Q1. How should one approach the project?

- Read the problem statement carefully.
- Go through the rubric criteria and descriptions for evaluation.
- Download the database and import it into a Python notebook (use Google Colab).
- Connect the database using SQL Agent
- Build the chat agent with well-defined tools and detailed prompts.
- Conclude the project with key findings and business recommendations.

Q2. How do I show the chatbot loop structure in my low-code submission?

As the code is already written for this - you can simply draw a flowchart or list the steps in bullets. The chatbot loop should clearly show the **end-to-end journey of a query**, for example:

- **User input** (what the learner types)
- **Input guardrails** (check if the query is safe and valid)
- **Processing with the LLM** (model generates the response)
- **Output guardrails** (check the response for sensitive or invalid content)
- **Final response to user** (safe, clear output)

This makes it easy for reviewers to see how your chatbot handles queries from start to finish.

Q3. As we have to upload the database, how can we access details in it?

You can use the SQL agent to ask questions in plain English, and the agent will fetch the details for you. For example:

- *Show me all the details for order ID O12486.*
- *Which customer placed order O12486?*
- *What items are in that order?*

This way, you interact with the data conversationally, while the SQL agent does the technical work.

Q4. How can I load the database in Google Colab?

- First, upload the database file to your Colab directory or Google Drive.
- Then, use the following code snippet to load the database

```
from langchain_community.utilities.sql_database import SQLDatabase


order_db = SQLDatabase.from_uri("sqlite:////your_file_path")
```

This happens when the tool is defined to take only one input, but multiple inputs are being passed. You can fix this by:

- Making sure the tool prompt is written to accept the whole query as a single string input (e.g., "Fetch all columns for order ID O45636").
- If you really need multiple inputs, define the tool accordingly, but in most cases, combining everything into one properly structured input string is enough.

Q6: I am getting extra details in the output that I never asked for. What should I do?

This happens when the model "hallucinates." You can fix it by adjusting your prompt:

- Add clear rules like: 'Only answer from the provided data'
- If the answer is not available, reply 'Not available'.
- Break the task into smaller steps so the model stays focused.
- Give sample responses in the prompt to show the format you expect.

Q7: My output guardrails are failing and sensitive information is still being shown. How can I handle this?

If sensitive details slip through, add an extra check in your prompt:

- Tell the model: "Do not include personal, financial, or sensitive information in the output."
- As a safeguard, include a human approval fallback response.

Q8: My input guardrails are not catching problematic queries. What should I do?

If bad inputs aren't being blocked, you can strengthen your prompts:

- Add rules like: *"If the question is harmful, irrelevant, or asks for private data, respond with 'This request cannot be processed.'"*
- *Ask the model first to check validity: "Step 1: Is this input safe and relevant? If yes, go to Step 2: Provide the answer."*