expect mapping
```

```
expect_mapping = {
    '10%-20%': 0.15,
    '20%-30%': 0.25,
    '30%-40%': 0.35,
    '40%-50%': 0.45,
    '50%-60%': 0.55,
    '60%-70%': 0.65,
    '70%-80%': 0.75,
    '80%-90%': 0.85,
    '90%-100%': 0.95
}


# %%
data['Expect_Numeric'] = data['Expect'].map(expect_mapping)
female_expect = data[data['gender'] == 'Female']['Expect_Numeric']
male_expect = data[data['gender'] == 'Male']['Expect_Numeric']


# %%

female_mean = female_expect.mean()
male_mean = male_expect.mean()
female_std = female_expect.std()
male_std = male_expect.std()
n_female = len(female_expect)
n_male = len(male_expect)
z_score = (female_mean - male_mean) / np.sqrt((female_std**2 / n_female) +
(male_std**2 / n_male))
p_value = 2 * (1 - norm.cdf(abs(z_score)))

# %%
print(f'z_score: {z_score}')
print(f'p_value: {p_value}')
```

Output:

```
z_score: 0.17883434996869604
p_value: 0.8580677726715031
```

The P-Value here is is very high as compared to 0.05 so we fail to reject the Null Hypothesis(H0)

6. Conclusion

- This experiment enables an understanding of both basic and advanced data visualizations using **D3.js**, giving insights into the finance domain through visual exploration.
- By performing hypothesis testing, you can statistically confirm relationships between variables, which is critical for data-driven decision-making in Finance/Banking/Insurance/Credit sectors.