2024 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE) | 979-8-3503-7623-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/TALE62452.2024.10834288

Mixing up Gemini and AST in ExplainS for Authentic SQL Tutoring

Hunter Clark
Department of Computer Science
University of Idaho, USA
clar9853@vandals.uidaho.edu

Hasan M. Jamil ✉
Department of Computer Science
University of Idaho, USA
jamil@uidaho.edu

***Abstract*—Mastering SQL is a key data science competence. While most large language models are able to translate natural language queries to SQL, their ability to tutor learners and authentically assess student assignments are at the least fragile. In this paper, we introduce *ExplainS* as an experimental prototype. In this web-based system, we augment Gemini with abstract syntax tree (AST) to enhance Gemini's semantic analysis power to be able to assist and tutor students better. This edition of ExplainS provides a collection of exercises with varying difficulty levels, covering core SQL concepts. Users interact with a dynamic schema display, and their queries are validated against carefully crafted solutions. To provide context-aware personalized feedback, ExplainS leverages Gemini and the SQLglot library to analyze query AST differences between user queries and correct solutions, pinpointing the root cause of errors. This emerging research is part of a wider Data Science effort, and in this paper, we only focus on the meaningful feedback generation component of the ExplainS system.**

***Index Terms*—SQLglot, Authentic Assessment, Tutoring, Personalized Feedback, Large Language Models.**

I. Introduction

SQL (Structured Query Language) is a foundational technology for interacting with relational databases. Its widespread use emphasizes the importance of effective SQL learning. However, traditional learning methods often fall short, providing static examples and limited opportunities for guided practice, and many students find it difficult to learn SQL and manage data using it. Tutoring systems have been developed to teach SQL [23], help students learn [3, 21, 26] and assess their progress [14]. Though generating feedback in the learning process has been a complicated issue [11, 22], several systems [16, 17] offer interesting solutions to this problem.

Assessment of students' SQL assignments [8, 9, 13, 19] has also met with serious challenges and continues to remain so. Query understanding [10] and query equivalence [6] have been logical approaches to SQL assignment grading. Assessment systems [12] have leveraged query equivalence with limited success due to theoretical limitations. But, recent research suggests that SQL equivalence, and thus assessment of SQL assignments, may have a brighter future using large language models (LLM) [4, 29].

However, exploiting the powers of LLMs such as ChatGPT, Gemini, or Llama remained largely focused on text-to-SQL generation [7, 15, 24, 28], and not many research have utilized LLMs for tutoring or assessment purposes of SQL assignments though opportunities emerged recently [1, 5, 18, 20, 27]. In this paper, we introduce a new LLM-based SQL tutoring system, called *ExplainS* (stands for Explainable SQL), to address these limitations through a web-based platform that combines interactive exercises with personalized feedback. Users engage with dynamic schema representations while constructing their queries. ExplainS's integration of LLMs enables the generation of insightful guidance tailored to individual learner's errors and misconceptions.

II. Related Works

Learning is about acquiring the knowledge to understand and solve problems, and just not knowing the answers to problems. The process of learning vary widely between students and not one size fits all. One of the major goals of a tutoring and assessment system is to support students in their learning process with tools to sharpen their creative abilities and improve analytical capacities. It is, therefore, not sufficient to just solve an SQL problem for a student [7, 15, 26, 28], but help them learn how to write one.

The tutoring and assessment systems that do aim to improve learning outcomes often fail to address the main challenges of tutoring and assessment. For example, a long series of research on SQL-Tutor [25] addresses learning outcome by focusing on issues that are tangential to actual crafting of SQL statements and the problems students face to learn coding in SQL. SQLTOR [26], however, leveraged AST to analyze student's queries toward generating guidance, and a progressive study plan using machine learning from task difficulties they face mainly to help the instructors design SQL tutorials. An interesting tutoring system, SQLearn [9], also leveraged cosine similarity of ASTs to measure students' query similarity with correct/reference solutions. A similar AST inspired tutoring system is ItsSQL [21] which has similar goals.

SQL Tester [13] shows that use of an assessment system helps improve student engagement. The aSQLg system's [14] assessment of SQL assignments largely are of syntactic in nature, and helps student learn the correct syntax, but does not offer much support to learn the semantic deficiencies. In contrast to these systems, ExplainS combines AST with Gemini to fill in gaps in these systems. It also serves both as a tutor and a grader. In the sections to follow, we discuss the features and techniques used in ExplainS in details.

