Analyzing Common Student Errors in SQL Query Formulation to Enhance Learning Support

Discussion Paper

Davide Ponzini[1,2,*,†], Abdolhamid Livani[1,†], Giovanna Guerrini[1,†], Barbara Catania[1] and Mauro Coccoli[1]

[1]University of Genoa, Dipartimento di Informatica, Bioingegneria, Robotica e Ingeneria dei Sistemi, via Dodecaneso 35, Genoa, 16145, Italy

[2]University of Genoa, Dipartimento di Lingue e Culture Moderne, piazza Santa Sabina 2, Genoa, 16124, Italy

Abstract

Learning SQL is challenging for many undergraduate students, leading to recurring errors during query formulation. This paper systematically analyzes common SQL errors made by undergraduate students enrolled in Database courses at the University of Genoa during the 2023-24 academic year. Employing a comprehensive taxonomy, we evaluated 561 student queries collected from written exams, laboratory assignments, and unsupervised contexts. Results reveal that syntax errors are the most prevalent, especially in exam settings where queries cannot be executed. Logical errors and complications were also frequently identified, underscoring issues not only with syntax but also with logical reasoning and optimization. These insights highlight the need for pedagogical strategies that balance syntactic mastery and logical problem-solving skills. The paper concludes by proposing directions for enhancing learning support through automated error detection and personalized, AI-driven tutoring tools.

Keywords

Data Education, learning SQL, SQL misconceptions, educational data analysis, computer-assisted learning.

1. Introduction

In tertiary education, relational databases are an integral part of many undergraduate and postgraduate degree programmes in computer science, computer engineering, data science, business informatics and related subjects. The practical use of the Structured Query Language (SQL) is part of introductory relational database courses in such programmes, as SQL is the standard language for manipulating relational databases. Thus, students are asked to produce simple queries and, in some cases, to develop more significant projects manipulating data in SQL.

Despite its importance, many students find SQL difficult, and the queries they produce contain repeated errors that reflect common misunderstandings. These difficulties include grasping the ideas of relational databases, understanding difficult query structures, and applying theoretical ideas to real-world problems. One of the main barriers to learning SQL is the transfer of knowledge acquired in other contexts, such as mathematics and programming [1]. Students tend to apply familiar thinking patterns and language structures to SQL, and errors can also result from previous experience with other programming languages. In addition to syntax errors, errors can also occur at the semantic level, revealing an incomplete mental model of SQL [1]. In some cases, students see the database management system as a *black box* without understanding its underlying principles. Typical concepts that are prone to error are `JOIN` clauses, `GROUP BY`, sub-queries and self-joins [1, 2, 3, 4].

SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16–19, 2025, Ischia, Italy

*Corresponding author.

†These authors contributed equally.

davide.ponzini@edu.unige.it (D. Ponzini); s5437973@studenti.unige.it (A. Livani); giovanna.guerrini@unige.it (G. Guerrini); barbara.catania@unige.it (B. Catania); mauro.coccoli@unige.it (M. Coccoli)

0009-0006-4282-9652 (D. Ponzini); 0000-0001-9125-9867 (G. Guerrini); 0000-0002-6443-169X0 (B. Catania); 0000-0001-5802-138X (M. Coccoli)

Since the development of targeted teaching tactics relies a deep analysis and understanding of common mistakes, some approaches have been proposed in the literature aiming on one side at categorizing common errors in taxonomies [5] and on the other at identifying the most common misconceptions [1].

In this paper, we analyze the mistakes that our students make when learning SQL and categorize them according to the most comprehensive established taxonomy [5]. The reference target consists of second year Bachelor students enrolled in database courses in the Computer Science and Computer Engineering Bachelor Degrees at the University of Genova in a.y. 2023-24. The research seeks trends in errors made by students, to gain insights about the areas that can be boosted to improve students learning. The classification involved 561 queries collected from 27 different query demands, provided by a total of 81 students/teams, formulated in different settings. Analysis of errors across different contexts and student groups highlights common difficulties in query construction. Syntax errors are the most common, accounting for almost half of all errors, in settings in which the students cannot execute the queries, highlighting the need to reinforce syntax rules and debugging skills. Logical errors are also common, suggesting that students struggle not only with syntax, but also with query logic, optimization, and interpreting the constraints in the request.

