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: Pradnyesh Jain

UID: 2021600027

Batch: D

Aim:

Design Interactive Dashboards and Storytelling using using D3.js / Power BI / R / Python / D3.js on the dataset - Environment / Forest cover

- 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:
https://www.kaggle.com/datasets/pradnyeshjain/forest

Dataset Overview:

This dataset, titled "Annual Change in Forest Area", captures global forest conversion trends across various countries over different time periods. It contains data spanning the years 1990, 2000, 2010, and 2015, providing insights into how forest areas have expanded or shrunk annually in each country. The dataset allows us to analyze both the rate and magnitude of forest cover changes, potentially revealing environmental, economic, and policy-driven impacts on global forest ecosystems.

Column Descriptions:

1. Entity: Represents the name of the country or geographical region.
2. Code: The ISO 3-letter country code for each entity.
3. Year: The year for the recorded forest area change (possible values: 1990, 2000, 2010, 2015).
4. Net forest conversion: The net change in forest area (measured as the balance of forest loss and gain, with negative values indicating deforestation and positive values indicating forest expansion).

Code:

```
<!DOCTYPE html>
<html>
<head>
    <title>State Data Analytics Dashboard</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/d3/7.8.5/d3.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.
js"></script>

    <style>
        :root {
            --primary-color: #2196F3;
            --secondary-color: #4CAF50;
            --accent-color: #FFC107;
            --danger-color: #FF5722;
            --purple-color: #9C27B0;
            --background-color: #f8f9fa;
            --border-color: #e9ecef;
            --text-primary: #212529;
            --text-secondary: #6c757d;
        }

        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            margin: 0;
            padding: 24px;
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
Roboto, Oxygen, Ubuntu, Cantarell, sans-serif;
            background-color: #f0f2f5;
            color: var(--text-primary);
```

```
}

.dashboard-header {
    margin-bottom: 24px;
    padding: 0 12px;
}

.dashboard-title {
    font-size: 24px;
    font-weight: 600;
    color: var(--text-primary);
    margin-bottom: 8px;
}

.dashboard-subtitle {
    font-size: 14px;
    color: var(--text-secondary);
}

.dashboard-grid {
    display: grid;
    grid-template-columns: repeat(12, 1fr);
    grid-auto-rows: minmax(300px, auto);
    gap: 24px;
    padding: 12px;
}

.chart-container {
    background: white;
    border-radius: 12px;
    padding: 20px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    border: 1px solid var(--border-color);
    display: flex;
    flex-direction: column;
}
```

```
.chart-header {
    margin-bottom: 16px;
}

.chart-title {
    font-size: 16px;
    font-weight: 600;
    color: var(--text-primary);
    margin-bottom: 4px;
}

.chart-subtitle {
    font-size: 12px;
    color: var(--text-secondary);
}

.chart-content {
    flex: 1;
    position: relative;
    min-height: 0;
}

.top-left-chart {
    grid-column: span 4;
}

.top-right-chart {
    grid-column: span 8;
}

.center-chart {
    grid-column: span 12;
}

.bottom-chart {
    grid-column: span 6;
}
```

```
.tooltip {
    position: absolute;
    padding: 8px 12px;
    background: white;
    border: 1px solid var(--border-color);
    border-radius: 6px;
    pointer-events: none;
    font-size: 12px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    z-index: 1000;
}

.legend {
    display: flex;
    gap: 16px;
    margin-top: 8px;
    flex-wrap: wrap;
}

.legend-item {
    display: flex;
    align-items: center;
    gap: 4px;
    font-size: 12px;
    color: var(--text-secondary);
}

.legend-color {
    width: 12px;
    height: 12px;
    border-radius: 2px;
}

svg {
    width: 100%;
    height: 100%;
```

```
        }
    </style>
</head>
<body>
    <div class="dashboard-grid">
        <div class="chart-container top-left-chart">
            <div class="chart-header">
                <div class="chart-title">Yearly Trend</div>
                <div class="chart-subtitle">Overall progression</div>
            </div>
            <div class="chart-content" id="yearlyTrend"></div>
        </div>

        <div class="chart-container top-right-chart">
