**Documentation for CSV Agent Using LangChain and Google Generative AI**

**Overview**

provides a detailed explanation CSV agent leveraging LangChain's experimental agents and Google Generative AI (Gemini 2.0 model). The agent enables querying CSV files interactively.

**Key Features**

* **Google Generative AI Integration**: Uses the Gemini 2.0 model for natural language processing.
* **CSV Agent**: Processes CSV data using LangChain's create\_csv\_agent functionality.
* **Interactive Querying**: Users can ask questions about the CSV data in natural language.
* **Zero-Shot React Description**: Implements a zero-shot approach to understand and generate responses based on the CSV content.

**Required Libraries and Modules**

The following libraries are utilized in the code:

1. **os**
   * To set environment variables for the Google API Key.
2. **langchain\_experimental.agents.agent\_toolkits.csv.base**
   * Provides the create\_csv\_agent function to create an agent that processes CSV data.
3. **langchain\_google\_genai**
   * Contains the ChatGoogleGenerativeAI class to interface with Google Generative AI.
4. **langchain.agents.agent\_types**
   * Provides predefined agent types, including ZERO\_SHOT\_REACT\_DESCRIPTION.

**Code Workflow**

Step 1: **Set Up Google API Key**

os.environ["GOOGLE\_API\_KEY"] = "<your\_google\_api\_key>"

The API key for Google Generative AI is stored as an environment variable.

**Step 2: Initialize the Agent**

The initialize\_agent function:

* Initializes the ChatGoogleGenerativeAI model with the gemini-2.0-flash-exp configuration.
* Creates a CSV agent using the create\_csv\_agent function, specifying the model, file path, and agent type.

llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash-exp", max\_tokens=None)

agent = create\_csv\_agent(

llm=llm,

path=csv\_file\_path,

agent\_type=AgentType.ZERO\_SHOT\_REACT\_DESCRIPTION,

verbose=True,

allow\_dangerous\_code=True,

)

**Step 3: Query the Agent**

The ask\_question function sends a question to the agent and retrieves the response:

response = agent.invoke({"input": question})

If successful, the response is displayed; otherwise, an error message is printed.