Assignment 1

Dong min Kim
dkim614@gatech.edu

1 RESEARCH LOG

1.1 Background

In today's world, coding has become a fundamental skill that people pursue regardless of their career paths or academic backgrounds. Learning to code effectively, however, requires a clear understanding of both the field of application and the programming languages themselves. While learning to code in the past often meant spending a significant amount of money on academies or online courses, the landscape has changed. These days, there are countless apps available that provide everything from basic exercises to advanced, hands-on challenges. Many of my friends have shared positive experiences with these tools, citing their accessibility and flexibility as major advantages.

I personally tried one of these apps when I wanted to pick up the basics of Python. It was helpful at first, but over time, I lost interest because the app lacked personalized feedback or deeper guidance that would have kept me engaged. That experience made me realize how important adaptive support and feedback are in sustaining learning motivation. Recently, as I've been exploring Kotlin for Android development, I've been thinking about creating an app that offers clearer explanations, structured exercises, and personalized feedback to make learning SQL more engaging and effective. My goal for this project is not only to build my technical skills but also to contribute to the educational technology space by developing a functional prototype of an SQL tutoring app.

1.2 Papers

1) The behavior of tutoring systems (VanLehn, 2006)

This paper explains how tutoring systems work in two loops: the outer loop picks the problems, and the inner loop walks you through them step by step. It really made me see why smaller steps make learning less overwhelming. For my SQL app, I'm thinking of starting with simple SELECT queries and then slowly building up to joins and aggregations. Tracking progress at each step just feels right to make it more personal and adaptive.

2) Cognitive tutors: Lessons learned (Anderson, Corbett, Koedinger, & Pelletier, 1995)

This one talks about how cognitive tutors adapt to the learner, making sure you only move forward when you're ready. It really drives home how important it is to build a strong foundation first. For my app, I'll have people master the basics before moving on to more advanced SQL topics. It's a simple idea but one that makes the whole learning experience smoother and less frustrating.

3) An overview of intelligent tutoring system authoring tools (Murray, 2003)

This paper reviews the tools people use to build tutoring systems and shows the pros and cons of each. The big takeaway is that it's better to start simple and then build up. For my SQL tutor, I'll keep things lightweight in the beginning so I can make changes easily. Once it's stable, I can add cool features like adaptive feedback and analytics.

4) An effective metacognitive strategy (Aleven & Koedinger, 2002)

This one points out that explaining your thinking as you solve problems helps you really understand what you're doing. That made me think about adding a simple feature in my app where users can type a quick note on why they wrote a query a certain way. It's a small thing, but it could help reinforce what they're learning and help me see where people are getting stuck.

5) Designing for metacognition (Roll, Aleven, McLaren, & Koedinger, 2007)

This paper is all about asking for help the right way. If you just keep hitting the hint button without trying, you don't really learn. That hit close to home because I've done that myself. In my app, I want to design hints that only show up after you've made an honest attempt, so you're still thinking and not just relying on the system.

6) Adaptive hypermedia (Brusilovsky, 2001)

This paper talks about how to make learning adaptive, adjusting content based on what the user is doing. But it also says don't make it so strict that people can't explore on their own. For my SQL tutor, I'll personalize the lessons but still let users jump around if they want to review or skip ahead. That flexibility will make it feel less restrictive and more engaging.

7) An intelligent SQL tutor on the web (SQLT-Web) (Mitrović, 2003)

This one is actually about a web-based SQL tutor, which is super relevant to me. It looks at your query, figures out what's wrong, and gives feedback that actually makes sense. Plus, it handles

multiple correct answers, which is huge. I want to bring that same kind of flexibility into my app so users don't feel boxed in.

8) New potentials for data-driven intelligent tutoring systems (Koedinger, Brunskill, Baker, McLaughlin, & Stamper, 2013)

This paper is all about using data to make tutoring systems better over time. Collect the data, look at the patterns, make changes, and repeat. That's exactly what I want to do with my SQL tutor. From day one, I'll track clean, organized data so I can make smart updates based on how people are actually using the app.

