Name: Omkar Shirwadkar
UID: 2021600062
Batch: D
Class: CSE(AI & ML)

EXPERIMENT NO: 7
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:**

a. **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

a. **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.

b. **Data Preprocessing:**

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

c. **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).

d. 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.

e. 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).

f. 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. Program and Code:

index.html:

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

```html
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Interactive Financial Data
Visualization</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/d3/7.4.4/d3.m
in.js"></script>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Financial Data Visualization with D3.js</h1>
    <div id="bar-chart" class="chart-container"></div>
    <div id="timeline-chart"
class="chart-container"></div>
    <div id="scatter-plot" class="chart-container"></div>
    <div id="box-whisker" class="chart-container"></div>
    <!-- <div id="bubble-plot"
class="chart-container"></div> -->
    <script src="script2.js"></script>
</body>
</html>
```

A. script.js:

```javascript
// Bar Chart with Tooltip
d3.csv("data.csv").then(data => {
    const svg =
d3.select("#bar-chart").append("svg").attr("width",
500).attr("height", 300);
    const margin = { top: 20, right: 30, bottom: 40,
left: 40 };
```

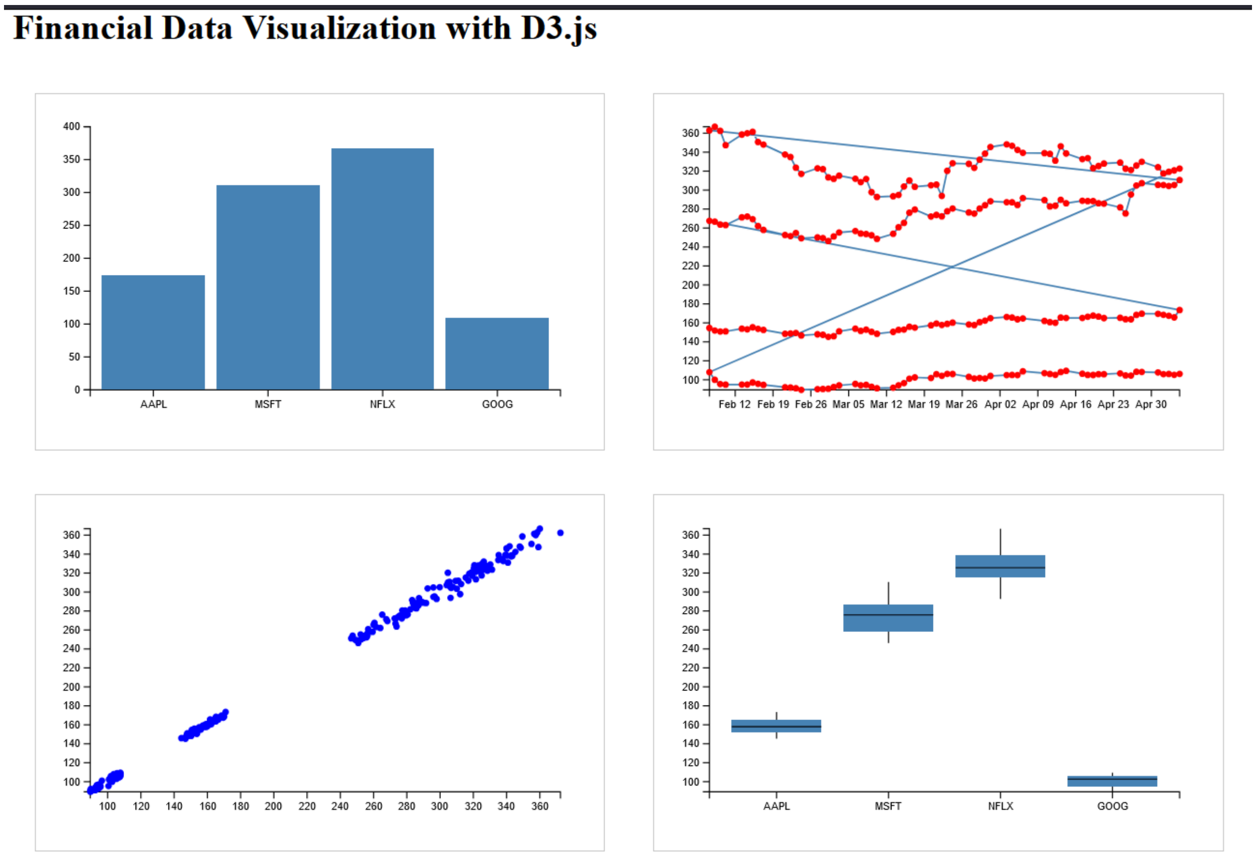
```
    const width = 500 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const x = d3.scaleBand().domain(data.map(d =>
d.Ticker)).range([0, width]).padding(0.1);
    const y = d3.scaleLinear().domain([0, d3.max(data, d
=> +d.Close)]).nice().range([height, 0]);

    const g = svg.append("g").attr("transform",
`translate(${margin.left},${margin.top})`);
    g.append("g").call(d3.axisLeft(y));
    g.append("g").attr("transform",
`translate(0,${height})`).call(d3.axisBottom(x));

    const tooltip =
d3.select("body").append("div").attr("class", "tooltip");

    g.selectAll(".bar")
        .data(data)
        .enter()
        .append("rect")
        .attr("class", "bar")
        .attr("x", d => x(d.Ticker))
        .attr("y", d => y(+d.Close))
        .attr("width", x.bandwidth())
        .attr("height", d => height - y(+d.Close))
        .attr("fill", "steelblue")
        .on("mouseover", (event, d) => {
            tooltip.style("visibility", "visible")
                   .text(`Ticker: ${d.Ticker}, Close:
${d.Close}`);
```

```
            d3.select(event.target).attr("fill",
"orange");
        })
        .on("mousemove", event => {
            tooltip.style("top", (event.pageY - 10) +
"px")
                   .style("left", (event.pageX + 10) +
"px");
        })
        .on("mouseout", (event) => {
