

# **LABLENS**

## **Interactive Blood-Work Explorer**

### **What's Lablens :**

**LabLens** is a comprehensive blood-work analytics platform that transforms raw laboratory test data into actionable insights through three integrated capabilities: data exploration, statistical analysis, and AI-powered querying.

The idea is to let users answer questions about the data **quickly, safely, and transparently**, even if they are not expert programmers.

### **The core problem it solves :**

Medical laboratories generate massive volumes of test results, but analyzing patterns across patients, tests, and time is challenging. LabLens makes it easy to:

- Find patterns in who orders which tests
- Identify repeated testing (potential waste or monitoring)
- Understand typical panel compositions
- Query complex relationships using plain English

### **END goals (what users can do) :**

#### **Load and validate datasets**

- Upload a blood-work file (synthetic during development, real later).
- Ensure it matches the expected schema (columns, types, date format).

#### **Subset and filter data flexibly**

- Filter by any attribute: patient ID (**numorden**), gender (**sexo**), age (**edad**), test name (**nombre**), secondary name (**nombre2**), or date (**Date**).
- Combine filters interactively and save/share these “cohort views” for future use.

#### **Perform descriptive statistics and visual exploration**

- Get numeric summaries (mean, std, quantiles) for numeric tests.
- Get qualitative summaries (counts, rates) for text-based results.
- Examine missing data patterns.

#### **Analyze panels and repeated tests**

- See how many tests each patient has per day (“panels”).
- Track repeat testing for the same patient over time.
- Identify common co-ordered tests on the same day.

#### **Visualize trends and distributions**

- Interactive charts for distributions, time trends, and co-occurrence.
- Heatmaps for co-ordered tests by service (**nombre2**).

#### **Query the dataset using natural language with an LLM assistant**

- Ask questions in plain language (e.g., “Show me all male patients over 50 with repeated glucose tests”).
- LLM converts NL into a safe, auditable query (SQL or Pandas DSL).
- See results in tables/charts along with an **Explain** describing how the query was executed.

### Export and share results

- Export filtered datasets, panels, or analytics to CSV/XLSX.
- Save cohort views to share with colleagues.

### Operate safely and efficiently

- In dev mode, work with synthetic data; in prod mode, respect privacy rules, role-based access, and read-only execution.
- Ensure good performance (fast queries) and stability even for large datasets (~500k rows).

### Technical pipeline :

This is basically the **step-by-step process the app follows to handle and analyze the data.**

#### Ingest & Validate

- Load a dataset (CSV file).
- Make sure the **columns are exactly** {numorden, sexo, edad, nombre, textores, nombre2, Date}.
- Parse the **Date** in day-first format (dd/mm/yyyy).
- Make **edad** an integer.
- Keep **textores** flexible (can be numbers or text).  
→ *This ensures the app only works with well-structured data.*

#### Indexing

- Create **indices** (like fast shortcuts) on **numorden** (patient ID), **nombre** (test), **nombre2** (service/department), and **Date** to speed up queries.
- Precompute **panel aggregates** (so some calculations like “how many tests per patient/day” are ready to go).

#### Cohort Engine

- Lets users **filter/subset the data** with a visual builder (clicks, dropdowns) or raw SQL.
- Users can **save a filter as a “view”** to use later or share with others.
- **View:** a saved cohort/filter with all the rules applied.
  - Example: “All male patients over 50 with repeated glucose tests.”

- Stored in **View** table: {id, name, owner, filter\_dsl, created}.

## Descriptive Stats

- Count how many rows there are, and check for missing values.
- Numeric summaries for numeric columns (mean, std, quantiles).
- Rates for qualitative columns (**textores**).
- → *Basically, quick insights about your dataset.*

## Panels & Repeats

- **Panel:** all the tests a patient has on a single day.
  - Example: Patient 123 had **Glucose**, **Cholesterol**, **Hemoglobin** on **01/01/2025** → this is one panel.
- Count the **number of tests per patient/day**, **unique tests per day**, and **repeated tests across days**.
- Helps see testing patterns over time.

## Co-Ordering

- Identify **tests that are often done together** on the same day.
- Can create **heatmaps** by service (**nombre2**).
- Useful to spot patterns like “patients who get Glucose also often get Cholesterol”.

## LLM Layer

- Users can **ask questions in natural language (NL)**.
- The LLM converts NL into a **safe query (SQL/Pandas DSL)**.
- It runs a **dry-run validator** to prevent unsafe operations.
- Provides an **Explain**: shows exactly what the query did.

## Privacy and security :

- **Dev mode:** uses a **synthetic dataset** to avoid exposing real patient data.
- **Prod mode:** uses **real data** safely:
  - Role-based access control (different users see different things).
  - Encryption at rest (data is encrypted on disk).
  - Read-only queries from the app.
  - Full audit logs.
  - LLM queries run in a **sandbox** with an Explain.

## Minimal Schemas :

- **Result:** one row = one test result for a patient.
  - {numorden, sexo, edad, nombre, textores, nombre2, Date}
- **Panel:** all tests a patient had on a single day.

- {numorden, date, tests[], n\_tests}
- **View:** a saved filter/cohort definition.
  - {id, name, owner, filter\_dsl, created}
- **LLMRun:** logs of LLM queries for auditing.
  - {ts, user, prompt, query\_dsl, rowcount, explain}