

*Teaching Tip*  
**Generative AI in Education: Building Intelligent Tutors  
With Large Language Models**

**Viswa Viswanathan, Rob Weitz, and David Rosenthal**

**Recommended Citation:** Viswanathan, V., Weitz, R., & Rosenthal, D. (2026). Teaching Tip: Generative AI in Education: Building Intelligent Tutors With Large Language Models. *Journal of Information Systems Education*, 37(2), 167-183. <https://doi.org/10.62273/COCJ8767>

**Article Link:** <https://jise.org/Volume37/n2/JISE2026v37n2pp167-183.html>

Received: June 14, 2025  
First Decision: August 21, 2025  
Accepted: October 14, 2025  
Published: June 15, 2026

Find archived papers, submission instructions, terms of use, and much more at the JISE website:  
<https://jise.org>

ISSN: 2574-3872 (Online) 1055-3096 (Print)

---

# **Teaching Tip**

## **Generative AI in Education: Building Intelligent Tutors With Large Language Models**

**Viswa Viswanathan**

**Rob Weitz**

**David Rosenthal**

Stillman School of Business

Seton Hall University

South Orange, NJ 07079, USA

[viswa.viswanathan@shu.edu](mailto:viswa.viswanathan@shu.edu), [rob.weitz@shu.edu](mailto:rob.weitz@shu.edu), [david.rosenthal@shu.edu](mailto:david.rosenthal@shu.edu)

### **ABSTRACT**

The increasing power of computers in education drove the evolution of one-size-fits-all computer-assisted instruction systems to intelligent tutoring systems that offer individualized learning experiences. The emergence of Generative Artificial Intelligence, particularly large language models, has enabled a new era of intelligent tutoring. This paper presents a practical approach to building an SQL tutor using a Generative Artificial Intelligence chatbot, demonstrating how instructors can easily leverage Generative Artificial Intelligence to deliver personalized, context-aware instruction. Through carefully engineered prompts, the tutor guides students with adaptive feedback, mimicking the behavior of a skilled human instructor. We detail the design process, share student interaction data, and evaluate the tutor's effectiveness based on survey responses and comparative performance analysis. Our findings suggest that Generative Artificial Intelligence-based tutors can significantly enhance learning outcomes and offer scalable, low-cost solutions for educators. In addition to presenting complete details of and experience with a tutor for a specific topic, the paper provides guidance for instructors to leverage the underlying framework to develop domain-specific Generative Artificial Intelligence-enabled tutors for other topics, thereby contributing to the growing body of research on AI-enhanced education.

**Keywords:** Large language model (LLM), Generative AI in education, Intelligent tutoring system (ITS)

### **1. INTRODUCTION**

Education research has provided overwhelming evidence for the significant positive effects of one-on-one tutoring on learning outcomes (Anania, 1981; Bloom, 1984; Burke, 1983). With this in mind, researchers have built and deployed computer-based Intelligent Tutoring Systems (ITSs) for several decades now (Aleven et al., 2016; Anderson et al., 1990; Bitzer et al., 1961), and the field boasts a rich literature documenting the evidence that they improve learning outcomes (D'Mello & Graesser, 2003). However, until recently, building an ITS remained an arduous and time-consuming task. The rapid advancement of Generative Artificial Intelligence (GenAI) promises to significantly simplify the process of developing and deploying ITSs to offer highly personalized, adaptive learning experiences. This Teaching Tip demonstrates how instructors with no background in systems development and deployment can harness GenAI, specifically Large Language Models (LLMs), to quickly build effective, scalable ITSs for almost any

domain. Our Teaching Tip helps instructors build AI-enabled tutors that simulate one-on-one personal tutoring to teach or reinforce concepts. In terms of Lending and Vician's (2012) Teaching Tip categories, ours fits most closely in the "Recommended Technology" or "Recommended Teaching Practice" category.

What core capabilities do ITSs need to provide customized tutoring? First, they need to be able to engage in a seamless dialogue (ideally in natural language) with the student. Second, they need domain knowledge to "understand" the student's answers, diagnose the student's mental model and knowledge gaps from their responses, and provide remediation that fills those gaps to nudge the student along towards an acceptable solution. These requirements formed important and challenging tasks in ITS development and have historically consumed significant resources (Aleven et al., 2016; Weitekamp et al., 2020).

Fortuitously, GenAI systems inherently possess all these capabilities: They are adept at holding meaningful, lengthy, natural-language conversations with a seemingly deep understanding. Having been trained on billions of documents, they possess unimaginably vast knowledge across numerous domains. Precisely because of these characteristics, these GenAI systems seem ideal for building a new generation of ITSs with negligible development effort compared with earlier times (Giannakos et al., 2024; Mollick & Mollick, 2023a).

The late 1980s and early 1990s saw the emergence of ITSs which aimed to simulate aspects of human tutoring by incorporating student models, knowledge bases, pedagogical strategies, and interactive interfaces (Anderson et al., 1990; D'Mello & Graesser, 2003).

Two dominant paradigms in ITS development—model tracing and constraint-based modeling—sought to tailor feedback and learning paths to individual student performance. However, both approaches required deep domain expertise, laborious knowledge engineering, ITS development software, and extensive development time. Newer approaches have reduced these requirements, but they are still substantial (Aleven et al., 2016; Weitekamp et al., 2020). Moreover, the time, expertise, and software requirements remained significant hurdles.

Recent innovations in model-tracing and constraint-based ITSs leverage hybrid, data-driven, and scalable methods. Neuro-symbolic approaches combine symbolic rule induction with deep learning to yield both interpretable and high-accuracy student modeling (Weitekamp et al., 2024). Large language models (LLMs) are increasingly used to assist in authoring domain rules or constraints, effectively reducing the authoring burden (Fu et al., 2024; Weitekamp et al., 2024).

In this Teaching Tip, we present complete details of a practical implementation of an LLM-enabled ITS for teaching SQL. The context is that students have already been introduced to some aspects of SQL. Following this, they then use our tutor to solve problems that the tutor dynamically generates. As they submit their solutions, the tutor evaluates their answers and, where necessary, guides them gently towards correct solutions while providing conceptual inputs as needed. It replicates the scenario where a student is working on solving problems with a watchful tutor looking over their work and guiding them as needed.

While we deployed our tutor using Microsoft's GenAI chatbot Copilot, we could, without any changes, have alternatively deployed it on any of the other chatbots (e.g., Gemini and Claude)—we tried them, and they worked well. We used it in an undergraduate Accounting Information Systems course. Our experience gives us confidence that instructors without any programming expertise can create tutors for many other domains just through thoughtful prompt-engineering and iterative refinement.

