Here's a **detailed roadmap** for your SQL engine project, covering **all phases (1, 2, and 3)** while aligning with the **project structure** I provided earlier. This roadmap is designed for **copy-pasting into your project folder** so you can track your progress easily.

**📌 ProSquare SQL - Project Roadmap**

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

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

graphql

CopyEdit

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 # (Later) Implements transactions & ACID

│ │ ├── index.py # (Later) Manages indexes for performance

│ │ ├── optimizer.py # (Later) Query optimizer

│ │ ├── schema.py # (Later) Manages table schemas

│ ├── cli/ # Command-line interface (CLI)

│ │ ├── repl.py # Interactive shell for SQL commands

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

│ ├── 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 (5-6 Weeks)**

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

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

✅ Create the **GitHub repository**  
✅ Set up **project structure**  
✅ Write a **README.md** with project goals  
✅ Implement config.py in utils/ for basic configurations

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

📂 **Modify: src/engine/storage.py**  
✅ Define a **table storage system** (JSON or CSV initially)  
✅ Implement functions:

* create\_table(table\_name, schema) → Stores table schema
* insert(table\_name, data) → Adds rows to a table
* fetch\_all(table\_name) → Retrieves all rows

📂 **Modify: src/utils/file\_manager.py**  
✅ Functions to handle **reading/writing tables** to disk

📂 **Modify: tests/test\_storage.py**  
✅ Write unit tests for **storage engine functions**

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

📂 **Modify: src/engine/sql\_parser.py**  
✅ Parse **basic SQL commands**:

* CREATE TABLE table\_name (col1 TYPE, col2 TYPE);
* INSERT INTO table\_name VALUES (...);
* SELECT \* FROM table\_name;

📂 **Modify: tests/test\_parser.py**  
✅ Write unit tests for SQL parsing

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

📂 **Modify: src/engine/query\_executor.py**  
✅ Implement logic for executing parsed queries  
✅ Handle **basic SELECT operations**

📂 **Modify: tests/test\_executor.py**  
✅ Unit tests for query execution

**Step 5: Command-Line Interface (Week 5)**

📂 **Modify: src/cli/repl.py**  
✅ Build an **interactive shell** to execute queries

📂 **Modify: src/cli/cli\_commands.py**  
✅ Implement command-handling functions

**Phase 1 Completion**

✅ Working **SQL engine** that supports:

* Table creation
* Data insertion
* Simple SELECT \* queries  
  ✅ Publish **Phase 1 on GitHub** 🎉

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

🔹 **Goal:** Enhance query execution with filtering, joins, indexing, and transactions.

**Step 6: Improve Data Storage (Week 6-7)**

📂 **Modify: src/engine/storage.py**  
✅ Switch from JSON/CSV to a **binary file format** for efficiency

📂 **Modify: src/utils/file\_manager.py**  
✅ Implement **compressed storage**

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

📂 **Modify: src/engine/query\_executor.py**  
✅ Implement SELECT \* FROM table WHERE column=value;

📂 **Modify: tests/test\_executor.py**  
✅ Unit tests for filtering

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

📂 **Modify: src/engine/index.py**  
✅ Implement a **hash index** for faster lookups

📂 **Modify: tests/test\_index.py**  
✅ Unit tests for indexing

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

📂 **Modify: src/engine/query\_executor.py**  
✅ Support INNER JOIN queries

📂 **Modify: tests/test\_executor.py**  
✅ Unit tests for joins

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

📂 **Modify: src/engine/transaction.py**  
✅ Implement a **basic transaction log**  
✅ Ensure ACID compliance

**Phase 2 Completion**

✅ Working **SQL engine** with:

* Indexing
* Query filtering (WHERE)
* Joins (JOIN)
* Basic transactions  
  ✅ Publish **Phase 2 on GitHub** 🎉

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

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

**Step 11: Query Optimization (Week 15-16)**

📂 **Modify: src/engine/optimizer.py**  
✅ Implement **query optimization** strategies

**Step 12: Improve Transactions (Week 17-18)**

📂 **Modify: src/engine/transaction.py**  
✅ Implement **rollback & recovery**

**Step 13: Support Multi-User Concurrency (Week 19-20)**

📂 **Modify: src/engine/query\_executor.py**  
✅ Implement **locking mechanisms**

**Step 14: Implement a Basic Network Interface (Week 21-22)**

📂 **Modify: src/server.py**  
✅ Create a **server that listens for SQL queries**

**Phase 3 Completion**

✅ Fully functional **SQL engine**  
✅ Publish **Phase 3 on GitHub** 🎉

**🎯 Final Notes**

* **Phase 1 (5-6 weeks)** → 🎯 **Minimal working SQL engine**
* **Phase 2 (8-9 weeks)** → 🎯 **Optimized SQL engine with indexing, joins, transactions**
* **Phase 3 (10-12 weeks)** → 🎯 **Production-like SQL engine with concurrency & networking**

🚀 **You now have a complete SQL engine project!** 🚀

Let me know when you're ready to set up your GitHub repo and start **Step 1**! 🎯