ADV EXP - 7

Aim - To create Visualizations using D3.js on a Finance Dataset.

Objectives:

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

Implementation:

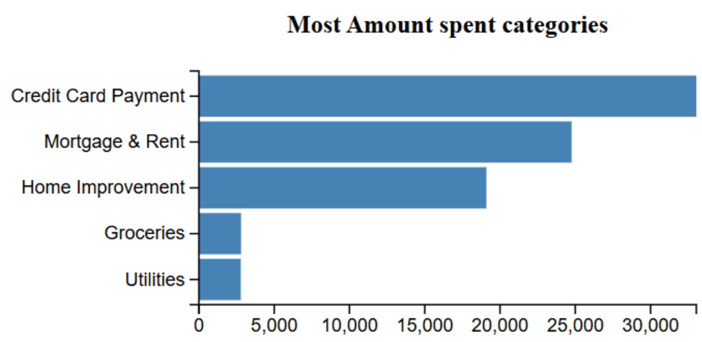
Barplot:

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

<head>
    <meta charset="UTF-8">
    <title>Bar Chart</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
</head>

<body>

    <svg></svg>

    <script>

        // Define the dimensions and margin
        const margin = { top: 50, right: 30, bottom: 40, left: 150 };
        const width = 500 - margin.left - margin.right;
        const height = 300 - margin.top - margin.bottom;
        const barHeight = 30;

        // Create the SVG element with specified width and height
        const svg = d3.select("svg")
```

```
            .attr("width", width + margin.left + margin.right)
            .attr("height", height + margin.top + margin.bottom)
            .append("g")
            .attr("transform", `translate(${margin.left},
${margin.top})`);

        // Load the CSV data
        d3.csv("personal_transactions.csv").then(data => {

            // Filter data for 'debit' transactions and parse the
Amount as an integer
            const debitData = data.filter(d => d["Transaction Type"]
=== "debit")
                                .map(d => ({ ...d, Amount: +d.Amount
}));

            // Calculate the total amount for each category
            const categoryTotals = d3.rollups(
                debitData,
                v => d3.sum(v, d => d.Amount),
                d => d.Category
            );

            // Sort categories by total amount in descending order and
select the top 5
            const top5Spends = categoryTotals.sort((a, b) => b[1] -
a[1]).slice(0, 5);

            // Convert to a more readable format if needed
            const top5SpendsFormatted = top5Spends.map(([category,
amount]) => ({ category, amount }));

            console.log(top5SpendsFormatted);



            // Define x-scale for subscriber count
            const x = d3.scaleLinear()
                        .domain([0, d3.max(top5SpendsFormatted, d =>
d.amount)])
                        .range([0, width]);

            // Define y-scale for course names
```

```
            const y = d3.scaleBand()
                        .domain(top5SpendsFormatted.map(d =>
d.category))
                        .range([0, top5SpendsFormatted.length *
barHeight])
                        .padding(0.1);

            // Adjust SVG height based on data length
            d3.select("svg").attr("height", top5SpendsFormatted.length
* barHeight + margin.top + margin.bottom);

            // Add the bars
            svg.selectAll("rect")
               .data(top5SpendsFormatted)
               .enter()
               .append("rect")
               .attr("x", 0)
               .attr("y", d => y(d.category))
               .attr("width", d => x(d.amount))
               .attr("height", y.bandwidth())
               .attr("fill", "steelblue");

            // Add a title to the graph
            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("Most Amount spent categories");

            // Add x-axis
            svg.append("g")
               .attr("transform", `translate(0,
${top5SpendsFormatted.length * barHeight})`)
               .call(d3.axisBottom(x).ticks(5))
               .selectAll("text")
               .style("font-size", "12px");

            // Add y-axis with course names
            svg.append("g")
               .call(d3.axisLeft(y))
               .selectAll("text")
```

```
            .style("font-size", "12px");