III. Complementing LLMs with AST

Reid et al. [21] demonstrate that providing multiple reference solutions to an SQL query assignment improves the chances of establishing equivalence of students' solution with a correct one. The subtext of this research is that many tutoring and assessment systems fail to accurately tutor and grade student assignments. The theoretical query rewriting system Cosette [6], and systems based on AST similarity [2] try to address these limitations significantly differently. Cosette though accurate in determining query equivalence, it can do so for very simple queries, and its it only can return a true-false response. AST similarity on the other hand is heuristic driven and in some cases it can return decisions that are incorrect or inaccurate. Because structural dissimilarity does not entail non-equivalence as ItsSQL [21] suggests.

A. AST as an Aid for Feedback Generation

Traditional SQL feedback systems often rely on simplistic error messages that merely point out the existence of a problem, leaving learners to decipher the underlying cause. ExplainS takes a more insightful approach by utilizing ASTs to pinpoint the exact structural differences between a user's query and the correct solution. An AST represents an SQL query's underlying grammatical structure as a hierarchical tree. The Python SQLglot tool has a function called AST Diff. When two queries are compared for differences, AST diff prints what a test query needs to change to match the solution query's AST. This allows for a test query to be compared with a correct solution and discover clues to provide insightful feedback. In ExplainS, when a user submits a (test) query to try and solve the exercise, SQLglot AST Diff is used to check what a user query would need to change to match the solution query.

We discuss the idea using the example **Q1** below.

> **Q1**: List all pets (PetID, Name, TypeofPet) living in South Alex, NE and owned by a minor who has no income. List the pets in ascending order of their names and in descending order of their pet type when the names are identical.

The assumed correct or reference query to compute the response is the SQL query below in the context of the three tables in the database with the schemes shown in Fig 3.

```
SELECT pets.PetID, pets.Name, pets.TypeofPet
FROM pets LEFT JOIN owns ON
  pets.PetID = owns.PetID
  LEFT JOIN owners ON owns.OID = owners.OID
WHERE pets.City = 'South Alex' AND
  pets.State= 'NE' AND owners.Age < 18 AND
  owners.AnnualIncome <= 0
ORDER BY pets.Name asc , pets.TypeofPet desc
```

Now consider the student query or test SQL query as follows submitted as a solution. One obvious discrepancy or difference with the reference query is that the test query does not have a filter for age to exclude minors as required by the query. Our expectation is that ExplainS should be able to identify this semantic error and suggest a remedy.

```
SELECT pets.PetID, pets.Name, pets.TypeofPet
FROM pets LEFT JOIN owns ON
  pets.PetID = owns.PetID
  LEFT JOIN owners ON owns.OID = owners.OID
WHERE pets.City = 'South Alex' AND
  pets.State= 'NE' AND
  owners.AnnualIncome <= 0
ORDER BY pets.Name asc , pets.TypeofPet desc
```

B. AST Similarity

It is easy to demonstrate that while two identical ASTs represent identical queries, and thus will compute identical responses on the same database (computationally equivalent), dissimilarity of ASTs does not entail non-equivalence of queries. Therefore, regardless of the similarity functions used to discover AST edit distances, as was done in SQLearn system [9], dissimilarity or a high edit distance will not be useful to discover functional equivalence as in Cosette [6]. Such similarity may help queries such as **Q1**, but will not for queries such as **Q2** below,

> **Q2**: List all pets (PetID, Name, State) which ate every most expensive foods in each class of food (ClassofFood) produced by Purina.

for which several equivalent SQL queries could be written.

This is essentially a division query. To solve this query, we need to identify the most expensive food items in every category produced by Purina in a set S. Then, we need to find a pet which has eaten each one of these food items in S, and do so for all pets, and list them.

```
SELECT p.PetID, p.Name, p.State
FROM Pets p NATURAL JOIN Owns o
WHERE
  (SELECT COUNT(*)
  FROM (SELECT DISTINCT FoodID
        FROM Purchases pu NATURAL JOIN
            (SELECT FoodID, ClassofFood,
                MAX(Price) as mPrice
            FROM Foods
            WHERE Brand='Purina'
            GROUP BY ClassofFood) m
        WHERE p.PetID = pu.PetID) =
            (SELECT COUNT(*)
            FROM (SELECT FoodID, ClassofFood,
                    MAX(Price) as mPrice
                FROM Foods
                WHERE Brand='Purina'
                GROUP BY ClassofFood));
```

Figs 1(a) and 1(b) were generated by ChatGPT as candidates for query **Q2**. Query in Fig 1(a) happens to be incorrect, as this query will not generate pets who ate all the most expensive foods in each category. It will list a pet if it ate any of the most expensive foods, but query in Fig 1(b) is correct.