Although the numbers of the study are significant and allow for meaningful initial insights, achieving scalability and improved reliability would require automated support. The paper also discusses how the classification can be exploited and incorporated in other tools currently under development for assisting and enhancing student learning.

The remainder of the paper is organized as follows. After discussing related work in Section 2, the methodology adopted for the study is presented in Section 3. Results are presented and discussed in Sections 4 and 5, respectively. The limitations and ongoing directions to enhance learning support are discussed in Section 6 while Section 7 concludes.

2. Related Work

In this section, we first review the most relevant approaches related to the analysis of typical errors and misconceptions and then briefly survey tools providing automatic feedback or proposing personalized exercises and learning paths.

2.1. Analysis of SQL Query Errors

Learning SQL presents difficulties for students, both in terms of syntax and semantics [6]. Establishing a robust framework for categorizing these errors is essential in order to address them by proposing targeted material and exercises. Cagliero et al. [7] highlight the importance of providing specific feedback based on a thorough assessment of errors to address these issues and support a data-driven approach to learning SQL. They highlight the importance of providing specific feedback based on a thorough assessment of errors to address these issues and support a data-driven approach to learning SQL. By identifying these errors, teachers can create more successful lesson plans and resources that target specific student misconceptions. Ahadi et al. [8] classify common SQL errors into seven distinct categories, providing a structured framework for identifying and correcting specific errors. The taxonomy proposed by Taipalus et al. in [5] is quite comprehensive and is adopted in our study. Other approaches that focus on categorizing errors and finding patterns of errors have been proposed by Poulsen et al. [9] to guide the development of educational materials and [4] on the provision of repair strategies.

From syntactic errors to more advanced semantic and logical misinterpretations, misconceptions in SQL query development can take many forms. Understanding the thinking aspects of learning SQL is key to understanding why students make mistakes so often. A significant amount of work is therefore focused on trying to understand the causes of problems in formulating SQL queries. Investigating the causes of SQL query formulation errors is the focus of [3]. Many of these errors, they discover, result from ignorance of the basics and a poor understanding of SQL syntax and semantics. They find that even small errors can lead to major misinterpretations. Their studies highlight the importance of a

strong foundational knowledge base in both avoiding and correcting these errors. In the same vein, Green and Petre [10] contrast procedural and non-procedural query languages. They emphasize that SQL, as a non-procedural language, requires a different cognitive approach, which may be particularly difficult for students familiar with procedural programming languages [11]. A study focusing on typical errors in (simple) queries by K-12 students who are not programmers is presented in [12].

In line with this, Miedema et al. [1] conducted think-aloud research to highlight typical beginner mistakes and they collected and analyzed the hypotheses of SQL experts about the causes of student errors [13]. One of the main obstacles in learning SQL is the transfer of knowledge acquired in other contexts, such as mathematics and programming [1]. Students tend to apply familiar thought patterns and language structures to SQL, generating misconceptions. For example, the presence of keywords in SQL that resemble natural language can lead to incorrect linguistic transfers and syntactic errors. Experience with other programming languages can also cause misconceptions (such as the use of the '==' operator instead of '=' in SQL queries, students misuse it due to familiarity with other languages) [1]. Other examples of misconceptions among novice students include the inappropriate use of auto-joins to retrieve pairs of records from the same table, without realizing the need to reference the table twice in the query, revealing an incomplete mental model of SQL [1]. In some cases, students see the database management system as a 'black box', without understanding its underlying principles. In particular, the literature shows that sub-queries, GROUP BY clauses and JOIN clauses are the main areas where errors and misconceptions most frequently occur [2, 1, 4].With regard to the GROUP BY clause, there are problems of placement within the query, as well as confusion as to when it is needed. Finally, other misconceptions emerging from the literature include the misconception that the DISTINCT clause selects the first tuple in the table, inappropriate use of primary keys, syntactic errors such as remembering only part of a keyword (e.g., GROUP instead of GROUP BY) or using synonyms interchangeably (e.g., SUM with COUNT, SORT with ORDER BY) [1, 3].