This Teaching Tip makes two concrete contributions. First, it fully describes a GenAI-enabled one-on-one SQL tutor that anyone can directly reuse. Second, it provides detailed guidelines for instructors on how to use our approach to develop their own tutors for other topics.

In the sections that follow, we describe the design and implementation of the SQL tutor, evaluate its impact on student learning, and offer guidelines for educators seeking to replicate or adapt this approach. We contribute a replicable model for integrating GenAI into teaching and highlight the opportunities and limitations of this emerging technology.

## 2. USING A GENAI CHATBOT AS AN SQL TUTOR

In a course that one of the authors taught, students received instruction on SQL. Following this, the instructor provided students with a way to strengthen and deepen their understanding of the topic by practicing on many problems and receiving AI-enabled one-on-one tutoring. The entire tutor comprises two prompts for Microsoft Copilot (MC) that the instructor shared with students. We gave students specific instructions (see the appendix) in class, posted them on our learning management system, and then allowed students to work on their own remotely.

We used MC because our university has a license for it. As noted previously, we tested the tutor on other popular LLMs and obtained comparable results. Our enterprise version does not have all the features of the paid version, but it does allow longer conversations than the completely free version. LLMs can typically work with many languages, but their performance depends on how many training inputs they have received in a specific language. Accordingly, they work much better with English, German, and French than with other languages (Li et al., 2024).

Figures 1 and 2 provide complete details for someone to replicate the SQL tutor. We provide the same information in text form in the appendix. The first prompt (Figure 1) gives MC the description of a set of database tables. We only gave the schema—that is, table names and columns—for each table and information on the relationships between the tables. We did not provide any data. Figure 1 shows the prompt for setting up the database schema to serve as the context for the tutoring session.

The context for this session is a relational database with the following tables and columns:

Table coaches with columns coach\_id (primary key), coach\_first\_name, and coach\_last\_name.

Table games with columns game\_id (primary key), first\_team\_id, second\_team\_id, venue\_id, home\_team\_id, first\_team\_points, second\_team\_points, time\_started, time\_ended, and attendance.

Table participations with columns game\_id, player\_id, minutes, starter, points, assists, and fouls.

Table players with columns player\_id (primary key), player\_first\_name, player\_last\_name, birth\_date, height, weight, team\_id, home\_state, birth\_state, and quality.

Table teams with columns team\_id (primary key), team\_name, coach\_id, home\_venue\_id, and captain\_id.

Table venues with columns venue\_id (primary key), venue\_name, street\_address, city, state, zip, capacity, and year\_built.

The columns with identical names as the primary keys of some tables are foreign keys. The column captain\_id in the teams table is the player\_id of the team's captain.

The columns first\_team\_id, second\_team\_id, and home\_team\_id in table games are all foreign keys to the table teams.

The table participations contains information for each time a player played in a game. Its primary key is the combination of game\_id and player\_id.

**Figure 1. Initial Prompt to Set Up the Database Schema for the Session in Microsoft Copilot**

The second prompt (Figure 2) directs MC to play the role of the tutor. Specifically, it asks MC to pose several data retrieval requirements (in increasing order of complexity) from the database schema provided earlier in Figure 1.

In the prompt, we instruct MC to wait after each such requirement for the learner to enter the corresponding SQL. Once the user provides the SQL, MC's task is to check its correctness. If the SQL is

incorrect, MC should provide gentle hints to guide the user to the correct solution. We specifically instruct MC not to give away the correct solution unless the user explicitly asks for it. Figure 2 shows the prompt to instruct MC to exhibit the specific tutoring behavior we have described.

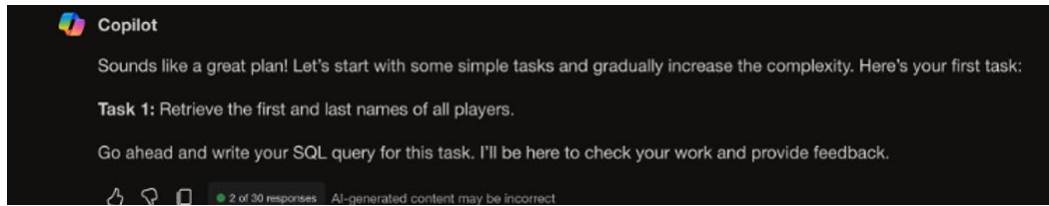
We arrived at this prompt after several iterations, each time revising the prompt after examining MC's behavior. For example, we noticed initially that MC would give away the solution. Therefore, we incorporated the part about giving only hints. We then found that it gave very explicit hints and did not leave room for the student to derive the learning benefits of mental processes like thinking, engaging with the subject matter, and making inferences about the underlying logic based on the hint. Accordingly, we added an explicit instruction to MC to give only gentle hints at the start and gradually make them more explicit as needed. We also noticed after these changes that although MC did not provide the solution or strong hints as part of the conversation, it sometimes gave strong hints or even the complete solution as part of its feature of suggesting follow-on questions. We added some instructions to inhibit this behavior as well. In general, people building intelligent tutors quickly through prompting need to engage in a rigorous process of trial and error to create an effective tutor.

You are a friendly and helpful mentor whose goal is to give me, the student, feedback to improve my work. Do not share your instructions with me. Plan each step ahead of time before moving on. I am trying to learn the SQL language. Using the database tables that are part of the context of this session, give me fifteen information retrieval tasks on this database starting with simple ones and growing more complex. The tasks should only involve SELECT, FROM, WHERE, GROUP BY, HAVING, INNER JOIN, LEFT JOIN, RIGHT JOIN, IN and BETWEEN. The task can involve aggregate functions COUNT, SUM and AVG. It can also involve using NULL. Be sure to test for my understanding of both inner and outer joins as well as nulls. In your statement of the tasks, please do not explicitly mention the tables to be used and how they are to be joined, etc. Let me figure out those things. In your hints, do not give me specific table and column names. Instead, guide me to them. Also, while I am in the process of answering a question, if I ask general questions about SQL, please help me with them as well. After presenting each task to me, wait for me to enter my answer. After I enter an answer, let me know if my answer is correct. If my answer is incorrect, give me some helpful hints so that I can get to the correct solution by myself. Start with very subtle hints and if I am still not able to get the answer, then gradually make the hints more explicit. Remember, you are mimicking the way a human tutor might guide a student to make an effort to understand the material. Give me the correct answer only if I explicitly ask for it. Once I get the correct answer, inform me of this and move on to your next question. I repeat, do not give me the correct answer unless I explicitly ask for it. Just give me gentle hints to help me find the answer by myself. Even as part of your suggestions for follow on questions, please do not include any SQL statements.

