```
mariadb greenleaf

MariaDB [greenleaf]> SELECT c.name, SUM(i.quantity * i.unit_price) AS sales
    -> FROM order_items i JOIN plants p USING (plant_id)
    -> JOIN categories c USING (category_id) GROUP BY c.name;

+----------------+--------+
| name           | sales  |
+----------------+--------+
| Herbs          |  25.50 |
| Houseplants    | 143.00 |
| Outdoor Shrubs |  29.00 |
| Succulents     |  39.00 |
+----------------+--------+
4 rows in set (0.001 sec)
```



SQL · DESIGN · ADMINISTRATION

MariaDB from Zero

Learn SQL, Database Design, and Administration Step by Step

No experience needed · Every query tested on a real MariaDB server with one continuous example database

QUERIES · JOINS · INDEXES · TRANSACTIONS · BACKUPS

THENEXBOOKS

MariaDB from Zero

Learn SQL, Database Design, and Administration Step by Step

First Edition

THENEXBOOKS

MariaDB from Zero
Learn SQL, Database Design, and Administration Step by Step

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. The code in this book may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Database software changes over time; consult the official documentation for the versions you use.

This book is an independent publication and is not affiliated with, sponsored by, or endorsed by MariaDB plc or the MariaDB Foundation. MariaDB is a trademark of its respective owners. MySQL is a trademark of Oracle and/or its affiliates. Other product names are trademarks of their respective owners; their use here is for identification only. Greenleaf Plant Nursery, the town of Willow Creek, and all people and data in the examples are fictional.

Preface

Almost every application you use keeps its data in a database: the shop that remembers your orders, the clinic that books your appointment, the app that saves your settings. MariaDB is one of the most widely deployed open-source databases for that job. It speaks SQL, the standard language for working with data, it runs on everything from a laptop to a large cluster, and it is free to use.

This book teaches MariaDB from zero. You do not need to know SQL or have any database experience. We start by installing MariaDB and asking it simple questions, then add one idea at a time: creating tables, inserting and changing data, filtering and sorting, functions, grouping, joins, subqueries, and window functions. From there we move on to designing good databases, keeping data correct with constraints, making queries fast with indexes, and programming the database with transactions, stored procedures, triggers, and events. The final parts cover the administration skills every MariaDB user needs — accounts and privileges, backups, monitoring, replication, and security — and show how applications connect to the database.

Every chapter uses the same realistic example: the database of **Greenleaf Plant Nursery**, a small family business with an online shop. You will create it in Chapter 3, query it, extend it, protect it, back it up, and finally build a set of business reports on top of it. Every query result in this book was produced by running the SQL against a real MariaDB server, so what you see on the page is what you should see on your screen.

Who This Book Is For

- **Complete beginners** who want to learn SQL and databases — no experience is assumed.
- **Students** in database courses who want a clear, practical companion.
- **Developers** who need to use MariaDB in their applications and want to understand it properly.
- **Junior administrators** who look after MariaDB servers and want a solid foundation.

What You Will Learn

- How to install MariaDB, connect with the `mariadb` client or a graphical tool, and run SQL.
- How to create tables, choose data types, and insert, update, delete, and query data.
- Functions, grouping, joins, subqueries, common table expressions, and window functions.
- How to design a database, enforce integrity with keys and constraints, and speed up queries with indexes.
- Transactions, stored procedures, triggers, scheduled events, JSON, and system-versioned tables.
- Users and roles, backup and restore, configuration, monitoring, replication, security, and connecting from applications.

How This Book Is Organized

Part I — Getting Started (Chapters 1–5) installs MariaDB, creates the Greenleaf database, and teaches the basic statements. **Part II — Working with Data** (Chapters 6–10) covers functions, aggregation, joins, subqueries, and window functions. **Part III — Designing Databases** (Chapters 11–15) covers design, constraints, indexes, and views, and ends with a project that builds out the Greenleaf database. **Part IV — Programming the Database** (Chapters 16–19) covers transactions, stored procedures and functions, triggers, events, JSON, and system-versioned tables. **Part V — Administration** (Chapters 20–24) covers users and privileges, backup, configuration and monitoring, replication, and security. **Part VI — Applications and Next Steps** (Chapters 25–28) connects applications, builds a reporting project, explores storage engines and MariaDB-specific features, and points the way forward. The appendices provide a cheat sheet, a reference of data types and functions, checklists, and a glossary.