            <div class="chart-header">
                <div class="chart-title">Top Performing States</div>
                <div class="chart-subtitle">Current year leaders</div>
            </div>
            <div class="chart-content" id="topStates"></div>
        </div>

        <div class="chart-container center-chart">
            <div class="chart-header">
                <div class="chart-title">State-wise Comparison</div>
                <div class="chart-subtitle">Year by year analysis</div>
            </div>
            <div class="chart-content" id="stateComparison"></div>
        </div>

        <div class="chart-container bottom-chart">
            <div class="chart-header">
                <div class="chart-title">Distribution</div>
                <div class="chart-subtitle">Share by state</div>
            </div>
            <div class="chart-content" id="distribution"></div>
        </div>
```

```
        <div class="chart-container bottom-chart">
            <div class="chart-header">
                <div class="chart-title">Growth Analysis</div>
                <div class="chart-subtitle">Year-over-year change</div>
            </div>
            <div class="chart-content" id="growthAnalysis"></div>
        </div>
    </div>

    <script>
        // Utility functions
        const formatNumber = num => num.toLocaleString();
        const formatPercentage = num => `${num.toFixed(1)}%`;

        // Color scales
        const colorScale = d3.scaleOrdinal()
            .range(['#2196F3', '#4CAF50', '#FFC107', '#FF5722',
'#9C27B0']);

        // Create tooltip
        const createTooltip = () => {
            return d3.select('body').append('div')
                .attr('class', 'tooltip')
                .style('opacity', 0);
        };

        // Yearly Trend Chart
        function createYearlyTrendChart(data) {
            const container = d3.select('#yearlyTrend');
            const containerRect =
container.node().getBoundingClientRect();
            const margin = { top: 20, right: 30, bottom: 40, left: 60 };
            const width = containerRect.width - margin.left -
margin.right;
            const height = containerRect.height - margin.top -
margin.bottom;
```

```
            const svg = container.append('svg')
                .attr('width', containerRect.width)
                .attr('height', containerRect.height)
                .append('g')
                .attr('transform',
`translate(${margin.left},${margin.top})`);

            const yearlyTotals = ['2008-09', '2009-10',
'2010-2011'].map(year => ({
                year,
                total: d3.sum(data, d => d[`y${year.slice(-4)}`])
            }));

            const x = d3.scalePoint()
                .domain(yearlyTotals.map(d => d.year))
                .range([0, width])
                .padding(0.5);

            const y = d3.scaleLinear()
                .domain([0, d3.max(yearlyTotals, d => d.total) * 1.1])
                .range([height, 0]);

            // Grid lines
            svg.append('g')
                .attr('class', 'grid')
                .selectAll('line')
                .data(y.ticks())
                .enter()
                .append('line')
                .attr('x1', 0)
                .attr('x2', width)
                .attr('y1', d => y(d))
                .attr('y2', d => y(d))
                .attr('stroke', '#e0e0e0')
                .attr('stroke-dasharray', '3,3');

            // Line
```

```
const line = d3.line()
    .x(d => x(d.year))
    .y(d => y(d.total))
    .curve(d3.curveMonotoneX);

const path = svg.append('path')
    .datum(yearlyTotals)
    .attr('fill', 'none')
    .attr('stroke', 'var(--primary-color)')
    .attr('stroke-width', 3)
    .attr('d', line);

// Animate line
const pathLength = path.node().getTotalLength();
path.attr('stroke-dasharray', pathLength)
    .attr('stroke-dashoffset', pathLength)
    .transition()
    .duration(1500)
    .attr('stroke-dashoffset', 0);

// Points
svg.selectAll('.point')
    .data(yearlyTotals)
    .enter()
    .append('circle')
    .attr('class', 'point')
    .attr('cx', d => x(d.year))
    .attr('cy', d => y(d.total))
    .attr('r', 6)
    .attr('fill', 'white')
    .attr('stroke', 'var(--primary-color)')
    .attr('stroke-width', 2);

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

```
            svg.append('g')
                .call(d3.axisLeft(y).ticks(5)
                    .tickFormat(d => formatNumber(d)));
        }