            tooltip.style("visibility", "hidden");
            d3.select(event.target).attr("fill",
"steelblue");
        });
});

// Timeline Chart with Tooltip
d3.csv("data.csv").then(data => {
    const svg =
d3.select("#timeline-chart").append("svg").attr("width",
500).attr("height", 300);
    const margin = { top: 20, right: 30, bottom: 40,
left: 40 };
    const width = 500 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    data.forEach(d => d.Date = new Date(d.Date));

    const x = d3.scaleTime().domain(d3.extent(data, d =>
d.Date)).range([0, width]);
    const y = d3.scaleLinear().domain(d3.extent(data, d
=> +d.Close)).range([height, 0]);
```

```
    const g = svg.append("g").attr("transform",
`translate(${margin.left},${margin.top})`);
    g.append("g").call(d3.axisLeft(y));
    g.append("g").attr("transform",
`translate(0,${height})`).call(d3.axisBottom(x));

    const tooltip =
d3.select("body").append("div").attr("class", "tooltip");

    const line = d3.line().x(d => x(d.Date)).y(d =>
y(+d.Close));
    g.append("path").datum(data).attr("fill",
"none").attr("stroke", "steelblue").attr("stroke-width",
1.5).attr("d", line);

    g.selectAll(".dot")
        .data(data)
        .enter()
        .append("circle")
        .attr("class", "dot")
        .attr("cx", d => x(d.Date))
        .attr("cy", d => y(+d.Close))
        .attr("r", 3)
        .attr("fill", "red")
        .on("mouseover", (event, d) => {
            tooltip.style("visibility", "visible")
                   .text(`Date: ${d.Date.toDateString()},
Close: ${d.Close}`);
        })
        .on("mousemove", event => {
            tooltip.style("top", (event.pageY - 10) +
"px")
                   .style("left", (event.pageX + 10) +
"px");
        })
        .on("mouseout", () => tooltip.style("visibility",
"hidden"));
});

// Scatter Plot
d3.csv("data.csv").then(data => {
    const svg =
d3.select("#scatter-plot").append("svg").attr("width",
500).attr("height", 300);
    const margin = { top: 20, right: 30, bottom: 40,
left: 40 };
    const width = 500 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const x = d3.scaleLinear().domain(d3.extent(data, d
=> +d.Open)).range([0, width]);
    const y = d3.scaleLinear().domain(d3.extent(data, d
=> +d.Close)).range([height, 0]);

    const g = svg.append("g").attr("transform",
`translate(${margin.left},${margin.top})`);
    g.append("g").call(d3.axisLeft(y));
    g.append("g").attr("transform",
`translate(0,${height})`).call(d3.axisBottom(x));

    g.selectAll(".dot")
        .data(data)
        .enter()
        .append("circle")
        .attr("cx", d => x(+d.Open))
        .attr("cy", d => y(+d.Close))
        .attr("r", 3)
        .attr("fill", "blue");
});

// Bubble Plot
d3.csv("data.csv").then(data => {
    const svg =
d3.select("#bubble-plot").append("svg").attr("width",
500).attr("height", 300);
    const x = d3.scaleLinear().domain(d3.extent(data, d