Every chapter ends with **Try It Yourself** exercises and a **Key Takeaways** box.

Welcome to Greenleaf Plant Nursery

Greenleaf Plant Nursery is a small (and fictional) family business in the town of Willow Creek. It grows and sells houseplants, succulents, herbs, and outdoor shrubs, both from its greenhouse shop and online. Hana Sato owns the nursery, Marco Diaz runs the greenhouse and keeps track of stock, and Olivia Grant looks after the online shop and the reports. Their orders used to live in a spreadsheet; now they are moving to MariaDB, and you are going to build and run the database for them.

How to Use This Book

Type the statements yourself and run them against your own copy of the Greenleaf database. The setup script in Chapter 3 rebuilds the database from scratch, so you can experiment freely and reset whenever you like. Compare your results with the book; if something differs, read the error message carefully and check the statement character by character. Results that depend on the current date or time, and timings such as `(0.002 sec)`, will naturally be different on your machine.

Conventions Used in This Book

> **TIP**
>
> A recommended practice or a shortcut that makes your work easier.

> **NOTE**
>
> Background information, version differences, or a useful aside.

> **WARNING**
>
> A common mistake or something that can cause errors or lost data.

SQL keywords are written in UPPERCASE and names in lowercase. Statements appear in dark boxes, often with a file name above them; results appear in lighter boxes labeled **Result** or **Terminal**. The examples were tested on a recent MariaDB release; nearly all of them also work on the long-term support releases, and features that need a newer version are pointed out.

Contents

Preface

PART I — GETTING STARTED

1 What Is MariaDB, and Why Learn It?

2 Installing MariaDB and Connecting

3 Databases, Tables, and Data Types

4 Inserting, Updating, and Deleting Data

5 Querying with SELECT

PART II — WORKING WITH DATA

6 Built-in Functions: Strings, Numbers, and Dates

7 Aggregation with GROUP BY and HAVING

8 Joining Tables

9 Subqueries and Common Table Expressions

10 Window Functions

PART III — DESIGNING DATABASES

11 Database Design and Normalization

12 Keys, Constraints, and Referential Integrity

13 Indexes and Query Performance Basics

14 Views

15 Project: Building Out the Greenleaf Database

PART IV — PROGRAMMING THE DATABASE

16 Transactions and Locking

17 Stored Procedures and Functions

18 Triggers and Scheduled Events

19 JSON and System-Versioned Tables

PART V — ADMINISTRATION

20 Users, Roles, and Privileges

21 Backup and Restore

22 Configuration and Monitoring

23 Replication and High Availability

24 Security Hardening

PART VI — APPLICATIONS AND NEXT STEPS

25 Connecting from Applications

26 Project: Reporting and Analytics for Greenleaf

27 Storage Engines and MariaDB-Specific Features

28 Next Steps

APPENDICES

A SQL and MariaDB Cheat Sheet

B Data Types and Functions Reference

C Checklists

D Glossary

PART I

Getting Started

Install MariaDB, connect to it, create the Greenleaf database, and learn the statements that add, change, and read data.

1 · What Is MariaDB, and Why Learn It?
2 · Installing MariaDB and Connecting
3 · Databases, Tables, and Data Types
4 · Inserting, Updating, and Deleting Data
5 · Querying with SELECT

CHAPTER 1 · PART I — GETTING STARTED

What Is MariaDB, and Why Learn It?

Before we type a single command, we need a clear picture of what a database is, what SQL does, and where MariaDB fits in. In this chapter you will learn the core vocabulary of the whole book and meet Greenleaf Plant Nursery, the small business whose database we will build, query, and run together.

Data Is Everywhere

Every business, club, school, and hobby runs on information. A bakery keeps track of orders and ingredients. A sports club records members and membership fees. A plant nursery knows which plants it sells, how many are in the greenhouse, who its customers are, and what each customer bought. That information is called *data*: facts that are recorded so they can be used later.

