**Frontend Engineering Challenge: Banking Data Visualization**

This exercise is designed to evaluate your proficiency in building a robust and scalable frontend application using React and TypeScript. Your task is to develop a data processing and visualization solution. We will assess your skills in data manipulation, performance optimization, error handling, and architecting a clean, maintainable codebase.

The dataset, Comprehensive\_Banking\_Database.csv, contains the following columns:

* **Customer ID**: Unique identifier for each customer.
* **Transaction Date**: Date of the transaction.
* **Transaction Type**: Type of transaction (e.g., deposit, withdrawal, transfer).
* **Transaction Amount**: Amount of money involved in the transaction.
* **Account Balance**: Balance in the customer's account after the transaction.
* **Customer Age**: Age of the customer.
* **Customer Gender**: Gender of the customer.
* **Account Type**: Type of account (e.g., savings, checking).
* **Branch Code**: Identifier for the branch where the transaction occurred.
* **Account Opening Date**: Date when the account was opened.

**Submission Guidelines**

1. **Code Quality**: Use proper TypeScript typing, clear variable names, and include comments
2. **Error Handling**: Address edge cases and invalid data scenarios
3. **Performance**: Consider efficiency for large datasets
4. **Business Context**: Explain your decisions from a banking/finance perspective
5. **Testing**: Describe how you would test your implementation

Your final submission should be a fully implemented frontend system. To ensure consistency and privacy across all candidates, we are using GitHub Classroom for submissions. Please follow the steps below carefully:

1. Access the Assignment

* Click the GitHub Classroom link provided: [HFC Analyst Assessment](https://classroom.github.com/a/Q7KHwkSK)
* Sign in with your GitHub account.

1. Repository Setup

* Once you accept the assignment, GitHub Classroom will automatically create a private repository for you under our organization.
* This repository will only be visible to you and our review team.

1. Work on Your Solution

* Clone the repository to your local environment.
* Implement the assignment requirements as outlined in this document.
* Ensure that your code is well-structured, documented, and runnable.

1. Submission

* Commit and push your work to the private repository before the submission deadline.

**Important Notes**

You are only required to submit through GitHub Classroom; please do not email or send code separately.

We will review your code directly from your private repository.

# **Part 1: Foundational Implementation and Data Integrity (45 points)**

**Question 1: Data Normalization and Type Conversion (15 points)**

Implement the data cleaning logic for the following transformations:

// Convert string to number safely

function parseAmount(amount: string): number {

// Handle cases like: "$1,234.56", "1234.56", "", "N/A", null

}

// Standardize gender values

function normalizeGender(gender: string): 'Male' | 'Female' | 'Other' {

// Handle cases like: "M", "male", "FEMALE", "f", "", "Non-binary"

}

// Parse date safely

function parseDate(dateString: string): Date | null {

// Handle various date formats: "2023-12-15", "12/15/2023"

}

**Question 2: Data Validation and Filtering (20 points)**

Design and implement a robust data validator module. This module should validate the dataset and filter out invalid records. Your implementation should:

* Validate that Transaction Amount is positive for deposits.
* Ensure that Account Balance is a logical value after a transaction.
* Confirm that Customer Age is within a reasonable range (e.g., 18-120).
* Verify data consistency for a single customer across multiple transactions (e.g., consistent Customer Age).
* Provide a clear explanation of your filtering criteria for invalid records.

**Question 3: Edge Case Management (10 points)**

Describe your strategy for handling the following data quality issues. Provide a code snippet or a high-level design for each solution:

* A single customer has different ages recorded across transactions
* The account balance does not mathematically reconcile with the previous balance.
* Duplicate transaction records are present in the dataset

# **Part 2: Business Logic & Scalable Analysis (40 points)**

**Question 4: Business Insights and Aggregation (25 points)**

Write functions to calculate these business metrics:

// Calculate monthly transaction volume by branch

function getMonthlyVolumeByBranch(data: CleanedTransaction[]): Map<string, Map<string, number>>

// Find customers with unusual spending patterns

function detectAnomalousTransactions(data: CleanedTransaction[]): CleanedTransaction[]

// Calculate customer lifetime value

function calculateCustomerLTV(customerId: number, data: CleanedTransaction[]): number

**Question 5: Strategic Data Aggregation (15 points)**

A senior manager needs to answer the following strategic questions. Describe your data analysis and aggregation approach for each:

1. Underperforming branches: What metrics would you use to identify branches that are underperforming?
2. High-Value Customer segments: How would you segment customers to identify those with the highest transaction volumes?
3. Seasonal trends: How would you analyze the data to find seasonal trends in banking activity? What visualizations would you use to support your findings?

# **Part 3: Performance, Visualization & Architecture (45 points)**

**Question 6: Performance Optimization for Large Datasets (15 points)**

Assume thedataset grows to 10 million transactions.

* Identify potential performance bottlenecks in your cleaning function
* Suggest optimizations (data structures, algorithms, processing strategies)
* How would you handle memory constraints?

**Question 7: Visualization and Reporting (20 points)**

Select 5 key visualizations that you would present for a senior manager. For each visualization, provide the following details:

* Chart type (bar, line, pie, scatter, heatmap, etc.)
* X and Y axes (if applicable)
* Data aggregation needed
* Business insight it provides
* Sample code snippet showing data preparation

**Example Format:**

Visualization 1: Branch Performance Dashboard

- Chart: Horizontal bar chart

- Data: Total transaction volume by branch (last 6 months)

- Insight: Identifies top/bottom performing branches

- Code: aggregateByBranch(filterByDateRange(data, last6Months))

**Question 8: Real-Time Data Architecture (10 points)**

A future requirement is to upgrade the dashboard to be "real-time."

* What architectural challenges does the current static CSV-based approach pose for real-time updates?
* Propose a high-level system design for a real-time data flow. What technologies would you consider?
* How would your existing functions and data interfaces need to be modified to support a continuous stream of data?