2.2. Automatic Feedbacks to SQL Queries and Personalization

Several approaches and prototypes have been designed to provide automatic feedback and correction for SQL teaching and learning. The tools aim to enhance learning outcomes by identifying errors, offering constructive feedback, and supporting self-guided improvement [14]. Many approaches target automatic correction and grading of SQL queries. The support ranges from the integration of CodeRunner in a Learning Management System like Moodle for providing immediate feedback by executing the query formulated by the student [15], to automatic grading systems [16, 17] and automatic correction based on Large Language Models [18].

The full potential of such tools in enhancing learning can only be leveraged, if they extend beyond grading efficiency by also providing tutoring capabilities to the students. Hint generation is, for instance, the focus of [19, 20]. In [21] fine-tuned GPT models to detect and provide feedback on semantic errors in SQL queries. Another possible use of generative AI explored in [22] is for the automatic exercise generation.

3. Methodology

In this study, a systematic analysis of SQL query errors produced by students is carried out. The student-generated SQL queries were collected from the following sources:

- Written exams from the Databases courses in Computer Science and Computer Engineering. In this context, students were not allowed to execute any queries or use any tools to provide support.
- Laboratory assignments specifically designed for the Database course in Computer Science. In this setting, students had the opportunity to execute queries, observe the corresponding results, and compare them with the expected answers, thus allowing for iterative refinement.
- The queries described in Section 4 of [23], which took place in an unsupervised environment wherein students autonomously decide whether or not to execute their queries. However, in this

	Computer Engineering Exam	Computer Science Exam	Computer Science Laboratories	Miedema Thesis [23]
Student Group (Degree)	Computer Engineering	Computer Science	Computer Science	Computer Science
Number of Queries	3	8	9	7
Number of Subjects (Students/Teams)	12	25	23	21
Language	Italian	Italian	Italian	English
Data Collection Method	Written Exam	Written Exam	Online Submission	Online Submission
Answer Time	Limited	Limited	Not Limited	Not Limited
Sample Data Provided	No	No	Yes	Yes
Work in Teams	No	No	Yes	No*
Possibility of executing the Queries	No	No	Yes	Yes
Expected Results Available	No	No	Yes	No

* The queries were formulated in an unsupervised context, precluding the possibility of determining whether students worked alone.

Table 1
Query collection sources and modalities

setting, no reference results were provided to students for comparison.

Each collected query was subjected to a manual review process, which entailed testing the query execution against the corresponding database to ascertain its correctness and identify any errors. Queries were evaluated based on their execution validity, the accuracy of their results, and their alignment with the requests outlined in the query specification. The manual analysis entailed evaluating each query for specific errors by comparing the student queries against the expected correct solutions.

Errors were identified and categorized following the established and comprehensive framework by Taipalus [5], distinguishing four main categories:

- **Syntax errors:** queries that are not executable due to incorrect SQL grammar.
- **Semantic errors:** queries that can be executed but are incorrect, irrespective of the specific query request.
- **Logical errors:** queries that can be executed but are not aligned with the specific query request.
- **Complications:** queries that accurately align with the specified query request, yet exhibit unnecessary complexity.

Additionally, queries were categorized based on their complexity: **simple** queries involve basic selection, filtering, and limited joins; **medium complexity** queries introduce aggregation, sorting, or conditional logic; **hard** queries involve advanced SQL features such as complex joins, nested conditions, and multi-table aggregations.

This structured approach allowed us to systematically record, categorize, and analyze errors, facilitating detailed statistical analysis and deeper insights into the error patterns and common misconceptions encountered by students. For a more thorough exposition on the modalities of data collection, refer to Table 1. Further details are presented in [24].

4. Results

Our analysis identified a total of 723 SQL errors across 561 queries. Specifically, the Database Course Exam for Computer Engineering students contained 82 errors in 29 queries, while 196 errors were identified in 183 queries from the Database Course Exam for Computer Science students. The Database

