BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to University of Mumbai)
[Knowledge is Nectar]
Department of Computer Engineering

Name: Nishant Zaveri

UID: 2022601009

Batch: A

Aim:

Create basic and advanced charts using D3.js / Power BI / R / Python / D3.js on the dataset - Finance / Banking / Insurance / Credit

- Basic - Bar chart, Pie chart, Histogram, Time line chart, Scatter plot, Bubble plot
- Advanced - Word chart, Box and whisker plot, Violin plot, Regression plot (linear and nonlinear), 3D chart, Jitter
- Write observations from each chart

Theory:

Dataset:

banking-dataset-with-customer-churn-prediction

Dataset Description:

This dataset contains information about bank customers and their churn status, which indicates whether they have exited the bank or not. It is suitable for exploring and analyzing factors influencing customer churn in banking institutions and for building predictive models to identify customers at risk of churning.

Columns Description:

1. **RowNumber:** The sequential number assigned to each row in the dataset.
2. **CustomerId:** A unique identifier for each customer.
3. **Surname:** The surname of the customer.
4. **CreditScore:** The credit score of the customer.
5. **Geography:** The geographical location of the customer (e.g., country or region).
6. **Gender:** The gender of the customer.
7. **Age:** The age of the customer.

8. **Tenure:** The number of years the customer has been with the bank.
9. **Balance:** The account balance of the customer.
10. **NumOfProducts:** The number of bank products the customer has.
11. **HasCrCard:** Indicates whether the customer has a credit card (binary: yes/no).
12. **IsActiveMember:** Indicates whether the customer is an active member (binary: yes/no).
13. **EstimatedSalary:** The estimated salary of the customer.
14. **Exited:** Indicates whether the customer has exited the bank (binary: yes/no).

Code:

Basic Charts

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>D3.js Data Visualizations</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }

        .chart {
```

```
            margin: 20px 0;
        }
    </style>
</head>

<body>
    <h2>Basic Visualizations</h2>

    <h3>Bar Chart</h3>
    <div class="chart" id="bar-chart"></div>

    <h3>Pie Chart</h3>
    <div class="chart" id="pie-chart"></div>

    <h3>Histogram</h3>
    <div class="chart" id="histogram"></div>

    <script>
        // Load the data from the CSV file
        d3.csv("Churn_Modelling.csv").then(data => {
            // Format the data, converting necessary fields to numbers
            data.forEach(d => {
                d.CreditScore = +d.CreditScore;
                d.Age = +d.Age;
                d.Tenure = +d.Tenure;
```

```
            d.Balance = +d.Balance;

            d.NumOfProducts = +d.NumOfProducts;

            d.HasCrCard = +d.HasCrCard;

            d.IsActiveMember = +d.IsActiveMember;

            d.EstimatedSalary = +d.EstimatedSalary;

            d.Exited = +d.Exited;

        });


        // Call the functions to create the visualizations

        createBarChart(data);

        createPieChart(data);

        createHistogram(data);

    });


    // Helper function to create scales

    function createAxis(svg, x, y, height) {

        svg.append("g")

            .attr("transform", `translate(0,${height})`)

            .call(d3.axisBottom(x));


        svg.append("g")

            .call(d3.axisLeft(y));

    }


    // Create Bar Chart
```

```
        function createBarChart(data) {
            const margin = { top: 20, right: 30, bottom: 40, left: 50 },
                width = 600 - margin.left - margin.right,
                height = 400 - margin.top - margin.bottom;

            const svg = d3.select("#bar-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 Geography
            const geographyCounts = d3.rollup(data, v => v.length, d =>
d.Geography);
            const x = d3.scaleBand()
                .domain(Array.from(geographyCounts.keys()))
                .range([0, width])
                .padding(0.1);

            const y = d3.scaleLinear()
                .domain([0, d3.max(geographyCounts, d => d[1])])
                .nice()
                .range([height, 0]);
```

```
    createAxis(svg, x, y, height);

    svg.selectAll("rect")
        .data(Array.from(geographyCounts))
        .enter()
        .append("rect")
        .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", "steelblue");
}

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

    const svg = d3.select("#pie-chart")
        .append("svg")
        .attr("width", width)
        .attr("height", height)
        .append("g")
```