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

</body>

</html>
```



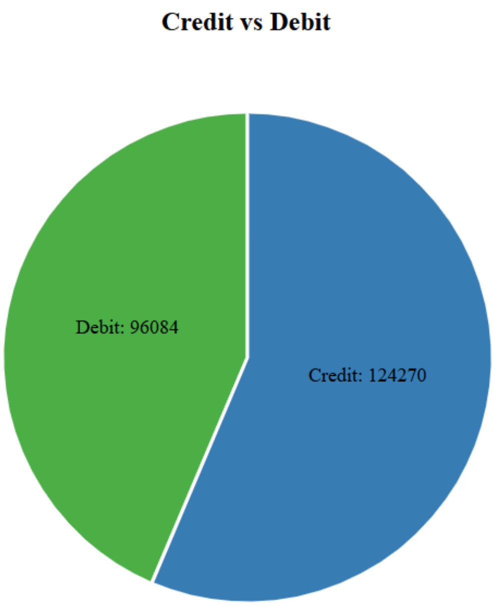
This bar chart shows that the highest spending category is "Credit Card Payment," followed by "Mortgage & Rent" and "Home Improvement." Categories such as "Groceries" and "Utilities" account for significantly lower spending amounts compared to the top categories.

Pie Chart:

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

<head>
    <meta charset="UTF-8">
    <title>Pie Chart</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
</head>

<body>

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

    <script>

        // Load the CSV data
        d3.csv("personal_transactions.csv").then(data => {
```

```
            // Calculate totals and round off the amounts
            const debitTotal = Math.round(d3.sum(data.filter(d =>
d["Transaction Type"] === "debit"), d => +d.Amount));
            const creditTotal = Math.round(d3.sum(data.filter(d =>
d["Transaction Type"] === "credit"), d => +d.Amount));

            // Data for the pie chart
            const pieData = [
                { label: "Debit", value: debitTotal },
                { label: "Credit", value: creditTotal }
            ];

            // Set up the pie chart dimensions and radius
            const width = 300, height = 500, radius = Math.min(width,
height) / 2;
            const svg = d3.select("svg")
                          .attr("width", width)
                          .attr("height", height)
                          .append("g")
                          .attr("transform", `translate(${width / 2},
${height / 2 })`);

            // Generate the pie and arcs
            const pie = d3.pie().value(d => d.value);
            const arc = d3.arc().innerRadius(0).outerRadius(radius);

            // Create color scale
            const color = d3.scaleOrdinal()
                            .domain(pieData.map(d => d.label))
                            .range(["#4daf4a", "#377eb8"]);

            // Add pie chart arcs
            svg.selectAll("path")
               .data(pie(pieData))
               .enter()
               .append("path")
               .attr("d", arc)
               .attr("fill", d => color(d.data.label))
               .attr("stroke", "white")
               .style("stroke-width", "2px");

            // Add labels to each slice
```

```
            svg.selectAll("text")
               .data(pie(pieData))
               .enter()
               .append("text")
               .text(d => `${d.data.label}: ${d.data.value}`)
               .attr("transform", d => `translate(${arc.centroid(d)})`)
               .style("text-anchor", "middle")
               .style("font-size", "12px");

            // Add a title to the graph
            svg.append("text")
                .attr("x", -1)
                .attr("y", -200)
                .attr("text-anchor", "middle")
                .style("font-size", "16px")
                .style("font-weight", "bold")
                .text("Credit vs Debit");

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

</body>

</html>
```



This pie chart illustrates the comparison between credit and debit amounts, with credit transactions totaling 124,270 and debit transactions totaling 96,084. Credit transactions make up a larger portion, indicating higher inflow compared to outflow.

