Github Repo: https://github.com/timagonch/fast_cars

Modal: https://tgonchar--fastest-cars-dashboard-serve.modal.run/

Report:

For this project I scraped and standardized a dataset of the world's fastest production cars, including details like make/model, year, horsepower, top speed, engine displacement, and engine type. I converted the raw JSON into a structured pandas DataFrame, keeping all columns and preserving null values. From there, I set up a Supabase table with the right schema and wrote Python code to batch-insert the cleaned data. I then connected the Supabase backend to a Streamlit dashboard, where users can explore the cars interactively. The app shows a scatterplot of top speed vs. year (with hover details like engine type and displacement) and a live table of the latest 10 entries ordered by timestamp. Finally, I containerized everything with Modal so it could be deployed as a fully serverless web app.

Using an LLM was a good fit here because I was able to iterate quickly through each stage without pausing to dig through scattered docs. My prompts were direct (e.g., "convert JSON into a DataFrame but don't drop nulls," "design a Supabase table schema for this data," "make a Streamlit scatterplot with hover info"), and the outputs were production-ready snippets I could paste and run. Instead of spending hours troubleshooting naming mismatches, Supabase client quirks, or Modal deployment configs, the LLM guided me to concise fixes (like the cache issue with Streamlit arguments) and helped me move from raw data to a polished, deployed app much faster than if I had pieced it together manually.