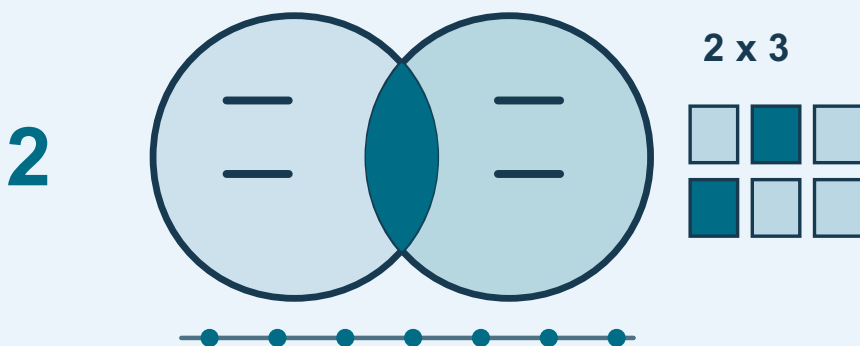




PYTHON AND DATA WORKFLOWS

Debug SQL Joins and Table Grain

Trace row multiplication and preserve ticket measures through one-to-many enrichment



Sanjaay Gowda T S

FREE SAMPLE / INTRODUCTION + CHAPTER 1 / EDITION 1.0.0

sanjaaygowdats.ai

About this free sample

Debug SQL Joins and Table Grain

Sanjaay Gowda T S

First edition 1.0.0, 2026-10-13

Copyright 2026 Sanjaay Gowda T S. This is a labelled free sample, not the complete ebook. You may share this sample unchanged. It contains the cover, sample permissions, contents, introduction, Chapter 1 and a closing product link. The remaining chapters, worked solutions and companion files are not included. The full ebook text, diagrams and artwork have separate single-reader terms; this sample permission does not authorize sharing the full purchased PDF or EPUB.

The companion source code, tests and fictional data are separately MIT licensed. You may adapt or share that project with its included licence notice. MIT does not apply to the ebook text or book artwork. The books and companion projects have no technical activation requirement or buyer-specific watermark.

This first edition was prepared with AI assistance. The case studies are fictional teaching examples. Any measured outputs come from the supplied examples, not production customer data. The teaching scope, assumptions and limitations are explained in the chapters. Independent human technical, editorial and assistive-technology reader review is pending. Automated tests and accessibility checks do not establish certification or suitability for every use.

Run examples in a separate working folder using the supplied synthetic data. Review inputs and calculations before adapting the techniques to real operations. Model software, service access, subscriptions, ongoing updates and consulting are not included.

Author website: sanjaaygowdats.ai

Ebook terms: sanjaaygowdats.ai/ebooks/license

Inside this free sample

Start here.....	4
1. Name the row represented.....	5
Continue with the full guide	6

This free sample contains the introduction and Chapter 1 only. The closing link opens the complete guide. The companion project and worked solutions are supplied with the complete ebook.

Start here

A query can be syntactically valid, return convincing rows and still multiply a measure several times. This often happens when an analyst adds useful descriptive relationships to a table and then sums a measure that belongs to the original entity. The cure begins with grain: the entity or combination of entities represented by one row at each stage.

The fictional case is Pine Support. Its ticket table records one effort-minute value per ticket. A ticket can have several tags and several comments. All tickets, teams, labels and effort values are invented. The effort numbers are arithmetic fixtures, not observations about worker productivity or organizational performance.

The companion creates an in-memory SQLite database, runs an intentionally unsafe raw join for diagnosis, and compares it with ticket-grain controls and independently aggregated enrichment counts. It also exposes duplicate tag relationships and orphan tag or comment records. It never modifies a live support system.

```
python main.py --input example-input.json
python -m unittest discover -s tests -v
```

Read Blue team's two totals. The correct effort is sixty minutes across two tickets. The raw joined effort is 240 minutes across eight joined combinations. T101 has two tags and three comments, making six joined rows. T102 has two duplicate tag links and one comment, making two. T103 has no partners and survives the left joins as one row.

This book teaches you to predict those results before running SQL, inspect intermediate relations and repair the model before applying final totals. Each exercise changes a cardinality or a count definition. The solutions explain the invariant behind the expected result. The aim is an inspectable query contract, not a universal query optimizer.

1. Name the row represented

Grain is a sentence

Write one row represents one ticket beside the ticket table. Write one row represents one ticket-tag relationship beside tags, and one row represents one comment beside comments. These sentences describe different grains even though all tables contain ticket identities. A shared key does not make their rows interchangeable.

The ticket effort measure belongs to ticket grain. A tag has no share of that effort unless an allocation rule explicitly assigns one. A comment also has no automatic effort share. Copying ticket effort onto joined rows is a representation convenience, not evidence that each copy should participate in a sum.

Relation	Intended grain	Candidate identity
ticket	One ticket	ticket_id
tag	One ticket-label link	ticket_id plus tag
comment	One comment	comment_id

The fixture deliberately violates the intended tag-link uniqueness for T102. This allows the diagnostic to distinguish duplicate relationships from legitimate multiple tags. Ticket and comment identities are constrained before staging, while tag duplicates remain visible as a quality finding.

Worked case: a misunderstood export

Suppose a new ticket export contains one row per technician assignment. Repeated ticket identities may now be legitimate. The old effort measure might describe ticket total repeated on every assignment, or assignment-specific effort. The same column name cannot settle this question. Obtain the source definition and choose either a ticket-grain relation or an allocation model before joining anything.

Exercise: write three grain statements

Describe the grain of the input tables and of the raw joined result. Include both partner identities in the raw-result statement. Explain why one joined row should not be called one ticket even though it contains a ticket identity.

Grain worksheet: distinguish measures and attributes

List each column and identify whether it is an identity, an attribute or a measure. Ticket identity establishes the entity. Team is an attribute in this snapshot. Effort minutes is a measure owned by that entity. Tag and comment identities describe relationships or partner entities. This classification helps detect a common error: treating every numeric column as something that may be summed at every resulting grain. An identifier containing digits is not a measure. A percentage may be a derived measure with a separate denominator. An amount repeated on relationship rows remains owned by its original entity unless an allocation contract says otherwise.

Now imagine the ticket table also contains customer satisfaction score. Summing scores is probably not the intended team measure, and averaging them after a raw join weights tickets by tag and comment multiplicity. The grain issue affects more than totals. Before choosing an aggregate, define what the measure means and which entities should receive equal weight. Keep the source-grain relation available as a control. A query with a clean GROUP BY clause can still answer the wrong question if the aggregated values were already repeated or weighted incorrectly upstream.

Continue with the full guide

This free sample ends after the introduction and Chapter 1. The complete guide contains nine teaching chapters, worked exercise solutions, primary references and a separate Python companion project. The full PDF, reflowable EPUB and project files are supplied together.

Use the complete guide to work through the remaining exercises and inspect the supplied examples. The companion code and fictional data are separately MIT licensed; the complete ebook has the single-reader terms described on its product page.

The examples are teaching fixtures, not production customer results. Independent human technical, editorial and assistive-technology reader review remains pending. Software access, subscriptions, consulting and guaranteed results are not included.

[View the complete guide and purchase options.](#)

You may share this labelled free sample unchanged. Keep the full purchased PDF and EPUB private. Thank you for reading.