It should be apparent that their ASTs will be vastly different that the SQL query above. In particular, the ASTs of the SQL query above and the query in Fig 1(b) though very different, they will compute identical responses as they are functionally identical. The question is, could a tutoring system identify the similarity of these two queries by analyzing the two corresponding ASTs, and report an error and recommend corrective steps for the pairs involving Fig 1(a). The answer to

```sql
SELECT p.PetID, p.Name, p.State
FROM Pets p
INNER JOIN Owns o ON p.PetID = o.PetID
INNER JOIN Purchases pu ON pu.PetID = p.PetID
INNER JOIN Foods f ON pu.FoodID = f.FoodID
WHERE f.Brand = 'Purina'
  AND f.Price = (
    SELECT MAX(f2.Price)
    FROM Foods f2
    WHERE f2.ClassofFood = f.ClassofFood
      AND f2.Brand = 'Purina'
  );
```

(a) SQL Query Written by ChatGPT for Query **Q2**.

```sql
SELECT p.PetID, p.Name, p.State
FROM Pets p
INNER JOIN Owns o ON p.PetID = o.PetID
WHERE NOT EXISTS (
    SELECT f.ClassofFood
    FROM Foods f
    WHERE f.Brand = 'Purina'
    GROUP BY f.ClassofFood
    HAVING MAX(f.Price) > 0
    AND MAX(f.Price) > (
        SELECT MAX(f2.Price)
        FROM Foods f2
        WHERE f2.Brand = 'Purina'
        AND f2.ClassofFood = f.ClassofFood
    )
    AND NOT EXISTS (
        SELECT NULL
        FROM Purchases pu
        WHERE pu.PetID = p.PetID
        AND pu.FoodID = (
            SELECT f3.FoodID
            FROM Foods f3
            WHERE f3.Brand = 'Purina'
            AND f3.ClassofFood = f.ClassofFood
            ORDER BY f3.Price DESC
            LIMIT 1
        )
    )
);
```

(b) Amended SQL Query for Query **Q2**.

Fig. 1. SQL queries written by ChatGPT.

this question is substantially difficult to answer without more research, which we plan to do. But, currently we are able to answer this question for non-equivalent query ASTs.

C. LLM as an Error Message Interpreter

A possible set of suggestions for the students could include messages such as

> Your query has some differences from the expected solution. Here are some suggestions to improve it.
>
> - Add the condition `AND owners.Age < 18` to the WHERE clause.
> - Change the ORDER BY clause to `ORDER BY pets.Name ASC, pets.TypeofPet DESC`.

Our goal is to use SQLglot's AST Diff function to first identify the query difference, then discover the steps that are needed to make the test query similar to the reference query in ways similar to the concept of "edit distance", and then use extensive prompt engineering and potentially training an LLM on AST Diff to generate the needed suggestions. We believe that there is an opportunity to create an SQL self-learning platform that provides feedback on how a user can improve their query. It also provides best practice recommendations as shown above with the capitalization of DESC and ASC in the above LLM recommendations. In the next few sections, we discuss how we approach the implementation of ExplainS to acheive this goal.

IV. Architecture of ExplainS

The logic model of the ExplainS is shown in Fig 2. Once students work on an SQL assignment available in the assignment database, they submit their solutions in two possible modes – tutoring or assessment, by first selecting one of the three difficulty levels – Beginner, Intermediate or Hard. Once submitted, the query is analyzed, executed and compared with reference solutions before a feedback is generated. While the assessment currently is at the true/false level, a part credit attribution is also possible.

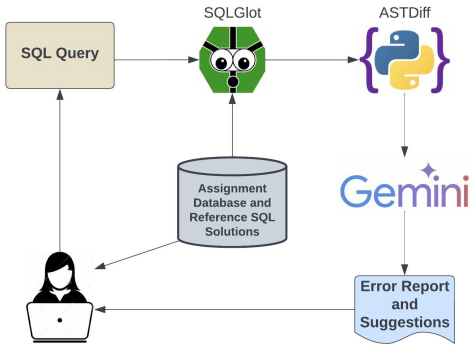

Fig. 2. ExplainS Tutoring Pipeline.

A. System Design

ExplainS leverages the Django web framework for its data management and templating capabilities with a goal to providing a scalable foundation for the interactive learning environment. The front-end, while currently basic, employs out-of-the-box CSS and Bootstrap for a clean and organized look. The system's core functionality is built around the following key components.

1) Database Support: SQL assignments and tests in the assignment database are always based on live database schemes and instances against which they are tested. ExplainS uses the Python Faker library to generate a comprehensive mock dataset for each of its database scheme in the SQL exercises, and provides functionalities to create realistic data. This approach fosters an engaging learning experience by allowing users to practice against datasets that resemble real-world scenarios. The database used is PostgreSQL. Once students choose an assignment or test for use, the correspond database is initiated and loaded to support the tutoring or assessment functions.