```
            .attr("transform", `translate(${width / 2}, ${height /
2})`);

        const customerExitedCounts = {
            "Exited": data.filter(d => d.Exited === 1).length,
            "Stayed": data.filter(d => d.Exited === 0).length
        };

        const pie = d3.pie().value(d =>
d[1])(Object.entries(customerExitedCounts));
        const arc = d3.arc().innerRadius(0).outerRadius(radius);

        svg.selectAll('path')
            .data(pie)
            .enter()
            .append('path')
            .attr('d', arc)
            .attr('fill', (d, i) => i === 0 ? 'red' : 'green')
            .attr("stroke", "white")
            .style("stroke-width", "2px");

        // Add labels
        svg.selectAll('text')
            .data(pie)
            .enter()
```

```
                .append('text')
                .text(d => `${d.data[0]}: ${d.data[1]}`)
                .attr("transform", d => `translate(${arc.centroid(d)})`)
                .attr("dy", "0.35em")
                .style("text-anchor", "middle")
                .style("fill", "white");
        }

        // Create Histogram
        function createHistogram(data) {
            const margin = { top: 20, right: 30, bottom: 40, left: 50 },
                width = 600 - margin.left - margin.right,
                height = 400 - margin.top - margin.bottom;

            const svg = d3.select("#histogram")
                .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), d3.max(data, d =>
d.Age)])
```

```
                .range([0, width]);


            const bins =
d3.bin().domain(x.domain()).thresholds(10)(data.map(d => d.Age));
            const y = d3.scaleLinear()
                .domain([0, d3.max(bins, d => d.length)])
                .range([height, 0]);


            createAxis(svg, x, y, height);


            svg.selectAll("rect")
                .data(bins)
                .enter()
                .append("rect")
                .attr("x", 1)
                .attr("transform", d =>
`translate(${x(d.x0)},${y(d.length)})`)
                .attr("width", d => x(d.x1) - x(d.x0) - 1)
                .attr("height", d => height - y(d.length))
                .style("fill", "#69b3a2");
        }
    </script>
</body>