When a business is tiny, data usually lives in a notebook or a spreadsheet. That works for a while. Then the business grows, and the problems start to appear one by one:

- **Duplicates and contradictions.** A customer's address is typed on three different sheets, and after she moves, only one of them gets corrected. Which one is right?
- **Typing mistakes nobody catches.** Someone enters a price of `4,50` instead of `4.50`, or writes "Sucullents" as a category. The spreadsheet accepts it without complaint.
- **Several people at once.** Two staff members open the same file, both edit it, and one person's changes silently overwrite the other's.
- **Hard questions.** "Which customers from Willow Creek spent more than $50 last month?" is easy to ask and slow to answer by scrolling and filtering by hand.
- **Growth.** A file with a few hundred rows is fine. A file with a few million rows is not.

Databases were invented to solve exactly these problems. They store data once, in one organized place, enforce rules about what is allowed, let many people and programs work at the same time safely, and answer questions quickly, even when there is a lot of data.

What Is a Database?

A **database** is an organized collection of data that is stored so it can be searched, updated, and protected. The software that manages databases is called a **database management system**, or DBMS. MariaDB is a DBMS. People often say "the database" when they mean either the data or the software; the context usually makes it clear, and we will be precise whenever it matters.

There are several families of databases. Some store documents, some store key–value pairs, some store graphs of connected things. MariaDB belongs to the oldest and most widely taught family: the **relational database**. In a relational database, all data lives in **tables**, and tables are connected to each other through shared values. Once you understand tables, you understand most of how a relational database thinks.

Tables, Rows, and Columns

A table looks a lot like a spreadsheet tab. Each **column** describes one kind of fact, such as a name or a price. Each **row** holds the facts about one thing, such as one plant. Here are the first few rows of the `plants` table we will create in Chapter 3, simplified to four columns:

plant_id	common_name	price	stock_qty
1	Swiss Cheese Plant	34.00	18
2	Snake Plant	22.00	30
3	Fiddle Leaf Fig	48.00	6
7	Zebra Haworthia	9.00	0

There are three important differences between a database table and a spreadsheet:

1. **Every column has a data type.** The `price` column only accepts numbers with two decimal places. If you try to store the word "cheap" there, the database refuses. Chapter 3 is all about choosing the right types.
2. **Columns have names that matter, not letters.** There is no "column C". You always refer to `price` or `stock_qty` by name, which makes your instructions readable.
3. **Rows have no fixed position.** A spreadsheet has row 1, row 2, row 3. A table is more like a bag of rows. If you want them in a particular order, you ask for that order every time. We will see a surprising example of this in Chapter 3.

Keys and Relationships

How do we point at exactly one row? We give each row a value that is guaranteed to be unique, called the **primary key**. In the table above it is `plant_id`. No two plants can have the same `plant_id`, and it can never be empty. Names are not good keys: two plants could share a common name, and names change. A plain number that never changes is simple and reliable.

Now imagine an order. Instead of copying the customer's name, email, and city into every order, the order simply stores the customer's number. That number is a **foreign key**: a column whose values must match the primary key of another table. The database checks it for us. If an order refers to customer 99 and there is no customer 99, the database rejects the order. This one idea — store each

fact once, and connect tables through keys — removes the duplicate-and-contradiction problem we described above. Chapter 11 explains the design ideas in depth, and Chapter 12 covers every kind of key and constraint.

> **NOTE**
> The word "relational" comes from the mathematical idea of a *relation*, which is essentially what we call a table. It does not come from "relationships between tables", although relational databases happen to be very good at those too.

What Is SQL?

To talk to a relational database, we use **SQL**, short for Structured Query Language. Some people say each letter, "S-Q-L", and some say "sequel"; both are fine. SQL is a standard language shared by MariaDB, MySQL, PostgreSQL, SQL Server, Oracle, SQLite, and many other systems. Each system adds its own extras, but the core is the same. What you learn in this book will help you with any relational database you meet later.