A syntax analyzer rigorously parses user queries before onward processing to ensure validity of submitted queries, and generates first-level errors messages that are syntactic in nature using PostgreSQL engine. Error handling mechanisms is graceful and manages invalid SQL syntax (caught during parsing) or database-level execution errors. Informative messages are provided to the user in both cases.

B. User Interface

ExplainS's user interface prioritizes a seamless learning experience. This section will provide a visual tour of its key elements, highlighting features that promote intuitive interaction and effective feedback delivery.

a) Query Input Area: In ExplainS, a user is provided an exercise problem, a schema, an area to write an sql query (see Fig 3). There are also three button options. The options are *preview, submit query* and *next*. Preview allows a user to see data output of the query they have written without Gemini feedback. Submit query is what provides Gemini feedback to the user along with the output of the user generated query. Next takes a user to another exercise

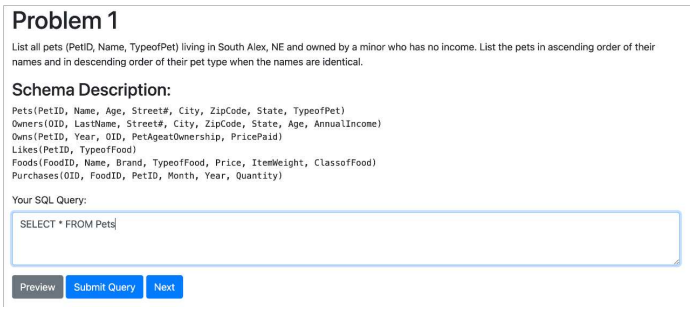

Fig. 3. ExplainS's Query Input Area.

b) Schema Visualization: ExplainS is designed with the understanding that learning is also visual. For this reason a preview button is provided, as shown in Fig 4, so a user can preview the data output of the query prior to getting feedback from Gemini to provide user's the opportunity to make adjustments to their query prior to Gemini providing the correction steps.

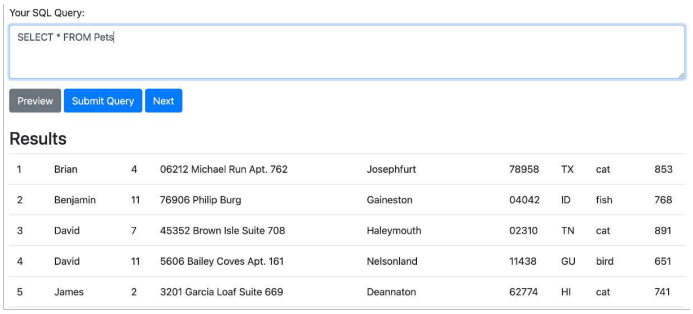

Fig. 4. Interactive Schema Visualization in ExplainS.

C. Gemini Prompt Construction

To generate the most helpful feedback, ExplainS carefully crafts prompts for Gemini. These prompts include the following components.

- User Query: The original query submitted by the user.
- AST Difference: The transformations calculated by SQLglot, highlighting discrepancies between the user's query and the solution.
- Exercise: The exercise problem the user is asked to solve.

The system works by fetching the solution query from the exercises dataset and when a user submits a query, SQLglot's AST Diff is called. The output of AST diff is fed to Gemini through a prompt, along with the user query and the exercise problem. Gemini then interprets the AST Diff and provides steps on how a user can make corrections to their query to correctly solve the exercise problem. A sample prompt is shown in Fig 5.

D. Feedback Generation

This core module leverages an LLM, Gemini, in conjunction with the SQLglot library to provide context-aware and personalized feedback. It analyzes the AST diff between a user's query and the correct solution to provide feedback on what needs to change about the user's query to be correct.

a) The Role of SQLglot: The SQLglot library is crucial for pinpointing specific differences between the user's query and the correct solution. It parses both queries into ASTs, as shown in Fig 6, and calculates the transformations needed to convert the user's AST into the solution AST. These transformations highlight the precise areas where the user's understanding might have diverged.

b) Integrating the Large Language Model: Gemini, the LLM, is used to interpret the AST differences identified by SQLglot. Based on this analysis, the module generates guidance tailored to the specific errors or misconceptions

```
# AST Generation with SqlGlot
user_ast = parse_one(user_query)
solution_ast = parse_one(solution_query)
# Calculate AST Diff
transformations = diff(user_ast, solution_ast)
# Construct a prompt based on the transformations
prompt = f"""{exercise}
User Query: {user_query}
**Task:**
Identify the specific changes needed in the User Query to make it produce the exact same output
as the Solution Query.  Analyze the User Query, Solution Query, and the provided AST Diff to
pinpoint these required modifications.
then provide guidance to the user on how they need to adjust their query to match the solution
**Output Format:**
Provide clear instructions on what the user query needs to match the solution query and the exercise,
in a step-by-step format.
Only provied correction steps no AS diff output. Do Not provde output of transformations.
Do Not provide the solution query only the steps needed to match the solution.
Only provide the changes needed to match the solution query. Try to not repeat yourself.
AST Diff: {transformations}"""



response = model.generate_content(prompt)

# Process Gemini's response
gemini_feedback = response.text

# Integrate Gemini's feedback
feedback = f"Your query has some differences from the expected solution. Here are some suggestions: {gemini_feedback}"
```

Fig. 5. Example of a Prompt Structure Sent to Gemini.