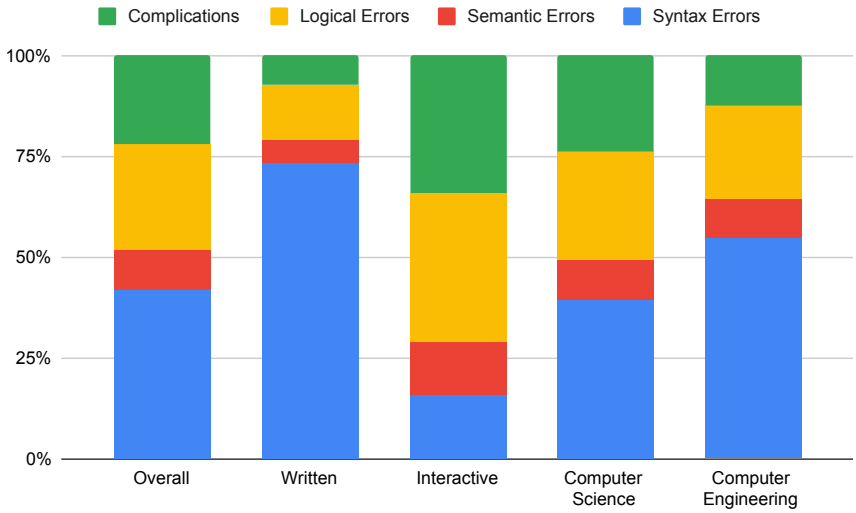

Figure 1: Overall error distribution and distributions by context, comparing different modalities and student groups.

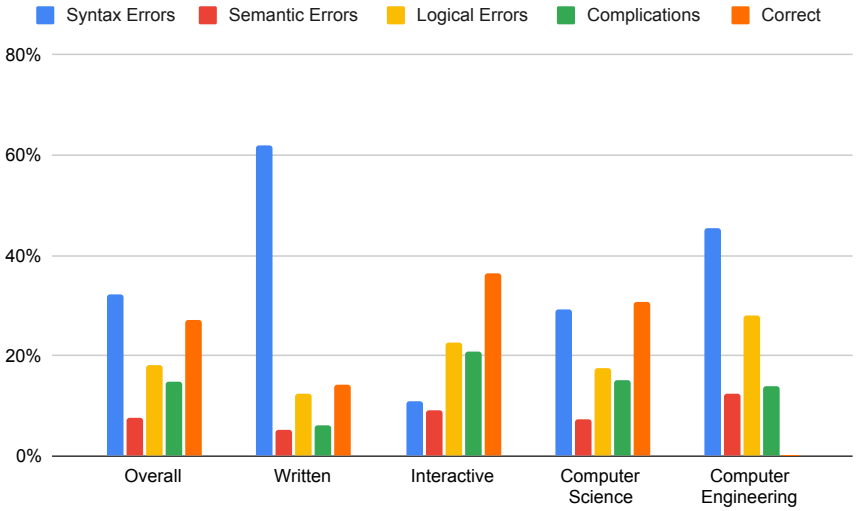

Figure 2: Error type distributions by query. Multiple errors of the same type in a single query are only counted once. Note that the percentages can add up to more than 100%, since the same query can present multiple types of error.

Laboratory for Computer Science students accounted for 262 errors in 209 queries, and the Miedema dataset included 183 errors in 140 queries.

The analysis revealed that 66% of the queries presented at least one error. Of these errors, 42% were classified as Syntax Errors, 10% as Semantic Errors, 26% as Logical Errors, and 22% as Complications. Figure 1 shows the error distributions overall and by each context, comparing the different modalities and student groups. Overall, syntax errors are the most prevalent type of error, followed by logical errors and complications. Semantic errors are the least prevalent type of error. The most common errors are referencing non-existing objects, followed by the inclusion of superfluous columns in the SELECT clause or the omission of requested columns and the use of non-standard keywords or operators. A comparative analysis of written and interactive assignments reveals distinct error patterns. Written assignments, which include both Computer Engineering and Computer Science students' exams, are associated with a significantly higher prevalence of syntax errors. In contrast, interactive assignments, which consist of laboratories and the queries from the Miedema dataset, predominantly exhibit logical

