Introduction

This is a book about SQL Patterns. Patterns describe problems that occur over and over in our professional settings. A pattern is a like a template that you can apply to different problems. Once you learn each one, you can apply them to solve problems faster and make your code more readable.

We can illustrate this with an example. In fiction writing, authors rarely write from scratch. They use character patterns like: “antihero”, “sidekick”, “mad scientist”, “girl next door”. They also use plot patterns like romantic comedy, drama, red herring, foreshadowing, cliffhangers. This helps them write better books, movies and TV shows.

Each pattern consists of four elements:

1. The **pattern name** is a handle that describes the problem and potential solutions
2. The **problem** describes when you should apply the pattern and in what context
3. The **solution** describes the elements of the design for the solution to the problem
4. The **tradeoffs** are the consequences of applying that specific solution

Who am I

I’ve been writing SQL for ~15 years. I’ve seen and written hundreds of thousands of lines of code. Over time I noticed a set of patterns and best practices I always come back to when writing queries. These patterns made my code more efficient, easier to understand and a breeze to maintain.

Why did I write this book

I have a background in computer science. As part of the curriculum we learn how to make our code more efficient, more readable and easy to debug. As I started to write SQL, I applied many of these lessons to my own code.

When reviewing other people's code I would often spot the same mistakes. There were chunks of code that would repeat everywhere. The queries were long, complex and slow. I would often have rewrite them so I could understand what they were doing.

I looked around for a book or course that taught these patterns but couldn't find one, so I decided to write it myself.

Who this book is for

This book is for anyone who is familiar with SQL and wants to take their skills to the next level. We won't cover any of the basic syntax here so make sure you have that down pat. I expect you to already know how to join tables and do basic filtering and aggregation.

If you find that your SQL code is often inefficient and slow and you want to make it faster, this book is for you.

If you find that your SQL code is long, messy and hard to understand and you want to make it cleaner, this book is for you

If you find that your SQL code breaks easily when data changes and you want to make it more resilient, this book is for you.

What you'll learn in this book

I'm a huge fan of project-based learning. You can learn anything if you can come up with an interesting project to use it thing in. I used a project when I taught myself data science.

That's why for this book I wanted to come up with an interesting and useful data project to organize it around. I explain each pattern as I walk you through the project.

This will ensure that you learn the material better and remember it the next time you need to apply it.

We'll be working with the Stackoverflow dataset that's available in BigQuery for free. You can access it here.

BigQuery offers 1TB/month free of processing so you can complete this entire course for free. I've made sure that the queries are small and limited in scope so you won't have to worry about running out.

Using this dataset we're going to build a table which calculates reputation metrics. You can use this same type of table to calculate a customer engagement score or a customer 360 table.

As we go through the project, we'll cover each pattern when it arises. That will help you understand why we're using the pattern at that exact moment. Each chapter will cover a select group of patterns while building on the previous chapters.

How this book is organized

In **Chapter 1** we introduce the project we'll be working on throughout the book. We'll make sure you have access to the dataset and can run the queries. We also get a basic understanding of the dataset by looking at the ER diagram

In **Chapter 2** we cover *Core Concepts and patterns*. These patterns act as our basic building blocks that will serve us throughout the book. I explain each one using the StackOverflow dataset since we'll be using every one in our final query.

The remaining patterns are grouped into four categories and each has its own chapter.

In **Chapter 3** we cover *Query Decomposition* patterns. We start off by learning how to decompose large queries into smaller pieces to make it easy to solve just about any complex problem. They are important to learn first because all the other patterns flow from them.

In **Chapter 4** we cover *Query Maintainability* patterns. These patterns teach you how to decompose a query and organize your code in ways that make it efficient. This will ensure your code is easier to read, understand and maintain in the future.

In **Chapter 5** we cover *Query Performance* patterns. They teach you ways to make your code more faster without sacrificing clarity. It's a delicate balance because performant code can sometimes look really messy.

In **Chapter 6** we cover *Query Robustness* patterns. They teach you ways to make your code resistant to messy data, such as duplicate rows, missing values, unexpected NULLs, etc.

The project is interwoven throughout the book. I make sure that each chapter covers some section of the final query.

In **Chapter 7** we wrap up our project and you get to see the entire query. By now you should be able to understand it and know exactly how it was designed. I recap the entire project so that you get another chance to review all the patterns. The goal here is to allow you to see all the patterns together and give you ideas on how to apply them in your day-to-day work.

In **Chapter 8** we cover a few special case patterns. We dive deeper into window functions and string manipulation, such as regular expressions and JSON parsing. Even though this is not related to our project, I wanted to make sure I enrich your vocabulary of patterns beyond what's in the project.

With that out of the way, let's dive into the project.