AST Diff

SQLGlot can calculate the semantic difference between two expressions and output changes in a form of a sequence of actions needed to transform a source expression into a target one:

```
from sqlglot import diff, parse_one
diff(parse_one("SELECT a + b, c, d"), parse_one("SELECT c, a - b, d"))
```

```
[
  Remove(expression=Add(
    this=Column(
      this=Identifier(this=a, quoted=False)),
    expression=Column(
      this=Identifier(this=b, quoted=False)))),
  Insert(expression=Sub(
    this=Column(
      this=Identifier(this=a, quoted=False)),
    expression=Column(
      this=Identifier(this=b, quoted=False)))),
  Keep(
    source=Column(this=Identifier(this=a, quoted=False)),
    target=Column(this=Identifier(this=a, quoted=False))),
  ...
]
```

Fig. 6. SQLglot AST Diff Example.

present in the user's query. This feedback focuses not only on correcting the syntax but also on improving the user's conceptual understanding of SQL constructs.

When a user submits the query, Gemini reads the SQLglot AST Diff and identifies what changes are needed for the user submitted query to match the solution query and provides correction steps (e.g., Fig 7).

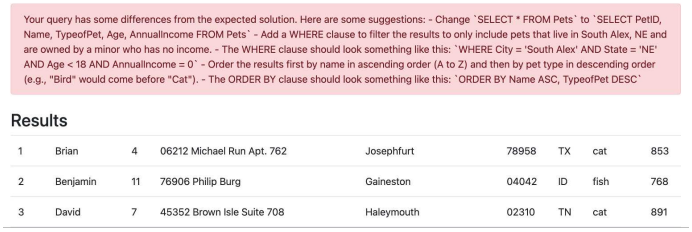

Fig. 7. Gemini Feedback

.

E. Feedback Panel

The Feedback panel appears in red, as shown in Fig 7, and provides correction steps to the user on how to fix their query to match the solution query. If the user query is correct, as shown in Fig 8, and matches the solution query's AST then feedback saying the user query matches the solution in green.

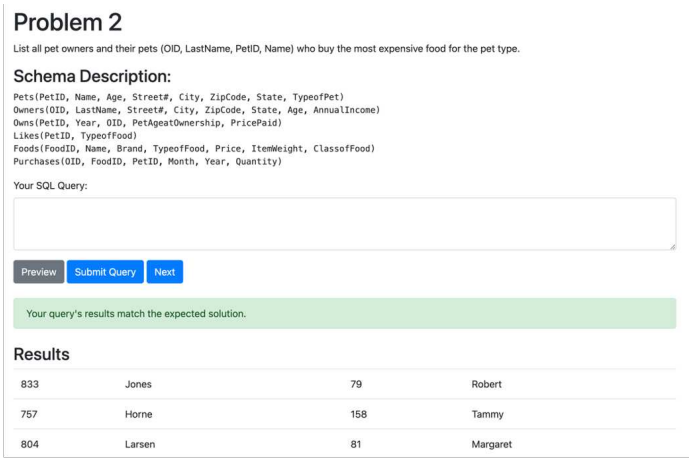

Fig. 8. Example of ExplainS's Feedback Panel for a Correct Query.

V. Current Functionalities

ExplainS offers the following core features to support a focused and interactive SQL learning experience:

A. Guided Practice:

A structured exercise set introduces fundamental SQL concepts (SELECT, WHERE, aggregation, joins, etc.), gradually increasing in complexity. This measured progression fosters a strong foundation and facilitates the development of problem-solving skills.

1) Dynamic Schema Display: An interactive visualization of the relevant database structure provides context for query formulation. By clearly presenting table relationships and attributes, this feature reduces guesswork and empowers learners to build queries with confidence.

2) Personalized Feedback: Leveraging Gemini, ExplainS analyzes submitted queries and compares them to solution queries. AST transformations are pinpointed, and the feedback mechanism generates tailored correction steps. This guidance promotes error understanding and helps users bridge the gap between their attempts and correct solutions.

