QuixLab

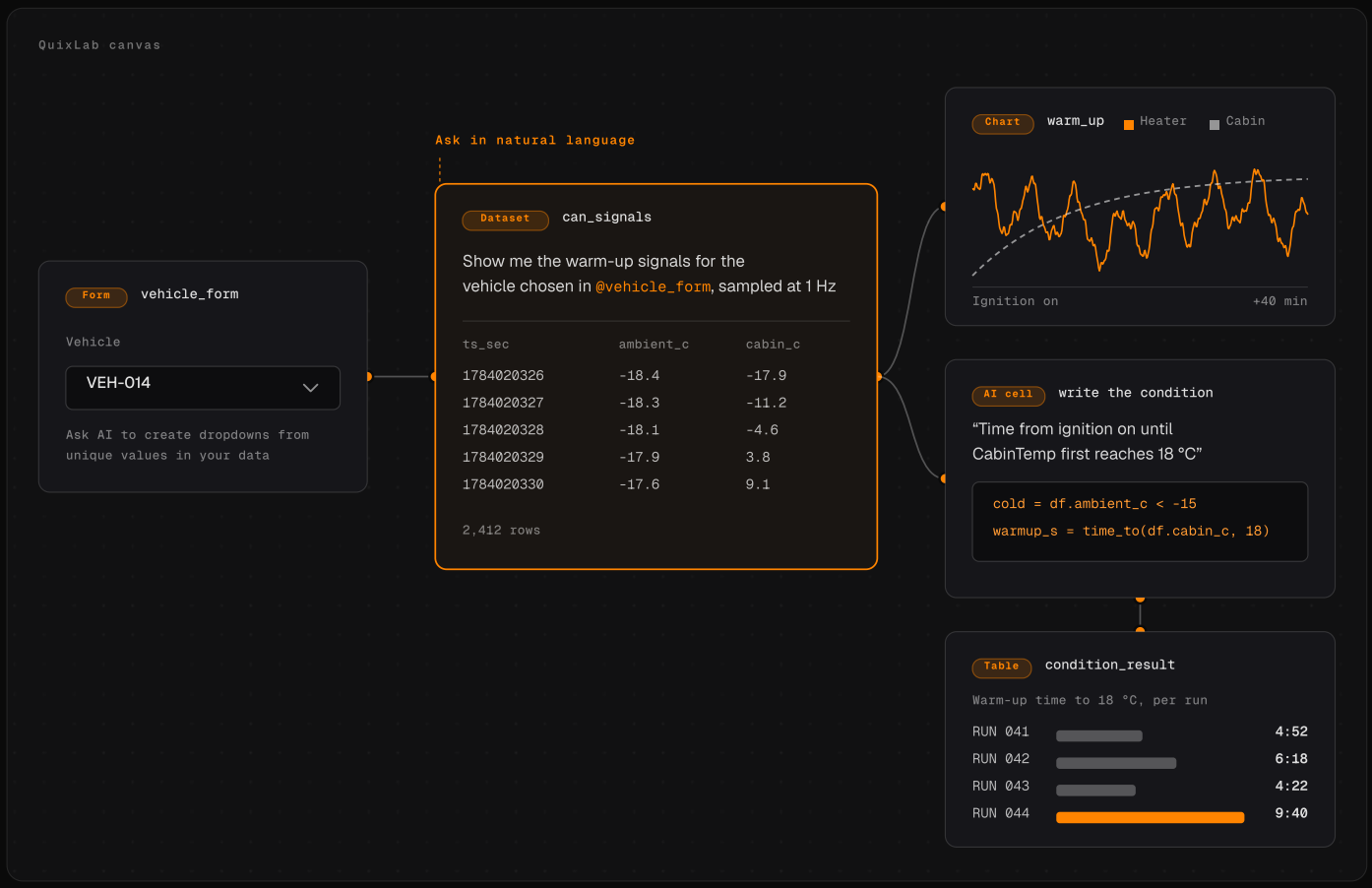
Explore high-resolution test data in the browser, using natural language and no-code tools

A low-code workspace for analyzing high-frequency vehicle test data

QuixLab lets engineers prototype analysis dashboards on data already stored in QuixLake, and share them with others. It is a browser-based canvas with QuixAI built in: write SQL or Python yourself, or have QuixAI write it for you. QuixLab runs inside the Quix platform, so there is no export step and no software to install. Because it queries through the QuixLake API, you can explore and analyse terabytes of test data without waiting minutes for results.

R&D WORKSPACE

QuixLab canvas — query, visualise, and analyse



Query your data, plot what the query returned, then write a filter condition and apply the condition across the campaign. Add a second branch to the canvas whenever you want to try a different idea.

The whole workspace is an ordinary Python file, so you can keep it in version control and run again a year later.

Test data is hard to compare when you have to load it from different sources

Most test engineers analyse their data in a script on their own machine, or in MATLAB, or in DIAdem — for good reason. You can try an idea in five minutes without asking anyone for permission, and concentrate on the engineering question: which channels to compare, and what counts as a failure. Desktop analysis tools are indispensable.

But some analysis is difficult to do at one workstation. Comparing four hundred runs across a whole campaign is much harder than comparing three. Engineers end up loading large batches of files from shared network drives with deep, nested folder structures — by hand, or with custom scripts. QuixLab loads no files at all, because it works with data already ingested into QuixLake.

[01] Compare and analyze data across every run in a campaign

Suppose an engineer wants to know how long the cabin took to reach a usable temperature during a cold-climate durability campaign. In QuixLab that is four steps:

[01]

Pick the runs

Query QuixLake by campaign, vehicle and software version, using the configuration attached when the data was captured. Pipe it into a dropdown node to toggle runs.

[02]

Check the signals

CAN traces were decoded against the DBC file on the way into QuixLake, so messages are already named signals. Plot AmbientTemp, CabinTemp and HeaterPower for one run to confirm.

[03]

Write the condition

A cold start is any run beginning below −15 °C; warm-up time runs from ignition on until CabinTemp reaches 18 °C. State it in plain language and QuixAI writes the Python.

[04]

Run the comparison

Apply the same condition to every run and get one row each: starting ambient temperature, warm-up time, and the energy the heater used. Overlay the results in a plot node.

[02] Have QuixAI write the complex queries

Describe what you want in plain language — lap statistics for each of these sessions. The assistant looks at the data you actually loaded, then writes the code. The code stays visible, so you can read what it did, and you get a preview of the results to check against a run you already know.

[03] Share dashboards without rewriting anything

Useful analysis should be shared so the work is not duplicated twice over. Convert a workspace into a read-only application, exposing a restricted subset of fields colleagues can change, then publish it and share a link. They open it as a dashboard; you rewrite nothing.

QuixLab runs in your own cloud account, or inside your own network as part of an on-premise installation. The Quix platform does the heavy computation for complex queries and returns the result to your browser.

SEE IT WITH YOUR OWN DATA

Book a product demonstration and query full-resolution test data straight from your own cloud storage.

BOOK A DEMO

SEE QUIXLAB IN ACTION

Watch a demo of the QuixAI integration with QuixLab and see how simple it is to explore high-resolution test data.

WATCH THE VIDEO