**Project: AI chat interface for DataSpell (Data Analysis IDE)**

**Requirement Description**

This project required the implementation of two abstraction layers:

A collection of predefined commands for data manipulation (such as selecting columns and filtering rows based on conditions).

An AI system utilizing an LLM that can produce sequences of those commands driven by user input articulated in natural language.

The primary aim is for the user to submit a requirement, and the program will generate and execute the corresponding transformations on a DataFrame utilizing Python and pandas.

**Implementation Method**

The project was developed in the form of organized Python scripts following a modular structure. I segmented the project into multiple files to uphold the principles of modularity and clarity of responsibilities.

**Layer 1: Predefined transformations for handling data**

This layer consists of fundamental functions for managing a DataFrame. The functions comprise:

**filter\_rows\_by\_predicate**: Filters rows in a DataFrame according to specified predicates.

**select\_columns:** Chooses designated columns within a DataFrame.

**apply\_transformations**: Executes a sequence of transformations generated on a DataFrame.

**Layer 2: Engaging with the LLM**

I am utilizing LLM Studio alongside the llama-3.2-1b-instruct model, which is hosted on my local machine.

I developed an LLMInterface class to interact with the LLM server, transmitting prompts and receiving structured responses in JSON format.

**Main Flow**

The user inputs requirements using natural language.

The LLM produces a set of JSON transformations based on the user's needs.

The transformations are then validated and applied to a DataFrame using the functions from Layer 1.

The final outcome is presented to the user.

**Technologies and Libraries Used**

**Python:**

The primary programming language for the project.

**pandas:**

For data manipulation and transformation tasks on DataFrames.

**requests:**

For HTTP communication with the locally hosted LLM server.

**LLM Studio:**

The local AI server.

**jsonschema:**

For validating the responses generated by the LLM to ensure the consistency of transformations.

**Key Features**

**Column Selection:**

Users can request to select specific columns.

Example: "Select columns 'name' and 'age'"

Row Filtering:

Filtering can be conducted based on conditions such as equality, inequality, or text patterns.

Example: "Filter rows where 'salary' > 60000"

Combined Transformation Sequences:

Supports complex commands that incorporate both column selection and row filtering.

Example: "Select columns 'name', 'salary' and filter rows where 'salary' > 70000"

**Friendly Error Messages:**

Errors, such as the absence of a necessary column for filtering, are communicated with user-friendly messages.

**Local LLM Hosting:**

The llama-3.2-1b-instruct model operates locally on my laptop via LLM Studio, ensuring rapid responses and eliminating reliance on cloud services.