AI Data Scientist Case Study: Automated LLM Report Generation

Objective

Design and implement an LLM-powered workflow that automates business reporting and visualization from structured data. The goal is to build a lightweight prototype that uses LLM to analyze the dataset, generate plots, and produce a cohesive written report — combining analytical reasoning, GenAI orchestration, and sound engineering practices.

Data Description

You will work with the dataset on synthetic BMW sales data, covering multiple regions and vehicle characteristics to explore regional demand, pricing dynamics, and global sales trends.

Core Requirements

1. **LLM-Driven Analysis and Visualization**

   Use LLM of your choice (via API or self-hosted) to generate both plots and narrative insights(e.g. executive summary → analysis sections → recommendations).

   - You may perform **data preprocessing in python** before feeding data into the LLM **and generate plots in python**.
   - You may use any model (OpenAI, Hugging Face, Ollama, etc.).
   - **Note:** Google's *Gemini API* offers free quota suitable for development and testing.

   The LLM should be guided to:

   - Identify and describe sales performance trend over time (e.g., by year or region)
   - Highlight top-performing and underperforming models or markets
   - Explore key drivers of sales (e.g., price, market segment, or model type)
   - Include 1–2 additional insights of your own choice that demonstrate creativity or business understanding.

   Produce a cohesive report (Markdown, HTML, Word, or PDF) combining visuals and commentary. The report should be reproducible when re-run on the same dataset — minor differences are expected due to the stochastic nature of LLMs, but structure and coverage should remain consistent.

2. **Bonus (Optional but Encouraged)**

- Production-grade practices: Clean, modular, well-documented, and reproducible code (e.g., functions, logging, clear folder structure).
- Evaluation framework: Simple method to assess report quality, such as correctness, completeness, readability.

Deliverables

1. **A Codebase** with all necessary files (well-documented).
2. **Generated report output** (word/pdf/markdown/HTML).
3. **A concise executive summary** (up to 2 pages) OR a slide deck (5–8 slides) for a technical audience describing:
    - Your approach and design choices
    - How your evaluation framework works (if included)

Submission Instructions:

- **Commit all deliverables to a public GitHub repository** in your personal account. *(We do not accept zipped submissions attached in emails.)*
- **DO NOT include this document** or sensitive terms like "Singlife" in the repository.
- **DO NOT include API keys or credentials** in your submission.
- **Email the GitHub repository link** to the sender of this test. Submission deadline is communicated by the sender of the test, please check your email.

Evaluation Criteria:

- **Solution Design:** Structure and automation of the workflow
- **Code Quality:** Cleanliness, reproducibility, organization of code
- **Communication:** Clarity and effectiveness in presenting findings through the executive summary or slide deck.
- **Business Impact:** Relevance and depth of analysis revenue.

Key Notes

- Time Commitment: This assessment is expected to take 1-2 days. We understand candidates may have full-time roles & other commitments. Therefore, we welcome partial submissions.
- Assumptions: If certain details are unspecified, make reasonable assumptions and proceed accordingly.
- Tool Requirement: All submissions must use Python as the primary programming language.
- For Candidates with Production Code Experience: **While not required, candidates who demonstrate production-ready code will be recognized**. This includes writing tests, managing dependencies, containerizing workflows, modularizing Jupyter notebooks, utilizing experimentation frameworks, proper documentation, etc.