SQL for Finance

Overview

What this Course Is

This is a SQL course with applications in finance. It does not require any previous programming, mathematics, or finance knowledge. The goal of this course is to instill a practical understanding of SQL in the context of how it is used in the financial industry. Students can expect to come out with the ability to query in SQL for their first day on the job as well as impress interviewers with their SQL knowledge.

What this Course Is Not

This is not an end-to-end academically-oriented SQL course. It does not contain any Computer Science theory and will not give a deep understanding of the mechanics of SQL. Students will not learn about permissioning, variables, or other SQL tools not traditionally used in finance.

Prerequisites

None

Syllabus

LEVEL 1: BASIC QUERIES

1.1: INTRODUCTION TO SQL

1.2: SELECT

1.3: FILTERS

1.4: MULTIPLE FILTERS

1.5: HOMEWORK GUIDELINES

LEVEL 2: WRITING DATA

2.1: DELETE

2.2: DELETE PART II

2.3: UPDATING DATA

2.4: ADDING DATA

LEVEL 3: JOINS

3.1: TOP/ALIASES

3.2: INTRODUCTION TO JOINS

3.3: INNER JOIN

3.4: LEFT/RIGHT JOIN

3.5: FULL OUTER JOIN

LEVEL 4: UNIONS

4.1: INTRODUCTION TO UNIONS

4.2: UNION

4.3 UNION ALL

4.4: FULL JOIN WORKAROUND

LEVEL 5: CLOSING

5.1: SUBQUERIES

5.2: FUNCTIONS/CLOSING

How to Progress

- Watch all videos for a single chapter. You should proceed in the provided chapter order.
- Try out the examples demonstrated in the videos
    1. Play around with the examples; make changes, etc., to get an intuitive feel for the queries.
- Ask questions to your instructor or in the comments section of a video until everything is clear.
- Do the HW for the level. Feel free to ask questions on the video or to the instructor.
- Repeat for the next level.

How to Ask Questions

- Questions may be on the lectures, material, or exercises.
- Search the related video to see if your question has already been asked/answered.
- If it has already been answered, feel free to ask any follow-up questions to clarify if still unclear.
- If it has not already been answered, post a new question on the video, or to your instructor.
- Posts related to exercises should not contain any code; they should be conceptual in nature and/or descriptions (in English) of issues encountered. Instructor responses will also be conceptual in nature – they will guide you to finding the solution on your own, without providing the solution. Posts that do not conform to this standard will be removed.
- Students are encouraged to respond to other students' queries, publicly and within these guidelines.

Recommended Study Planner

Total time for the course is 4 weeks. After 4 weeks, you will lose instructor support, the ability to earn the certificate, and access to the course material.

You may work at your own pace within the allotted 4 weeks, but the below is the recommended pace per level:

Week 1: Level 1 & 2

Week 2: Level 3

Week 3: Level 4 & 5

Week 4: Catchup/Review

Submission Instructions

- Submissions should be for an entire chapter at a time (no individual sections or exercises).
- Every exercise should be submitted as its own script(s).
- You must wait for feedback on your submissions prior to advancing to the next chapter, as it will be locked until the instructor unlocks the next chapter

Grading Policy

- Once you submit your exercises for a given level, your instructor will provide detailed feedback on your code. This will usually occur within 24-48 hours of submission; often quicker, rarely slower.
- Each exercise is assigned a grade out of 100. The score for a chapter is the straight average of all the exercises in that chapter.
- Each chapter (and the final project) is assigned an overall course weight. Your final score is the weighted average of all the chapters and final project. The chapter weights are equal.
- You will need a minimum of a 70 weighted average score to earn the certificate.

Grading Criteria

- Each exercise is graded out of 100. An exercise that is code correct and meets the specification of the instructions gets an 80. The extra 20 points are devoted to code commenting, code format, code conciseness/clarity, efficiency of code, and taking the optimal coding approach (within what was already taught).
- You are expected to build upon your knowledge as you progress through the course. Therefore, once a concept is taught, you are expected to incorporate it into your future code when applicable. Taking an earlier (non-optimal) approach will cost you points.