Christian Asdourian
DS210
5/1/25

Final Project Written Report

A. Project Overview

- **Goal:** This project aims to analyze a board game dataset, allowing users to explore trends such as average ratings and player count preferences, and make game recommendations based on different criteria.

- **Dataset:** The dataset used is from BoardGameGeek (22,000+ records), containing details such as game names, player counts, ratings, and playtime. The dataset can be found here.

B. Data Processing

- **How it was loaded into Rust:** The data was loaded from a CSV file using the csv crate. It was deserialized into a vector of BoardGame structs using serde.

- **Cleaning or transformations applied:**

    - The dataset may contain malformed data, so fields like MinPlayers and MaxPlayers are wrapped in Option<u8> to handle missing or invalid entries.

    - Invalid rows with non-numeric values (e.g., "N/A") are skipped during loading.

C. Code Structure

- **Modules:**

    - main.rs: The entry point that parses command-line arguments and runs the appropriate analysis or recommendation logic.

    - models.rs: Contains the BoardGame struct and related types.

    - analysis.rs: Contains functions for analyzing game data (e.g., average ratings, extreme ratings).

- recommend.rs: Functions for recommending games based on player count or rating.

- reader.rs: Provides the function for loading CSV data.

- **Key Functions & Types:**

  - **BoardGame struct:** Represents a board game with fields like Name, YearPublished, MinPlayers, MaxPlayers, MfgPlaytime, and AvgRating.

  - **analyze_average_rating() function:** Calculates the average rating of all games with a valid rating.

  - **recommend_games_by_players() function:** Recommends games based on the desired number of players.

  - **load_boardgames() function:** Loads and deserializes the board game data from a CSV file.

D. Tests

- **cargo test output:**
  Run cargo test to check if all functions are correctly implemented. The tests ensure the accuracy of analysis (e.g., average rating) and recommendations (e.g., player count).

- **What each test checks:**

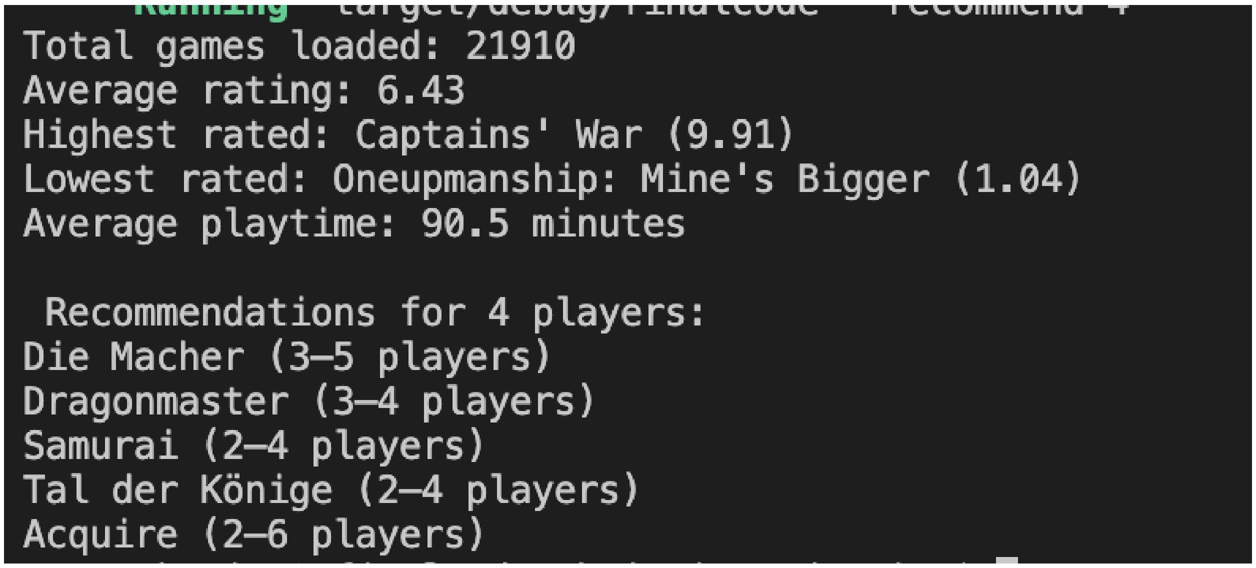
  - **test_analyze_average_rating()**: Ensures the correct calculation of the average rating.

  - **test_recommend_games_by_players()**: Verifies that the game recommendations are correct based on player count.

E. Results

- **Outputs:**
  When run with a valid CSV and command arguments (e.g., cargo run -- --recommend 4), the program will print the recommended games for 4 players, their ratings, and player

counts.



```
Total games loaded: 21910
Average rating: 6.43
Highest rated: Captains' War (9.91)
Lowest rated: Oneupmanship: Mine's Bigger (1.04)
Average playtime: 90.5 minutes

 Recommendations for 4 players:
Die Macher (3–5 players)
Dragonmaster (3–4 players)
Samurai (2–4 players)
Tal der Könige (2–4 players)
Acquire (2–6 players)
```

- **Interpretation:**
  The program successfully filters games based on the user's input (e.g., desired number of players), and the average ratings are calculated correctly. The game recommendations are reasonable based on the criteria.

F. Usage Instructions

- **How to build and run the code:** cargo build
- To run the program with a specific command: cargo run -- --recommend 4
  - Use the --recommend <players> flag to filter games by the number of players.
- **Expected runtime:**

  The runtime is fast for this dataset (~10-15 seconds for full processing).

G. AI-Assistance Disclosure and Other Citations

- ChatGPT was used for multiple instances of debugging.