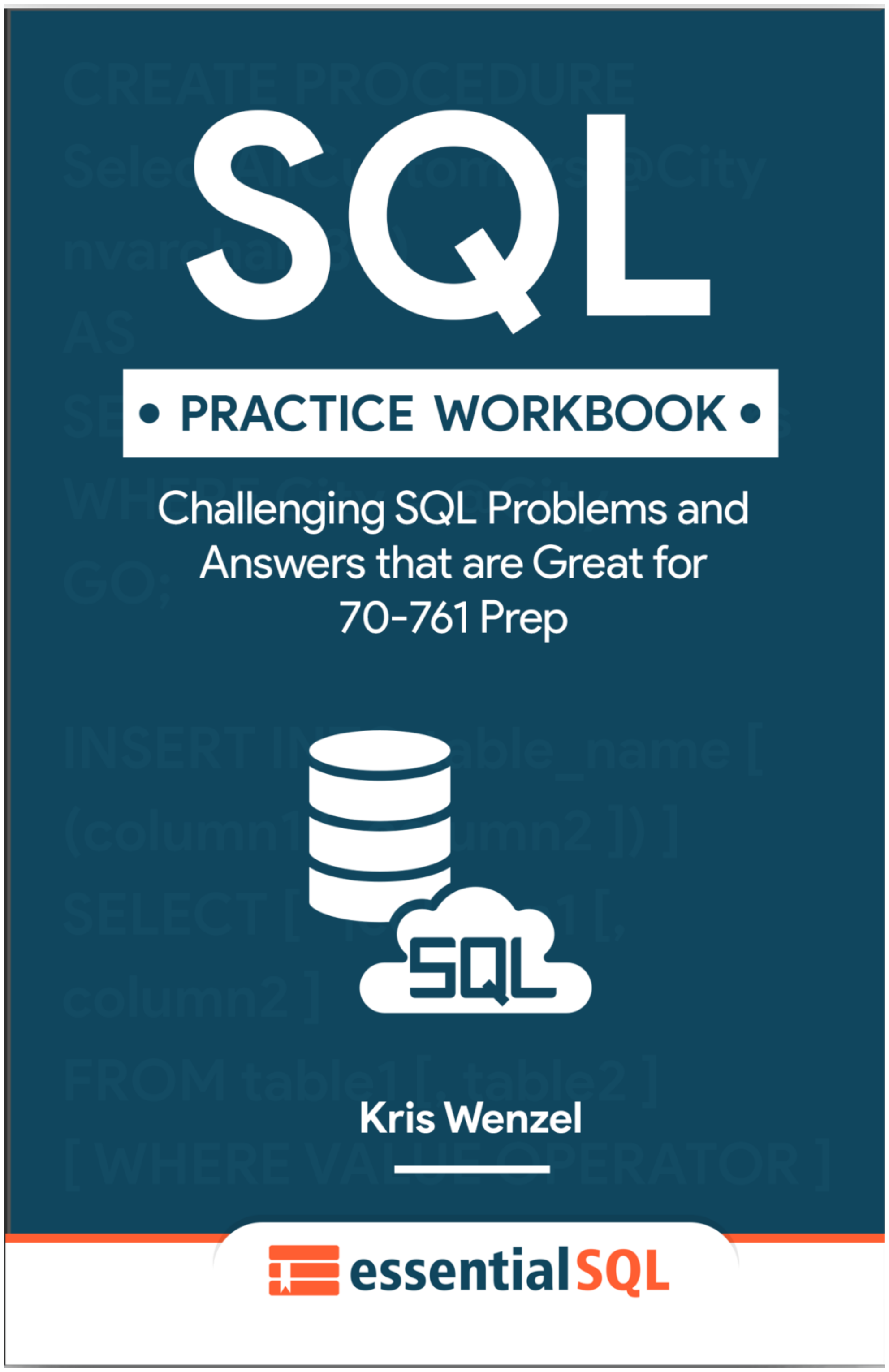


SQL

• PRACTICE WORKBOOK •

Challenging SQL Problems and Answers that are Great for 70-761 Prep

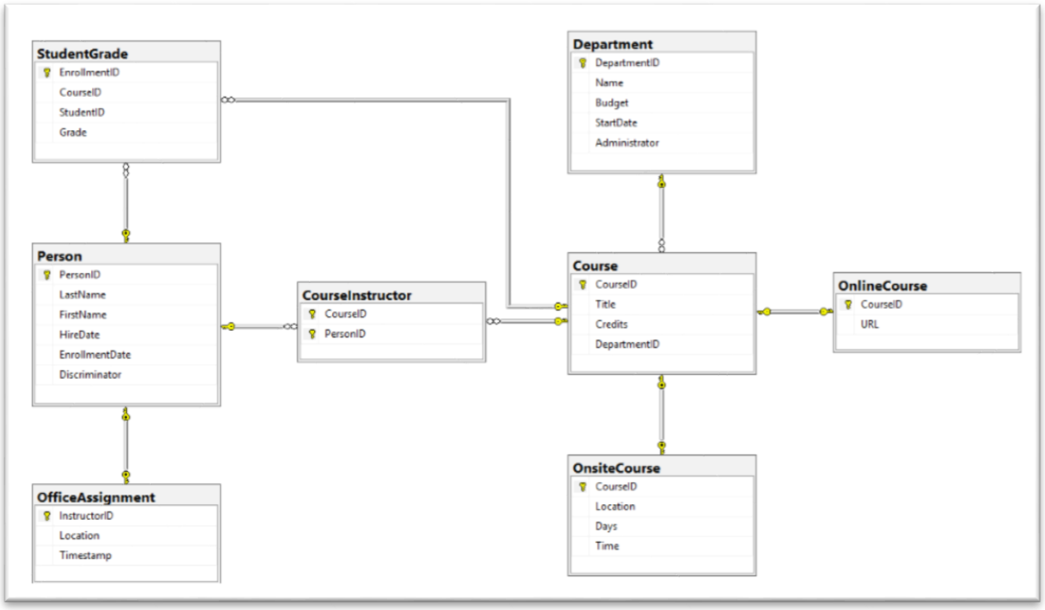
Kris Wenzel

essentialSQL

SQL Practice Workbook

These practice problems I've put together are based on a simple database named School. You can get the script to make the database from my site.

To help answer the problems, use this diagram I created:



So, give each problem a whirl. Then compare your answer to mine. Also, don't forget to check out my videos, where I solve the problems in "real-time."

Tip! Get the School Sample Database Here!

Problem 1

The schools would like to know which student has the highest GPA. To qualify students, need to have taken two or more courses.

List the Student Name, number of courses taken, and overall GPA.

Keep it simple. You may find the solution is easier that you first though!

Here is how I solved the problem. I was able to solve it mainly using GROUP BY and HAVING.

```sql
SELECT TOP 1 Person.FirstName + ' ' + Person.LastName AS StudentName,
             COUNT(StudentID) AS CoursesTaken,
             AVG(Grade) AS OverallGPA
FROM StudentGrade
     INNER JOIN Person ON StudentGrade.StudentID = Person.PersonID
GROUP BY Person.FirstName + ' ' + Person.LastName
HAVING COUNT(StudentID) >= 2
ORDER BY OverallGPA DESC;
```

At first, I thought I would need a subquery to count the number of courses a student had, but then I realized that it didn't need to be so complicated.

Watch the video, see me solve the problem in action.

Problem 2

In order to plan for some new hires, the school would like to know which course is most popular.

Create a cross tab counting the number of courses taken. The row should be student enrollment year, the columns, course title.

Before you get hung up creating the pivot table, think about what source data you'll need to create it. Focus on that query first.

I used a Pivot table to solve this problem. I had just written an article on them, so it helped me remember the syntax. Read Create a Pivot table in six steps to cross tabulate your data.

```
SELECT EnrollmentYear,
       Calculus,
       Chemistry,
       Composition,
       Literature,
       Macroeconomics,
       Microeconomics,
       Physics,