</html>
```

[redacted]

Advanced Charts:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Word Cloud, Box Plot, and Regression Plot</title>

    <script src="https://d3js.org/d3.v6.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/d3-
cloud/1.2.5/d3.layout.cloud.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

        }

        .box {

            fill: steelblue;

            opacity: 0.7;

        }

        .median {

            stroke: black;

            stroke-width: 2;

        }
```

```
        .regression-line {
            stroke: red;
            stroke-width: 2;
        }
        .scatter {
            fill: blue;
            opacity: 0.7;
        }
    </style>
</head>
<body>
    <h2>Word Cloud of Surnames</h2>
    <div id="wordCloud"></div>

    <h2>Box and Whisker Plot of Balance</h2>
    <div id="boxPlot"></div>

    <h2>Regression Plot of Age vs Estimated Salary</h2>
    <div id="regressionPlot"></div>
    <script>
        d3.csv("Churn_Modelling.csv").then(function(data) {
            let surnameCount = {};
            data.forEach(d => {
                let surname = d.Surname;
                if (surname in surnameCount) {
```

```
            surnameCount[surname]++;
        } else {
            surnameCount[surname] = 1;
        }
    });

    let wordData = Object.keys(surnameCount).map(surname => ({
        text: surname,
        size: surnameCount[surname] * 10
    }));

    const width = 800;
    const height = 400;

    d3.layout.cloud()
        .size([width, height])
        .words(wordData)
        .padding(5)
        .rotate(() => ~~(Math.random() * 2) * 90)
        .fontSize(d => d.size)
        .on("end", drawCloud)
        .start();

    function drawCloud(words) {
        const svg = d3.select("#wordCloud").append("svg")
```

```
                    .attr("width", width)

                    .attr("height", height)

                    .append("g")

                    .attr("transform", `translate(${width / 2},${height /
2})`);


                svg.selectAll("text")

                    .data(words)

                    .enter().append("text")

                    .style("font-size", d => `${d.size}px`)

                    .style("fill", () => `hsl(${Math.random() * 360},
100%, 50%)`)

                    .attr("text-anchor", "middle")

                    .attr("transform", d => `translate(${[d.x, d.y]})
rotate(${d.rotate})`)

                    .text(d => d.text);

            }

            let balanceValues = data.map(d => +d.Balance).filter(d =>
!isNaN(d)); // Convert Balance to number


            // Calculate summary statistics for box plot

            const q1 = d3.quantile(balanceValues.sort(d3.ascending),
0.25);

            const median = d3.quantile(balanceValues, 0.5);

            const q3 = d3.quantile(balanceValues, 0.75);

            const interQuantileRange = q3 - q1;
```

```
            const min = Math.max(0, q1 - 1.5 * interQuantileRange);

            const max = Math.min(d3.max(balanceValues), q3 + 1.5 *
interQuantileRange);

            const boxWidth = 800;

            const boxHeight = 400;

            const margin = { top: 20, right: 30, bottom: 30, left: 40 };

            const svgBox = d3.select("#boxPlot").append("svg")

                .attr("width", boxWidth + margin.left + margin.right)

                .attr("height", boxHeight + margin.top + margin.bottom)

                .append("g")

                .attr("transform",
`translate(${margin.left},${margin.top})`);

            const xScale = d3.scaleLinear()

                .domain([0, d3.max(balanceValues)])

                .range([0, boxWidth]);

            svgBox.append("g")

                .attr("transform", `translate(0, ${boxHeight})`)

                .call(d3.axisBottom(xScale));

            svgBox.append("rect")

                .attr("class", "box")

                .attr("x", xScale(q1))

                .attr("width", xScale(q3) - xScale(q1))

                .attr("y", boxHeight / 4)

                .attr("height", boxHeight / 2);

            svgBox.append("line")
```

```
        .attr("class", "median")
        .attr("x1", xScale(median))
        .attr("x2", xScale(median))
        .attr("y1", boxHeight / 4)
        .attr("y2", 3 * boxHeight / 4);
    svgBox.append("line")
        .attr("x1", xScale(min))
        .attr("x2", xScale(q1))
        .attr("y1", boxHeight / 2)
        .attr("y2", boxHeight / 2)
        .attr("stroke", "black");

    svgBox.append("line")
        .attr("x1", xScale(q3))
        .attr("x2", xScale(max))
        .attr("y1", boxHeight / 2)
        .attr("y2", boxHeight / 2)
        .attr("stroke", "black");
    svgBox.append("line")
        .attr("x1", xScale(min))
        .attr("x2", xScale(min))
        .attr("y1", boxHeight / 3)
        .attr("y2", 2 * boxHeight / 3)
        .attr("stroke", "black");
```

```
            svgBox.append("line")
                .attr("x1", xScale(max))
                .attr("x2", xScale(max))
                .attr("y1", boxHeight / 3)
                .attr("y2", 2 * boxHeight / 3)
                .attr("stroke", "black");
            let ageValues = data.map(d => +d.Age).filter(d => !isNaN(d));
            let salaryValues = data.map(d => +d.EstimatedSalary).filter(d
=> !isNaN(d));

            const svgRegression =
d3.select("#regressionPlot").append("svg")
                .attr("width", boxWidth + margin.left + margin.right)
                .attr("height", boxHeight + margin.top + margin.bottom)
                .append("g")
                .attr("transform",
`translate(${margin.left},${margin.top})`);
            // X-scale for age
            const xScaleReg = d3.scaleLinear()
                .domain([d3.min(ageValues), d3.max(ageValues)])
                .range([0, boxWidth]);
            // Y-scale for estimated salary
            const yScaleReg = d3.scaleLinear()
                .domain([d3.min(salaryValues), d3.max(salaryValues)])
                .range([boxHeight, 0]);
```