9) The relative effectiveness of human tutoring and ITS (VanLehn, 2011)

This one says intelligent tutoring systems, when done well, can be almost as good as human tutors. That's really motivating. It made me think about making my SQL app more interactive and detailed so it feels like you've got someone guiding you through each step. That kind of step-by-step feedback will make the learning process a lot more effective.

10) AutoTutor (Graesser, Chipman, Haynes, & Olney, 2005)

AutoTutor is this system that talks to you in a conversational way, helping you understand concepts better. It made me think that maybe one day my SQL app could have a chat-like feature to guide users. For now, though, I'll stick with simple text explanations and build from there once I get the basics right.

11) AutoTutor and family (Nye, Graesser, & Hu, 2014)

This review looks at 17 years of research on AutoTutor. The big lesson is that natural language tutoring can be amazing but also really complex to build and maintain. So, I'll start with structured, template-based feedback first. Later, if my app is solid, I can slowly try adding natural language for more personalized interactions.

12) Intelligent tutoring goes to school in the big city (Koedinger, Anderson, Hadley, & Mark, 1997)

This paper talks about bringing tutoring systems into real classrooms and what makes them work. Teacher support and aligning with the curriculum are super important. If my SQL tutor ever gets tested in classrooms, I'll need features like dashboards to track progress and maybe offline mode for areas with weak internet.

13) Classroom integration of intelligent tutoring systems (Ritter, Koedinger, Hadley, Corbett, & Lilly, 2004)

This one looks at how tutoring systems fit into classrooms and what problems can pop up during integration. It really showed me that having good teacher training and a smooth setup process is key. For my app, I'm thinking about syncing with LMS platforms and adding reports that show common mistakes so teachers can help more efficiently.

14) ItsSQL: Intelligent Tutoring System for SQL (Reid, Kammer, Kunz, Pellekoorne, Siepermann, & Wölfer, 2023)

This one is about a modern SQL tutor called ItsSQL that uses a solution pool to evaluate multiple correct answers. I love how scalable that is. For my app, I'll try to build something similar from the start so I don't have to redo everything later. It's like setting things up now for smoother growth down the line.

15) Qr-Hint: Actionable hints towards correcting wrong SQL queries (Hu, Gilad, Stephens-Martinez, Roy, & Yang, 2024)

This paper is about a system that gives actionable, step-by-step hints for fixing SQL queries. It doesn't just say "wrong" — it guides you through fixing it. That's exactly the kind of experience I want to build. I'll start with simpler hints and then eventually add visuals like query comparisons to make it even clearer.

1.3 Synthesis

After reviewing 15 papers, several clear themes stood out that directly connect to my SQL tutoring app idea. The first is the importance of step-by-step learning and feedback. VanLehn (2006) explained the concept of outer and inner loops, which fits perfectly with how I want to structure my app. Instead of just checking if the final answer is correct, the app should break SQL problems down into smaller pieces like SELECT, JOIN, and GROUP BY and provide targeted feedback for each step. Anderson et al. (1995) reinforced this with their discussion of mastery-based learning, showing that learners build a stronger foundation when they only move forward after mastering the basics.

Another strong theme is personalization and adaptivity. Brusilovsky (2001) and Koedinger et al. (2013) showed how adaptive systems that respond to user progress lead to higher engagement and better outcomes. This insight tells me my app should be dynamic, adjusting problem difficulty and hint detail depending on how the learner is performing. Mitrović (2003) and Hu

et al. (2024) provided concrete examples in the SQL domain, highlighting the need to handle multiple correct answers and provide actionable, context-aware hints. Those systems showed me that SQL tutoring is not just about syntax correction but about building an interactive and flexible experience.

