LearnSQL: Impact of an Automatic Judge in Database Learning

ENRIQUE MARTIN-MARTIN, MANUEL MONTENEGRO, ADRIÁN RIESCO, and RUBÉN RUBIO, Dpto. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain
FERNANDO SÁENZ-PÉREZ, Dpto. Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain

Databases are a key topic in many technical university degrees. As databases have a strong practical nature, students are expected to solve many exercises before mastering the different aspects involved: querying and modifying the database, writing procedural code (functions and procedures), and defining triggers, among others. In this scenario, it is very important to have a substantial number of exercises available but also a timely feedback to detect and fix mistakes. Therefore, automatic judges that execute students' solutions and generate immediate feedback are valuable tools to include in the teaching practice.

In this article, we assess the real impact of using an online automatic judge for free practice in a database course over four academic years. For this purpose, we have contrasted the marks obtained in one academic year, without the automatic judge, against the three following years in which the automatic judge was used. The results show that final marks are statistically higher during the years when students make use of the automatic judge, thus showing an overall positive impact on database learning. Similarly, the results show that the more students use the automatic judge, the higher their final marks are. Besides these two insights, we have also studied if the impact of the automatic judge is the same in groups of high-profile students, concluding that this tool is less effective when improving learning in top-performing, highly self-motivated students in a database course.

CCS Concepts: • **Applied computing** → **Education**; **Computer-assisted instruction**;

Additional Key Words and Phrases: Automatic judge, database learning, SQL, evaluation

ACM Reference format:
Enrique Martin-Martin, Manuel Montenegro, Adrián Riesco, Rubén Rubio, and Fernando Sáenz-Pérez. 2025. LearnSQL: Impact of an Automatic Judge in Database Learning. *ACM Trans. Comput. Educ.* 26, 1, Article 3 (November 2025), 37 pages.
https://doi.org/10.1145/3769852

Funding support for this article was provided by the Complutense University of Madrid (INNOVA-Docencia 2020-21/18 and 2021-22/387).

Authors' Contact Information: Enrique Martin-Martin (corresponding author), Dpto. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain; e-mail: emartinm@ucm.es; Manuel Montenegro, Dpto. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain; e-mail: montenegro@fdi.ucm.es; Adrián Riesco, Dpto. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain; e-mail: ariesco@ucm.es; Rubén Rubio, Dpto. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain; e-mail: rubenrub@ucm.es; Fernando Sáenz-Pérez, Dpto. Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain; e-mail: fernan@sip.ucm.es.

This work is licensed under Creative Commons Attribution-NonCommercial-ShareAlike International 4.0.

© 2025 Copyright held by the owner/author(s).
ACM 1946-6226/2025/11-ART3
https://doi.org/10.1145/3769852

1 Introduction

Databases form a foundational aspect of numerous technical disciplines and serve as the backbone of modern information systems. Mastery of database concepts and skills is essential for students pursuing careers in fields such as computer science (see, e.g., the computer science curricula [19]), software engineering, and information technology. Among the various types of databases, *relational databases* have been the most widely utilized over the past decades. These databases are primarily managed and queried using the **Structured Query Language (SQL)**, a standardized language designed for interacting with relational data. While its syntax may initially appear similar to natural language, mastering SQL requires addressing its inherent complexities [52] and overcoming common misconceptions, which can be categorized into four distinct groups [27]: misconceptions arising from prior course knowledge, stemming from incomplete or inaccurate mental models, generalization, as well as language-based ones. SQL education has been extensively researched, as demonstrated by the mapping study conducted by Taipalus and Seppänen [54]. A key aspect emphasized in their findings is the importance of practical work in enhancing learning outcomes [21, 34]. Therefore, the acquisition of database skills involves solving a considerable number of exercises to reinforce theoretical knowledge and foster practical competence, which appear in higher levels of Bloom's taxonomy [2, 5]. However, providing timely and comprehensive feedback on student solutions presents a significant challenge for educators, particularly as class sizes grow and resource constraints become more pronounced. To address this challenge, automatic judges have proven to be effective tools for improving the learning of practical programming skills in general [35, 37] and database manipulation in particular (e.g., [31, 42, 43]). These platforms enable students to submit their solutions to database exercises, which are then automatically evaluated and provided with immediate feedback. By leveraging the capabilities of automatic judges, educators can augment the learning experience by offering students timely guidance and assessment, thereby facilitating the iterative process of skill development.