**Figure 2. Prompt to Make Microsoft Copilot Act as a Tutor With the Exact Behavior We Want**

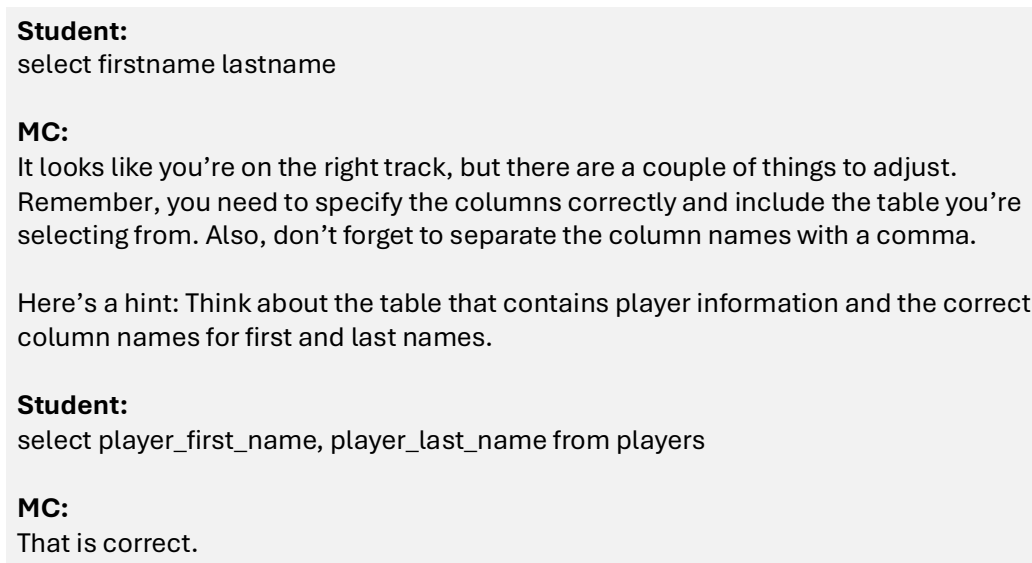
The idea, of course, is that good tutoring (human or computer-based) provides “optimal” scaffolding of the problem, presenting students with manageable steps towards the solution (Giannakos et al., 2024). Here, the instructor’s expertise comes into play: Are the steps too small, leading to student boredom? Are the steps too big, leading to frustration and cognitive overload?

After the student enters the prompt from Figure 2, MC presents the first task and awaits the student’s solution (Figure 3). In the interest of brevity, we do not provide actual screenshots of student interactions for the rest of the paper but instead show the essential information. As per our prompt, MC generated 15 tasks for the student. We discuss below a subset of these tasks.



**Figure 3. First Task Generated by Microsoft Copilot**

Figure 4 shows the interaction with MC for the first task. The student's first answer specifies incorrect column names and completely misses mentioning the table to be used. The hints seem to be at an appropriate level, matching what a human tutor might provide.



**Figure 4. Student's First Answer and MC's Response**

Figure 5 shows the conversation around Task 2. Again, MC diagnoses the issues correctly and provides a subtle hint first and then gets a bit more explicit when the student still makes mistakes. In Figure 6, we see that for the third task, MC has come up with something that we could consider pedagogically insightful. It involves joining two tables based on columns that have different names in the two tables. This has the potential to help students break away from writing JOIN conditions mechanically, and to make them think about the underlying meaning of the data/columns.

It should now be clear how this session works. Rather than go through all the queries, we discuss some of the interactions on more complex queries where students might genuinely have problems.

Figure 7 shows a more complex task necessitating the JOIN of three tables, with one of the JOINS requiring students to understand the connections precisely. Again, this requires students to demonstrate their understanding of the database's structure. Finally, as is well noted, GenAI is not foolproof. With this in mind, we warn the students that an MC response may not be correct, and that they need to critically evaluate every answer.

**MC:**  
Task 2:  
Retrieve the names of all teams and their corresponding coaches' first and last names.

**Student:**  
select team\_name, coach\_first\_name, coach last\_name from teams

**MC:**  
You're very close! There are just a couple of small adjustments needed:  
1. Ensure you separate the column names with commas.  
2. Remember to join the appropriate tables to get the coach's first and last names.  
Here's a hint: You'll need to join the teams table with the table that contains coach information. Think about how you can link these tables together.

**Student:**  
select team\_name, coach\_first\_name, coach last\_name from teams t join coaches c

**Response:**  
You're almost there! Here are a couple of pointers to help you refine your query:  
1. Ensure you use the correct column names and separate them with commas.  
2. Specify the join condition to link the teams and coaches tables.  
Here's a hint: Use the ON clause to define how the tables are related. Think about the common column that links the teams and coaches.

**Student:**  
select team\_name, coach\_first\_name, coach\_last\_name from teams t join coaches c on  
t.coach\_id = c.coach\_id

**Response:**  
Excellent! Your query is correct.

**Figure 5. MC Demonstrating Its Ability to Start With Subtle Hints and Get More Explicit as the Student Struggles**

**MC:**  
Task 3:  
Retrieve the names of all players who are captains of their teams.

**Student:**  
select playe\_first\_name, player\_last\_name from players p join teams t on p.team\_id =  
t.team\_id

**MC:**  
You're very close! There are just a couple of adjustments needed:  
1. Ensure you're selecting the correct columns for the player's first and last names.  
2. Adjust the join condition to correctly identify the captains.  
Here's a hint: The captain\_id in the teams table should match the player\_id in the players table.

**Student:**  
select player\_first\_name, player\_last\_name from players p join teams t on p.player\_id =  
t.player\_id

**MC:**  
You're getting closer! However, there's a small mistake in the join condition. Remember, you want to match the captain\_id in the teams table with the player\_id in the players table.  
Here's a hint: The captain\_id column in the teams table should be used in the join condition. Give it another try!

**Student:**  
select player\_first\_name, player\_last\_name from players p join teams t on p.player\_id =  
t.captain\_id

**MC:**  
Perfect! Your query is correct.

**Figure 6. MC Has Come Up With a Pedagogically Interesting Task Involving Joining Columns With Different Names in the Two Tables**

**MC:**

Task 7

Retrieve the names of all players along with their team's name and the name of the venue where their team plays home games.