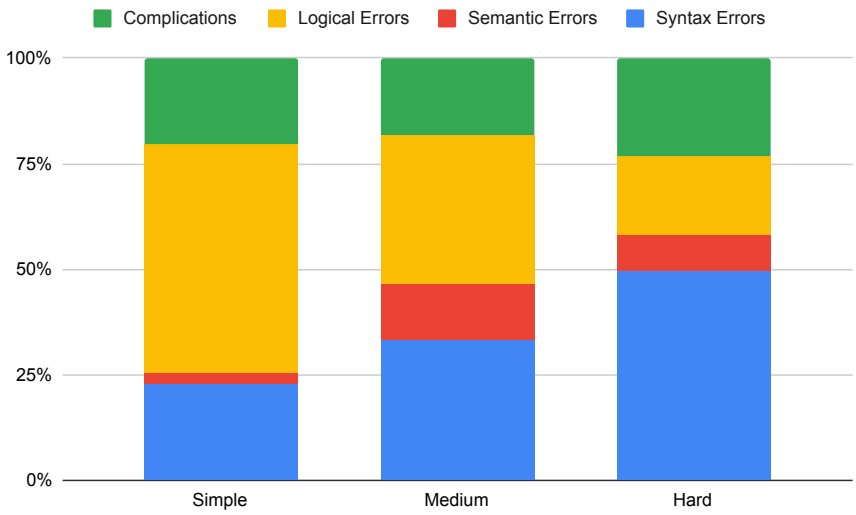

Figure 3: Error distributions by difficulty.

errors and complications. For written tests, the most common errors are referencing non-existing objects, while the most common complication is performing join operations with tables not needed for solving the request. For interactive assignments, the most common error is returning too many duplicate rows when not requested. The most prevalent complication is analogous to the written tests. A comparison of Computer Engineering and Computer Science students reveals that the former tend to make a higher frequency of syntax errors, whereas the latter tend to write overcomplicated queries more often. The most prevalent errors and complications exhibited by both student groups are analogous to the overall errors previously delineated.

As illustrated in Figure 2, the error type distribution for each query is compared across different modalities and student groups. It is important to note that a single query can exhibit multiple error types. In instances where a query exhibits multiple errors of the same type, it is considered as a single occurrence. Correct queries are defined as those that do not present any error or complication.

Overall, the majority of queries exhibit at least one syntax error. After excluding these errors, the majority of the remaining queries demonstrate no errors or complications. Logical errors are also frequent, followed by complications. The least prevalent error type is semantic errors. During written exams most queries contain syntax errors, with a mean value of 0.83 syntax errors per query. At the same time they present few semantic or logical errors. Complications are uncommon, with a mean value of 0.08 complications per query. Interactive assignments present a different distribution: most of them are completely correct, followed by logical errors and complications. A small amount of queries presents syntax errors. The least common error type is semantic errors. Syntax, logical and semantic errors are more common in Computer Engineering students when compared to Computer Science students. Both student groups tend instead to write complications with a similar frequency. Interestingly, in the data we analyzed from Computer Engineering students, we were unable to find a query that did not present any error or complication.

Figure 3 analyzes error distribution by query difficulty. A close examination of the data reveals that syntax errors are more prevalent in complex queries, while logical errors are more prevalent in simpler queries. Semantic errors appear to be more frequent in medium-complexity queries. Complications are present in all queries, regardless of their complexity.

5. Discussion

The analysis of SQL errors across various educational contexts and student groups provides significant insights into common issues students encounter when constructing database queries. The prevalence of errors identified across contexts underscores the challenges students face, particularly with syntax and logical reasoning in query construction.

Overall, syntax errors emerged as the most common type of error, representing nearly half of all errors. This highlights a fundamental issue students experience in mastering SQL syntax, suggesting a need for reinforcing syntax rules and debugging skills in teaching strategies. Logical errors and complications were also frequent, emphasizing that students not only struggle with basic syntax but also with query logic, optimization, and adherence to problem constraints.

When comparing written and interactive assignment modalities, a distinct pattern emerges. Written tests predominantly exhibited syntax errors, indicative of difficulties in recalling precise SQL syntax without interactive feedback. The high occurrence of referencing non-existent database objects suggests that students might benefit from additional practice in accurately memorizing and understanding database schemas. Conversely, interactive assignments showed fewer syntax errors but more logical errors and complications, suggesting that while interactive environments reduce syntactical mistakes, students might over-complicate solutions or misunderstand query requirements when immediate feedback is available. Therefore, teaching methodologies could integrate both written and interactive elements to balance syntax learning and logical problem-solving.