SQL is **declarative**. You describe the result you want, and the database works out how to get it. You do not write loops that walk through the rows one by one. Here is the smallest useful SQL statement there is:

first_query.sql

```
SELECT 2 + 3 AS answer;
```

Result

```
+--------+
| answer |
+--------+
|      5 |
+--------+
1 row in set (0.000 sec)
```

Even this tiny example shows the shape of every query in the book. `SELECT` means "give me". `2 + 3` is an expression to calculate. `AS answer` gives the result column a name. The semicolon marks the end of the statement. The result comes back as a small table with a header row and one data row, followed by a line that says how many rows were returned and how long it took. Timings such as `(0.000 sec)` differ on every machine and every run, so never worry if your numbers are not the same as ours.

Here is a query that does real work. It asks for the herbs Greenleaf sells, with their prices and how many are in stock:

herbs.sql

```
SELECT common_name, price, stock_qty
FROM plants
WHERE category_id = 3;
```

Result

```
+-------------+-------+-----------+
| common_name | price | stock_qty |
+-------------+-------+-----------+
| Sweet Basil |  4.50 |        60 |
| Rosemary    |  6.00 |        45 |
| Peppermint  |  5.00 |        50 |
+-------------+-------+-----------+
3 rows in set (0.000 sec)
```

Read it aloud and it almost sounds like English: "Select the common name, price, and stock quantity from plants where the category id is 3." That readability is one of SQL's greatest strengths. You will write queries like this confidently by the end of Chapter 5.

The Four Jobs of SQL

SQL statements fall into a few groups. You do not need to memorize the abbreviations, but they appear in documentation, so it helps to recognize them:

Group	What it does	Example statements	Chapters
Data definition (DDL)	Creates and changes structures	CREATE, ALTER, DROP	3, 11–14
Data manipulation (DML)	Adds, changes, removes rows	INSERT, UPDATE, DELETE	4
Queries (DQL)	Reads and analyzes data	SELECT	5–10
Data control (DCL)	Controls who may do what	GRANT, REVOKE	2, 20

There is also a group for transactions, which bundle several changes into one all-or-nothing unit. We introduce them in Chapter 16.

Clients and Servers

MariaDB is a **client/server** system. That phrase describes how the pieces fit together, and it explains many things that confuse beginners, such as why you need a password, a host name, and a port number just to "open the database".

- The **server** is a program that runs in the background, usually all the time. On most systems its program file is called `mariadbd` (older versions call it `mysqld`). The server owns the data files on disk. Nobody else touches them directly.

- A **client** is any program that connects to the server, sends it SQL, and receives results. The command-line program `mariadb` is a client. A graphical tool such as HeidiSQL or DBeaver is a client. The Greenleaf web shop, written in Python or PHP, is also a client.
- Clients and server talk over a **connection**. A connection can go over the network, normally to **port 3306**, or through a local channel on the same computer (a socket file on Linux and macOS, or a named pipe on Windows).

Because the server sits in the middle, it can enforce the rules for everyone. It checks every user's password and privileges. It makes sure two clients that change the same row at the same moment do not corrupt each other's work. It writes changes safely to disk so a power cut does not destroy the data. The client's only job is to send requests and show results.

In practice, the server and the client often run on the same computer while you learn. That is exactly the setup we create in Chapter 2. In a company, the server usually runs on a dedicated machine, and dozens or thousands of clients connect to it from elsewhere. The SQL is identical in both cases; only the connection details change.

> **TIP**
> When something goes wrong, ask yourself which side the problem is on. "Can't connect" errors are about the connection between client and server. Errors that start with a number such as `ERROR 1146` come from the server, which means you connected successfully and it did not like your SQL. Knowing the difference saves a lot of time.

Where MariaDB Came From

To understand MariaDB, it helps to know a little about MySQL. MySQL is an open-source relational database that became enormously popular for websites and applications from the late 1990s onward. It was simple to install, fast for common workloads, and free to use.

In 2009, some of MySQL's original developers, led by Michael "Monty" Widenius, started a community-developed fork of MySQL called **MariaDB**. A fork is a new project that begins from a copy of an existing project's source code and then develops independently. The goal was to keep a fully open, community-developed version of the database that would stay compatible with MySQL while adding improvements. The name comes from Monty's younger daughter, Maria. (MySQL itself had been named after his older daughter, My.)

