METHOD 2: Direct SQL Query (SQLite) - Complete Guide

This method lets you inspect your SQLite database directly using the `sqlite3` command-line tool.

STEP-BY-STEP IMPLEMENTATION

STEP 1: Verify SQLite is Installed

SQLite comes pre-installed with Python, so you should already have it.

Check if sqlite3 is available:

```
sqlite3 --version
```

Expected output:

```
3.40.1 2022-12-28 14:03:47...
```

If you get "command not found", install it:

- **Windows**: Download from sqlite.org/download.html
- **Mac**: `brew install sqlite3`
- **Linux**: `sudo apt install sqlite3`

STEP 2: Locate Your Database File

Your database should be at:

```
E:\AgentSync\data\sqlite\agentsync.db
```

Verify it exists:

```
# Windows PowerShell
Test-Path data\sqlite\agentsync.db

# Windows CMD or Git Bash
```

```
dir data\sqlite\agentsync.db

# Expected: True or file listing
```

STEP 3: Open SQLite Database

Navigate to your project root:

```
cd E:\AgentSync
```

Open the database:

```
sqlite3 data/sqlite/agentsync.db
```

You should see:

```
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite>
```

You're now in SQLite interactive mode!

STEP 4: Basic Database Exploration

Before querying contexts, let's verify the database structure:

4.1: List all tables

```
.tables
```

Expected output:

```
agent_contexts    agents            alembic_version   context_requests
```

4.2: See table structure

```
.schema agent_contexts
```

Expected output:

```
CREATE TABLE agent_contexts (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    agent_id INTEGER NOT NULL,
```

```
    context_key VARCHAR NOT NULL,
    context_summary TEXT NOT NULL,
    embedding_id VARCHAR,
    metadata_json TEXT,
    relevance_score FLOAT DEFAULT 1.0,
    created_at DATETIME,
    expires_at DATETIME,
    FOREIGN KEY(agent_id) REFERENCES agents (id)
);
```

4.3: Enable better formatting

```
.mode column
.headers on
.width 5 10 20 30 20 20
```

What these do:

- `.mode column` - Displays results in columns
- `.headers on` - Shows column names
- `.width` - Sets column widths for readability

STEP 5: Query Agent Contexts

Now let's run the actual queries!

Query 1: View Recent Contexts (Summary)

```
SELECT
    id,
    agent_id,
    context_key,
    substr(context_summary, 1, 50) as summary_preview,
    embedding_id,
    created_at
FROM agent_contexts
ORDER BY created_at DESC
LIMIT 10;
```

What this does:

- Shows first 50 characters of summary
- Orders by newest first
- Limits to 10 results

Expected output:

```
id     agent_id    context_key              summary_preview
-----  ----------  -----------------------  ----------------------------------------------
1      1           email_20251123_203353    - The task was executed by an unspecified age
```

Query 2: Count Contexts by Agent

```
SELECT
    agent_id,
    COUNT(*) as context_count
FROM agent_contexts
GROUP BY agent_id;
```

What this does:

- Groups contexts by agent
- Counts how many each agent has

Expected output:

```
agent_id    context_count
----------  -------------
1           1
```

Query 3: Get Most Recent Context (Full Details)

```
SELECT * FROM agent_contexts
ORDER BY created_at DESC
LIMIT 1;
```

What this does:

- Shows ALL columns
- Gets the newest context

Expected output:

```
id = 1
agent_id = 1
context_key = email_20251123_203353
context_summary = - The task was executed by an unspecified agent type capable of handlin
- Specifically, the action performed was "send_email" which involved sending corresponder
embedding_id = ctx_1_6cba3dbc6bad
metadata_json = {"agent_type": "email", "user_input": "send a friendly mail to vishnu3sgk
relevance_score = 1.0
created_at = 2025-11-23 20:33:53.304337
expires_at =
```

Query 4: View Full Summary (Readable)

```
SELECT
    id,
    agent_id,
    context_summary
FROM agent_contexts
WHERE id = 1;
```

To see full summary without truncation:

```
.mode list
SELECT context_summary FROM agent_contexts WHERE id = 1;
.mode column
```

Query 5: Parse Metadata JSON

SQLite has JSON functions:

```
SELECT
    id,
    json_extract(metadata_json, '$.agent_type') as agent_type,
    json_extract(metadata_json, '$.user_input') as user_input,
    json_extract(metadata_json, '$.timestamp') as timestamp
FROM agent_contexts
ORDER BY created_at DESC
LIMIT 5;
```

