Tech Test Brief: CSV Analytics Dashboard

Context

You are building a lightweight "Customer Insights" dashboard for internal stakeholders (account managers and senior leadership). The dashboard must read a provided CSV dataset, validate it, analyse it, and present useful insights in a frontend UI.

This is not a pixel perfect UI challenge. We care far more about structure, clarity, and correctness than styling.

Goal:

Produce a working full stack solution that:

1. Ingests a CSV file (no database required).
2. Validates and normalises the data into a clear schema.
3. Exposes the data and derived analytics via a well-designed API.
4. Displays data and insights in a frontend that a non-technical stakeholder can understand.

What we provide

A starter template with:

- A FastAPI backend with CORS enabled for the Angular dev server and a basic / and /health endpoint.
- A basic Angular frontend that calls the backend root endpoint and renders the returned message.

You can keep the stack as provided and use that or you can change/switch it if you explain why and keep the setup simple.

Requirements & Marking:

Backend requirements

You must:

- Load the CSV data (from a file in the repo). You choose whether ingestion happens on startup, via an endpoint upload, or both.
- Validate the data against a defined schema.
    - Handle missing or invalid fields gracefully.

    - Decide what “invalid” means and document it.
    - Provide meaningful error responses.
- Normalise the data into typed models (for example Pydantic models) so the rest of your code is predictable.
- Provide an API that supports the frontend use cases.

Suggested API features (you can adjust if you justify it):

- A “summary” endpoint that returns high level metrics for leadership.
- A “records” endpoint that supports pagination and filtering.
- One or more “analytics” endpoints for charts (group by, time series, breakdowns).
- Clear response shapes and consistent naming.

Constraints:

- No database required. In memory is fine.
- Keep the API fast and predictable.
- Include OpenAPI docs (FastAPI gives this by default, keep it tidy).

Frontend requirements

You must:

- Fetch data from the backend API, handle loading and error states.
- Present the data in a way that makes sense to account managers and leadership.
- Include at least:
    - One chart (bar, line, pie, etc).
    - One table or list view of raw or lightly processed records.
    - At least one filter or interaction (search, date range, category filter, drill down, etc).

UX expectations:

- Clear labels, units, and definitions.
- Make it obvious what the data means and what action someone could take from it.
- Avoid dumping raw CSV rows without context.

Engineering expectations

You must:

- Provide clear README instructions to run backend and frontend – especially if you decide to change the tech stack provided.
- Include tests. We are scoring testing as part of code quality and backend quality.
- Use consistent formatting and sensible project structure.

Deliverables

Submit a repo that includes:

- Backend code + tests
- Frontend code + tests (at least basic component or service tests)