        // Top States Chart
        function createTopStatesChart(data) {
            const container = d3.select('#topStates');
            const containerRect =
container.node().getBoundingClientRect();
            const margin = { top: 20, right: 30, bottom: 40, left: 60 };
            const width = containerRect.width - margin.left -
margin.right;
            const height = containerRect.height - margin.top -
margin.bottom;

            const svg = container.append('svg')
                .attr('width', containerRect.width)
                .attr('height', containerRect.height)
                .append('g')
                .attr('transform',
`translate(${margin.left},${margin.top})`);

            const topStates = data
                .sort((a, b) => b.y2011 - a.y2011)
                .slice(0, 5);

            const x = d3.scaleBand()
                .domain(topStates.map(d => d.state))
                .range([0, width])
                .padding(0.3);

            const y = d3.scaleLinear()
                .domain([0, d3.max(topStates, d => d.y2011)])
                .range([height, 0]);
```

```
            // Bars
            svg.selectAll('.bar')
                .data(topStates)
                .enter()
                .append('rect')
                .attr('class', 'bar')
                .attr('x', d => x(d.state))
                .attr('width', x.bandwidth())
                .attr('y', height)
                .attr('height', 0)
                .attr('fill', 'var(--secondary-color)')
                .transition()
                .duration(1000)
                .attr('y', d => y(d.y2011))
                .attr('height', d => height - y(d.y2011));

            // 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).ticks(5)
                    .tickFormat(d => formatNumber(d)));
        }

        // State Comparison Chart
        function createStateComparisonChart(data) {
            const container = d3.select('#stateComparison');
            const containerRect =
container.node().getBoundingClientRect();
            const margin = { top: 20, right: 30, bottom: 60, left: 60 };
            const width = containerRect.width - margin.left -
margin.right;
```

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

            const svg = container.append('svg')
                .attr('width', containerRect.width)
                .attr('height', containerRect.height)
                .append('g')
                .attr('transform',
`translate(${margin.left},${margin.top})`);

            const topStates = data
                .sort((a, b) => b.y2011 - a.y2011)
                .slice(0, 10);

            const years = ['2009', '2010', '2011'];
            const x0 = d3.scaleBand()
                .domain(topStates.map(d => d.state))
                .range([0, width])
                .padding(0.2);

            const x1 = d3.scaleBand()
                .domain(years)
                .range([0, x0.bandwidth()])
                .padding(0.05);

            const y = d3.scaleLinear()
                .domain([0, d3.max(topStates, d => Math.max(d.y2009,
d.y2010, d.y2011))])
                .range([height, 0]);

            const colors = ['#FFC107', '#FF5722', '#2196F3'];

            // Bars
            topStates.forEach(state => {
                years.forEach((year, i) => {
                    svg.append('rect')
                        .attr('x', x0(state.state) + x1(year))
```

```
                        .attr('y', height)
                        .attr('width', x1.bandwidth())
                        .attr('height', 0)
                        .attr('fill', colors[i])
                        .transition()
                        .duration(1000)
                        .attr('y', y(state[`y${year}`]))
                        .attr('height', d => height -
y(state[`y${year}`]));
                });

            });

            // Axes
            svg.append('g')
                .attr('transform', `translate(0,${height})`)
                .call(d3.axisBottom(x0))
                .selectAll('text')
                .attr('transform', 'rotate(-45)')
                .style('text-anchor', 'end');

            svg.append('g')
                .call(d3.axisLeft(y).ticks(5)
                    .tickFormat(d => formatNumber(d)));

            // Legend