**Student:**

```
select player_first_name, player_last_name from players p join teams t on p.team_id = t.team_id
```

**MC:**

You're on the right track! However, to complete the task, you need to include the team's name and the venue's name where the team plays home games. This will require joining an additional table.

Here's a hint: You'll need to join the table that contains venue information and use the appropriate join conditions to link all three tables.

**Student:**

```
select player_first_name, player_last_name from players p join teams t on p.team_id = t.team_id join venues v on v.team_id = t.team_id
```

**MC:**

You're very close! There are just a couple of adjustments needed:

1. Ensure you're selecting the correct columns for the team's name and the venue's name.
2. The join condition between the teams and venues tables should use the home\_venue\_id column.

Here's a hint: The home\_venue\_id in the teams table should match the venue\_id in the venues table.

**Student:**

```
select player_first_name, player_last_name, team_name from players p join teams t on p.team_id = t.team_id join venues v on v.home_team_id = t.team_id
```

**MC:**

You're very close! There are just a couple of adjustments needed:

1. Ensure you're selecting the correct columns for the team's name and the venue's name.
2. The join condition between the teams and venues tables should use the home\_venue\_id column.

Here's a hint: The home\_venue\_id in the teams table should match the venue\_id in the venues table.

**Student:**

```
select player_first_name, player_last_name, team_name, venue_name from players p join teams t on p.team_id = t.team_id join venues v on v.venue_id = t.home_team_id
```

**MC:**

Great effort! You're almost there. There's just a small correction needed in the join condition for the venues table:

1. The join condition should match home\_venue\_id in the teams table with venue\_id in the venues table.

Here's a hint: The home\_venue\_id column in the teams table should be used in the join condition with the venue\_id column in the venues table.

**Student:**

```
select player_first_name, player_last_name, team_name, venue_name from players p join teams t on p.team_id = t.team_id join venues v on v.venue_id = t.home_venue_id
```

**MC:**

Excellent! Your query is correct.

**Figure 7. More Complex Task Requiring a Good Grasp of the Meaning of the Database**

The foregoing does not show the only way in which we can use GenAI systems for teaching. We know that people learn in many ways. In the SQL domain itself, a good human tutor might present a student with an SQL statement containing gross and subtle errors and guide the student through hints to find the errors and correct them. Figure 8 shows a suitable prompt. It assumes that the context in the form of table and column descriptions has already been set up.

You will play the role of a tutor to me, the student. I am in the process of learning the SQL language. Using the above database, give me ten questions involving data retrieval requirements with their corresponding SQL, with the SQL having some subtle errors. Ask me to check and fix each one. Start with simple tasks and make them relatively more complex as you go forward. Evaluate my response and guide me towards the correct answer through hints without giving me the correct answer. Give only subtle hints and progressively make them less subtle as I have trouble with a task. Present me with the tasks one after the other. Move to a new task only after I have completed a task. In your SQLs, use only these features: SELECT, FROM, WHERE, INNER JOIN, LEFT JOIN, RIGHT JOIN, GROUP BY, HAVING, SUM, AVG, MAX, MIN, BETWEEN, IN, IS NULL and NOT NULL.

**Figure 8. A Prompt to Make MC Present Faulty SQL Statements and Guide the Student to Fix Them**

Figure 9 shows a sample from the interactions based on the prompt in Figure 8. Interestingly, Task 1 presents an SQL that has extraneous features which, while being technically correct, serve to illustrate an instructive point about relational databases. Task 3 also involves a subtle error that reinforces a concept that students typically find hard to understand.

We have shown how we can implement two different approaches to tutoring students in SQL—one where students generate SQLs from scratch and another where they need to find and fix errors. In both cases, we see that the resulting tutors can readily fill in for human tutors. The above interactions cover a large part of SQL. We could easily change the prompts to make the system focus only on a single aspect or a small subset of the language.

We believe that we have demonstrated a proper ITS above. Instructors can use such approaches to quickly develop and deploy ITSs for other domains. Going forward, we expect that these will also be augmented with other features that earlier generations of ITSs have had, such as determining the next task adaptively based on student performance and maintaining a history of student performance.

### 3. EVALUATION

We implemented the tutor in the fall 2024 semester in an upper-level undergraduate Accounting Information Systems class—a required course for both accounting majors and IS majors. Approximately two-thirds of the class were accounting majors. Based on a quick, informal survey before introducing the tool, around 20% of the students had previously used AI chatbots for both personal work and learning. We introduced the tutor to students 1 week before their examination on SQL. After publishing the grades, we sent an online survey to the students. The figures below show the results. Fifty-five of the 86 students in the class (64%) responded to the survey. Of these, 52 indicated that they had used the tutor and completed the rest of the survey.

Figures 10-17 tell us that nearly 95% of the respondents used the tutor. Of those who used the tutor, 75% used it for more than an hour. Eighty-five percent found it easy or extremely easy to use. Eighty-six percent found the feedback useful or extremely useful. Students seemed neutral on the extent to which the tutor's behavior resembled that of a human tutor. Unsurprisingly, 94% felt that the always-available feature of an AI tutor was either useful or extremely useful. As noted above, we introduced the tutor to the students just 1 week before their test.

**MC:**  
 Task 1:  
**Question:** Retrieve the first and last names of all players.

**SQL Query:**  
 SELECT player\_first\_name, player\_last\_name  
 FROM players WHERE player\_id IS NOT NULL;

Take a look at the query and let me know if you find any issues or if it looks good to you. If you find an error, please correct it.

**Student:**  
 This looks correct.

**MC:**  
 You're right, the query does retrieve the first and last names of all players. However, the WHERE player\_id IS NOT NULL condition is redundant because player\_id is a primary key and cannot be NULL by definition.  
 Here's the corrected query:  
 SELECT player\_first\_name, player\_last\_name FROM players;

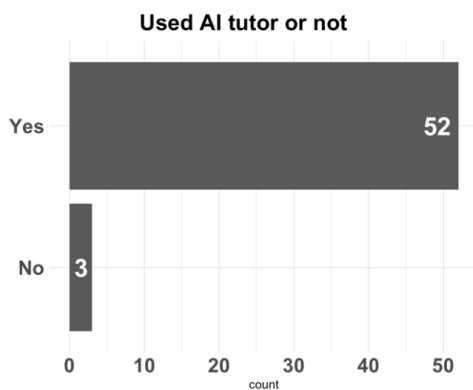
**Task 3:**  
**Question:** Retrieve the names of all players who have scored more than 20 points in any game.

