Below is the take-home assignment we'd like you to complete as part of our hiring process. Please read it carefully and submit your solution according to the guidelines provided.

Take-Home Assignment: LLM-Based Ticket Reply Evaluation

Objective

You are given a CSV file (tickets.csv) containing two columns:

1. ticket – A (fictional) customer support ticket message.
2. reply – A response generated by an AI system.

Your task is to **use an OpenAI Large Language Model (LLM)** (e.g., GPT-4o or o1) to **evaluate** each reply along two dimensions:

- **Content** (relevance, correctness, completeness)
- **Format** (clarity, structure, grammar/spelling)

You will then produce a **new CSV** containing four additional columns:

1. content_score
2. content_explanation
3. format_score
4. format_explanation

Please use the scoring scale **1-5** for the "_score" fields and provide **a short textual explanation** in each of the "_explanation" fields.

Assignment Requirements

Input

A CSV file named tickets.csv with the columns:

- ticket (the customer's message)
- reply (the AI-generated response)

Processing with an LLM

You will write a Python script or Jupyter notebook that:

1. Reads tickets.csv.

2. For each row, sends both ticket and reply to an LLM with a suitable prompt asking the model to evaluate the reply's content and format.
3. Parses the LLM's output and extracts two numeric scores (content_score, format_score) and two explanations (content_explanation, format_explanation).
4. Feel free to engineer your prompt in a way that you find most effective.
5. Ensure that your code can handle potential errors (e.g., missing data or API issues).

Output

A CSV file named tickets_evaluated.csv with the following columns:

- ticket
- reply
- content_score
- content_explanation
- format_score
- format_explanation

Deliverables

Notebook or Script

Provide a Jupyter notebook (.ipynb) or a Python script (.py) containing your solution.

tickets_evaluated.csv

The final output with the added scoring columns.

README

- Briefly describe how to set up and run your code.
- Mention any dependencies, environment variables (e.g., for API keys), or library installations needed.

(**Optional but Encouraged**) **Tests**

If you have time, include a small test suite (e.g., using pytest or unittest) that covers your core functions (reading CSV, calling the LLM, writing output, etc.).

Handling API Keys

Please do **not** commit any private keys to your repository or submission.

Use environment variables or a .env file (excluded from version control) for storing sensitive credentials.

Development Guidelines

Write **clean, maintainable, and well-documented** Python code.

You can use any packages you prefer; just ensure they are listed (e.g., in requirements.txt or pyproject.toml).

Explanations can be brief (one or two sentences).

Submission

Send us a link to a public GitHub repository, a zipped folder, or any other method that allows us to easily review your code.

Include instructions in your README on how to run the solution.

Evaluation Criteria

Code Quality

Readability, structure, and adherence to best practices (e.g., PEP-8, error handling).

Prompt Engineering

How effectively you prompt the LLM to produce useful scores and explanations.

Correctness of Output

Your approach to reading/writing CSVs, generating valid numeric scores, and capturing explanations.

Documentation & Tests

Clarity of your README and any test coverage you provide.

We expect this assignment to take about **2–4 hours** in total, depending on your familiarity with LLM APIs. If you have any questions or need clarifications, feel free to reach out.

Good luck, and we look forward to seeing your solution!