# TL;DR: Using Cursor for SQL & DB

Writing, debugging, and optimizing SQL doesn't have to be tedious. By organizing schemas, queries, and migrations, you provide context that makes SQL generation and analysis faster.

With Cursor, you can: - Translate plain-language instructions into structured SQL - Modify existing queries or optimize slow ones - Explain complex or legacy queries - Identify syntax, logical, and performance issues

Always review suggestions, validate results, and follow security best practices. Cursor works best as a guided assistant alongside your expertise.

---

# How to Set Up Cursor for SQL & Database Work

Setting up Cursor correctly ensures accurate and reliable assistance. Proper structure gives the AI the context it needs to generate, explain, and optimize SQL without confusion.

## Opening SQL Projects and DB Repos

Open your SQL project folder or repository in Cursor. Include all relevant files such as schemas, seed data, and migrations. This allows Cursor to reference table structures, constraints, and historical changes.

Organize files logically. Separate folders for schemas, migrations, and queries improve clarity and reduce errors. Cursor does not connect to live databases automatically; it relies on the files you provide, keeping you in full control.

## Best File Structure for AI Context

**Recommended Project Layout (PDF-safe):**

| Folder | Purpose |
|---|---|
| schemas/ | Table definitions and constraints |
| migrations/ | Versioned schema changes |
| queries/ | Reusable SQL queries |
| README.md | Project context and notes |

**Mini Checklist** - Schema files included - Migrations accessible - Queries clearly scoped

This structure ensures Cursor can work effectively even with legacy databases.

---

# Writing SQL Queries Using Cursor (Core Workflow)

Cursor speeds up repetitive SQL tasks while keeping you in control. You guide the process, review the output, and finalize the query.

## Generating Queries from Natural Language

Instead of writing SQL manually, describe your goal in plain language and let Cursor generate a draft query.

**Workflow Steps:** 1. Highlight relevant schema or table files 2. Write instructions in plain language 3. Ask Cursor to generate a query 4. Review and refine the SQL

**Prompt Examples (Separated for PDF):**

| Purpose | Prompt / Instruction |
|---|---|
| Data retrieval | Fetch all users who placed more than five orders in the last 30 days |

This reduces manual effort while preserving accuracy.

## Modifying Existing Queries

Cursor can refactor existing SQL without starting from scratch. Highlight the query, describe the change, and review the suggested edits.

| Use Case | Instruction |
|---|---|
| Change reporting period | Modify this monthly orders query to return weekly results |

This is especially useful for legacy SQL.

## Inline SQL Completion vs Full Rewrite

- **Inline completion:** Small changes such as WHERE clauses or JOIN updates
- **Full rewrite:** Structural or performance improvements

Always test changes before deployment.

---

# Explaining and Debugging SQL with Cursor

Cursor helps analyze complex or error-prone SQL while you retain decision-making authority.

## Explaining Legacy or Complex Queries

Use Cursor to break down unreadable SQL into understandable steps.

**Workflow:** 1. Open the query 2. Highlight the section 3. Ask for a plain-language explanation

| Task | Prompt |
|---|---|
| Query explanation | Explain what this query does step by step |

This helps with onboarding, audits, and refactoring.

### Debugging Common SQL Errors

Cursor can identify syntax and runtime issues such as ambiguous columns or missing commas.

| Error Type | Instruction |
|---|---|
| Ambiguous join | Identify and fix ambiguous column references |

Review and test all fixes in a safe environment.

### Finding Logical Bugs

Logical bugs occur when queries run but return incorrect data.

| Scenario | Prompt |
|---|---|
| Incorrect aggregation | Identify potential logical issues causing incorrect totals |

Compare suggestions with expected results before accepting changes.

---

## SQL Query Optimization and Performance Analysis

Cursor assists in spotting inefficiencies but does not replace performance testing.

### Optimizing Slow Queries

| Step | Action |
|---|---|
| 1 | Highlight the slow query |
| 2 | Ask Cursor for optimization ideas |
| 3 | Test changes with EXPLAIN and metrics |

Cursor may suggest filtering earlier, rewriting joins, or refactoring subqueries.

### Index Recommendations

Cursor can suggest index candidates based on joins and filters.

| Use Case | Instruction |
|---|---|
| Join performance | Suggest indexes for frequently joined columns |

Always validate index impact before production use.

### Readability vs Performance Tradeoffs

Balance clarity and efficiency consciously.

| Decision Area | Guidance |
|---|---|
| Readability | Use CTEs for clarity |
| Performance | Collapse joins when needed |

Cursor helps evaluate these tradeoffs with context.

### Example Task Comparison (PDF-Stable Table)

| Task | Manual Time | With Cursor |
|---|---|---|
| Query explanation | 10 min | 2 min |
| Debugging joins | 15 min | 4 min |
| Optimization ideas | 20 min | 6 min |

*Illustrative benchmarks only.*

---

## Using Cursor with Schemas, Migrations, and ORMs

Structured schemas and migration history reduce guesswork and errors.

### Schema-Aware Query Generation

| Step | Action |
|---|---|
| 1 | Provide schema files |
| 2 | Describe query goal |
| 3 | Review generated SQL |

This ensures joins follow correct relationships.

### Working with Migrations

Cursor can reference migrations to prevent conflicts.

| Task | Instruction |
|---|---|
| Schema updates | Check migrations for renamed or removed columns |

This avoids runtime errors in dependent queries.

### SQL Inside ORMs

Cursor can analyze SQL embedded in ORM code.

| Use Case | Prompt |
|---|---|
| ORM optimization | Translate this ORM query into optimized SQL |

You control final implementation.

---

## Best Practices for Cursor in SQL & DB Projects

### Prompting Do’s and Don’ts

**Do:** - Specify tables and fields - Describe expected output - Ask for explanations when needed

**Don’t:** - Assume access to live data - Accept suggestions blindly - Use vague prompts

| Poor Prompt | Improved Prompt |
|---|---|
| Optimize this query | Optimize this query joining users and orders from the last 30 days |

### Validation Workflow

| Step | Action |
|---|---|
| 1 | Generate or modify SQL |
| 2 | Compare with expected results |
| 3 | Test in staging |
| 4 | Review performance metrics |

## Security Considerations

Avoid sharing sensitive data in prompts.

| Risk Area | Mitigation |
|---|---|
| Credentials | Use anonymized data |
| Production data | Test with samples |
| Permissions | Limit write access |

Following these practices keeps Cursor safe and effective.