3) Query Previewing: The ability to preview query results before submission empowers learners to experiment and iteratively refine their solutions. This immediate feedback loop reinforces understanding and encourages a self-directed learning approach

B. Exercise Navigation

Users can navigate through a curated series of beginner-level exercises. This structured pathway provides a well-defined starting point and a sense of achievable progress within the learning journey.

VI. Discussion

A. Novelty and Impact

ExplainS distinguishes itself from traditional SQL learning tools in several aspects:

1) Dynamic Schema Visualization: ExplainS's dynamic schema visualization transcends the limitations of static examples. Rather than merely observing a diagram, learners actively engage with the data model. Below are key aspects that set this visualization apart:

a) Beyond Static: Static examples often present a single, isolated view of a database. In contrast, ExplainS's dynamic visualization empowers users to interactively explore relationships (through filtering, utilizing joins, cte's, window functions, etc.), developing a comprehensive understanding of the data model's structure.

b) Query Formulation as Discovery: The visualization becomes a discovery tool, enabling learners to identify valid joins, relevant columns, and reduce guesswork in query construction. This hands-on approach promotes confidence in query formulation.

c) Conceptual Understanding: A strong mental model of the schema is crucial for writing efficient and accurate SQL. ExplainS's visualization facilitates the development of this mental model, fostering long-term SQL proficiency.

2) Personalized Feedback: ExplainS utilizes an LLM, like Gemini, to deliver personalized feedback that goes far beyond the capabilities of traditional error messages. Here's how this integration creates a more impactful learning experience:

a) Pinpointing the Root Cause: Generic errors often simply state that something is wrong ("Incorrect syntax near X"). In contrast, LLM-powered feedback leverages SQLglot's AST diff analysis to identify the precise conceptual or syntactic source of the error (misconceived join, improper aggregation, etc.).

b) Guiding Towards Resolution: Instead of merely pointing out a problem, ExplainS's feedback aims to guide the learner step-by-step towards the solution. This guidance could involve breaking down a complex transformation, suggesting alternative SQL techniques, or providing hints grounded in the error's context.

c) Addressing Misconceptions: Over time, LLM feedback can be tailored to address recurring patterns in user errors. This allows the system to provide explanations that target common SQL misconceptions, actively promoting the development of strong conceptual understanding.

d) Adapting to the Learner: LLMs can be fine-tuned to match a user's current skill level. Feedback could adjust its complexity, whether providing high-level guidance for beginners or more nuanced explanations for those with some experience.

B. The Ideal Vision

ExplainS envisions a future where Gemini acts as a true SQL coach. A tiered feedback system would be implemented alongside a grading mechanism. On initial incorrect submissions, the feedback would prioritize guidance with escalating specificity:

1) Attempt 1: High-level hints ("Check your joins," "Re-examine your WHERE clause").

2) Attempt 2: More targeted advice ("You're not joining correctly, check the table...," "Your WHERE clause has too many/too few filters.").

3) Attempt 3: If still unresolved, provide the correct solution with an explanation of the changes made.

C. Focus on Guided Practice

ExplainS adopts a structured approach to practice, recognizing its paramount importance in the SQL learning journey. Contrasting with the often overwhelming nature of unstructured resources, guided exercises provide a framework that promotes mastery by fostering a logical progression of skills and a deep understanding of core SQL concepts.

Exercises are meticulously designed to span three distinct difficulty levels:

1) Beginner: Introduces foundational concepts (SELECT, FROM, WHERE, simple aggregations, basic joins). Exercises focus on clear problem statements and emphasize building core query-writing mechanics.

2) Intermediate: Assumes fluency with the basics and introduces more complex scenarios. Exercises incorporate multiple joins, nested queries, advanced aggregations, and the use of window functions. The focus shifts towards problem-solving and applying SQL constructs strategically.

3) Advanced: Challenges learners with real-world-inspired, open-ended problems, demanding the synthesis of various techniques. These exercises promote solution optimization, nuanced SQL reasoning, and encourage exploration beyond the explicitly taught concepts.

D. Tiered Exercise Structure

This tiered exercise structure fosters true mastery for several reasons:

1) Scaffolding Knowledge: By starting with the fundamentals and gradually increasing complexity, learners build upon a solid base. This prevents gaps in understanding that might hinder the application of advanced concepts later.

2) Targeted Practice: Each difficulty level reinforces specific skills. Exercises are crafted to expose common mistakes and misconceptions, allowing learners to address them directly.

3) Conceptual Solidification: Progressing through exercises translates to working with increasingly elaborate datasets and data relationships. This constant interaction reinforces the connection between SQL syntax and the underlying logic of data manipulation.

4) Motivation and Success: Clearly defined levels, coupled with achievable challenges within each tier provide learners with a sense of accomplishment and the drive to advance further.