Metacognition was another recurring topic. Aleven and Koedinger (2002) and Roll et al. (2007) emphasized how prompting learners to explain their reasoning and to request help effectively can lead to deeper understanding. For my app, this means adding simple reflection prompts — for example, asking learners to explain why they wrote a query in a particular way — and structuring hint usage so that students try solving the problem before requesting help.

Lastly, the importance of data-driven design came through in almost every paper. Koedinger et al. (2013) stressed how logging user data allows for iterative improvements, and Reid et al. (2023) demonstrated how a reference solution pool can manage the complexity of multiple valid SQL solutions. These findings convinced me that building strong analytics into the app from the beginning will make it easier to refine the system over time with real data.

Overall, the literature made it clear that an effective SQL tutor app should focus on incremental learning, personalized feedback, metacognitive prompts, and data-driven iteration. These four elements will guide how I approach the design and prototyping phases of my project.

1.4 Reflection

Reading these papers was a lot more eye-opening than I expected. At first, I thought I was just gathering general ideas about intelligent tutoring systems, but the more I read, the more I started seeing concrete patterns that apply directly to my project. Understanding the importance of step-level feedback and mastery-based progression gave me a much clearer sense of how to structure the learning flow for the app. VanLehn's concept of inner and outer loops, in particular, felt like the missing piece I needed to visualize the learning experience I want to build.

It wasn't all easy. Some of the more technical papers, especially the ones on data-driven optimization and natural language tutoring, took multiple reads to fully understand. But that process helped me prioritize what's essential for now and what can wait for later iterations. The biggest realization was that I don't need to build something overly complex from the start; instead, I can launch with a simple, well-structured version and gradually add more advanced features as I learn.

What I found most exciting was connecting the research to my own experiences. I remembered how, when I tried to learn Python using a basic app, I lost motivation because it didn't adapt to me or offer useful feedback. Seeing solutions in these papers for problems I've personally faced made this research feel much more meaningful.

1.5 Planning

For next week, I'm still not ready to start building anything yet. I want to keep digging into research so I can shape a clearer direction for my SQL tutor idea. I'm especially planning to look deeper into papers on Game-Based Learning and Large Language Models because I think combining those ideas could make the app more engaging and adaptive. I'll also keep refining the structure of how I want the lessons to flow — like what a beginner's path would look like versus an intermediate one — and start thinking about how the feedback loop should actually feel for users. By the end of the week, I want to have a more detailed outline or "blueprint" for the project, along with notes on how game-like elements and AI-driven features could fit into the design.

2 ACTIVITY

1) VanLehn, K. (2006). The behavior of tutoring systems

Need:
When I first started thinking about my app, I knew I needed to understand how tutoring systems actually guide students. This paper digs into why step-by-step guidance is so important.

Method:
VanLehn looked at different tutoring systems and came up with a model that splits them into two parts: the outer loop that picks problems and the inner loop that helps students step by step.

Audience:
Mostly people building or researching tutoring systems, but honestly, it's super helpful for anyone designing a learning tool.

Results:
The main point is that students learn better when they get small, incremental feedback instead of just being told if their answer is right or wrong at the end.

Critique:

This was really useful for me. It's not SQL-specific, but it gave me a clear idea that my app should help learners progress one small step at a time, like starting with SELECT queries and slowly moving up to joins and nested queries.

2) Mitrović, A. (2003). An intelligent SQL tutor on the web (SQLT-Web)

Need:

I remember how frustrating it was when I was learning SQL and couldn't figure out what was wrong with my query. This paper addresses that exact problem.

Method:

SQLT-Web uses a constraint-based model. Basically, instead of only checking syntax, it looks at the logic behind a query and then gives feedback that actually helps you fix it.

Audience:

This was made for students learning SQL, but it's also a goldmine for someone like me trying to build a tutoring app.

Results:

Students got a lot better with SQL and felt more confident because they were actually understanding their mistakes, not just guessing.

Critique:

It feels a bit old-school now, but the core idea is still solid. I want to bring that same kind of logic-focused feedback into my app, but with a cleaner, more modern experience.