This article presents an online automatic judge for database learning, LearnSQL, and investigates the impact of its integration into the pedagogical framework of a database course over four academic years. Specifically, we empirically evaluate the efficacy of employing this automatic judge for free practice exercises and its influence on student learning outcomes. Through a comparative analysis of student performance across academic years, we aim to determine the extent to which the adoption of an automatic judge correlates with improvements in final marks. In addition, we explore how the frequency of student engagement with the automatic judge correlates with their overall academic achievement. Furthermore, this study examines whether the impact of the automatic judge remains consistent across groups of high-performing students. Ultimately, our findings aim to inform educators and curriculum designers about the potential benefits and considerations associated with integrating automated assessment mechanisms into database education.

To conduct our research on the impact of LearnSQL on database learning, we have collected evidence to answer the following three **research questions (RQ)**:

— *RQ1: Does providing an automatic judge for practicing database problems improve final marks in the ordinary exam?* Our main goal is to identify statistically relevant evidence supporting a positive impact of the availability an automatic judge for free practice in the students' learning outcomes, as obtained in the final ordinary exam, which is one of the most objective ways to measure the knowledge of the subject of each individual student. For that, we statistically compare the marks in the final ordinary exams before and after the automatic judge was introduced in the database class, whereas the remaining instructional aspects remain unchanged. This RQ assumes a binary distinction (students who had the automatic judge available vs. those who did not) without considering the intensity or frequency of its

use. RQ1 aims to provide evidence on the suitability of incorporating automatic judges into database classes.

— *RQ2: Do students who use the automatic judge more achieve higher marks in the ordinary exam?* Once the automatic judge is available, the second RQ focuses on whether using the automatic judge more intensively will lead to better learning outcomes. Although such a correlation may not imply a direct causality, it would provide more evidence about the positive impact of including an automatic judge for free practice in database classes. This RQ complements RQ1 by providing insight into whether increased engagement with the automatic judge is associated with better outcomes, i.e., how the automatic judge should be used for optimal results.

— *RQ3: Is the impact of the automatic judge dependent on students' profiles?* Research has identified distinct student profiles in Computer Science, particularly regarding motivation and strategic self-regulation [45], while other studies have distinguished between effective and ineffective students when using automatic judges in introductory programming courses [36]. Therefore, we also want to study the possible differences in the impact of the automatic judge in two groups with a highly different profile: single-degree students on Computer Science Engineering and double-degree students on Computer Science Engineering/Mathematics. Double-degree students are, in general, very self-motivated toward learning and obtain higher marks compared to single-degree students; so having additional learning tools may be less useful for them.

This article is an extension of the version [26] appeared on the SEKE 2022 international conference. The previous article presented a preliminary evaluation of the LearnSQL online automatic judge considering only the academic year 2021–2022 in the single degree, focusing only on the correlation between automatic judge usage and final marks. In the present article, we have significantly extended the evaluation of the automatic judge by expanding the number of academic years considered (2019–2020, 2021–2022, 2022–2023, and 2023–2024) and including students from the double degree on Computer Science Engineering/Mathematics. In addition to extending the study population, we have formulated more precise RQs to obtain higher-quality and more generalizable conclusions, applied more robust statistical analyses, conducted a detailed comparison with other automatic judges for database learning, and included an in-depth discussion addressing potential threats to validity and the practical implications identified throughout the development and evaluation of the automatic judge.

The rest of the article is organized as follows: Section 2 introduces LearnSQL, the automatic judge for database learning; while Section 3 presents the research design for its evaluation. Section 4 analyzes the impact of the automatic judge on students' performance across all studied scenarios and includes a qualitative examination of its usage. Section 5 presents some insights about the automatic judge extracted from a survey based on student questionnaires. In Section 6, we discuss the conclusions drawn from statistical evidence and student feedback, reflect on potential threats to validity, and outline practical implications that may be useful for educators and tool developers. Finally, Section 7 presents a detailed comparison between the automatic judge LearnSQL and other similar tools for database learning, and Section 8 concludes and presents some topics of future work.

2 The LearnSQL Automatic Judge

LearnSQL is an online automatic judge designed to practice database exercises through a user-friendly web interface. It is available as open source software[1] under the MIT license, enabling

[1]The source code of LearnSQL is available at https://github.com/emartinm/lsql.