=> +d.Volume)).range([20, 480]);
    const y = d3.scaleLinear().domain(d3.extent(data, d
=> +d.Close)).range([280, 20]);
    const size = d3.scaleSqrt().domain(d3.extent(data, d
=> +d.Volume)).range([4, 40]);

    svg.selectAll("circle")
        .data(data)
        .enter()
        .append("circle")
        .attr("cx", d => x(d.Volume))
        .attr("cy", d => y(d.Close))
        .attr("r", d => size(d.Volume))
        .attr("fill", "green");
});

// Box and Whisker Plot with Tooltip
d3.csv("data.csv").then(data => {
    const svg =
d3.select("#box-whisker").append("svg").attr("width",
500).attr("height", 300);
    const margin = { top: 20, right: 30, bottom: 40,
left: 40 };
    const width = 500 - margin.left - margin.right;
    const height = 300 - margin.top - margin.bottom;

    const tickers = Array.from(new Set(data.map(d =>
d.Ticker)));
    const x = d3.scaleBand().domain(tickers).range([0,
width]).padding(0.2);
    const y = d3.scaleLinear().domain([d3.min(data, d =>
+d.Close), d3.max(data, d => +d.Close)]).range([height,
0]);

    const tooltip =
d3.select("body").append("div").attr("class", "tooltip");

    const boxData = tickers.map(ticker => {
        const prices = data.filter(d => d.Ticker ===
ticker).map(d => +d.Close);
        return {
            ticker: ticker,
            min: d3.min(prices),
            q1: d3.quantile(prices, 0.25),
            median: d3.median(prices),
            q3: d3.quantile(prices, 0.75),
            max: d3.max(prices)
        };
    });
    const g = svg.append("g").attr("transform",
`translate(${margin.left},${margin.top})`);
    g.append("g").call(d3.axisLeft(y));
    g.append("g").attr("transform",
`translate(0,${height})`).call(d3.axisBottom(x));

    g.selectAll("g.box")
        .data(boxData)
        .enter()
        .append("g")
        .attr("transform", d =>
`translate(${x(d.ticker)},0)`)
        .each(function(d) {
            const gBox = d3.select(this);
            gBox.append("line").attr("x1", x.bandwidth()
/ 2).attr("x2", x.bandwidth() / 2).attr("y1",
y(d.min)).attr("y2", y(d.max)).attr("stroke", "black");
            gBox.append("rect").attr("x",
0).attr("width", x.bandwidth()).attr("y",
y(d.q3)).attr("height", y(d.q1) - y(d.q3)).attr("fill",
"steelblue");
            gBox.append("line").attr("x1", 0).attr("x2",
x.bandwidth()).attr("y1", y(d.median)).attr("y2",
y(d.median)).attr("stroke", "black");

            gBox.on("mouseover", () => {
                tooltip.style("visibility", "visible")
                    .text(`Ticker: ${d.ticker}, Min:
${d.min}, Max: ${d.max}, Median: ${d.median}`);
            })
            .on("mousemove", event => {
                tooltip.style("top", (event.pageY - 10) +
"px")
                       .style("left", (event.pageX + 10)
+ "px");
            })
            .on("mouseout", () => {
                tooltip.style("visibility", "hidden");
            });
        });
});
```