Today MariaDB is developed in the open by the community, with the non-profit MariaDB Foundation looking after the project and the company MariaDB plc offering commercial products and services around it. The server is released under the GNU General Public License, version 2, which means you can download, use, study, and modify it freely.

Because of its history, MariaDB still answers to many MySQL names. The client program used to be called `mysql`; it is now `mariadb`, and `mysql` remains as an alias on most installations. The same goes for `mariadb-dump` (formerly `mysqldump`), `mariadb-admin` (formerly `mysqladmin`), and others. In this book we always use the `mariadb` names, but if you follow an older tutorial that types `mysql`, it will usually still work.

> **NOTE**
>
> MariaDB is a trademark of MariaDB plc and the MariaDB Foundation; MySQL is a trademark of Oracle. We use the names only to identify the products.

MariaDB and MySQL: Cousins, Not Twins

For the everyday SQL in the first half of this book — creating tables, inserting rows, querying, joining, grouping — MariaDB and MySQL behave almost identically. Many applications written for MySQL run on MariaDB without changes, and the same programming-language drivers can often connect to either. That compatibility is useful: much of the advice you find about MySQL also applies to MariaDB.

The two projects have, however, grown apart over the years. Each has added features the other does not have, and some features with the same purpose work differently. This book is about MariaDB specifically, so we take full advantage of what it offers. Here are some of the MariaDB features you will meet:

Feature	What it gives you	Chapter
RETURNING	Get back the rows you just inserted or deleted	4, 27
CHECK constraints	Rules such as "price can't be negative"	3, 12
Sequences	Number generators independent of any table	12, 27
System-versioned tables	Ask what a row looked like last month	19
JSON_TABLE	Turn JSON documents into rows	19
Roles	Group privileges and hand them out together	20
Storage engines	InnoDB, Aria, ColumnStore and more	27
Galera Cluster	Several servers that all accept writes	23

Whenever MySQL behaves differently in a way that might trip you up, we mention it briefly. Otherwise we simply teach MariaDB.

Why Learn MariaDB?

You might be learning MariaDB because a job, course, or project requires it. You might be choosing a first database and wondering whether MariaDB is a good place to start. Here are solid reasons why it is:

- **It is free and open.** You can install the full server on your own laptop at no cost, with no trial period and no feature limits. Everything in this book runs on a free installation.
- **It is everywhere you look.** Many Linux distributions ship MariaDB as their standard MySQL-compatible database, so it is often already available on servers and in web hosting packages.
- **The skills transfer.** SQL, table design, indexing, transactions, backups, and user management work on the same principles across relational databases. Learn them well on MariaDB and you have a head start on any other system.
- **It scales with you.** The same server that runs on your laptop can run a busy production system. Later in the book we look at the tools that make that possible, from monitoring to replication.
- **It rewards curiosity.** MariaDB includes advanced features such as temporal tables and sequences that let you try professional techniques without buying anything.

Most importantly, databases are one of the most durable skills in software. Programming languages and frameworks come and go; the relational model and SQL have been at the heart of business software for decades, and that is not changing soon.

Meet Greenleaf Plant Nursery

Learning SQL with abstract examples such as "table A" and "column x" is hard. Learning it on a realistic business is much easier, because you always know what a query is for. Throughout this book, we work with one running example: **Greenleaf Plant Nursery**, a fictional family-run plant nursery with an online shop in the fictional town of **Willow Creek**.

Three people run Greenleaf, and each of them will need something different from the database:

- **Hana Sato** is the owner. She wants the big picture: sales per month, best-selling plants, and which customers come back again and again.
- **Marco Diaz** manages the greenhouse and the stock. He cares about what is running low, what arrived from suppliers, and what has sold out.
- **Olivia Grant** runs the online shop and the reports. She needs orders, shipping charges, loyalty points, and clean customer data.