Expected output:

```
id     agent_type    user_input                                          timestamp
-----  ------------  --------------------------------------------------  ----------------
1      email         send a friendly mail to vishnu3sgk08@gmail.com      2025-11-23T20:33
```

Query 6: Search by Agent Type

```
SELECT
    id,
    context_key,
    substr(context_summary, 1, 60) as summary,
    created_at
FROM agent_contexts
WHERE json_extract(metadata_json, '$.agent_type') = 'email'
ORDER BY created_at DESC;
```

STEP 6: Useful Advanced Queries

Check Context Expiration

```
SELECT
    id,
    context_key,
    created_at,
    expires_at,
    CASE
        WHEN expires_at IS NULL THEN 'Never expires'
        WHEN expires_at > datetime('now') THEN 'Active'
        ELSE 'Expired'
    END as status
FROM agent_contexts
ORDER BY created_at DESC;
```

Find Contexts by Date Range

```
SELECT
    id,
    agent_id,
    context_key,
    created_at
FROM agent_contexts
WHERE created_at BETWEEN '2025-11-23' AND '2025-11-24'
ORDER BY created_at DESC;
```

Count Total Contexts

```
SELECT COUNT(*) as total_contexts FROM agent_contexts;
```

Get Latest Context per Agent

```
SELECT
    agent_id,
    MAX(created_at) as latest_context_time,
    COUNT(*) as total_contexts
FROM agent_contexts
GROUP BY agent_id;
```

STEP 7: Export Results to File

Export query results to CSV:

```
.mode csv
.output contexts_export.csv
SELECT * FROM agent_contexts ORDER BY created_at DESC;
.output stdout
.mode column
```

This creates `contexts_export.csv` in your current directory.

STEP 8: Exit SQLite

```
.quit
```

Or press `Ctrl+D` (Linux/Mac) or `Ctrl+Z` then Enter (Windows)

COMPLETE WORKFLOW EXAMPLE

Here's a complete session from start to finish:

```
# 1. Open database
sqlite3 data/sqlite/agentsync.db

# 2. Setup formatting
sqlite> .mode column
sqlite> .headers on
sqlite> .width 5 10 25 50 25 25

# 3. Check what's there
sqlite> SELECT COUNT(*) as total FROM agent_contexts;

# 4. View recent contexts
sqlite> SELECT
   ...>     id,
   ...>     agent_id,
   ...>     context_key,
   ...>     substr(context_summary, 1, 50) as preview,
   ...>     created_at
   ...> FROM agent_contexts
   ...> ORDER BY created_at DESC
   ...> LIMIT 5;

# 5. Check agent distribution
sqlite> SELECT agent_id, COUNT(*) as count
   ...> FROM agent_contexts
   ...> GROUP BY agent_id;
```

```
# 6. Exit
sqlite> .quit
```

HELPFUL SQLITE COMMANDS REFERENCE

Command	Purpose
`.tables`	List all tables
`.schema TABLE_NAME`	Show table structure
`.mode column`	Column display mode
`.headers on`	Show column names
`.width N N N`	Set column widths
`.output FILE`	Write to file
`.output stdout`	Write to screen
`.quit`	Exit SQLite
`.help`	Show all commands

TROUBLESHOOTING

Problem: "unable to open database file"

Solution:

```
# Check you're in correct directory
pwd

# Should be: E:\AgentSync

# If not, navigate there
cd E:\AgentSync

# Then try again
sqlite3 data/sqlite/agentsync.db
```

Problem: Table doesn't exist

Solution:

```
-- Check if database was initialized
.tables

-- If empty, run migrations
```

```
-- Exit SQLite first (.quit)
-- Then in terminal:
alembic upgrade head
```

Problem: Results are truncated

Solution:

```
-- Increase column width
.width 5 10 30 100 30 30

-- Or use list mode for full text
.mode list
SELECT context_summary FROM agent_contexts WHERE id = 1;
```

VERIFICATION CHECKLIST

After running queries, verify:

- ✓ Contexts exist in `agent_contexts` table
- ✓ `agent_id` matches your agents (1, 2, 3...)
- ✓ `embedding_id` has format `ctx_X_XXXXXXXXXX`
- ✓ `context_summary` contains meaningful text
- ✓ `metadata_json` is valid JSON with expected keys
- ✓ `created_at` has recent timestamps

NEXT: Method 3

Want to implement **Method 3: ChromaDB Collection Inspection** next? Let me know!