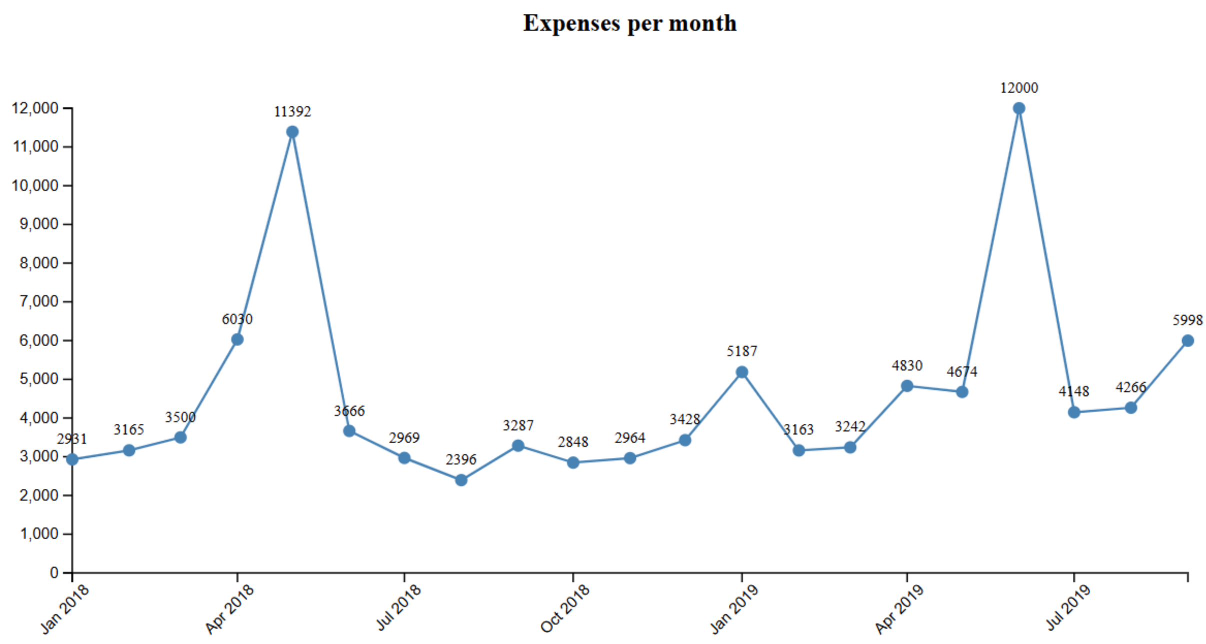
Timeline:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Timeline Chart</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
</head>
<body>
    <svg width="800" height="400"></svg>

    <script>

        // Load the CSV data
        d3.csv("personal_transactions.csv").then(data => {

            // Filter data for 'debit' transactions and parse the
Amount as an integer
            const debitData = data.filter(d => d["Transaction Type"]
=== "debit")
                                .map(d => ({ ...d, Amount: +d.Amount
}));

            // Parse date with mm-dd-yyyy format and convert Amount to
a number
            debitData.forEach(d => {
                d.Date = d3.timeParse("%m-%d-%Y")(d.Date);
                d.Amount = +d.Amount;
            });

            // Aggregate data by year-month, summing amounts for each
month
            const monthlyData = Array.from(
                d3.group(debitData, d =>
d3.timeFormat("%Y-%m")(d.Date)),
                ([key, values]) => ({
                    Date: d3.timeParse("%Y-%m")(key), // Convert key
back to date
                    Amount: Math.round(d3.sum(values, d => d.Amount))
// Sum the amount for that month
                })
            );
```

```
            // Set up chart dimensions and margins
            const margin = { top: 100, right: 30, bottom: 100, left: 50
},
                  width = 800 - margin.left - margin.right,
                  height = 500 - margin.top - margin.bottom;

            const svg = d3.select("svg")
                          .attr("width", width + margin.left +
margin.right)
                          .attr("height", height + margin.top +
margin.bottom)
                          .append("g")
                          .attr("transform",
`translate(${margin.left},${margin.top})`);

            // Set up scales
            const xScale = d3.scaleTime()
                             .domain(d3.extent(monthlyData, d =>
d.Date))
                             .range([0, width]);

            const yScale = d3.scaleLinear()
                             .domain([0, d3.max(monthlyData, d =>
d.Amount)])
                             .range([height, 0]);

            // Add x-axis
            svg.append("g")
               .attr("transform", `translate(0, ${height})`)
               .call(d3.axisBottom(xScale).tickFormat(d3.timeFormat("%b
%Y"))) // Format as Month Year
               .selectAll("text")
               .attr("transform", "rotate(-45)")
               .style("text-anchor", "end");

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

            // Add line path
            const line = d3.line()
                           .x(d => xScale(d.Date))
```

```
                    .y(d => yScale(d.Amount));

    svg.append("path")
        .datum(monthlyData)
        .attr("fill", "none")
        .attr("stroke", "steelblue")
        .attr("stroke-width", 1.5)
        .attr("d", line);