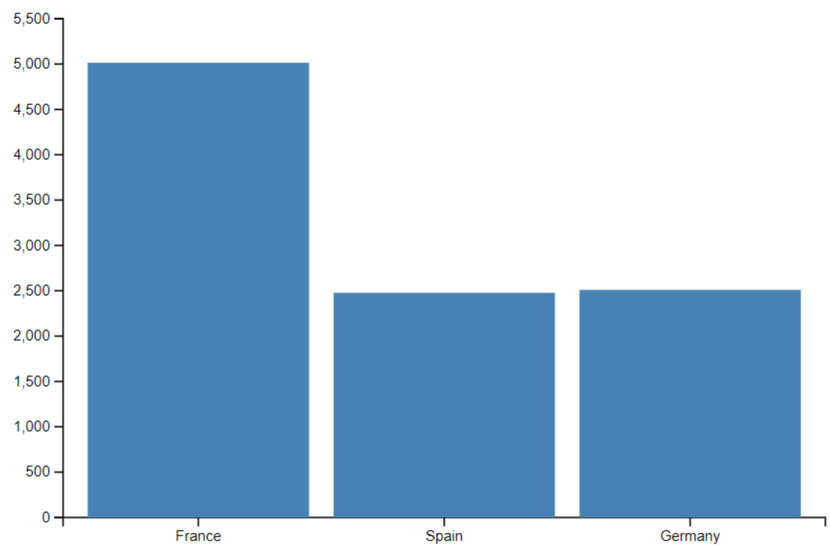
```
            // Add X-axis and Y-axis
            svgRegression.append("g")
                .attr("transform", `translate(0, ${boxHeight})`)
                .call(d3.axisBottom(xScaleReg));


            svgRegression.append("g")
                .call(d3.axisLeft(yScaleReg));
            svgRegression.selectAll(".dot")
                .data(data)
                .enter().append("circle")
                .attr("class", "scatter")
                .attr("cx", d => xScaleReg(d.Age))
                .attr("cy", d => yScaleReg(d.EstimatedSalary))
                .attr("r", 3);
            const regressionLine = d3.line()
                .x(d => xScaleReg(d.Age))
                .y(() => yScaleReg(d3.mean(salaryValues)));


            svgRegression.append("path")
                .datum(data)
                .attr("class", "regression-line")
                .attr("d", regressionLine);
        });
    </script>
</body>
```

```
</html>
```

Charts:

1. Bar Chart:

<u>Chart</u>:

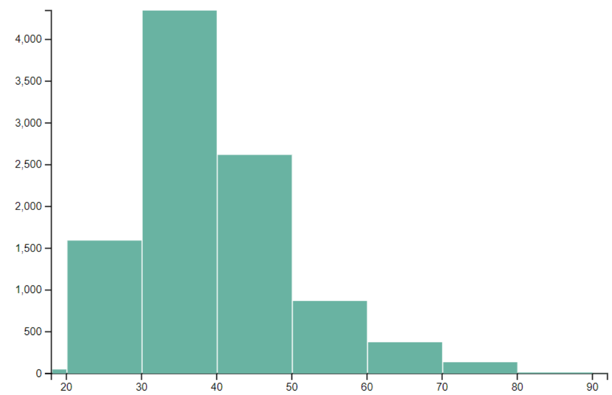


<u>Observations</u>:

- **France has the largest customer base:** The bar for France is significantly taller than the others, indicating that it has the largest number of customers compared to Spain and Germany.
- **Spain and Germany have comparable customer bases:** The bars for Spain and Germany are of similar height, suggesting that they have roughly the same number of customers.
- **France significantly outweighs Spain and Germany:** The difference in height between the France bar and the Spain/Germany bars is substantial, emphasizing that France has a much larger customer base.

2. Histogram:

<u>Chart</u>:

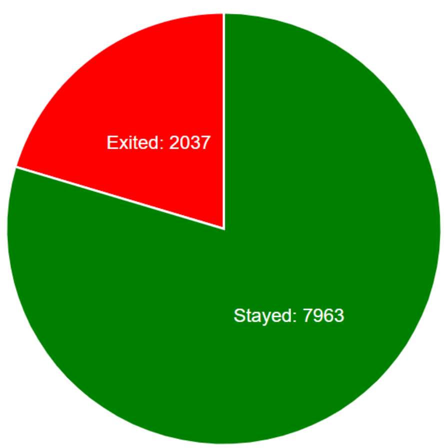


Observations:

- **Skewed Distribution:** The histogram shows a right-skewed distribution, meaning there's a longer tail on the right side. This indicates that there is a larger proportion of customers in the younger age groups compared to the older ones.
- **Concentration in Younger Age Groups:** The majority of customers are clustered in the age range of 30-40, with a significant portion also present in the 20-30 age range.
- **Smaller Proportion of Older Customers:** The number of customers in the older age groups (50+) gradually decreases, suggesting that the bank's customer base is relatively younger.