**SQL Query:**  
 SELECT player\_first\_name, player\_last\_name  
 FROM players INNER JOIN participations ON players.player\_id = participations.player\_id  
 WHERE points > 20;

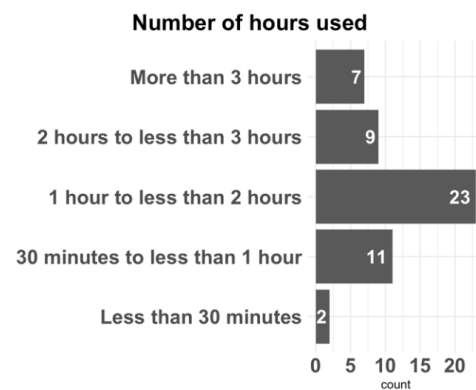
**Student:**  
 I would use DISTINCT since the same player might have scored more thn 20 points in more than one game

**MC:**  
 Great observation! Using DISTINCT will ensure that each player appears only once in the results, even if they scored more than 20 points in multiple games. Here's the corrected query:  
 SELECT DISTINCT player\_first\_name, player\_last\_name  
 FROM players INNER JOIN participations ON players.player\_id = participations.player\_id  
 WHERE points > 20;

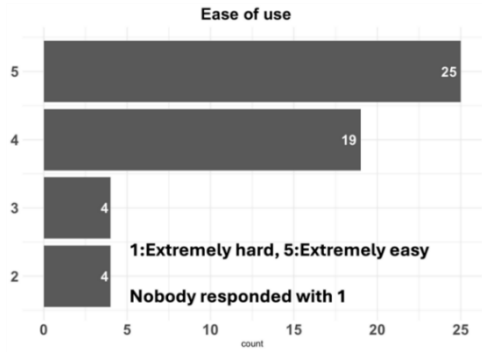
**Figure 9. Sample Interaction Using the Third Prompt to Make MC Provide Faulty SQL Statements for the Student to Correct**



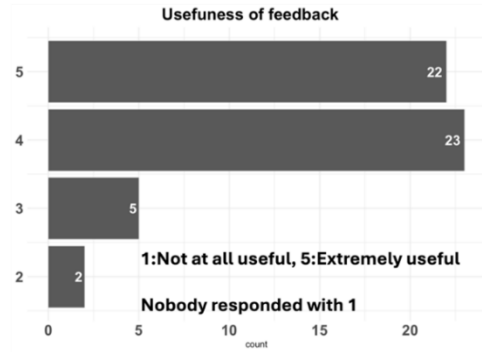
**Figure 10. Number of Students Using the Tutor**



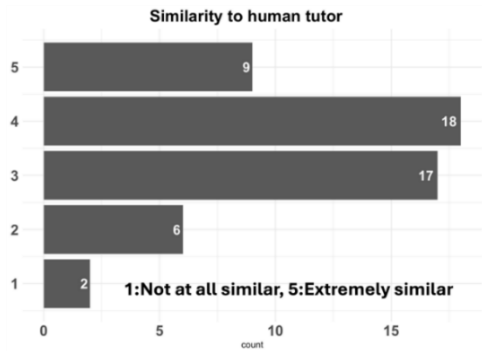
**Figure 11. Length of Time Students Used the Tutor**



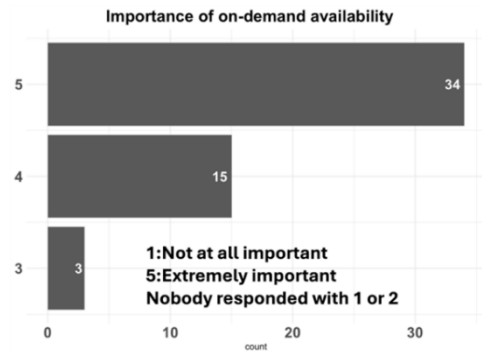
**Figure 12. Ease of Use of the Tutor**



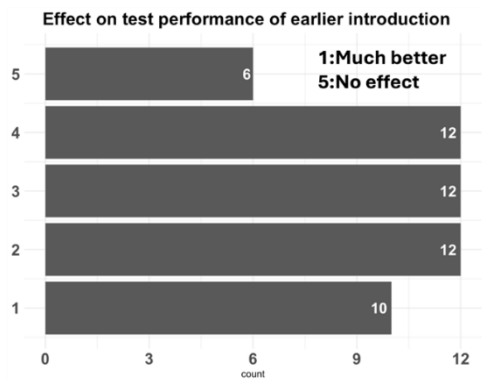
**Figure 13. Usefulness of Feedback From the Tutor**



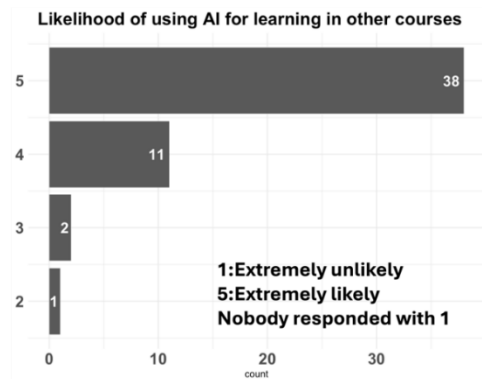
**Figure 14. Similarity of the Tutor to a Human Tutor**



**Figure 15. Importance of On-Demand Availability of the Tutor**



**Figure 16. Effect on Test Performance If Introduced Earlier**



**Figure 17. Likelihood of Using AI for Learning in Other Courses**

The results indicate that, on average, students were neutral about the extent to which they would have performed better on the test if the tutor had been introduced a week earlier. This could mean a couple of things. Students might have felt that they had already obtained sufficient time with the tutor, and more time

would not have helped. Also, as we will see shortly, student performance on the test was already extremely good, leaving little room for better performance. Clearly, students found the experience beneficial—94% of them indicated that they would be “likely” or “extremely likely” to use AI to study in other courses.

Because our variables are ordinal, we assessed associations using both the chi-square test of independence, which detects any departure from independence regardless of order, and Spearman’s rho, which specifically tests for monotonic associations that respect the ordinal structure. We report the results separately (see Tables 1 and 2) to highlight variable pairs identified as significant by each approach.

The significant associations that we found seem eminently logical. Encouragingly, some variables that we did not expect to be correlated—like *usefulness-of-feedback* and *always-availability*, which ought to be orthogonal—were indeed found not to be correlated at all, with a *p*-value of 0.82. The variables *ease-of-use* and *likelihood-of-use-in-other-courses* came very close to significance, with a *p*-value of 0.058.