    // Add points for each month
    svg.selectAll("circle")
        .data(monthlyData)
        .enter()
        .append("circle")
        .attr("cx", d => xScale(d.Date))
        .attr("cy", d => yScale(d.Amount))
        .attr("r", 4)
        .attr("fill", "steelblue");

    // Add labels for each point
    svg.selectAll("text.label")
        .data(monthlyData)
        .enter()
        .append("text")
        .attr("class", "label")
        .attr("x", d => xScale(d.Date))
        .attr("y", d => yScale(d.Amount) - 10)
        .attr("text-anchor", "middle")
        .style("font-size", "10px")
        .text(d => d.Amount);

    // Add a title to the graph
    svg.append("text")
         .attr("x", width/2)
         .attr("y", -50)
         .attr("text-anchor", "middle")
         .style("font-size", "16px")
         .style("font-weight", "bold")
         .text("Expenses per month");

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

```
    </script>
</body>
</html>
```



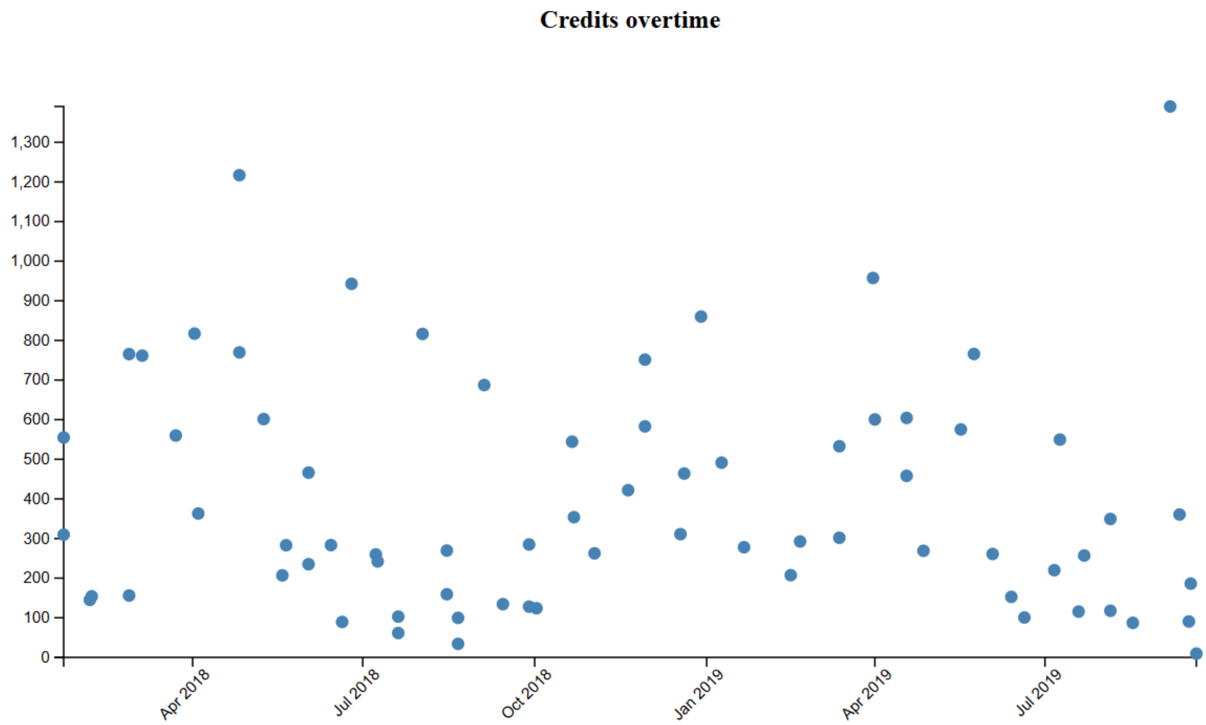
This timeline chart shows monthly expenses over time, with significant peaks in April 2018 and May 2019, where spending exceeded 10,000. The pattern suggests periodic spikes in expenditure, while most other months maintain a relatively stable range around 3,000 to 5,000.

