Section 1 – System Architecture & Module Design

- **End-to-End Flow (`main.py`)**
  - `setup_logging()` configures unified logging (file + stdout).
  - `main()` executes 5 steps in sequence: load data → basic analysis → trend analysis → structural/revenue analysis → chart generation → LLM report.

Section 2– Data Processing & Analytics Strategy

- **Data Loading & Normalization (`data_loader.py`)**
  - Read from local Excel (`DATA_FILE` constant), then normalize column names via `_normalize_column_names`.
- **Business Analysis Functions (`analyzer.py`)**
  - `analyze_basic`: prints row/column counts, missing value stats, and `describe(include="all")` as a **pre-flight data sanity check**.
  - `analyze_trend`:
    - Aggregates yearly `Sales_Volume` and computes `YoY_growth_%`.
    - When `Price_USD` exists, also computes yearly revenue and weighted ASP.
  - `analyze_mix`: model and regional structure analysis, enabling structural insights.
  - `analyze_revenue`: Constructs `Revenue_USD` and aggregates by model/region with a unified `Weighted_ASP_USD` calculation. Adds engine-size-level price analysis via `Engine_Size_L`.
- **Pre-Aggregation for LLM (`_build_summary_for_ai`)**
  - Instead of feeding the entire raw table into the LLM, `_build_summary_for_ai` first computes, in Python:
    - Yearly aggregated volume and growth (`yearly`)
    - Per-region volume / revenue / ASP (`region_summary`)

- Per-model volume / revenue / ASP, plus top/bottom models (`model_top_by_volume`, `model_bottom_by_volume`, `model_top_by_revenue`)
- The approximate relationships between price and engine size / mileage (`engine_size_vs_price`, `corr_price_mileage`)

It then bundles these **key metrics only** into a small JSON dictionary and passes that to the LLM to write the report.

Section 3 – Visualization & LLM Orchestration

- **Visualization Design (`visualizer.py`)**
  - **Year-level**: sales + YoY; revenue + weighted ASP.
  - **Model-level**: top 10 models by volume/revenue; weighted ASP by model.
  - **Region-level**: volume, revenue, and weighted ASP by region + Year×Region heatmap.
  - **Distribution/Relationship**: price distribution, engine size vs price, mileage vs price.
- **LLM Integration & Prompting (`llm_client.py`)**
  - **Section-based generation** via `sections_config`: Executive Summary, Sales Performance, Top & Underperformers, Key Drivers, Strategic Insights.
    - For each section:
      - A shared system prompt .
      - A tailored user prompt embedding `summary_json`, with explicit structural requirements.
      - A list of chart files to be attached to that section.
  - **Text vs Multi-modal calls** in `_call_api_for_section`:
    - If `input_images` are provided, uses `responses.create` with combined `input_text` and `input_image`. Otherwise falls back to `chat.completions.create` with `temperature=0.2` for stable, business-like prose.