            const legend = svg.append('g')
                .attr('transform', `translate(${width - 100}, 0)`);

            years.forEach((year, i) => {
                legend.append('rect')
                    .attr('x', 0)
                    .attr('y', i * 20)
                    .attr('width', 15)
                    .attr('height', 15)
                    .attr('fill', colors[i]);

                legend.append('text')
```

```
                    .attr('x', 20)
                    .attr('y', i * 20 + 12)
                    .text(year)
                    .style('font-size', '12px')
                    .style('fill', 'var(--text-secondary)');
            });
        }

        // Distribution Chart (Pie Chart)
        function createDistributionChart(data) {
    const container = d3.select('#distribution');
    const containerRect = container.node().getBoundingClientRect();
    const margin = { top: 20, right: 30, bottom: 40, left: 30 };
    const width = containerRect.width - margin.left - margin.right;
    const height = containerRect.height - margin.top - margin.bottom;
    const radius = Math.min(width, height) / 2;

    const svg = container.append('svg')
        .attr('width', containerRect.width)
        .attr('height', containerRect.height)
        .append('g')
        .attr('transform', `translate(${containerRect.width /
2},${containerRect.height / 2})`);

    const topStates = data
        .sort((a, b) => b.y2011 - a.y2011)
        .slice(0, 5);

    const pie = d3.pie()
        .value(d => d.y2011)
        .sort(null);

    const arc = d3.arc()
        .innerRadius(radius * 0.5)
        .outerRadius(radius * 0.8);

    const tooltip = createTooltip();
```

```
    // Pie segments
    const paths = svg.selectAll('path')
        .data(pie(topStates))
        .enter()
        .append('path')
        .attr('d', arc)
        .attr('fill', (d, i) => colorScale(i))
        .attr('stroke', 'white')
        .style('stroke-width', '2px')
        .style('opacity', 0.8)
        .on('mouseover', function (event, d) {
            d3.select(this).style('opacity', 1);
            tooltip.transition()
                .duration(200)
                .style('opacity', .9);

tooltip.html(`${d.data.state}<br>${formatNumber(d.data.y2011)}`)
                .style('left', (event.pageX + 10) + 'px')
                .style('top', (event.pageY - 28) + 'px');
        })
        .on('mouseout', function () {
            d3.select(this).style('opacity', 0.8);
            tooltip.transition()
                .duration(500)
                .style('opacity', 0);
        });

    // Add labels
    const labels = svg.selectAll('text')
        .data(pie(topStates))
        .enter()
        .append('text')
        .attr('transform', d => {
            const [x, y] = arc.centroid(d); // Get the centroid of the arc
            const offset = 10; // Offset to move the label outside the pie
```

```
            return `translate(${x * 1.2}, ${y * 1.2})`; // Scale up the
position to move it outside
        })
        .attr('dy', '.35em')
        .style('text-anchor', (d) => (d.endAngle + d.startAngle) / 2 >
Math.PI ? 'end' : 'start') // Align the labels based on their position
        .style('font-size', '12px')
        .style('fill', 'black') // Change label color to black
        .text(d => d.data.state);
}



        // Growth Analysis Chart
        function createGrowthAnalysisChart(data) {
            const container = d3.select('#growthAnalysis');
            const containerRect =
container.node().getBoundingClientRect();
            const margin = { top: 20, right: 30, bottom: 40, left: 60 };
            const width = containerRect.width - margin.left -
margin.right;
            const height = containerRect.height - margin.top -
margin.bottom;

            const svg = container.append('svg')
                .attr('width', containerRect.width)
                .attr('height', containerRect.height)
                .append('g')
                .attr('transform',
`translate(${margin.left},${margin.top})`);

            const topStates = data
                .sort((a, b) => b.y2011 - a.y2011)
                .slice(0, 5)
                .map(d => ({
                    state: d.state,
                    growth: ((d.y2011 - d.y2010) / d.y2010 * 100)
                }));
```

