dataProcessing.js

```
// Exported array to store life expectancy data from various countries
export var lifeData = [];

// Private variable to store the maximum value found in the dataset
var _maxVal = 0;

// Exported function to get the current maximum value stored
export function getMaxVal() {
    return _maxVal;
}

// Exported function to set a new maximum value
export function setMaxVal(value) {
    _maxVal = value;
}

// Processes raw data from CSV into a more structured format and updates the maximum value found
export function processData(data) {
    // Initialize the structure for a country's data
    let result = {
        country: data.Country,
        years: {}
    };
    let localMax = 0; // Temporary variable to find the maximum value in the current dataset
    // Loop over all properties in the data object
    Object.keys(data).forEach(key => {
        if (key !== 'Country') {
            let value = +data[key]; // Convert the data value to a number
            result.years[key] = { expec: value }; // Assign the value to the corresponding year
            if (value > localMax) localMax = value; // Update local maximum if current value is higher
        }
    });

    // If the local maximum for this dataset is higher than the global max, update the global max
    if (localMax > _maxVal) {
        _maxVal = localMax;
    }
    return result;
}

// Loads life expectancy data from a specified CSV file, processes each row, and stores it in lifeData
export function loadLifeData() {
    return d3.dsv(";", "./data/cleanedData/lifeExpectancy_cleaned_csv.csv", processData)
        .then(function(data) {
            lifeData = data; // Update the lifeData array with the processed data
        });
}

// Loads additional data (like GDP) from a specified CSV file and merges it with existing lifeData
export function loadData(csvPath) {
    return d3.dsv(";", csvPath, processData) // Load and process data from the CSV file
        .then(function(data) {
```

```
            let dataMap = new Map(data.map(item => [item.country, item.years])); // Create a map of country to data
            // Merge the loaded data with existing lifeData
            lifeData.forEach(item => {
                if (dataMap.has(item.country)) { // Check if new data exists for the country
                    let dataByYears = dataMap.get(item.country);
                    Object.keys(item.years).forEach(year => {
                        // If data for the year exists, merge it, otherwise set as null
                        item.years[year].gdp = dataByYears[year] ? +dataByYears[year].expec : null;
                    });
                }
            });
        });
}

```