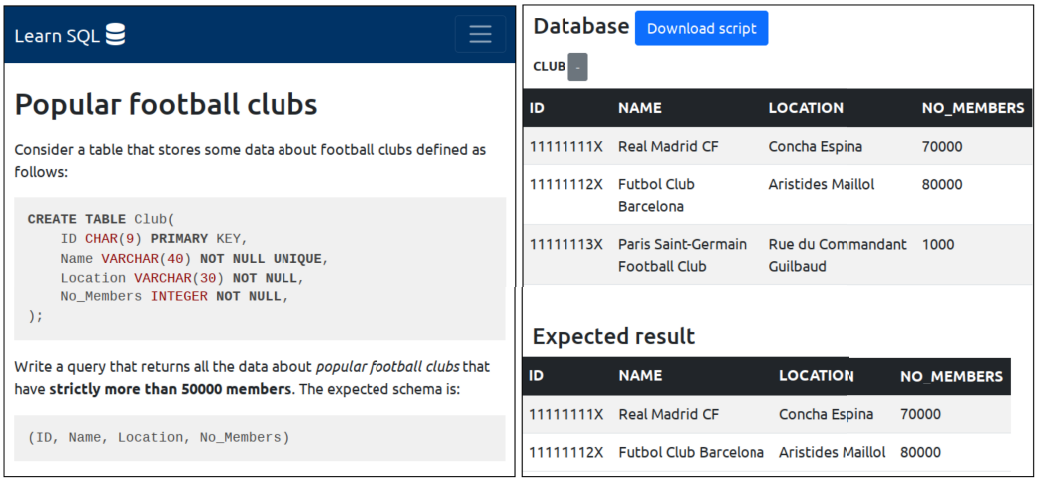

Fig. 1. Problem statement in LearnSQL (SQL query).

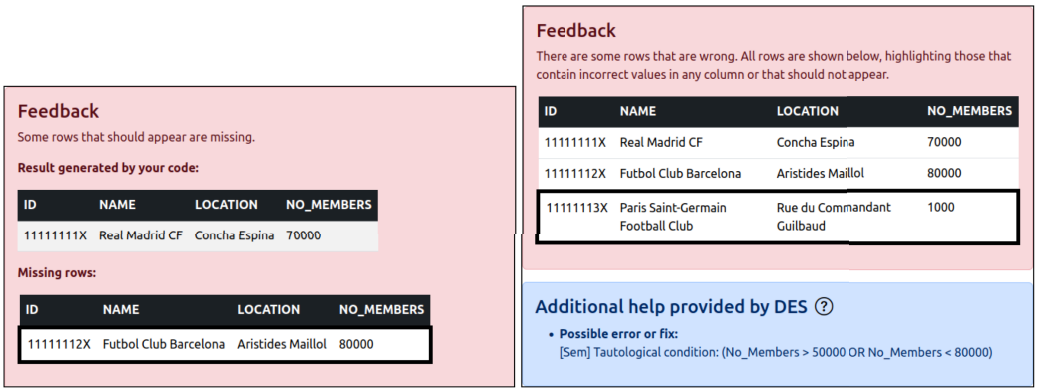

Fig. 2. LearnSQL feedback: missing rows (left) and DES additional hints (right).

instructors to deploy and customize it to suit their teaching needs. With LearnSQL, students can access collections of database problems, view problem statements, and submit their solutions directly through a text input area or uploading a text file. Figure 1 shows the statement of a problem as seen by the students: first a description of the task to solve (in this case, writing a query) followed by a sample database and the expected results of the query for that database. Students access the automatic judge using their usernames and passwords, allowing them to consult the history of all their attempts, see which problems they have solved, and explore other gamification features, such as their achievements and rankings.

A standout feature of LearnSQL is its role as an automatic judge tailored for educational purposes. When students submit their solutions, the system provides detailed feedback on any errors encountered. This feedback covers a wide range of issues, including syntax errors, schema discrepancies, and differences in expected results, as shown in the left part of Figure 2. By offering comprehensive feedback, LearnSQL aids students in understanding and rectifying their mistakes, thereby enhancing their learning experience.

LearnSQL supports different types of problems suitable for introductory database courses. The most usual kind of problem involves writing SQL queries that meet a particular requirement.

