DS210 - Transaction Analysis with Rust

For my final project, I built a Bank Transaction Analysis platform using Rust and the Charming visualization library. The project utilizes a Kaggle dataset of over 1 million financial transactions, processes the data to compute insights like regional statistics, time series trends, and anomaly detection, and generates an interactive HTML dashboard (using Charming) showcasing the results.

The project was separated into four Rust modules (models, parser, analysis, main), with unit testing implemented to ensure reliability.

Some challenges:

- Cleaning messy input
- Aggregating trends over time and regions
- Detecting anomalies for fraud or unusual activity
- Figuring out the Charming library (with much help from AI)

(More detailed project structure, along with the dataset and Charming library can be found in the README file)

Module:

- parser.rs: Parses the CSV file, handles whitespace cleanup, fixes typos (e.g., "RESTRAUNT" ➔ "RESTAURANT")
- models.rs: Defines core structs (Transaction, RegionStats, MonthStats)
- analysis.rs: Aggregates statistics, calculates medians/percentiles, detects anomalies using the IQR method

- main.rs: Loads the data, performs analysis, generates graphs and outputs an interactive HTML dashboard

---

Data Parsing & Cleaning (parser.rs):

- Set up a reader
- Initialize a new vector to store each row (except the first row being the header)
- Trim the whitespaces of domain and location (string cols)
- Manually replace the typo for “restaurant”
- Push the cleaned row (using the Transaction struct in models.rs) to the vector
- For unit testing, it is done manually by entering information for a single row and testing out the parsing function

---

Setting up Data Structures (models.rs):

- Custom NaiveDate parsing (with help of AI)
- RegionStats (for filtering based on region and performing mathematical computations), MonthStats (for time series graph later on), and the Transaction struct.

Statistical Analysis & Anomaly Detection (analysis.rs):

- By **Region**: Calculated total, average, median transaction value, and transaction counts per city.
- By **Month**: Aggregated transaction value and count over each month of 2022.
- Unit testing for calculating median & percentile.

For this part, I utilized hashmap to store both the region/month as key and a vector (total, average, etc.) as the value. I iterate through the Transaction struct and push either the region or month into the hashmap as key, value or (value, transaction_count) as the value. Following that, I initialize a new vector, loop over the hashmap, and use mathematical formulas to compute the total, average, median, etc. Finally, pushing the results into the RegionStats and MonthStats structs created in models.rs. Certainly, I've also created functions for calculating the median, percentile, and detecting anomalies using the IQR method, though I never end up using the anomalies detection in the visualization.

Creating Visualization Using Charming Library (main.rs):

- Imported all the rust functions
- Generated interactive charts (scatter plots, line charts) using the **Charming** library.
- Charts include:
    - Total Transaction Value by City
    - Average and Median Transaction Value by City
    - Total Number of Transactions by City
    - Monthly Trends in Transaction Value and Transaction Count
- The visualization can be found by opening the auto-generated HTML file in the live server.