E. Planned Future Developments

ExplainS envisions an evolution into a comprehensive SQL tutoring system. Key development goals include:

1) Advanced Feedback Mechanism: By training a specialized LLM on vast datasets of AST differences and queries, ExplainS aims to provide nuanced and explanatory feedback. This transition from simple correction steps to "tutoring-like" guidance would significantly enhance the platform's value for learners.

2) Automated Grading System: A robust automated grading system would streamline assessment and deliver rapid performance feedback. This feature would not only benefit learners by providing instant evaluation but also scale ExplainS's utility in classroom or self-learning settings.

3) Expanded Exercise Dataset: ExplainS aspires to offer a diverse range of exercises covering a wider spectrum of difficulty levels. Introducing more complex schemas and problem scenarios would cater to learners at various stages and keep the platform engaging.

4) LLM-Driven Solution Generation: Fine-tuning an LLM to independently generate accurate solution queries would transform the learning experience. Users and the AI could collaborate on solutions, fostering both critical thinking in learners and potentially refining the LLM's own SQL proficiency.

VII. Conclusion

ExplainS addresses the limitations of traditional SQL learning practices, especially feedback generation, by providing a dynamic and engaging environment for guided practice. Key innovations that distinguish ExplainS include interactive schema visualization, personalized feedback, and structured and progressive difficulty leveled based exercise. The current edition of ExplainS includes our initial design ideas of explainable SQL, semantic error trapping and feedback generation that we believe are more effective compared to contemporary

approaches. While several systems used AST and LLM for tutoring systems individually, we made them work together for an improved tutoring system.

ExplainS envisions a continued evolution into a comprehensive SQL tutoring system. As discussed earlier, future development goals include enhanced LLM feedback, automated grading, and inclusion of expanded learning resources. One of the most important enhancements will be development of an algorithm to test semantic equivalence of AST trees that is not addressed in this paper. Once available, automatic grading will be fairly accurate. We believe ExplainS strives to empower learners of all backgrounds to achieve SQL mastery. It holds the potential to change how SQL is taught, making the learning process more effective, engaging, and accessible.

ACKNOWLEDGEMENT

This Research was supported in part by a National Institutes of Health IDeA grant P20GM103408, a National Science Foundation CSSI grant OAC 2410668, and a US Department of Energy grant DE-0011014.

REFERENCES

[1] N. P. Bakas, M. Papadaki, E. Vagianou, I. T. Christou, and S. A. Chatzichristofis. Integrating llms in higher education, through interactive problem solving and tutoring: Algorithmic approach and use cases. In *EMCIS 2023, Dubai, United Arab Emirates, December 11-12, 2023, Proceedings, Part I*, volume 501 of *LNBIP*, pages 291–307. Springer, 2023.

[2] P. Bhuse, J. Jain, A. Shaju, V. John, A. Joshi, and R. Rajendran. Sqlearn: A browser based adaptive SQL learning environment. In *AIS 2021, Held as Part of the HCII 2021, Virtual Event, July 24-29, 2021*, volume 12792 of *Lecture Notes in Computer Science*, pages 139–152. Springer, 2021.

[3] I. Bider and D. Rogers. YASQLT - yet another SQL tutor - A pragmatic approach. In *ER 2016 Workshops, Gifu, Japan, November 14-17, 2016*, volume 9975 of *LNCS*, pages 197–206, 2016.

[4] N. Carr, F. R. Shawon, and H. M. Jamil. An experiment on leveraging chatgpt for online teaching and assessment of database students. In *IEEE TALE 2023, Auckland, New Zealand, November 28 - Dec. 1, 2023*, pages 1–8. IEEE, 2023.

[5] S. Chen, X. Xu, H. Zhang, and Y. Zhang. Roles of chatgpt in virtual teaching assistant and intelligent tutoring system: opportunities and challenges. In *WSSE 2023, Tokyo, Japan, September 22-24, 2023*, pages 201–206. ACM, 2023.

[6] S. Chu, C. Wang, K. Weitz, and A. Cheung. Cosette: An automated prover for SQL. In *CIDR 2017, Chaminade, CA, USA, January 8-11, 2017*. www.cidrdb.org, 2017.

[7] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, L. Chen, J. Lin, and D. Lou. C3: zero-shot text-to-sql with chatgpt. *CoRR*, abs/2307.07306, 2023.

[8] M. Fabijanic, G. Dambic, and J. Sasunic. Automatic, configurable, and partial assessment of student SQL queries with subqueries. In *MIPRO 2022, Opatija, Croatia, May 23-27, 2022*, pages 542–547. IEEE, 2022.

[9] S. Ganesan, T. Gong, and J. Lee. Sqlearn: Automated SQL statement assessment using structure-based analysis. In *SIGCSE 2024, Volume 2, Portland, OR, USA, March 20-23, 2024*, pages 1644–1645. ACM, 2024.

