SQL Table Relations - Recap

Introduction

Another section down, great job! In this section you learned about SQL queries that use `JOIN` clauses and subqueries to return data from multiple tables at once. You also got some additional SQL practice resembling what you might need to do in a technical interview.

Key Takeaways

Relational Data

One of the powerful aspects of SQL is that **a given database can contain multiple, related tables**. Records across tables can be connected based on **foreign keys** — values in one table that reference a unique identifier in another table.

Relational data means that you don't need to store multiple copies of the same information, and that updates can be made more easily and efficiently.

Sometimes multiple tables must be joined together to answer a single question, based on how the various tables are related to each other.

`JOIN` Clause

The `JOIN` clause is typically used to **connect records from multiple tables**. SQL will not automatically understand how the tables are related, so you need to specify this using either the `ON` clause (e.g. `ON employees.office_id = offices.id`), or, if the column names are identical in both tables, using the `USING` clause (e.g. `USING(office_id)`).

Often it is useful to **alias** table names (e.g. `JOIN orderdetails AS od`) when you are using `JOIN` clauses in order to shorten the syntax. Similar to aliases for columns, the `AS` is optional when using SQLite (and may be required or not allowed in other contexts).

Subqueries

In some cases **subqueries can be used for the same tasks as `JOIN` clauses**. To do that, you don't use `JOIN`, and instead add a nested `SELECT` query inside of your main `SELECT` query, which selects from another table.

Subqueries are also useful **if you want to filter or order by an aggregate, but not select an aggregate**. For example, if you had a database of guest appearances on a late night show, and you wanted to select the dates of appearances of guests who had been on the show 10 or more times (maybe you're theorizing that repeat guests are more frequent on Mondays), you couldn't just use `GROUP BY` and `HAVING` because that would return *guests* (all appearance dates grouped together) and you want the actual appearance dates.

As you can see from the example above, subqueries can be useful even if you are only working with a single table, if you want to use an aggregate in your analysis but not have your query return an aggregate.

Technical Interview Prep

You can expect to run into SQL questions in job interviews. Often times when asked to write a query, you'll be given a hypothetical situation and asked to write a query, without having an ERD or tables to look at. When working through these questions, it's important to clarify your assumptions, and to ask questions to make sure that your assumptions are actually correct!

Once you have the information you need, be sure to write clean, well-structured SQL code. This includes things like:

- Capitalizing SQL keywords (technically `select` and `SELECT` both work the same, but it's easier to read if you use capitalization consistently)
- Using whitespace consistently (unlike Python, whitespace is not important in SQL — try your best to use new lines and indentation to spread out your code and help you spot errors)
- Using semantic variable names (renaming things like `COUNT(*)` to something readable, and using sensible names for foreign keys such as `office_id` for a foreign key that links to the `offices` table)

This is also a situation where thinking out loud and engaging your interviewer is a great strategy. They're likely trying to evaluate how you would work on a team, just as much as they're trying to evaluate your technical knowledge or your ability to get data from a relational database.

Open-ended Questions in Interviews

You may also encounter open-ended SQL questions where there is no single correct solution. When faced with these sorts of interview questions, have an opinion, and be ready to back it up with your rationale! For these sorts of questions, interviewers are often trying to evaluate *how* you approach problems and evaluate your thought process, so helping them understand your thought process is very important!

The Internet Is Your Friend

All of the real-world interview questions you worked through in this section came from popular interview sites such as Glassdoor. If you still feel a bit shaky about your ability to answer SQL/relational database questions in an interview, that's okay — practice makes perfect! Don't be afraid to peruse sites like Glassdoor and others to look for the types of SQL questions you can expect to see during the interview process. Just make sure that when you look at the questions, you try to answer them yourself first before looking at the answers that others have posted!

Summary

Congratulations! You should have a good amount of SQL skills by now. The more practice you get, the more comfortable you will be when you're actually interviewing!