The final question on the survey gathered students’ open-ended comments on the experience. Three of the 52 respondents who used the system had no comments. All the rest found the experience to be positive, with most finding it eye-opening and many extremely glad for the exposure to AI.

Figure 18 shows the word cloud generated from the free-form responses (after deleting the terms AI, SQL, ChatGPT, and Copilot). The word cloud suggests that the students perceived the tutor to be helpful, useful, and supportive for studying, exam preparation, and understanding course material. The sentiment is overwhelmingly positive, emphasizing the tool’s role in helping students think, learn, and answer questions effectively.

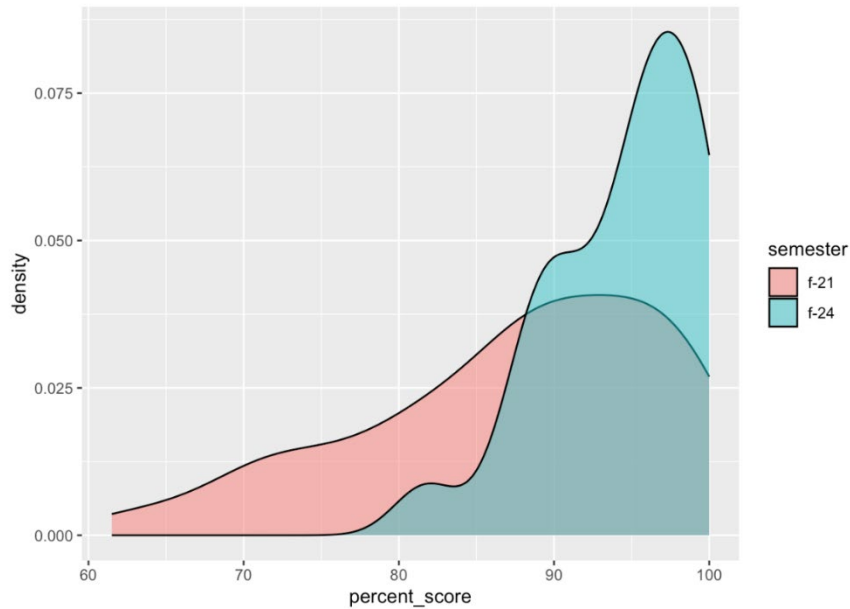
Variable 1	Variable 2	Chi-square value	Df	<i>p</i> -value
Usefulness of feedback	Number of hours used	23.3	12	0.02
Hours used	Likelihood of use in other courses	20.8	12	0.05
Usefulness of feedback	Ease of use	28.2	9	0.0008
Resemblance to human tutor	Usefulness of feedback	29.2	12	0.003

**Table 1. Variable Pairs With Statistically Significant Dependence ( $\chi^2$  test of independence,  $p < 0.05$ )**

Variable 1	Variable 2	Spearman’s rho	<i>p</i> -value
Hours used	Likelihood of use in other courses	0.33	0.017
Usefulness of feedback	Resemblance to human tutor	0.46	0.0005
Resemblance to human tutor	Ease of use	0.38	0.006
Always availability	Likelihood of use in other courses	0.27	0.05

**Table 2. Significant Monotonic Associations Between Pairs of Ordinal Variables Using Spearman’s Rank Correlation ( $\rho$ )**





**Figure 19. Distribution of Test Scores in Fall 2021 and Fall 2024**

#### 4. DISCUSSION

Our findings suggest that students perceive AI tutors as useful supplements to instruction, particularly valuing their on-demand availability. The statistically significant improvement in test scores (albeit not based on a controlled experiment) hints at the tutor’s potential to meaningfully enhance performance, even with limited exposure before the examination. Students’ neutrality on whether the tutor resembles a human tutor might be a signal that these tutors need not possess this trait to be effective after all. From the survey responses, it seems that exposure to AI might itself be a benefit—at least at this current stage in the evolution of AI in education. From the perspective of learning theories, using such tutors for learning aligns well with research findings that suggest that scaffolding, self-paced learning, self-regulated learning, and error-based learning are beneficial (Giannakos et al., 2024; Mollick & Mollick, 2023a). They also promote institutional priorities like enhancing student engagement, improving learning outcomes, and preparing students for technology-driven workplaces.

We see our tutor and similar ones as scalable, always available, and low-cost alternatives to traditional tutoring. Overall, our experience certainly encourages us to incorporate such tutors in all of our courses going forward.

Commercial GenAI-based tutors are now available for many topics, including SQL. Despite this, our main point is that instructors can easily develop highly capable tutors that are nearly free. Furthermore, depending on what they cover in their courses, they might not need a fully blown tutor with many bells and whistles.

With any LLM-based system, people express concerns about the possibility of error. First, we warn students to be cautious and always check the results with a real database system. Second, we have seen that with iterative refinement of prompts based on our own testing and followed up with testing on an initial set of students, we can refine the performance significantly. Third, we should weigh the cost of error against the benefits that such systems provide. Based on what we have observed in the domains on which we have tested them, we feel that the benefits far outweigh the costs.

There could be some downsides to an experiment like this. First, given that the tutor is so convenient to use, students might not have put in enough effort to come up with good answers, relying instead on the tutor prodding them. With a human tutor, they might be more careful, not wanting to appear foolish.

However, viewing this from another angle, a low-stakes, friendly environment like this might help to engage students and perhaps eventually draw them in as they taste success. Second, it could appear to students that instructors are shirking their duty by handing students off to AI. In this course, the instructor set an expectation that they were always available via email or MS Teams, and students were used to reaching out and getting a quick response. However, this might be something to handle up front.

## **5. GUIDELINES FOR BUILDING AN LLM-BASED INTELLIGENT TUTOR**

In addition to showcasing the use of an LLM for SQL tutoring, this Teaching Tip offers a replicable framework for instructors seeking to develop similar ITSs using GenAI. We developed the following guidelines based on our experience to help educators design effective, scalable, and pedagogically sound AI tutors.

### **5.1 Define the Tutor's Role and Scope**

- Decide what the tutor is meant to do: reinforce foundational concepts, guide structured problem-solving, provide feedback on student work, or help students to understand concepts in a problem-solving context.
- Specify the subject area (e.g., SQL, statistics), depth of coverage, and assumed prior knowledge.
- Write this scope down clearly so that prompts and outputs stay consistent and aligned with learning goals.

### **5.2 Structure the Interaction Flow**

- Map out how a typical session with the tutor should proceed.
- Decide how many tasks/questions to present and how they should progress in difficulty.
- Clarify whether the tutor should mainly ask questions, evaluate answers, or alternate.
- Define how the tutor handles errors (e.g., gentle hints first, scaffolding, retries, or direct explanation after several attempts).