Greenleaf has a few simple business rules, and the database will help enforce them:

- Shipping costs **$5.95 per order**, but it is **free when the item subtotal is $50.00 or more**.
- Customers earn **1 loyalty point per whole dollar** of item subtotal on orders that have shipped.
- An order is always in one of four states: `pending`, `paid`, `shipped`, or `cancelled`.

- Stock can never go below zero. You cannot sell a plant that is not in the greenhouse.

The Five Tables

The Greenleaf database is called `greenleaf`, and it starts with five tables. In Chapter 3 you will create all of them with a single script and read every line of it.

Table	One row describes	Starting rows
categories	A group of plants, such as Herbs	4
plants	A plant Greenleaf sells, with price and stock	12
customers	A person who shops at Greenleaf	6
orders	One order placed by one customer	5
order_items	One line of an order: a plant and quantity	9

Notice how the tables connect. Every plant belongs to a category, so `plants` stores a `category_id`. Every order belongs to a customer, so `orders` stores a `customer_id`. An order can contain several plants, and a plant can appear in many orders, so the `order_items` table sits between them: each of its rows says "in this order, on this line, this plant, this many, at this price". These connections are the foreign keys we described earlier, and the database enforces every one of them.

The four categories are Houseplants, Succulents, Herbs, and Outdoor Shrubs. The twelve plants range from a $4.50 pot of Sweet Basil to a $48.00 Fiddle Leaf Fig. Six customers have signed up so far, mostly from Willow Creek, with a few from the neighboring towns of Riverton and Oakdale. Five orders were placed in February and March 2026, numbered 1001 to 1005, and one of them was cancelled.

A First Look at What We Will Build

You cannot run these queries yet — you will install MariaDB in Chapter 2 and create the database in Chapter 3 — but it is motivating to see where we are going. Do not try to understand every word; just notice how directly each query answers a business question.

First, Marco's everyday question: "What do we have, and how much of it?"

plant_list.sql

```
SELECT plant_id, common_name, category_id, price, stock_qty
FROM plants;
```

Result

```
+----------+--------------------+-------------+-------+-----------+
| plant_id | common_name        | category_id | price | stock_qty |
+----------+--------------------+-------------+-------+-----------+
|        1 | Swiss Cheese Plant |           1 | 34.00 |        18 |
```

```
|        2 | Snake Plant        |           1 | 22.00 |        30 |
|        3 | Fiddle Leaf Fig    |           1 | 48.00 |         6 |
|        4 | Peace Lily         |           1 | 19.50 |        25 |
|        5 | Jade Plant         |           2 | 14.00 |        40 |
|        6 | Aloe Vera          |           2 | 12.50 |        35 |
|        7 | Zebra Haworthia    |           2 |  9.00 |         0 |
|        8 | Sweet Basil        |           3 |  4.50 |        60 |
|        9 | Rosemary           |           3 |  6.00 |        45 |
|       10 | Peppermint         |           3 |  5.00 |        50 |
|       11 | English Lavender   |           4 | 16.00 |        20 |
|       12 | Hydrangea          |           4 | 29.00 |         8 |
+----------+--------------------+-------------+-------+-----------+
12 rows in set (0.000 sec)
```

Next, a sharper version of the same question: "What is running low?" Here we keep only the plants with fewer than ten in stock and sort them so the most urgent comes first.

low_stock.sql

```
SELECT common_name, stock_qty
FROM plants
WHERE stock_qty < 10
ORDER BY stock_qty;
```

Result

```
+-----------------+-----------+
| common_name     | stock_qty |
+-----------------+-----------+
| Zebra Haworthia |         0 |
| Fiddle Leaf Fig |         6 |
| Hydrangea       |         8 |
+-----------------+-----------+
3 rows in set (0.000 sec)
```

The Zebra Haworthia is sold out, and two more plants need reordering soon. In a spreadsheet you would filter and sort by hand; here the question and the answer are both written down, and you can run the query again tomorrow to get tomorrow's answer.