LearnSQL also supports **data manipulation language (DML)** problems, where the goal is to insert, delete, or update rows in the tables to modify a database instance. Regarding the procedural languages used in databases, LearnSQL also provides problems to define functions, procedures, and triggers. Finally, LearnSQL supports problems devoted to discriminate SQL queries. In this kind of problems, given two similar but different SQL queries, students are expected to provide a set of rows for the involved tables such that the queries produce different results. These problems challenge students to discern between similar but non-equivalent queries, promoting critical thinking and problem-solving skills. In total, LearnSQL contains around 200 database problems, where the vast majority (86%) are related to SQL queries.

To assess the correctness of student submissions, LearnSQL executes the provided code in a **database management system (DBMS)** and compares the results against expected outcomes generated by the instructor's solution. This validation process can involve multiple test cases to improve accuracy. To enhance the applicability and flexibility of the automatic judge, all the communication with the DBMS is encapsulated in the SQL execution component. Currently, we have implemented this SQL execution component for Oracle databases,[2] but similar components could be easily developed for other relational systems such as PostgreSQL, MySQL, or Microsoft SQL Server.

In addition to execution-based evaluation, LearnSQL employs static analysis of student code using the **datalog educational system (DES)** [49]. DES provides more informative messages for syntax errors and, most importantly, it performs advanced analyses to identify semantic errors [50], such as unnecessary joins, inconsistent or tautological conditions, or unnecessary subqueries, empowering students to not only correct but also refine and simplify their SQL code. For example, Figure 2 (right) shows that the condition used by the student will always return all rows despite the actual values found in the column `No_Members`, therefore indicating a tautological condition.

Furthermore, LearnSQL is designed to support multiple languages. It currently supports Spanish and English, but any other language can be easily integrated by means of string translation files. By enabling gamification elements such as achievements, podiums, and rankings, instructors can motivate students to engage with the platform actively. Achievements are earned by completing sets of problems or specific problem types and are displayed as badges in a global ranking, encouraging healthy competition among students. As discussed in Section 7, all these features differentiate our tool from currently available alternatives.

3 Research Design

In this section, we present the context of the course in which we have applied the automatic judge, explaining the participants involved in the evaluation, as well as the data acquisition procedure and the statistical tools applied.

3.1 Context of the Course

We have used LearnSQL in an introductory course on databases, which is part of the degree programs offered by the Faculty of Computer Science at the Complutense University of Madrid, a research- and teaching-focused, publicly funded university located in Madrid, Spain. The Bachelor's degree on **computer science and engineering (CSE)** (in the following) covers 4 years, being the database course taught in the first half of the second year (third semester). The double Bachelor's degree on **computer science and engineering/mathematics (CSE/M)** (in the following) covers 5 years, and the database course is taught in the first half of the third year (fifth semester). Teaching in both degrees is conducted entirely in person.

[2] Tested with the systems Oracle XE 11.2, Oracle XE 18.0, Oracle XE 21.3, and Oracle Free 23.6.

The syllabus of the introductory course on databases covers the standard contents [18, 46]: relational model, entity-relationship model, SQL queries, procedural SQL (functions, procedures, and triggers), and transactions. The teaching schedule in the introductory course on databases consists of a total of 50 hours organized in either 30 sessions of 100 minutes each with an additional break of 10 minutes, or 15 sessions of 100 minutes (with break) and 30 sessions of 50 minutes (without break), depending on the degree. Approximately 50% of the sessions are lectures, while the remaining 50% consist of practical sessions in which students solve exercises that involve performing SQL queries and defining functions, procedures, and triggers in the database. Considering all the degrees in the faculty, there were 6 different groups in this course with 30–80 students in each group.

Regarding the evaluation, 70% of the course grade was obtained in a final written exam. This exam, with a total score of 10 points, was composed of 3 parts. The first part is devoted to the design of databases for a total of 3.5 points. The second part is the main component of the exam, with exercises on SQL queries and procedural SQL up to a total of six points. This is the part of the exam we will focus on when assessing the impact of LearnSQL on students' learning (see Section 4). Finally, there is a part about transactions with a value of 0.5 points. The final exam is exactly the same in all the groups of the course, for both single and double degrees.