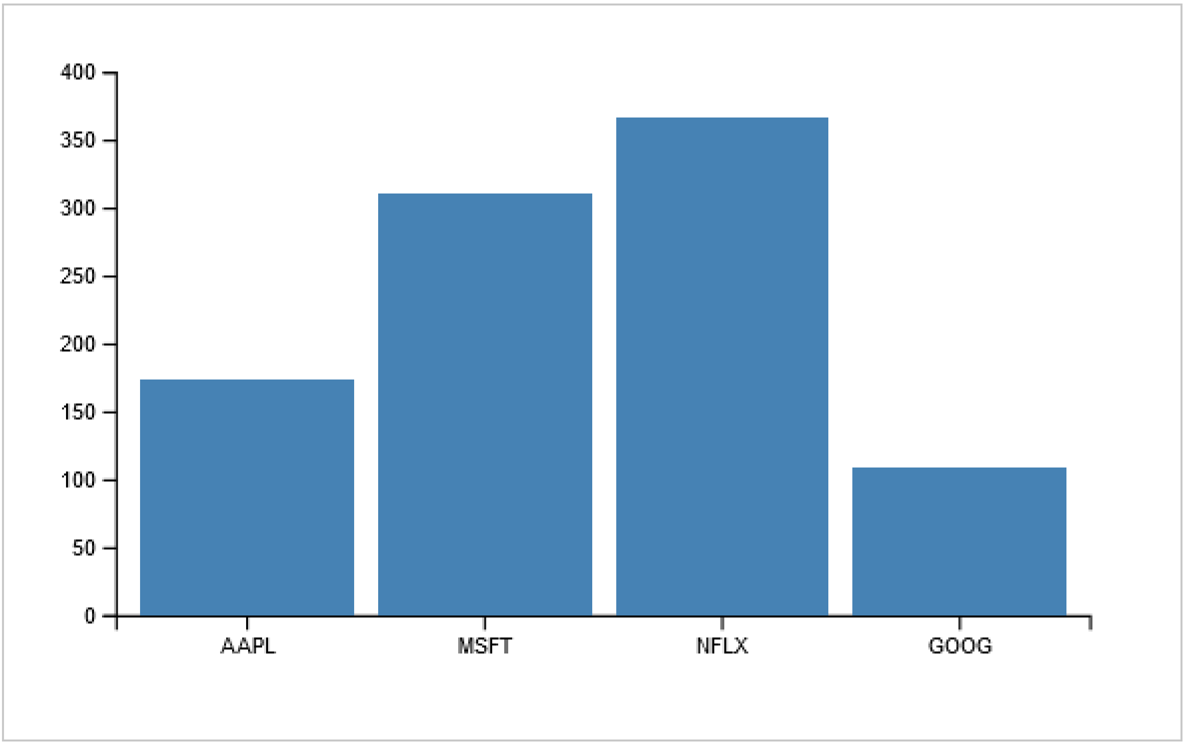
4. **Visualization and Observations:**

a. Dashboard:



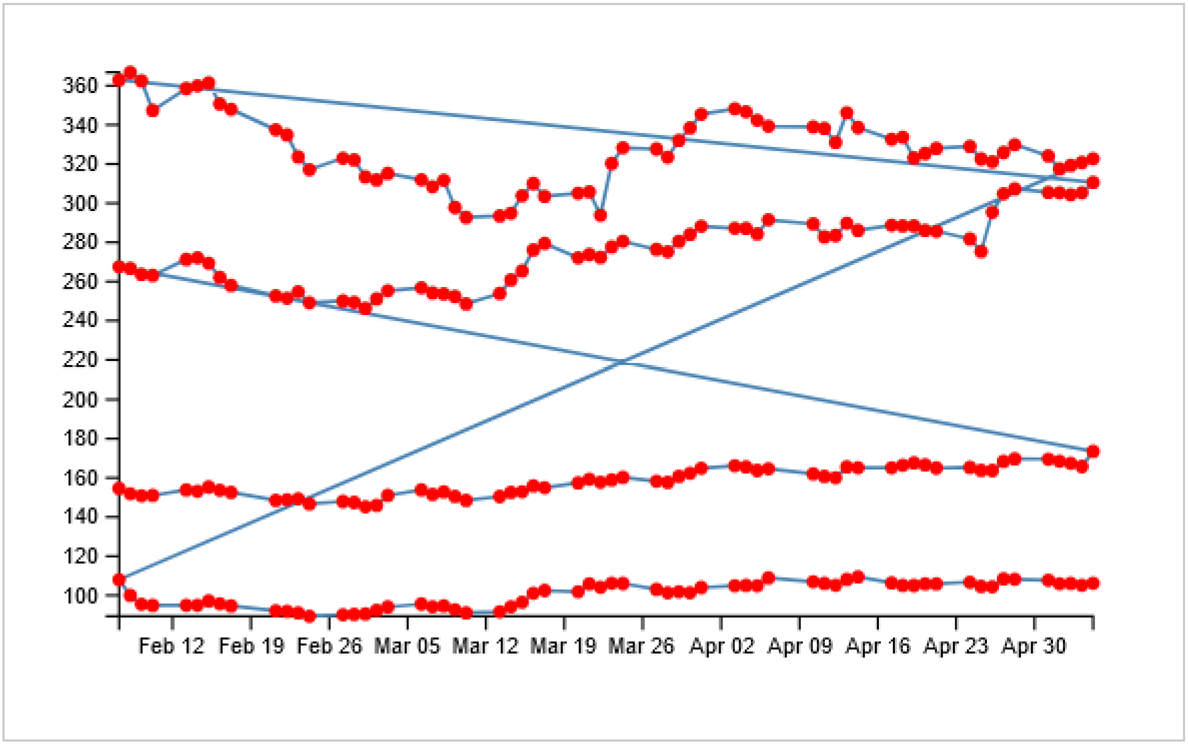
Financial Data Visualization with D3.js

b. Barchart:



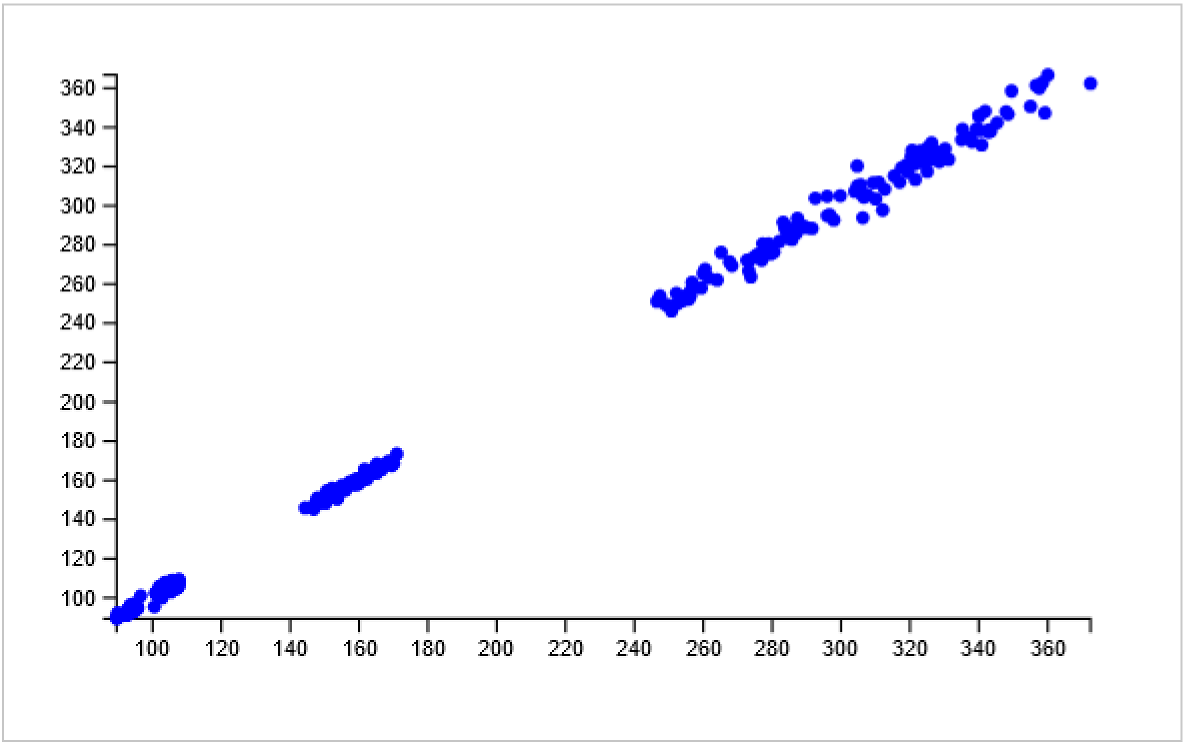
Shows the closing price for AAPL, MSFT, NFLX, GOOG shares.

c. TimeLine Chart:



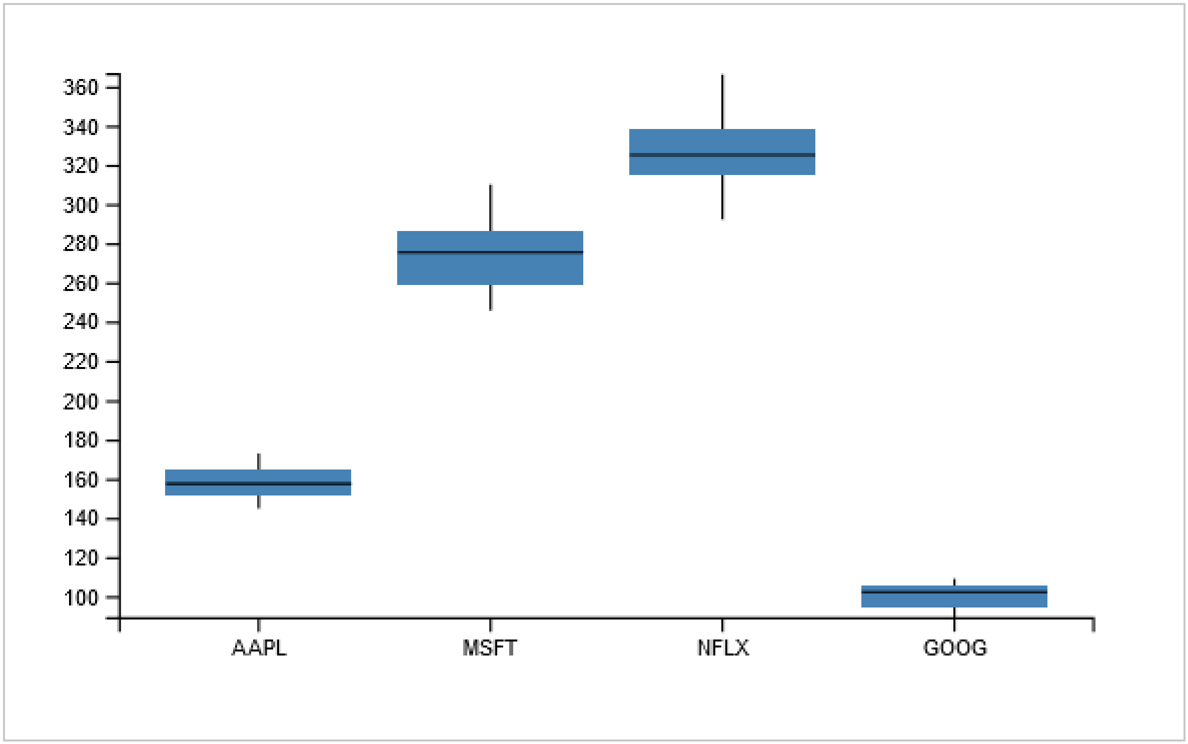
Displays closing prices over time.