Finally, a question that needs four tables at once. Olivia gets an email from Emma Brooks asking what she has ordered so far:

emma_orders.sql

```
SELECT c.first_name, o.order_id, p.common_name, i.quantity
FROM customers c
JOIN orders o ON o.customer_id = c.customer_id
JOIN order_items i ON i.order_id = o.order_id
JOIN plants p ON p.plant_id = i.plant_id
WHERE c.customer_id = 1;
```

Result

```
+------------+----------+--------------------+----------+
| first_name | order_id | common_name        | quantity |
+------------+----------+--------------------+----------+
| Emma       |     1001 | Swiss Cheese Plant |        1 |
| Emma       |     1001 | Peace Lily         |        2 |
| Emma       |     1005 | Snake Plant        |        1 |
+------------+----------+--------------------+----------+
3 rows in set (0.000 sec)
```

This query follows the foreign keys from the customer to her orders, from the orders to their lines, and from each line to the plant. It looks intimidating now. By the end of Chapter 8 you will write joins like this without a second thought, and you will also notice that order 1005 was cancelled and should probably be filtered out — a nice example of how real questions keep getting more precise.

The Journey Through the Book

Here is the road ahead, part by part, all on the Greenleaf database:

- **Part I — Getting Started (Chapters 1–5).** Install MariaDB, create the database, add and change data, and write your first real queries.
- **Part II — Working with Data (Chapters 6–10).** Functions, grouping and totals, joins, subqueries, and window functions. This is where Hana's monthly reports come to life.
- **Part III — Designing Databases (Chapters 11–15).** Good table design, keys and constraints, indexes for speed, and views. In Chapter 15 we extend Greenleaf with suppliers, deliveries, and reviews.
- **Part IV — Programming the Database (Chapters 16–19).** Transactions, stored procedures, triggers, scheduled events, JSON, and tables that remember their own history.
- **Part V — Administration (Chapters 20–24).** Users and privileges, backups, configuration and monitoring, replication, and security.
- **Part VI — Applications and Next Steps (Chapters 25–28).** Connecting from application code, a complete reporting project, storage engines, and where to go after this book.

How to Use This Book

This book is written for complete beginners, but it does not stay at the beginner level. Each chapter introduces an idea with a tiny example first and then applies it to Greenleaf. To get the most out of it:

- **Type the examples yourself.** Copying and pasting works, but typing builds memory. Small typos and the errors they cause are some of the best teachers you will ever have.
- **Read the results.** Do not skip over the output boxes. Check that the numbers make sense. Ask yourself why there are three rows and not four.

- **Experiment.** Change a value, remove a condition, sort in a different order. You cannot hurt anything: in Chapter 3 you will get a setup script that rebuilds the whole database from scratch in a second.
- **Do the exercises.** Every chapter ends with a short "Try It Yourself" section and a Key Takeaways box. The exercises are small and are designed to take a few minutes each.

Conventions

We use a few consistent conventions so you always know what you are looking at:

- SQL code appears in dark boxes, often with a file name above it. SQL keywords such as `SELECT` and `WHERE` are written in UPPERCASE, and names of tables and columns are written in lowercase with underscores, such as `stock_qty`. MariaDB does not require this style for keywords, but it makes code much easier to read.
- Results from MariaDB appear in boxes labeled "Result", "Output", or "Terminal". Every result in this book was produced by actually running the code on a MariaDB server. When something cannot be run in a book-writing environment — such as an installer on your particular operating system — we say so and describe it as output "similar to" what you will see.
- Timings such as `(0.004 sec)` in results depend on your computer and will be different for you.
- Boxes marked **Tip** give practical advice, **Note** boxes add background, and **Warning** boxes point out mistakes that can cost you data or time.

> **WARNING**
> Practice on your own computer, never on a database that holds real business data. Several chapters deliberately change and delete rows, drop tables, and rebuild the database. That is safe on a learning setup and potentially disastrous on a live system.

Our examples were tested on a recent MariaDB release. Most of them also work unchanged on the long-term support releases 10.11 and 11.x that many servers still run. When a feature needs a newer version, we tell you which one.

A Few More Words You Will Hear

Before moving on, here are some terms that appear again and again. You do not need to master them now; just let them become familiar.

Term	Meaning
Query	A request for data, usually a SELECT
Statement	Any complete SQL command ending in a semicolon
Schema	In MariaDB, simply another word for a database