       Poetry,
       Quantitative,
       Trigonometry
FROM
(
    SELECT YEAR(P.EnrollmentDate) EnrollmentYear,
           SG.EnrollmentID,
           C.Title
    FROM Person P
         INNER JOIN StudentGrade SG ON P.PersonID = SG.StudentID
         INNER JOIN Course C ON SG.CourseID = C.CourseID
) AS PivotData PIVOT(COUNT(EnrollmentID) FOR Title IN(Calculus,
                                                      Chemistry,
                                                      Composition,
                                                      Literature,
                                                      Macroeconomics,
                                                      Microeconomics,
                                                      Physics,
                                                      Poetry,
                                                      Quantitative,
                                                      Trigonometry)) AS PivotResult
ORDER BY EnrollmentYear;
```

Watch the video, see me solve the problem in action.

Problem 3

The facilities manager would like to create a directory of Instructor Offices. To help them out, you’re going to provide them a result containing the Building Name, Office Number, and Instructor Name.

The results should be ordered by Building Name and then Office Number.

Knowing how to use built-in functions can help you work through data formatting issues.

The trick to this problem is understanding how to use string functions to manipulate the Location. To break it into two pieces: the building and office number.

We used SUBSTRING, CHARINDEX, and LEN to help us. These are explained in my article Introduction to SQL Server's Common String Functions

```sql
SELECT   SUBSTRING(OA.Location,
                   CHARINDEX(' ', OA.Location) + 1,
                   LEN(OA.Location) - CHARINDEX(' ', OA.Location)) AS Building,
         LEFT(OA.Location, CHARINDEX(' ', OA.Location) - 1) AS OfficeNumber,
         P.FirstName + ' ' + P.LastName AS InstructorName
FROM     OfficeAssignment OA
            INNER JOIN Person P ON P.PersonID = OA.InstructorID
WHERE    P.Discriminator = 'Instructor'
ORDER BY Building,
         OfficeNumber;
```

Watch the video, see me solve the problem in action.