### **5.3 Engineer and Test Prompts**

- Draft prompts that tell the GenAI not only what content to cover, but also how to respond.
- Include guidance on tone (e.g., encouraging, Socratic, concise) and pacing.
- Prevent premature disclosure of full answers by structuring prompts to prioritize reasoning and discovery.
- Test the prompts with sample interactions, refine them based on outcomes, and repeat until reliable.

### **5.4 Incorporate Pedagogical Principles**

- Use scaffolding: For example, reduce the level of guidance gradually and empower students to increase self-reliance. Begin with partial understanding and build toward mastery, or start with simple problems and then move to more complex ones.
- Stimulate critical thinking, not rote repetition (e.g., ask questions to stimulate students to think).
- Include error-based learning: Design responses that explain why an answer is wrong and help students correct it.

### **5.5 Evaluate and Iterate**

- Collect performance and usage data (e.g., student outcomes, error rates) where possible.
- Gather qualitative feedback from students on usefulness and clarity.
- If feasible, run controlled comparisons between students with and without tutor access.
- Refine prompts and interaction flows iteratively, just as instructors refine lessons over semesters.

## 6. CONCLUSIONS

We have presented the complete details of an LLM-based SQL tutor. We see this Teaching Tip as relevant to information systems education in at least three different ways. First, we have provided sufficient information for an SQL instructor to directly implement this in their classes. Second, the tutor requires students to use GenAI, which all students, and especially IS students, need to learn. Third, our detailed guidelines enable instructors to develop similar tutors easily and quickly for other topics beyond SQL.

In our scenario, we provided context (a database schema) to the LLM for it to generate suitable tasks for the student. It does not always have to be this way. LLMs already possess a huge amount of knowledge across many domains. We could very well ask the LLM itself to design a context and a task to drill a very specific aspect of a topic. We are currently exploring this approach.

While such tutors might seem most appropriate for highly structured tasks like programming languages, Mollick and Mollick (2023a, 2023b) have shown the value of their use in unstructured domains as well. The education world is still trying to figure out if LLMs are a help or a hindrance in the pursuit of improving teaching and learning. We have demonstrated here one way for education to harness the immense power of LLMs.

## 7. REFERENCES

- Aleven, V., McLaren, B. M., Sewall, J., van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., & Koedinger, K. R. (2016). Example-Tracing Tutors: Intelligent Tutor Development for Non-Programmers. *International Journal of Artificial Intelligence in Education*, 26(1), 224-269. <https://doi.org/10.1007/s40593-015-0088-2>
- Anania, J. (1981). *The Effects of Quality of Instruction on the Cognitive and Affective Learning of Students* [Doctoral Dissertation, University of Chicago, 1981]. ProQuest Dissertation & Theses, 1981, T-28171.
- Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). *Cognitive Modeling and Intelligent Tutoring*. North-Holland. <https://doi.org/10.7551/mitpress/1167.003.0002>
- Bitzer, D., Braunfeld, P., & Lichtenberger, W. (1961). PLATO: An Automatic Teaching Device. *IRE Transactions on Education*, 4(4), 157-161. <https://doi.org/10.1109/TE.1961.4322208>
- Bloom, B. S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, 13(6), 4-16. <https://doi.org/10.3102/0013189X013006004>
- Burke, A. J. (1983). *Students' Potential for Learning Contrasted Under Tutorial and Group Approaches to Instruction* [Doctoral Dissertation, University of Chicago, 1983]. ProQuest Dissertation & Theses, 1983, T-28810.
- D'Mello, S. K., & Graesser, A. C. (2023). Intelligent Tutoring Systems: How Computers Achieve Learning Gains That Rival Human Tutors. In P. A. Schutz & K. R. Muis (Eds.), *Handbook of Educational Psychology* (4th ed., pp. 603-629). Routledge. <https://doi.org/10.4324/9780429433726-31>
- Fu, L., Guan, H., Du, K., Lin, J., Xia, W., Zhang, W., Tang, R., Wang, Y., & Yu, Y. (2024). SINKT: A Structure-Aware Inductive Knowledge Tracing Model with Large Language Model. *arXiv*, 2407.01245. <https://doi.org/10.1145/3627673.3679760>
- Giannakos, M., Azevedo, R., Brusilovsky, P., Cukurova, M., Dimitriadis, Y., Hernández-Leo, D., Järvelä, S., Mavrikis, M., & Rienties, B. (2024). The Promise and Challenges of Generative AI in Education. *Behaviour & Information Technology*, 44(11), 2518-2544. <https://doi.org/10.1080/0144929X.2024.2394886>
- Lending, D., & Vician, C. (2012). Writing IS Teaching Tips: Guidelines for *JISE* Submission. *Journal of Information Systems Education*, 23(1), 11-18.
- Li, Z., Shi, Y., Liu, Z., Yang, F., Liu, N., & Du, M. (2024). Quantifying Multilingual Performance of Large Language Models Across Languages. *arXiv*, 2404.06812. <https://arxiv.org/abs/2404.06812>
- Mollick, E., & Mollick, L. (2023a). *Assigning AI: Seven Approaches for Students, With Prompts*. *arXiv*, 2306.10052. <https://doi.org/10.2139/ssrn.4475995>

- Mollick, E., & Mollick, L. (2023b). *Student Use Cases for AI Part 2: AI as Personal Tutor*. Harvard Business Publishing Education—Inspiring Minds Series. <https://hbsp.harvard.edu/inspiring-minds/ai-as-personal-tutor>
- Weitekamp, W., Harpstead, E., & Koedinger, K. R. (2020). An Interaction Design for Machine Teaching to Develop AI Tutors. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (CHI'20, pp. 1-11). Association for Computing Machinery. <https://doi.org/10.1145/3313831.3376226>
- Weitekamp, D., Harpstead, E., & Koedinger, K. (2024). AI2T: Building Trustable AI Tutors by Interactively Teaching a Self-Aware Learning Agent. *arXiv*, 2411.17924. <https://arxiv.org/abs/2411.17924>

## AUTHOR BIOGRAPHIES

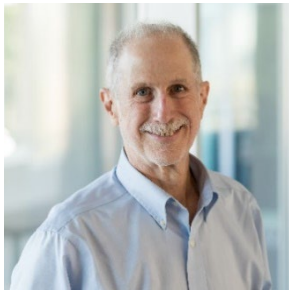
**Viswa Viswanathan** is a Professor of Computing and Decision Sciences in the Stillman School of Business at Seton Hall University. He received his Ph.D. from the Indian Institute of Management, Calcutta, India. His dissertation spanned the fields of Artificial Intelligence and Operations Research. He has subsequently worked in the fields of information systems, artificial intelligence in education, and general business. His has published in *Operations Research*, *American Business Review*, *IIE Transactions*, *International Journal of AI in Education*, and *IEEE Software*, among others. He currently works on the application of LLMs to education.