Differences between Computer Science and Computer Engineering students further enrich this analysis. Computer Engineering students produced more syntax and logical errors, possibly reflecting less familiarity or fewer practice opportunities with SQL compared to Computer Science students. Interestingly, complications were equally prevalent among both groups, suggesting a common tendency toward complexity in query design. Targeted interventions could thus be developed for Computer Engineering students focusing particularly on syntax practice and logical structuring of queries, whereas Computer Science curricula might benefit from emphasizing query simplification techniques.

Analysis by query difficulty reveals additional trends. Complex queries correlated strongly with increased syntax errors, pointing to students' struggles with managing intricate query structures. Conversely, simpler queries saw increased logical errors, indicating complacency or misunderstandings about fundamental database query logic. Semantic errors were notably higher in queries of medium complexity, perhaps because intermediate queries involve nuanced schema understanding and data relationships without the explicit complexity of more challenging problems. Educational approaches could thus differentiate between levels of query difficulty, tailoring instructions and practice exercises to student needs at each complexity level.

The presence of complications across all difficulty levels suggests a pervasive inclination toward unnecessarily complex query formulations. This underscores a pedagogical opportunity to reinforce principles of efficient and simplified query design, potentially through examples contrasting complicated and straightforward solutions.

With regard to Miedema's queries, we found only a partial correspondence with the problems highlighted in [1, 23], perhaps also due to the different settings in which the queries were formulated.

In conclusion, our findings advocate for pedagogical adjustments, including enhanced syntax training, targeted logical reasoning exercises, and focused interventions based on student backgrounds and assignment modalities. Such targeted educational strategies promise to effectively address the identified challenges, thereby improving overall proficiency in SQL query formulation among students.

6. Limitations & Ongoing Research

This study presents some limitations that must be acknowledged. The primary limitation is the restricted amount of data, particularly from Computer Engineering students, which impacts the generalizability of our findings. The dataset from Computer Engineering students was notably smaller than that from

Computer Science students, potentially skewing the analysis toward issues predominant within a single group. Additionally, the necessity of manually examining each query introduces potential for human error or bias in error categorization. Manual evaluation, while thorough, is time-intensive and not scalable to larger datasets without introducing considerable resource constraints.

Based on the findings of this study, future research should focus on the development and assessment of targeted educational interventions using innovative technologies. Currently, we are developing three distinct tools:

- **Automatic Error Identification and Categorization Tool:** This tool automatically analyzes SQL queries to detect and classify errors according to established taxonomies. This tool can be used to analyze more in depth students' weaknesses and to allow educators to rapidly pinpoint and address students' specific areas of weakness.
- **Generative AI-based Assignment Creator:** Leveraging generative AI, this tool generates tailored SQL assignments reflecting students' personal interests and targeting common error patterns. This personalized approach aims to improve student engagement and targeted skill development.
- **Interactive AI-based SQL Tutor:** An AI-driven interactive tutor designed to provide personalized guidance and immediate feedback on SQL queries. The tutor interacts dynamically with students, addressing misconceptions and facilitating a deeper understanding of SQL concepts.

These tools are anticipated to significantly enhance the precision of educational interventions, enabling educators to more effectively address specific SQL learning difficulties and misconceptions.

7. Conclusions

This study analyses common SQL errors made by undergraduate students, revealing significant challenges with syntax and logic. Syntax errors were most common in exams, while interactive assignments highlighted logical errors. Computer Engineering students had higher error rates, probably due to less exposure to SQL. These findings highlight the need for tailored teaching approaches. To address these issues, we are developing AI-driven tools for error detection, personalized practice and real-time feedback to improve SQL learning and student proficiency.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and DeepL in order to: Abstract drafting, Drafting content, Paraphrase and reword, Improve writing style, Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Miedema, E. Aivaloglou, G. Fletcher, Identifying SQL misconceptions of novices: Findings from a think-aloud study, ACM Inroads 13 (2022) 52–65.

[2] Semantic errors in SQL queries: A quite complete list, Journal of Systems and Software 79 (2006) 630–644.

[3] T. Taipalus, Explaining causes behind SQL query formulation errors, in: 2020 IEEE Frontiers in Education Conference (FIE), IEEE, 2020, pp. 1–9.