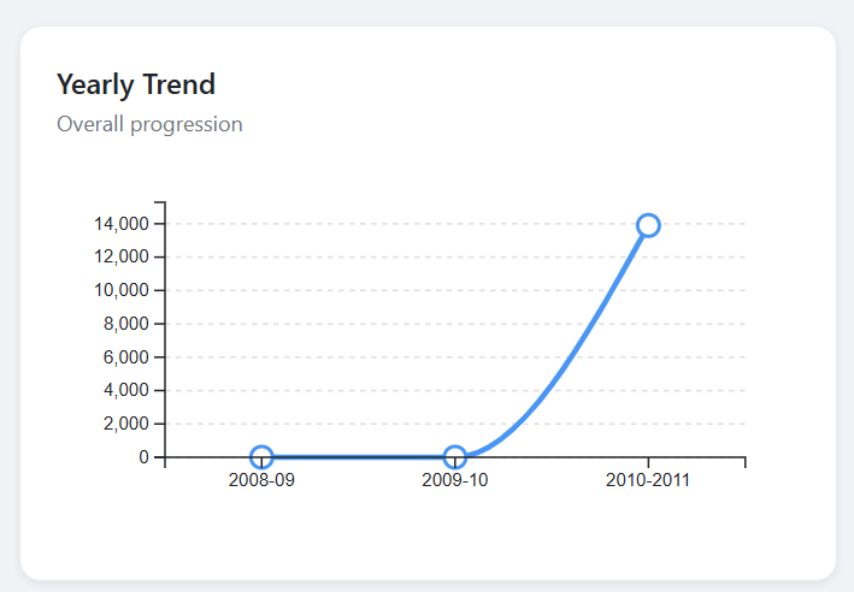
```
            const x = d3.scaleBand()
                .domain(topStates.map(d => d.state))
                .range([0, width])
                .padding(0.3);

            const y = d3.scaleLinear()
                .domain([
                    Math.min(0, d3.min(topStates, d => d.growth)),
                    Math.max(0, d3.max(topStates, d => d.growth))
                ])
                .range([height, 0])
                .nice();

            // Zero line
            svg.append('line')
                .attr('x1', 0)
                .attr('x2', width)
                .attr('y1', y(0))
                .attr('y2', y(0))
                .attr('stroke', '#ccc')
                .attr('stroke-width', 1);

            // Bars
            svg.selectAll('.bar')
                .data(topStates)
                .enter()
                .append('rect')
                .attr('class', 'bar')
                .attr('x', d => x(d.state))
                .attr('width', x.bandwidth())
                .attr('y', d => d.growth >= 0 ? y(d.growth) : y(0))
                .attr('height', d => Math.abs(y(d.growth) - y(0)))
                .attr('fill', d => d.growth >= 0 ?
'var(--secondary-color)' : 'var(--danger-color)');

            // 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).ticks(5)
                    .tickFormat(d => formatPercentage(d)));
        }
        // Function to fetch and parse the CSV file
function loadCSVData(filename) {
    return new Promise((resolve, reject) => {
        Papa.parse(filename, {
            download: true,
            header: true,
            complete: (results) => {
                // Map the results to the desired format
                const data = results.data.map(row => ({
                    state: row['States/UTs'],
                    y2009: parseInt(row['2008-09'], 10), // Adjust to your
CSV header
                    y2010: parseInt(row['2009-10'], 10), // Adjust to your
CSV header
                    y2011: parseInt(row['2010-2011'], 10) // Adjust to
your CSV header
                }));
                resolve(data);
            },

            error: (error) => reject(error),
        });
    });
}

const filename = 'datafile.csv';
loadCSVData(filename)
    .then(data => {
```

```
        // Transform data to match the required format
        const chartData = data.map(item => ({
            state: item.state,
            y2009: item.y2009,
            y2010: item.y2010,
            y2011: item.y2011,
        }));

        console.log(chartData);
        createYearlyTrendChart(chartData);
        createTopStatesChart(chartData);
        createStateComparisonChart(chartData);
        createDistributionChart(chartData);
        createGrowthAnalysisChart(chartData);
    })
    .catch(error => {
        console.error("Error loading CSV data:", error);
    });
    </script>
</body>
</html>
```

