Take-Home Exercise: SQL Proxy Service

Overview

In this exercise, you will design and implement a small SQL proxy–style service that sits between users and a relational database (MySQL or PostgreSQL).

The proxy analyzes SQL statements before they reach the database, enforces access policies, classifies sensitive data exposure, and produces meaningful audit information.

This exercise resembles real-world engineering work. We value design decisions, reasoning, and clarity over production readiness.

Goal

Build a service that receives SQL statements from users and:

- Analyzes DDL, DML, and SELECT statements
- Enforces access policies and user-level permissions
- Classifies sensitive data exposure
- Produces audit records
- Executes allowed statements against the database

General Guidelines

- Choose MySQL or PostgreSQL
- Expected effort: 1–2 working days
- You may use any libraries, tools, documentation, or AI assistants
- Focus on design, correctness, and reasoning
- Production readiness is not required

SQL Scope

The proxy should analyze:

- DDL statements (CREATE, ALTER, DROP)
- DML statements (INSERT, UPDATE, DELETE)
- Basic SELECT queries

Advanced SQL features are not required.

Execution Requirements

SELECT statements should be executable if allowed.

DDL and DML statements must be analyzed and may be allowed or rejected.

Please explain your decision in the README.

Data Setup

Create and seed a small schema for demonstration purposes.

Example tables: customers (id, name, email, phone) and orders (id, customer_id, amount, created_at).

You may add additional tables if needed.

Core Requirements

1. SQL Analysis

For each incoming SQL statement, determine (best effort):

- Statement type (DDL / DML / SELECT)
- Referenced database / schema / tables
- For SELECT: projection (selected expressions and/or columns)
- For DML: affected columns where reasonably possible

Perfect SQL parsing is not required. Document assumptions and limitations.

2. Source Table Access Policies (ALLOW / BLOCK)

Policies should be definable at database/schema and table level.

Rules: most specific match wins; BLOCK overrides ALLOW; default deny.

3. User Management

Support multiple users.

Every SQL statement must be evaluated in the context of the requesting user.

4. Query Execution

Allowed statements should be executed.

Denied statements must not be executed and should return a clear error.

5. Data Classification (Basic)

Classify sensitive data exposure based on result sets.

Required classifications: email → PII.Email, phone → PII.Phone.

6. Audit Logging

Every statement (allowed or denied) must produce an audit record.

Explain what you log and why it is meaningful.

Interaction Model

The proxy should support normal SQL usage.

It may be implemented as a transparent proxy or as a service-based interface.

Deliverables

Source code

SQL scripts (DDL + DML)

README explaining setup, rules, analysis approach, design decisions, and limitations

Evaluation Criteria

Correctness of SQL analysis

Clarity of access policy enforcement

User access handling

Classification logic

Audit quality

Overall design and explanation

Final Notes

- We care more about how you think than about a perfect implementation.
- Clear assumptions and honest tradeoffs are highly valued.