Aona

Synthetic SQLite join-debugging pack

Developers, DBAs and security engineering

Read the complete guide on aona.ai

Build the smallest invented dataset that preserves the relationships and constraints behind the SQL problem. Do not start by uploading a production dump and masking a few columns. The downloadable SQLite fixture reproduces a join overcount with fictional rows and verified expected output, so the query can be discussed without customer records.

Run a real relational fixture and compare an overcounting query with verified expected results.

Entirely synthetic dataset. SQL outputs and two constraints verified offline; no AI or endpoint-policy test performed.

Your review

Reviewer: _____ Date: _____

Organisation or team: _____

☐ **Keep the fixture invented**

Do not replace it with a dump, credentials or real customer records.

☐ **Compare amounts and aggregation counts**

The first customer exposes the join bug; the second is a useful control case.

☐ **Run only the supplied in-memory verifier**

It uses no network, rejects changed SQL and opens no existing database.

Checked items record your review, not a compliance approval or an installed-product test result.

Notes and next actions

Decision and scope: _____

Evidence to collect: _____

Owner and next review date: _____

Review matrix

Check	Expected fixture result	Verification
Correct totals	3,900 and 2,700 cents	Verified locally
Overcount example	4,200 and 2,700 cents	Verified locally
Orphan order	Foreign-key rejection	Verified locally
Zero item quantity	Check-constraint rejection	Verified locally

The relationship that reveals the error

- **Order 101:** Two items share one shipping charge

The naive join adds shipping twice

- **Correct level:** Aggregate items per order, then total per customer

Expected first total: 3,900 cents

- **Control case:** One item on order 201

Both queries return 2,700 cents

Synthetic SQLite debugging fixture

All rows, labels and amounts are invented. No customer records, credentials, external connection strings or API calls are included.

Question: why does adding order shipping after a join overcount the first customer?

The optional verifier uses only Python's standard sqlite3 module and an in-memory database. Run verify_fixture.py from this extracted pack to check the unchanged SQL, both expected query outputs and two constraints. It does not open an existing database or use a network. Changed SQL is rejected by hash.

Review fixture.sql, overcounting-query.sql and correct-query.sql with the expected CSV files. If you choose to share the example with an approved AI client, share only these synthetic files and the question. Treat a suggested query as something to validate locally, not a verified production fix.

Expected correct

Customer ID	Total cents	Order count
1	3900	2
2	2700	1

Expected overcounting

Customer ID	Total cents	Joined row count
1	4200	3
2	2700	1

Practice files

The complete download pack includes these original working files. Keep their original formats when running or inspecting the examples.

- Fixture (fixture.sql)
- Correct query (correct-query.sql)

- Overcounting query (overcounting-query.sql)
- Verify fixture (verify_fixture.py)

Sources and scope

Sources checked 2026-09-21. Review the current requirements and your actual agreement before using this workbook for a real decision. Aona does not provide database row permissions or guarantee inspection of every query, dump or MCP result. This fixture tests SQL locally, not an Aona policy.

SQLite: Foreign Key Support. Documents foreign-key relationships and enabling enforcement for a connection.

SQLite: SELECT. Documents joins, grouping and query evaluation used in the fixture.

OAIC: Use of commercially available AI products. Explains privacy considerations for personal information supplied to AI.