d. Scatter Plot:



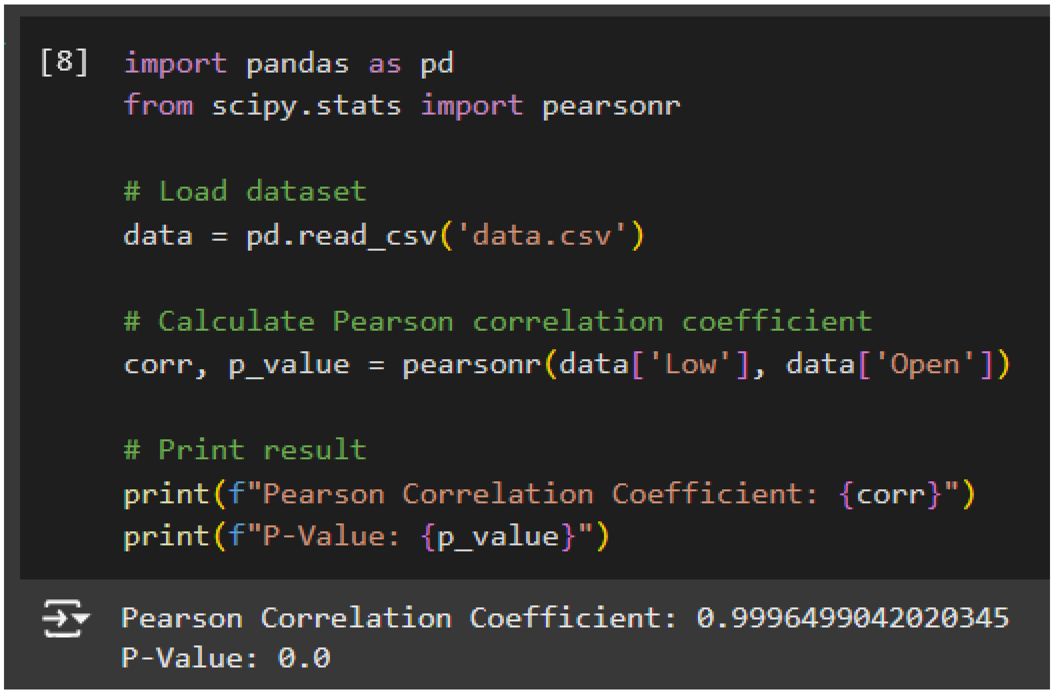
Shows the relationship between opening and closing prices.

e. Box and Whisker Plot:



Displays price distributions. Hovering over a box shows min, max, and quartiles.

f. Pearson Correlation:



```
[8] import pandas as pd
    from scipy.stats import pearsonr

    # Load dataset
    data = pd.read_csv('data.csv')

    # Calculate Pearson correlation coefficient
    corr, p_value = pearsonr(data['Low'], data['Open'])

    # Print result
    print(f"Pearson Correlation Coefficient: {corr}")
    print(f"P-Value: {p_value}")
```

```
Pearson Correlation Coefficient: 0.9996499042020345
P-Value: 0.0
```

The Low and Open column are strongly dependent on each other