3) Reid, S. A., Kammer, F., Kunz, J., Pellekoorne, T., Siepermann, M., & Wölfer, J. (2023). ItsSQL

Need:

After reading about SQLT-Web, I realized there's still a need for something more modern and scalable. That's where ItsSQL comes in.

Method:

ItsSQL uses something called a reference solution pool. Basically, it stores multiple correct solutions so the system can handle different approaches to the same problem.

Audience:

It's meant for today's learners and teachers who want smarter, more flexible tutoring tools.

Results:
The system gave better, more accurate feedback and kept improving because it was designed to update based on real user data.

Critique:
This one got me thinking a lot. I'm not ready to build something as complex as ItsSQL, but I like the idea of starting small and designing the app in a way that can grow into something like this over time.

4) Hu, Y., Gilad, A., Stephens-Martinez, K., Roy, S., & Yang, J. (2024). Qr-Hint

Need:
One of the things that frustrated me most when I was learning SQL was knowing my query was wrong but not understanding why. This paper is all about fixing that.

Method:
Qr-Hint compares your query to a correct one and gives you step-by-step hints to help you fix it. It's not just about syntax — it focuses on the logic of your query.

Audience:
Perfect for students learning SQL or developers like me trying to make better feedback systems.

Results:
The hints made students more confident and helped them solve problems faster. They weren't just guessing anymore; they understood what needed to change.

Critique:
This is exactly the kind of experience I want in my app. I know I can't build something this advanced right away, but I can start with simpler hints and work up to something like Qr-Hint as I collect more data.

5) Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors

Need:
Back then, tutoring was either one-size-fits-all or really hard to scale. This paper tackled the problem by showing how systems could adapt to each learner.

Method:
The Cognitive Tutor framework builds a model of what the student knows and uses that to guide what problems they should do next, giving feedback in real time.

Audience:

It was written for educators and developers, but as someone trying to build a SQL tutor, it was a really helpful starting point.

Results:

Students using these tutors learned faster and retained more knowledge compared to traditional classrooms.

Critique:

This made me realize how important it is to build a mastery-based flow in my app making sure learners really understand the basics before they move on. The challenge will be finding the right balance so users don't feel stuck, but this paper gives me a good direction to follow.

3 REFERENCES

1. VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227–265.
2. Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167–207.
3. Murray, T. (2003). An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In *Authoring tools for advanced technology learning environments* (pp. 491–544). Springer.
4. Aleven, V., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science, 26*(2), 147–179.
5. Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2007). Designing for metacognition: Applying Cognitive Tutor principles to the tutoring of help seeking. *Metacognition and Learning, 2*(2), 125–140.
6. Brusilovsky, P. (2001). Adaptive hypermedia: From intelligent tutoring systems to web-based education. *User Modeling and User-Adapted Interaction, 11*(1–2), 87–110.
7. Mitrović, A. (2003). An intelligent SQL tutor on the web (SQLT-Web). *International Journal of Artificial Intelligence in Education, 13*(2–4), 173–197.
8. Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., & Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine, 34*(3), 27–41.
9. VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist, 46*(4), 197–221.

10. Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education, 48*(4), 612–618.
11. Nye, B. D., Graesser, A. C., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education, 24*(4), 427–469.
12. Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*(1), 30–43.
13. Ritter, S., Koedinger, K. R., Hadley, W., Corbett, A. T., & Lilly, M. (2004). Classroom integration of intelligent tutoring systems for algebra and geometry. *Carnegie Learning Report*.
14. Reid, S. A., Kammer, F., Kunz, J., Pellekoorne, T., Siepermann, M., & Wölfer, J. (2023). ItsSQL: Intelligent Tutoring System for SQL. *arXiv preprint arXiv:2311.10730*.
15. Hu, Y., Gilad, A., Stephens-Martinez, K., Roy, S., & Yang, J. (2024). Qr-Hint: Actionable hints towards correcting wrong SQL queries. *Proceedings of ACM SIGCSE 2024*.