Engineering Interview Task

Objective

You will build a small API that interacts with a trivia dataset and visualize the questions. You can also verify answers and optionally create agents to play the game. You can spend more time on data, backend, frontend.

Expected Duration: 3-4 hours

Task Overview

You will:

1. **Ingest Data** from a publicly available **Jeopardy dataset**.
2. **Store Data** using **PostgreSQL** with **SQLAlchemy ORM**.
3. **Create a REST API** using **FastAPI**.
4. **Use LLM (optional bonus)** to verify answers.
5. **Create AI agents** to play the game.
6. **Add instructions to deploy the project**.
7. **Provide proper documentation and testing**.

---

1. Dataset

Download the dataset from: [Jeopardy Dataset]

Dataset Structure:

Column Name	Description
Show Number	Unique identifier for the show
Air Date	The show air date (YYYY-MM-DD)
Round	One of: "Jeopardy!", "Double Jeopardy!", "Final Jeopardy!"

Category	The question category (e.g., "HISTORY")
Value	The monetary value of the question (e.g., "$200")
Question	The trivia question
Answer	The correct answer

Task:

- Store **subset with question values up to $1200** into a PostgreSQL database using **SQLAlchemy ORM**.

---

2. API Development (FastAPI)

You need to create an API with the following endpoints:

Endpoints:

1. Get a Random Question

GET /question/?round=Jeopardy!&value=$200

- Returns a random question based on the provided **Round** and **Value**.

Example response:

```
{
  "question_id": 3,
  "round": "Jeopardy!",
  "category": "HISTORY",
  "value": "$200",
  "question": "For the last 8 years of his life, Galileo was under house arrest for espousing this man's theory"
}
```

2. Verify an Answer Using AI

POST /verify-answer/

Request Body:

```
{
  "question_id": 7,
  "user_answer": "The answer is Copernics"
}
```

For example the question is "For the last 8 years of his life, Galileo was under house arrest for espousing this man's theory" and the original answer in DB is "Copernicus". Note, users can reply in free text and even have a spelling error.

- This API endpoint should check whether the **user's answer** matches the correct **answer**.

Response Example:

```
{
  "is_correct": true,
  "ai_response": "Yes, Copernicus proposed the heliocentric theory."
}
```

3. Create AI Agents to Play (Optional)

POST /agent-play/

- Implement a basic AI agent that selects a question and attempts to answer it using an **LLM**. The agents can have different skill levels, so that they can mess up the answers sometimes.

Example response:

```
{
  "agent_name": "AI-Bot",
  "question": "Built in 312 B.C. to link Rome & the South of Italy, it's still in use today",
  "ai_answer": "The Appian Way",
  "is_correct": true
}
```

---

3. Database (PostgreSQL + SQLAlchemy)

- Use **SQLAlchemy ORM** to create a **Question** model and manage interactions with a PostgreSQL database.

---

4. Deployment

- Provide instructions on how to deploy the project.

---

5. Documentation

- Use **FastAPI's built-in Swagger UI** for API documentation.
- Provide **a README file** in the GitHub repository with setup instructions:
    - How to install dependencies
    - How to run the API

---

6. Testing (optional)

- Write **tests** for API endpoints

---

7. Submission Guidelines

- Push your code to **a public GitHub repository**.
- Provide clear setup instructions in the **README.md**.

Evaluation Criteria

Criteria	Description
Code Quality	Clean, well-structured, and documented code
Functionality	API correctly retrieves and verifies questions
Database Usage	Proper use of SQLAlchemy and PostgreSQL

API Design	Well-designed, RESTful API endpoints
Error Handling	Handles bad input gracefully
Testing	Sufficient unit tests
Deployment	Easy to deploy
Bonus	AI-based answer verification and agents

Bonus Points 🚀

- Testing and implementing agents using **OpenAI API, LlamaIndex or other framework**.

Good Luck!

Feel free to ask questions if anything is unclear. We look forward to seeing your submission! 🚀