3. Pie Chart:

Chart:

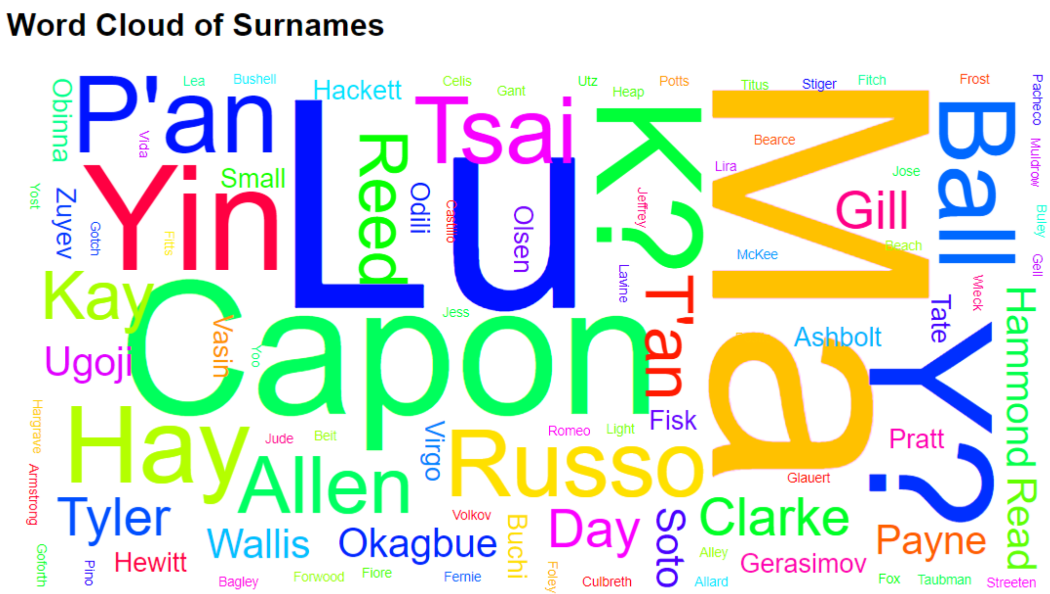


Observations:

- **Significant Churn Rate:** The pie chart indicates a relatively high churn rate, with approximately 20% of customers (2037 out of 10,000) exiting the bank.
- **Majority of Customers Stayed:** Despite the notable churn, the majority of customers (7963 out of 10,000) remained with the bank.
- **Customer Retention Focus:** The data suggests that the bank could benefit from implementing strategies to address the factors contributing to customer churn.

4. Word Cloud:

Chart:

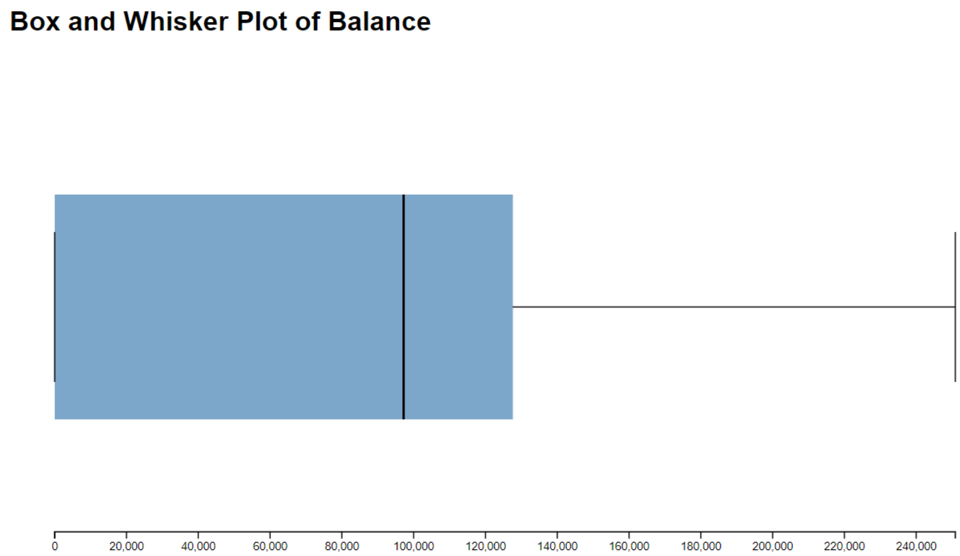


Observations:

- **Common Surnames:** The word cloud reveals the most frequent surnames among the customers, with larger font sizes indicating higher occurrences.
- **Surname Clusters:** Certain surnames seem to be clustered together, suggesting potential familial or regional connections among customers.
- **Surname Diversity:** The word cloud also showcases a diverse range of surnames, reflecting the bank's customer base from various backgrounds.