[10] A. Gilad, Z. Miao, S. Roy, and J. Yang. Understanding queries by conditional instances. In *SIGMOD '22, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 355–368. ACM, 2022.

[11] A. Hull and B. du Boulay. Motivational and metacognitive feedback in sql-tutor. *Comput. Sci. Educ.*, 25(2):238–256, 2015.

[12] M. Karimzadeh and H. M. Jamil. Visql: An intelligent online SQL tutoring system. In *IEEE ICALT 2022, Bucharest, Romania, July 1-4, 2022*, pages 212–213. IEEE, 2022.

[13] A. Kleerekoper and A. Schofield. SQL tester: an online SQL assessment tool and its impact. In *ACM ITiCSE 2018, Larnaca, Cyprus, July 02-04, 2018*, pages 87–92. ACM, 2018.

[14] C. Kleiner, C. Tebbe, and F. Heine. Automated grading and tutoring of SQL statements to improve student learning. In *Koli Calling '13, Koli, Finland, November 14-17, 2013*, pages 161–168. ACM, 2013.

[15] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. Chang, F. Huang, R. Cheng, and Y. Li. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[16] Z. Miao, T. Chen, A. Bendeck, K. Day, S. Roy, and J. Yang. I-Rex: An interactive relational query explainer for SQL. *Proc. VLDB Endow.*, 13(12):2997–3000, 2020.

[17] Z. Miao, S. Roy, and J. Yang. Explaining wrong queries using small examples. In *SIGMOD 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 503–520, 2019.

[18] B. D. Nye, D. Mee, and M. G. Core. Generative large language models for dialog-based tutoring: An early consideration of opportunities and concerns. In *AIED Workshop 2023, Tokyo, Japan, July 7, 2023*, volume 3487 of *CEUR Workshop Proceedings*, pages 78–88. CEUR-WS.org, 2023.

[19] J. R. Prior. Assesql: an online, browser-based SQL skills assessment tool. In *ITiCSE '14, Uppsala, Sweden, June 23-25, 2014*, page 327. ACM, 2014.

[20] J. Rajala, J. Hukkanen, M. Hartikainen, and P. Niemelä. "Call me Kiran" - ChatGPT as a tutoring chatbot in a computer science course". In *Mindtrek 2023, Tampere, Finland, October 3-6, 2023*, pages 83–94. ACM, 2023.

[21] S. A. Reid, F. Kammer, J. Kunz, T. Pellekoorne, M. Siepermann, and J. Wölfer. Itssql: Intelligent tutoring system for SQL. *CoRR*, abs/2311.10730, 2023.

[22] C. F. Reilly. Experience with active learning and formative feedback for a SQL unit. In *IEEE FIE 2018, San Jose, CA, USA, October 3-6, 2018*, pages 1–9. IEEE, 2018.

[23] U. Röhm, L. Brent, T. Dawborn, and B. Jeffries. SQL for data scientists: Designing SQL tutorials for scalable online teaching. *Proc. VLDB Endow.*, 13(12):2989–2992, 2020.

[24] S. Sun, Y. Zhang, J. Yan, Y. Gao, D. Ong, B. Chen, and J. Su. Battle of the large language models: Dolly vs llama vs vicuna vs guanaco vs bard vs chatgpt - A text-to-sql parsing comparison. In *EMNLP 2023, Singapore, December 6-10, 2023*, pages 11225–11238, 2023.

[25] F. Tahir, A. Mitrovic, and V. Sotardi. Investigating the causal relationships between badges and learning outcomes in sql-tutor. *Res. Pract. Technol. Enhanc. Learn.*, 17(1):7, 2022.

[26] I. Vagin, O. Havrylenko, J. P. M. Bastida, and A. Chukhray. Computer intelligent tutoring system "sqltor". In *ICT in Education, Research and Industrial Applications. Kherson, Ukraine, June 12-15, 2019*, volume 2387 of *CEUR*, pages 525–530, 2019.

[27] G. Yadav, Y. Tseng, and X. Ni. Contextualizing problems to student interests at scale in intelligent tutoring system using large language models. In *AIED 2023 Workshops, Tokyo, Japan, July 7, 2023*, volume 3487 of *CEUR*, pages 17–25, 2023.

[28] X. Zhang, K. Khedri, and R. Rawassizadeh. Can llms substitute sql? comparing resource utilization of querying llms versus traditional relational databases. *CoRR*, abs/2404.08727, 2024.

[29] F. Zhao, L. Lim, I. Ahmad, D. Agrawal, and A. E. Abbadi. Llm-sql-solver: Can llms determine SQL equivalence? *CoRR*, abs/2312.10321, 2023.