Scatter Plot:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Scatter Plot</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
</head>
<body>
    <svg width="800" height="400"></svg>

    <script>

        // Load the CSV data
        d3.csv("personal_transactions.csv").then(data => {
```

```
            // Parse date and amount, and filter for 'credit'
transactions
            data.forEach(d => {
                d.Date = d3.timeParse("%m-%d-%Y")(d.Date);
                d.Amount = +d.Amount;
            });

            const creditData = data.filter(d => d["Transaction Type"]
=== "credit" & d['Amount']<1500);

            // Set up chart dimensions and margins
            const margin = { top: 100, right: 30, bottom: 50, left: 50
},
                  width = 800 - margin.left - margin.right,
                  height = 500 - margin.top - margin.bottom;

            const svg = d3.select("svg")
                          .attr("width", width + margin.left +
margin.right)
                          .attr("height", height + margin.top +
margin.bottom)
                          .append("g")
                          .attr("transform",
`translate(${margin.left},${margin.top})`);

            // Set up scales
            const xScale = d3.scaleTime()
                             .domain(d3.extent(creditData, d =>
d.Date))
                             .range([0, width]);

            const yScale = d3.scaleLinear()
                             .domain([0, d3.max(creditData, d =>
d.Amount)])
                             .range([height, 0]);

            // Add x-axis
            svg.append("g")
               .attr("transform", `translate(0, ${height})`)
               .call(d3.axisBottom(xScale).tickFormat(d3.timeFormat("%b
%Y"))) // Format as Month Year
               .selectAll("text")
               .attr("transform", "rotate(-45)")
```

```
            .style("text-anchor", "end");

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

        // Add scatter plot points
        svg.selectAll("circle")
            .data(creditData)
            .enter()
            .append("circle")
            .attr("cx", d => xScale(d.Date))
            .attr("cy", d => yScale(d.Amount))
            .attr("r", 4)
            .attr("fill", "steelblue");

        // Add a title to the graph
        svg.append("text")
             .attr("x", width/2)
             .attr("y", -50)
             .attr("text-anchor", "middle")
             .style("font-size", "16px")
             .style("font-weight", "bold")
             .text("Credits overtime");

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



The scatter plot shows no noticeable upward or downward trend in credit amounts over time, indicating a relatively stable pattern. Credit transactions are mostly scattered below the 1,000 mark, with a few higher outliers over the timeline. The distribution of credits suggests sporadic, high-value transactions rather than consistent monthly increases or decreases.

Hypothesis Testing:

```
import pandas as pd
from scipy.stats import pearsonr

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

# Get the required data
reqdata = []
for index, rec in data.iterrows():
    if rec['Transaction Type'] == 'credit':
        reqdata.append(rec)

# Get date and covert it to numeric
date = []
for rec in reqdata:
    date.append(rec['Date'])
```

```
date_numeric = pd.to_datetime(date, format='%m-%d-%Y').astype('int64')
/ 10**9
print(date_numeric)
```

```
Index([1514937600.0, 1515715200.0, 1516320000.0, 1516579200.0, 1516579200.0,
       1517529600.0, 1517788800.0, 1517875200.0, 1518739200.0, 1519603200.0,
       ...
       1565913600.0, 1566000000.0, 1567123200.0, 1567728000.0, 1568160000.0,
       1568332800.0, 1568592000.0, 1568678400.0, 1568937600.0, 1569542400.0],
      dtype='float64', length=118)
```

```
# Get credit
credit = []
for rec in reqdata:
    credit.append(round(rec['Amount']))
print(credit)
```

```
[2298, 2000, 2000, 555, 310, 2000, 145, 154, 2000, 765, 156, 2000, 762, 2000,
```

```
# Calculate Pearson correlation coefficient
corr, p_value = pearsonr(date_numeric, credit)

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

```
Pearson Correlation Coefficient: -0.014825326904847773
P-Value: 0.8734020555859093
```

Pearson Correlation Coefficient (-0.0148): This value is very close to zero, which suggests that there is no linear relationship between the date and the credit amount.
P-Value (0.8734): A high p-value (typically above 0.05) means that the result is not statistically significant.