5. Box Plot:

Chart:



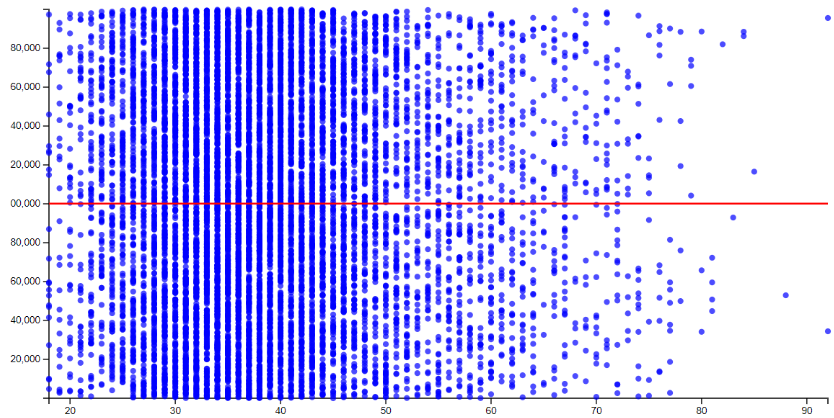
Observations:

- **Median Balance:** The median balance (represented by the line within the box) appears to be around 100,000, indicating that half of the customers have a balance below this value and the other half have a balance above it.
- **Distribution of Balances:** The box represents the interquartile range (IQR), showing the middle 50% of the data. The relatively short length of the box suggests that a significant portion of customers have balances within a specific range, with a smaller portion having balances outside this range.
- **Outliers:** The plot shows a single outlier on the right side of the whisker, indicating a customer with a significantly higher balance compared to the rest of the sample.

6. **Regression Plot:**

Chart:

Regression Plot of Age vs Estimated Salary



Observations:

- **Weak Correlation:** Age and estimated salary have a weak positive correlation.
- **Clustering and Outliers:** Data points are clustered and there are outliers, indicating individual variation.
- **Limited Predictive Power:** Age alone is not a strong predictor of estimated salary.

Hypothesis Testing:

We will perform a Pearson correlation test to check if there's a statistically significant linear relationship between Age and Balance.

1. Null Hypothesis (H_0): There is no correlation between Age and Balance (r = 0).
2. Alternative Hypothesis (H_1): There is a significant correlation between Age and Balance (r ≠ 0).

Code:

```
import pandas as pd

from scipy.stats import pearsonr

df = pd.read_csv('Churn_Modelling.csv')



# Calculate the Pearson correlation coefficient and p-value
```

```
corr_coefficient, p_value = pearsonr(df['Age'], df['Balance'])


print(f"Pearson correlation coefficient: {corr_coefficient}")

print(f"P-value: {p_value}")


alpha = 0.05

if p_value < alpha:

    print(f"The correlation is statistically significant (p < {alpha}).")

else:

    print(f"The correlation is not statistically significant (p >=
{alpha}).")
```

Output:

```
Pearson correlation coefficient: 0.0283083683274919 75
P-value: 0.004639542845 77124
The correlation is statistically significant (p < 0.05).
PS C:\Users\Sanjay\Desktop\ADV\exp 7>
```

Interpretation:

1. **Correlation coefficient (r = 0.0283)**:
   - The Pearson correlation coefficient is very close to **0**, which indicates a **very weak positive** linear relationship between **Age** and **Balance**. This means that as **Age** increases, **Balance** slightly increases, but the strength of this relationship is extremely weak.
2. **P-value (p = 0.0046)**:
   - The p-value is less than 0.05, meaning the result is **statistically significant**. This allows us to reject the **null hypothesis (H_0)**, indicating that there is indeed

some correlation between **Age** and **Balance**, even though the correlation is very weak.

Although the relationship between **Age** and **Balance** is statistically significant (since $p < 0.05$), the strength of the correlation is negligible (since the correlation coefficient is close to 0). This suggests that the linear relationship between these two variables is not practically meaningful.

Conclusion:

This experiment demonstrated the effective use of D3.js for data visualization and analysis. By creating various charts, such as bar charts, histograms, pie charts, word clouds, box plots, and regression plots, we were able to gain valuable insights into the customer data. These visualizations helped identify key trends, patterns, and relationships within the data, providing valuable information for decision-making and further analysis. The ability to customize and interact with the visualizations through D3.js enhanced our understanding of the data and facilitated effective communication of findings.