**📌 ProSquare SQL - Detailed Roadmap (with Security Features)**

🚀 **Goal:** Build a fully functional SQL engine from scratch, progressing from a simple data store to a fully optimized and secure database system.

**🔗 Project Structure (Referenced in roadmap):**

prosquare\_sql/

️— src/

️ ├️ engine/ # Core database engine

️ │ ├️ storage.py # Manages table storage & retrieval

️ │ ├️ query\_executor.py # Executes parsed SQL commands

️ │ ├️ sql\_parser.py # Parses SQL statements into operations

️ │ ├️ transaction.py # Implements transactions & ACID

️ │ ├️ index.py # Manages indexes for performance

️ │ ├️ optimizer.py # Query optimizer

️ │ ├️ schema.py # Manages table schemas

️ ├️ cli/ # Command-line interface (CLI)

️ │ ├️ repl.py # Interactive shell for SQL commands

️ │ ├️ cli\_commands.py # Helper functions for CLI

️ ├️ security/ # Security features

️ │ ├️ auth.py # Implements authentication & RBAC

️ │ ├️ firewall.py # Blocks SQL injection attacks

️ │ ├️ encryption.py # Handles secure data storage

️ ├️ utils/ # Utility functions

️ │ ├️ file\_manager.py # Handles file storage

️ │ ├️ logger.py # Logs queries and errors

️ │ ├️ config.py # Configuration settings

️ ├️ tests/ # Unit tests

️️— data/ # Stores database files

️️— examples/ # Sample SQL queries

️️— docs/ # Documentation and design notes

️️— README.md # Project overview

️️— requirements.txt # Dependencies

️️— .gitignore # Ignore unnecessary files

️️— setup.py # Packaging (if needed)

**🟢 Phase 1: Minimal SQL Engine (6-7 Weeks)**

🔹 **Goal:** Create a simple SQL engine that can store tables, execute basic queries, and provide foundational security.

**Step 1: Set Up Project (Week 1)**

**What to Do:**

* 👉 Create a **GitHub repository** for version control.
* 👉 Set up the **project structure** according to the layout above.
* 👉 Write a **README.md** explaining project goals and installation steps.
* 👉 Implement config.py in utils/ for basic configurations (default settings, logging paths, etc.).

**Step 2: Implement Storage Engine (Week 2)**

* Implement a **basic file-based storage system** using JSON.
* Define functions in storage.py to **create, insert, and fetch table data**.
* Implement file\_manager.py to handle **secure data storage operations**.

**Step 3: Implement SQL Parser (Week 3)**

* Implement **basic SQL parsing** in sql\_parser.py for:
  + CREATE TABLE table\_name (col1 TYPE, col2 TYPE);
  + INSERT INTO table\_name VALUES (...);
  + SELECT \* FROM table\_name;
* Write tests in test\_parser.py.

**Step 4: Implement Query Execution (Week 4)**

* Implement query\_executor.py to handle **basic query execution**.
* Support simple SELECT operations.

**Step 5: Security Enhancements (Week 5-6)**

* Implement firewall.py for **SQL Injection Prevention**.
* Implement auth.py for **Role-Based Access Control (RBAC)**.
* Implement **basic authentication for users and roles**.

**🟡 Phase 2: Intermediate SQL Engine (8-9 Weeks)**

🔹 **Goal:** Enhance query execution with filtering, joins, indexing, and **secure data storage**.

**Step 6: Improve Storage Format (Week 7)**

* Convert storage system from **JSON to binary format**.
* Implement **compressed storage**.

**Step 7: Implement Query Filtering (WHERE) (Week 8)**

* Implement WHERE clause support in query\_executor.py.
* Write tests for filtering.

**Step 8: Implement Indexing (Week 9-10)**

* Implement **hash-based indexing** in index.py.

**Step 9: Implement Joins (Week 11-12)**

* Implement INNER JOIN in query\_executor.py.
* Implement **index-based joins** for optimization.

**Step 10: Implement Transactions (Week 13-14)**

* Implement **ACID transactions** in transaction.py.
* Implement rollback & logging.

**Step 11: Implement Secure Storage (Week 15)**

* Implement **AES encryption** for sensitive data in encryption.py.

**🔴 Phase 3: Advanced SQL Engine (10-12 Weeks)**

🔹 **Goal:** Optimize query execution, add multi-user support, and scale performance securely.

**Step 12: Query Optimization (Week 16-17)**

* Implement **query optimization strategies** in optimizer.py.

**Step 13: Improve Transactions (Week 18-19)**

* Implement **multi-user transaction handling**.

**Step 14: Implement Multi-User Concurrency (Week 20-21)**

* Implement **locking mechanisms** to prevent conflicts.

**Step 15: Implement Network Security (Week 22-23)**

* Implement **a secure SQL server** in server.py to handle remote queries.

**Final Notes**

🚀 **Phase 1 (6-7 weeks)** → 🎯 **Basic SQL engine with security** 🚀 **Phase 2 (8-9 weeks)** → 🎯 **Optimized SQL with indexing, joins, transactions, encryption** 🚀 **Phase 3 (10-12 weeks)** → 🎯 **Full-scale SQL engine with networking & multi-user support**

✅ **Publish each phase on GitHub** 🎉  
🚀 **Let me know when you're ready to start!**