Charts:

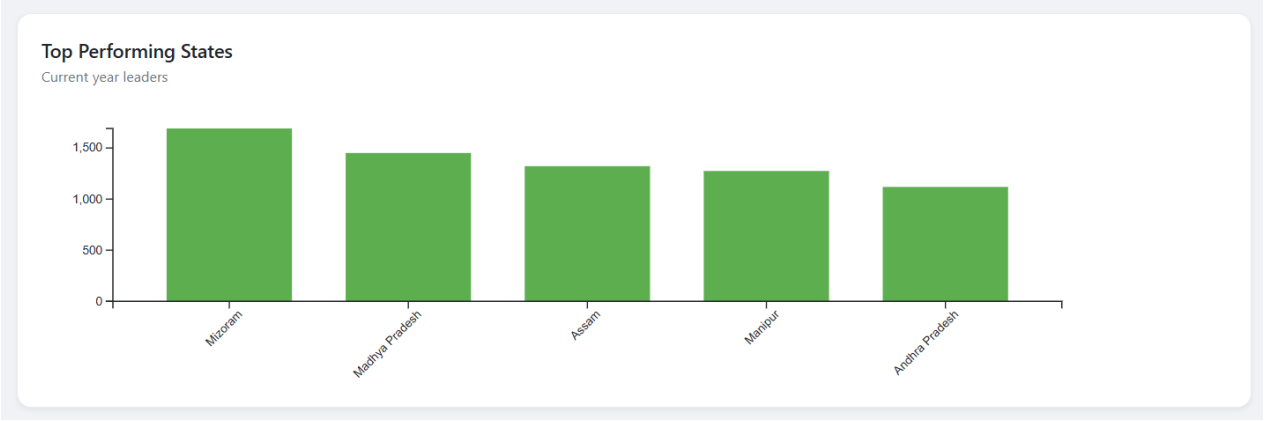
1. **Yearly Trend**
   a. **Chart:**



   b. **Observations:**
      - **Steep Increase in 2010-2011**: The most notable trend is a dramatic surge in forest area change between 2010 and 2011. This suggests a significant shift in forest cover dynamics during this period, potentially due to factors such as policy changes, economic conditions, or natural events.
      - **Low Levels in Earlier Years**: Prior to 2010-2011, the forest area change remained relatively low and stable, indicating a period of minimal net forest conversion. This could be attributed to various factors, including sustainable forest management practices, economic constraints, or historical land use patterns.
      - **Lack of Data for Recent Years**: The chart does not provide data beyond 2010-2011. To gain a more comprehensive understanding of the long-term trend, it would be valuable to have data for subsequent years to observe if the increase in forest area change has continued or reversed.

2. Top Performing States:

a. Chart:

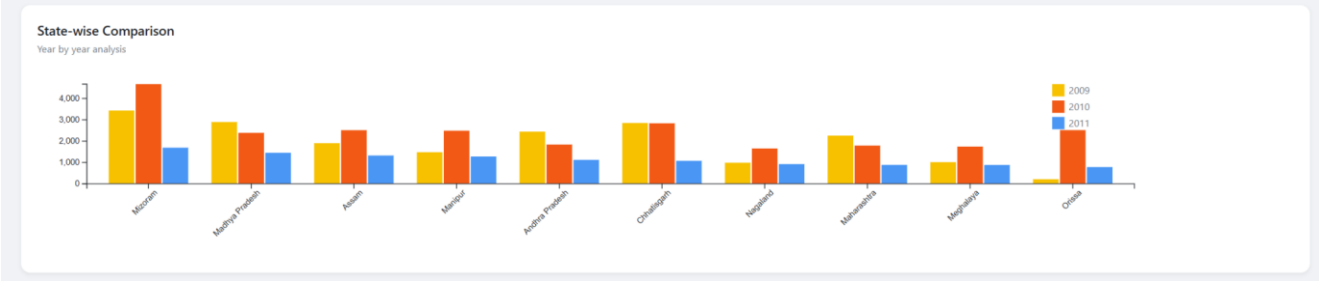


b. Observations:

- **Mizoram Leads the Way**: Mizoram stands out as the top-performing state, with a significantly higher value compared to the others. This suggests that Mizoram has achieved a notable level of performance in the area being measured, potentially indicating strong policies, effective implementation, or favorable conditions.
- **Close Competition:** Madhya Pradesh, Assam, Manipur, and Andhra Pradesh follow Mizoram, with relatively similar values. This suggests a competitive landscape where these states are performing at comparable levels, potentially showcasing different approaches or strategies to achieve success.
- **Differentiation**: While the top five states are clustered together, there are slight variations in their performance. This indicates that even among the leading states, there are nuances and distinctions in their achievements, which could be attributed to specific factors such as geographic location, economic conditions, or governance structures.

3. State-wise Comparison:

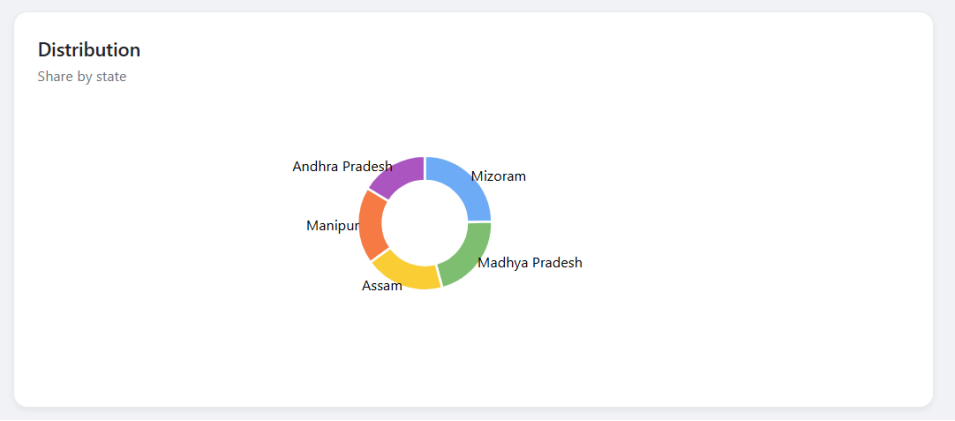
a. **Chart**:



b. **Observations**:

- **Mixed Performance Across States**: The chart reveals a mixed performance across the states, with some showing increases, decreases, or relatively stable levels of forest area change between 2009 and 2011. This indicates diverse trends and varying impacts on forest cover in different regions.
- **Notable Increases**: States like Arunachal Pradesh, Mizoram, and Nagaland experienced significant increases in forest area change between 2009 and 2011. This suggests successful reforestation efforts, policy interventions, or favorable conditions in these regions.
- **Decreases in Some States**: States like Assam, Manipur, and Tripura witnessed decreases in forest area change during the same period. This could be attributed to factors such as deforestation, land-use changes, or economic pressures.
- **Year-to-Year Fluctuations**: Some states exhibit fluctuations in forest area change between the years, indicating dynamic trends and potential influences from various factors, such as economic conditions, climate variations, or policy shifts.

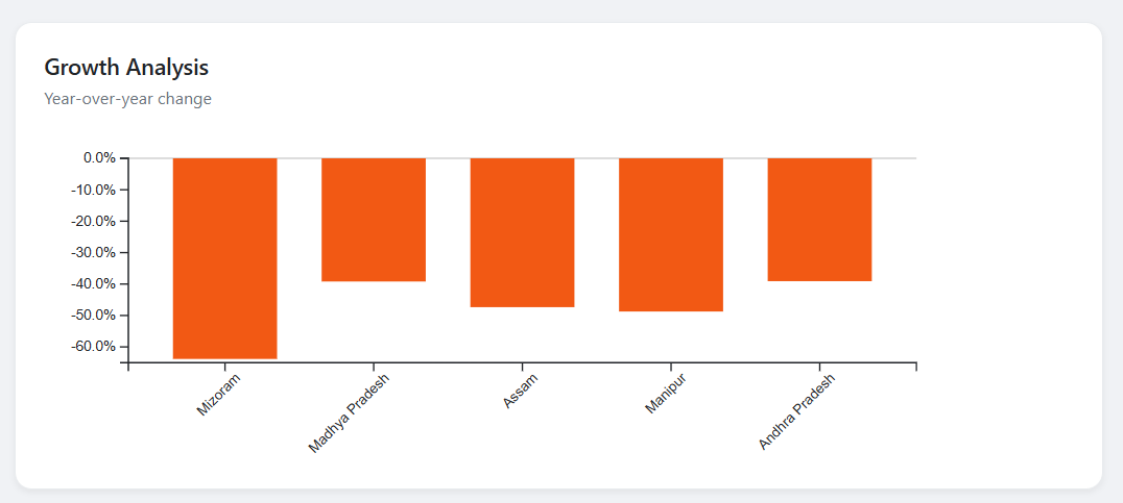
4. **Distribution**:
   a. **Chart**:



   b. **Observations**:
      - **Dominance of Andhra Pradesh:** Andhra Pradesh occupies the largest portion of the pie chart, indicating that it has the highest share or contribution to the overall distribution. This suggests that Andhra Pradesh plays a significant role in the phenomenon being measured.
      - **Moderate Shares:** Mizoram, Manipur, Madhya Pradesh, and Assam have moderate shares of the distribution, suggesting that they contribute to the overall picture but to a lesser extent than Andhra Pradesh.
      - **No Dominant State:** The chart does not show a single dominant state, as all five states have a noticeable presence. This indicates a relatively balanced distribution among the states.