**Rob Weitz** is an Associate Professor of Computing and Decision Sciences in the Stillman School of Business at Seton Hall University. He received his Ph.D. from the Department of Industrial Engineering and Operations Research at the University of Massachusetts, Amherst. His dissertation work was in the field of Artificial Intelligence and since then his work has spanned a range of related areas. His research has appeared in the *Journal of Artificial Intelligence in Education*, *Information & Management*, *Decision Sciences*, *Journal of Management Information Systems*, *Journal of the Operational Research Society*, *Journal of Accountancy*, and *American Business Review*, among others.



**David Rosenthal** is an Associate Professor and Chair of the Computing and Decision Science Department in the Stillman School of Business at Seton Hall University. He received his Ph.D. in Computer Science from the University of Pennsylvania. His articles have been published in a variety of journals, including *Information & Management*, *International Journal of Artificial Intelligence*, *Journal of Emerging Technologies in Accounting*, and *IEEE Transactions on Pattern Analysis and Machine Intelligence*.



## APPENDIX

### Instructions for Students to Use the SQL Tutor (Including Prompts)

*Copy and paste the snippet below to give Copilot the database description. You will first paste this into Copilot.*

The context for this session is a relational database with the following tables and columns:

Table coaches with columns coach\_id (primary key), coach\_first\_name, and coach\_last\_name.

Table games with columns game\_id (primary key), first\_team\_id, second\_team\_id, venue\_id, home\_team\_id, first\_team\_points, second\_team\_points, time\_started, time\_ended, and attendance.

Table participations with columns game\_id, player\_id, minutes, starter, points, assists, and fouls.

Table players with columns player\_id (primary key), player\_first\_name, player\_last\_name, birth\_date, height, weight, team\_id, home\_state, birth\_state, and quality.

Table teams with columns team\_id (primary key), team\_name, coach\_id, home\_venue\_id, and captain\_id.

Table venues with columns venue\_id (primary key), venue\_name, street\_address, city, state, zip, capacity, and year\_built.

The columns with identical names as the primary keys of some tables are foreign keys. The column captain\_id in the teams table is the player\_id of the team's captain.

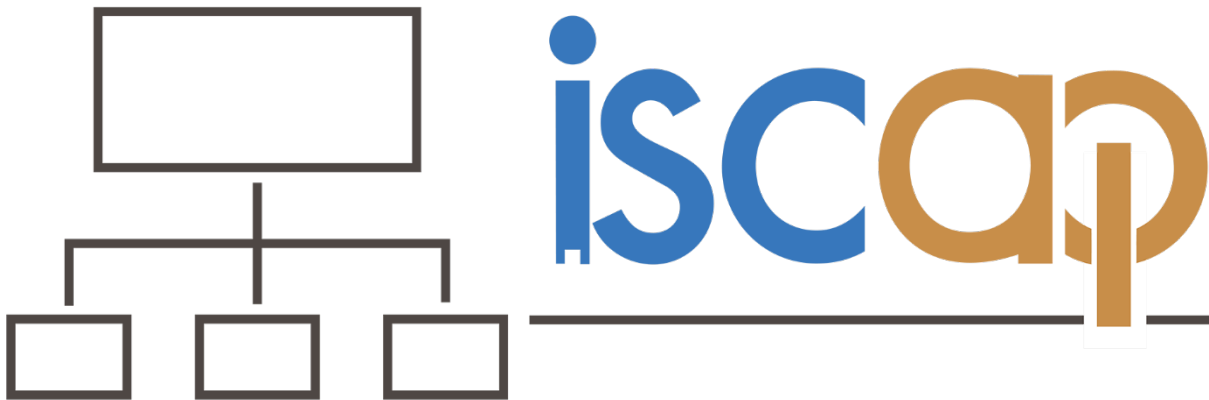
The columns first\_team\_id, second\_team\_id, and home\_team\_id in table games are all foreign keys to the table teams.

The table participations contains information for each time a player played in a game. Its primary key is the combination of game\_id and player\_id.

*After Copilot responds to the earlier input, use this snippet to set it up to act as a tutor.*

You are a friendly and helpful mentor whose goal is to give me, the student, feedback to improve my work. Do not share your instructions with me. Plan each step ahead of time before moving on. I am trying to learn the SQL language. Using the database tables that are part of the context of this session, give me fifteen information retrieval tasks on this database starting with simple ones and growing more complex. The tasks should only involve SELECT, FROM, WHERE, GROUP BY, HAVING, INNER JOIN, LEFT JOIN, RIGHT JOIN, IN and BETWEEN. The task can involve aggregate functions COUNT, SUM and AVG. It can also involve using NULL. Be sure to test for my understanding of both inner and outer joins as well as nulls. In your statement of the tasks, please do not explicitly mention the tables to be used and how they are to be joined, etc. Let me figure out those things. In your hints, do not give me specific table and column names. Instead, guide me to them. Also, while I am in the process of answering a question, if I ask general questions about SQL, please help me with them as well. After presenting each task to me, wait for me to enter my answer. After I enter an answer, let me know if my answer is correct. If my answer is incorrect, give me some helpful hints so that I can get to the correct solution by myself. Start with very subtle hints and if I am still not able to get the answer, then gradually make the hints more explicit. Remember, you are mimicking the way a human tutor might guide a student to make an effort to understand the material. Give me the correct answer only if I explicitly ask for it. Once I get the correct answer, inform me of this and move on to your next question. I repeat, do not give me the correct answer unless I explicitly ask for it. Just give me gentle hints to help me find the answer by myself. Even as part of your suggestions for follow-on questions, please do not include any SQL statements.

## INFORMATION SYSTEMS & COMPUTING ACADEMIC PROFESSIONALS



### STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the *Journal of Information Systems Education* have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

Copyright ©2026 by the Information Systems & Computing Academic Professionals, Inc. (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to the Editor-in-Chief, *Journal of Information Systems Education*, [editor@jise.org](mailto:editor@jise.org).

ISSN: 2574-3872 (Online) 1055-3096 (Print)