[4] K. Presler-Marshall, S. Heckman, K. Stolee, SQLRepair: Identifying and repairing mistakes in student-authored SQL queries, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), IEEE, 2021, pp. 199–210.

[5] T. Taipalus, M. Siponen, T. Vartiainen, Errors and complications in SQL query formulation, ACM Transactions on Computing Education (TOCE) 18 (2018) 1–29.

[6] P. Garner, J. Mariani, Learning SQL in steps, Learning 12 (2015) 23.
[7] L. Cagliero, L. De Russis, L. Farinetti, T. Montanaro, Improving the effectiveness of SQL learning practice: a data-driven approach, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), volume 1, IEEE, 2018, pp. 980–989.
[8] A. Ahadi, J. Prior, V. Behbood, R. Lister, Students' semantic mistakes in writing seven different types of SQL queries, in: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, 2016, pp. 272–277.
[9] S. Poulsen, L. Butler, A. Alawini, G. L. Herman, Insights from student solutions to SQL homework problems, in: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, 2020, pp. 404–410.
[10] C. Welty, D. W. Stemple, Human factors comparison of a procedural and a nonprocedural query language, ACM Transactions on Database Systems (TODS) 6 (1981) 626–649.
[11] H. Al-Shuaily, SQL pattern design, development & evaluation of its efficacy, Ph.D. thesis, University of Glasgow, 2013.
[12] D. Traversaro, Insegnare SQL a chi non ha mai programmato: Analisi delle misconcenzioni, in: ITADINFO Secondo Convegno Italiano sulla Didattica dell'Informatica (In Italian), 2024.
[13] D. Miedema, G. Fletcher, E. Aivaloglou, Expert Perspectives on Student Errors in SQL, ACM Trans. Comput. Educ. 23 (2022).
[14] C. Kenny, C. Pahl, Automated tutoring for a database skills training environment, in: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, 2005, pp. 58–62.
[15] A. Wójtowicz, M. Prill, Relational Database Courses with CodeRunner in Moodle: Extending SQL Programming Assignments to Client-Server Database Engines, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, 2025, pp. 1239–1245.
[16] K. Manikani, R. Chapaneri, D. Shetty, D. Shah, SQL Autograder: Web-based LLM-powered Autograder for Assessment of SQL Queries, International Journal of Artificial Intelligence in Education (2025) 1–31.
[17] B. Chandra, B. Chawda, B. Kar, K. M. Reddy, S. Shah, S. Sudarshan, Data generation for testing and grading SQL queries, The VLDB Journal 24 (2015) 731–755.
[18] Z. Chen, S. Chen, M. White, R. Mooney, A. Payani, J. Srinivasa, Y. Su, H. Sun, Text-to-SQL error correction with language models of code, arXiv preprint arXiv:2305.13073 (2023).
[19] C. Kleiner, F. Heine, Enhancing Feedback Generation for Autograded SQL Statements to Improve Student Learning, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2024, Association for Computing Machinery, New York, NY, USA, 2024, p. 248–254.
[20] Y. Hu, A. Gilad, K. Stephens-Martinez, S. Roy, J. Yang, Qr-hint: Actionable hints towards correcting wrong sql queries, Proceedings of the ACM on Management of Data 2 (2024) 1–27.
[21] A. AlRabah, S. Yang, A. Alawini, Optimizing Database Query Learning: A Generative AI Approach for Semantic Error Feedback, in: ASEE Annual Conference and Exposition, Conference Proceedings, American Society for Engineering Education, 2024.
[22] W. Aerts, G. Fletcher, D. Miedema, A Feasibility Study on Automated SQL Exercise Generation with ChatGPT-3.5, in: Proceedings of the 3rd International Workshop on Data Systems Education: Bridging education practice with education research, 2024, pp. 13–19.
[23] D. E. Miedema, On learning SQL: Disentangling concepts in data systems education, ???? URL: https://pure.tue.nl/ws/portalfiles/portal/314131184/20240112_Miedema_hf.pdf.
[24] A. Livani, Do the errors produced by generative AI in formulating queries reflect students' misconceptions in learning SQL?, Master's thesis, MSc in Computer Science, University of Genova, 2024. URL: https://unire.unige.it/bitstream/handle/123456789/10623/tesi31530643.pdf?sequence=1.