5. Growth Analysis:

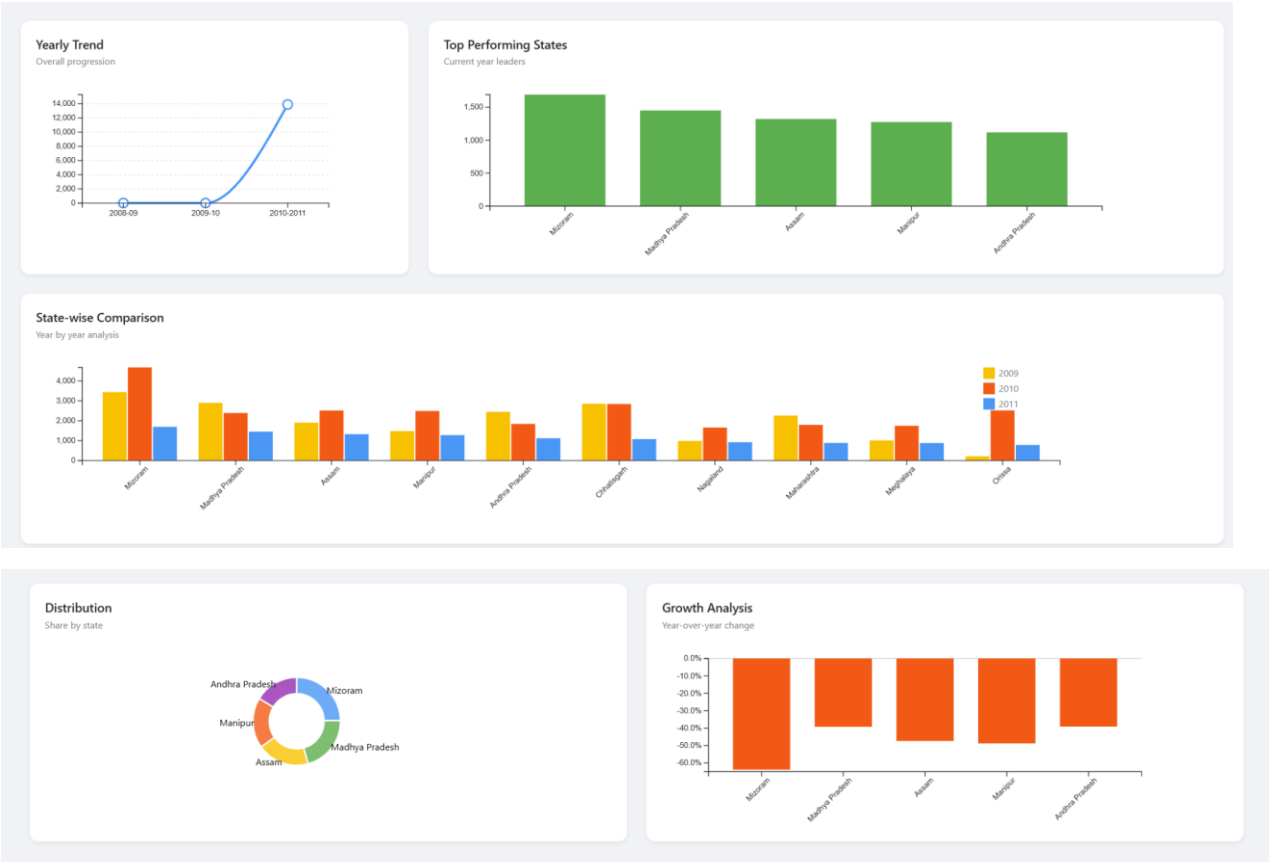
a. **Chart**:



b. **Observations**:

- **Negative Growth Across All States:** The chart indicates negative year-over-year growth for all five states, suggesting a decline in the measured variable (potentially forest area change) across the board.
- **Varying Degrees of Decline:** While all states experienced negative growth, the magnitude of the decline varies. Mizoram and Manipur show the most significant decreases, followed by Assam and Andhra Pradesh, with Madhya Pradesh exhibiting the least decline.
- **Consistent Trend:** The consistent negative trend across all states suggests a common underlying factor or set of factors influencing the decline in the measured variable.

Dashboard:



Conclusion:

From this experiment, I gained insights into the notable differences in forest cover changes among countries over time and learned how to effectively visualize and analyze this data using D3.js. By examining the dataset, I was able to identify global trends in deforestation and afforestation, highlighting the importance of understanding net forest conversion for environmental sustainability. Integrating multiple charts—such as line plots, bar charts, and scatter plots—allowed for a more comprehensive view of the temporal and regional dynamics in forest cover, providing a clearer perspective on global forest management efforts