Problem 4

There is a fear the Person table, which isn’t normalized, has a data issue. The column discriminator should either be “Instructor” or “Student” depending on whether HireDate or EnrollmentDate, respectively are filled.

Write a query to check the discriminator colum is correctly based on HireDate and EnrollmentDate. Your query should output the PersonID, LastName, DatePresent (Hire or Enrollment), Flag. The Flag will be “Discriminator OK” if the data check out, and “Discriminator Bad” if it doesn’t.

This is one of those problems that relies on logic. To solve it we use the built-in logic functions. At first, I was thinking I could use IIF, but though that's a great function, and compact, it would have been hard to read the logic we needed to put together

I used CASE instead. I find CASE to be a true T-SQL workhorse!

```
SELECT PersonID,
       LastName,
       COALESCE(HireDate, EnrollmentDate) AS DatePresent,
       HireDate,
       EnrollmentDate,
       Discriminator,
       CASE
           WHEN HireDate IS NOT NULL
                AND EnrollmentDate IS NOT NULL
           THEN 'Discriminator Bad'
           WHEN HireDate IS NULL
                AND EnrollmentDate IS NULL
           THEN 'Discriminator Bad'
           WHEN HireDate IS NOT NULL
                AND Discriminator = 'Instructor'
           THEN 'Discriminator OK'
           WHEN EnrollmentDate IS NOT NULL
                AND Discriminator = 'Student'
           THEN 'Discriminator OK'
           ELSE 'Discriminator Bad'
       END AS Flag
FROM Person
ORDER BY PersonID;
```

Watch the video, see me solve the problem in action.

Problem 5

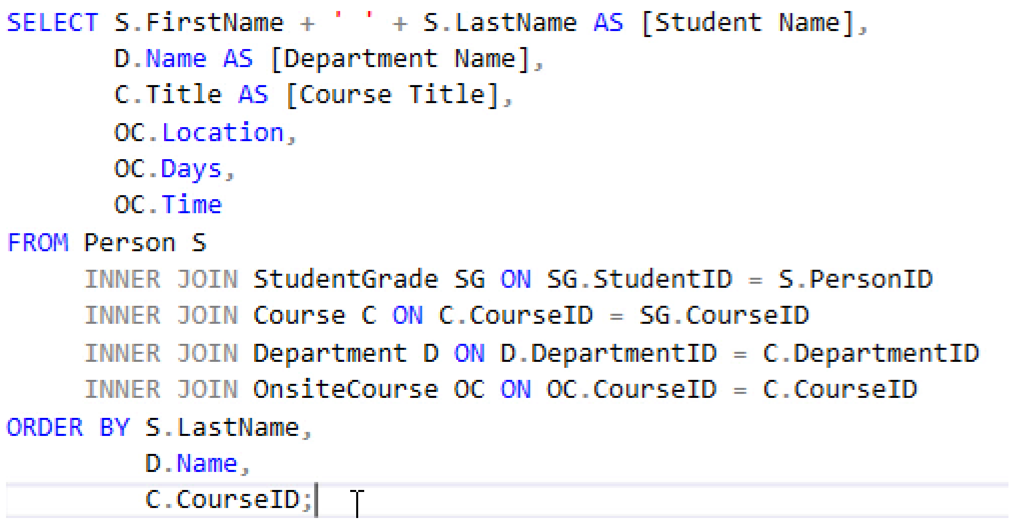
Create course schedules for all student’s onsite courses

Include the Student Name, Department Name, Course Title, Location, Days and Time.

Order the schedule by student last name, department name, and course id.

When dealing with queries that span many tables, it a good idea to get a lay of the land. What table are needed? How are they related?

This is straight forward problem, but a fun one! I always enjoy stringing together tables within SQL.



```
SELECT S.FirstName + ' ' + S.LastName AS [Student Name],
       D.Name AS [Department Name],
       C.Title AS [Course Title],
       OC.Location,
       OC.Days,
       OC.Time
FROM Person S
     INNER JOIN StudentGrade SG ON SG.StudentID = S.PersonID
     INNER JOIN Course C ON C.CourseID = SG.CourseID
     INNER JOIN Department D ON D.DepartmentID = C.DepartmentID
     INNER JOIN OnsiteCourse OC ON OC.CourseID = C.CourseID
ORDER BY S.LastName,
         D.Name,
         C.CourseID;
```

Watch the video, see me solve the problem in action.

Resources

There quite a bit that goes into a answering these questions. To help you get the most out of this guide, I put together some resources you may want to use to help with some of the "foundational" challenges, such as knowing how to navigate your database to find table relationships, and so on.

Each of these resources is a good starting point to learn more!

- School Sample Database
- GROUP By and HAVING
- Learn the Three Crucial Steps to Write Better SQL
- Introduction to Database Joins
- Create a Pivot table in six steps to cross tabulate your data
- Introduction to SQL Server's Common String Functions
- Introduction to SQL Server's Built-In Logical Functions

Also, I would encourage you to join my essential SQL learning group and join my active community of aspiring business analysts and accidental DBAs!

Have a great day – Kris.