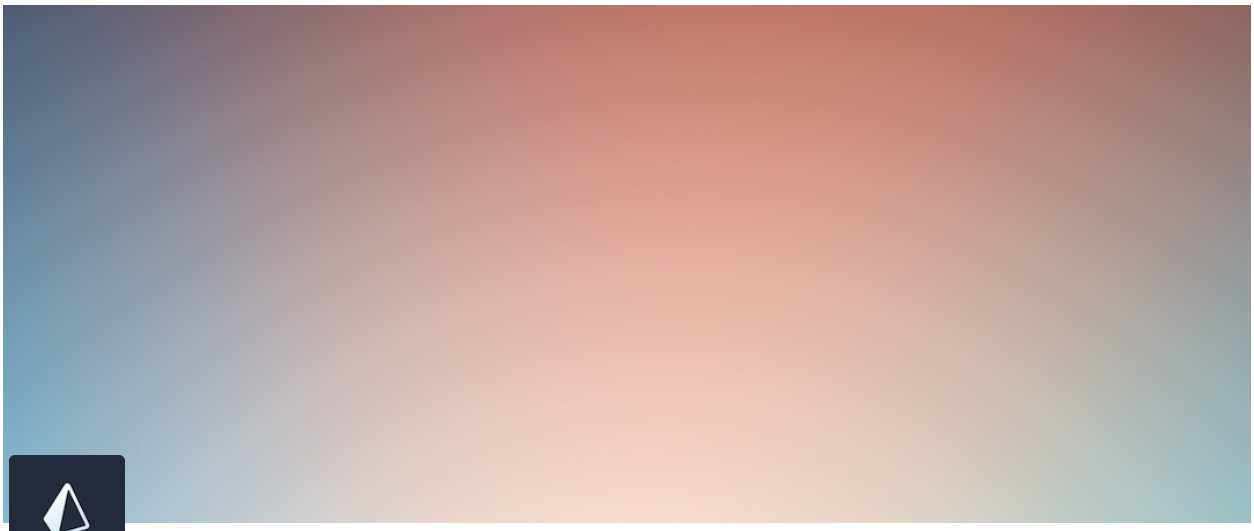

Take Home Challenge: Prisma Core Challenge

Overview

The Challenge

You are tasked with building a simplified query engine in Rust or TypeScript (depending on the role you applied for) for a small in-memory dataset. The goal is to demonstrate your skills in the selected programming language, basic database concepts, and query processing. While simplified, this task should be completed in approximately **5 hours.**

Detailed Requirements

1. **Data Loading:**
    - You need to process CSV into memory, with values of integers, or string type. (A column is of type number if all the cells contents' are strings composed by unicode characters representing digits 0-9). You can use a library or implement CSV parsing yourself.
2. **Query Language and Execution:**
    - Build a function that processes queries on the loaded data.
    - Support queries combining projections of columns and equality (=) and partial ordering filtering (>) by a single column (any column, number or string)
        - Example: `PROJECT col1, col2 FILTER col3 > "value"`
3. **Output:**
    - Implement a function to print query results to the console.
4. **Deliverable**:

    A GitHub project with the code, including a `README.md` answering the Questions below

Questions

- What were some of the tradeoffs you made when building this and why were these acceptable tradeoffs?
- Given more time, what improvements or optimizations would you want to add? When would you add them?
- What changes are needed to accommodate changes to support other data types, multiple filters, or ordering of results?
- What changes are needed to process extremely large datasets
- What do you still need to do to make this code production ready?

We're Interested In

- Ability to use the idiomatic features of TS/Rust that can make this project's code robust and scalable
- Performance-aware data traversal and filtering: try not to loop through data more than necessary
- Human-friendly errors: ensure that every thrown error is handled
- Code clarity and modularity
- Minimal documentation to run the demo and the tests
- Usage of popular tools in the TS/Rust ecosystem for building, testing, linting code
- Code that runs on Node.js LTS (20.18.0) or Rust 1.81.0

> 💡 **Important:** For us, the how and the why are important. Sending us your results somewhat incomplete is totally fine though.

Submission

Please submit your assignment using the link to Rippling provided in the email.

Questions

If you have any questions or concerns, please contact Martin Janse van Rensburg (rensburg@prisma.io)