3.2 Academic Years and Participants

For evaluating the learning improvement, we used the LearnSQL automatic judge in the academic years 2021–2022, 2022–2023, and 2023–2024, considering the year 2019–2020 as the baseline academic year in which no automatic judge was used. Although we have also gathered data from the academic year 2020–2021, we have consciously left it out from our research because of the special circumstances in which it took place. Due to the COVID-19 pandemic, the classes during the academic year 2020–2021 followed a hybrid modality with theoretical classes delivered remotely via video call and practical classes conducted in-person in the faculty laboratories. This change of modality in the academic year 2020–2021, combined with the serious impact the pandemic had on the mood and mental health of university students [12, 25, 57], strongly casts doubt on any statistical comparison involving that academic year. Note that the database course of the year 2019–2020 was not affected by the COVID-19 pandemic because it took place from September 2019 to January 2020, so it ended a few months before the large numbers of infections and lockdowns started. The academic year 2021–2022 returned to fully in-person teaching with some safety measures, but the pandemic was fading out, and the classes took place in an environment of relative normality.

For the evaluation of LearnSQL, we have considered two of the six groups of the introductory course on databases: one group from the single-degree CSE and the only group of the double-degree CSE/M. These groups were the only ones that have had the same instructors in the span 2019–2024 and have incorporated the automatic judge in their teaching. While the student cohorts naturally varied across academic years, the syllabus, content delivery methods, and evaluation criteria remained unchanged. As such, the introduction of the LearnSQL automatic judge represents the only major pedagogical change implemented in these groups between 2019–2020 and the academic years 2021–2022, 2022–2023, and 2023–2024. We introduced the automatic judge to the students in the first weeks of the course, and we encouraged them to use it to solve the exercises of the practical classes. However, the use of LearnSQL was completely voluntary for the students: the assignments in the practical classes were free practice exercises that were not assessed and did not have any impact on the final mark of the course. In other words, all students could freely use LearnSQL as a tool to practice SQL if they considered it was useful for them.

Table 1 shows the distribution of students among the groups that were evaluated, including the proportion of women students. In the single-degree CSE we have 191 students in total, with

Table 1. Distribution of Students among Academic Years, Including the Proportion of Women Students

Degree	2019–2020		2021–2022		2022–2023		2023–2024		Total	
	Women	All	Women	All	Women	All	Women	All	Women	All
CSE	13 (19.7%)	66	5 (12.2%)	41	8 (17.8%)	45	5 (12.8%)	39	31 (16.2%)	191
CSE/M	9 (33.3%)	27	4 (16.7%)	24	6 (26.1%)	23	5 (17.9%)	28	24 (23.5%)	102
Total	22 (23.7%)	93	9 (13.8%)	65	14 (20.6%)	68	10 (14.9%)	67	55 (18.8%)	293

Table 2. Student Performance across All Subjects in the Second Year (CSE) and Third Year (CSE/M)

Degree	**Metric**	**2019–2020**	**2021–2022**	**2022–2023**	**2023–2024**
CSE	Avg. mark	6.14	5.91	6.02	5.60
	Performance rate (%)	63.06	52.26	63.56	57.11
CSE/M	Avg. mark	7.70	7.52	7.48	7.27
	Performance rate (%)	92.18	93.69	86.14	86.70

31 women students (16.2%). In the double-degree CSE/M we have 102 students, with 24 women students (23.5%). Considering both degrees, we have a population of 293 students, with an overall 18.8% of women students. Therefore, we observe that women students are a clear minority, as usual in STEM careers [4, 48].

In Table 2, we present the students' performance in all the subjects of the same year as provided by the institutional intelligence center of the university. We show the two main performance metrics for both degrees. The average mark (value from 0 to 10) combines the final mark for all the students in the group considered in the evaluation, taking into account all the subjects in the same academic year, whereas the performance rate shows the number of academic credits passed (measured as European Credit Transfer and Accumulation System) among all the academic credits enrolled in every academic year. In the CSE curriculum, there are no elective courses in the second year, and a high percentage of students in the CSE group (over 70% in most academic years) take the database course for the first time. This suggests that CSE students are enrolled in a large number of second-year courses from the program in addition to the database course under study. This homogeneity is even more evident in the CSE/M group, where the percentage of students taking the database course for the first time is much higher (over 96%), and the program does not offer elective courses until the fifth year, so we can ensure that most of the students in the CSE/M group are also enrolled in all courses corresponding to the third year of the CSE/M program. To place these data in the context of the corresponding degree programs, Table 3 shows the figures for the years immediately preceding and following the one in which the databases course is taught in each program. That is, the data for the first and third years in the case of CSE, and for the second and fourth years in the case of CSE/M. The table also includes the overall performance rates of each degree program for every academic year between 2018–2019 and 2023–2024.

The information shown in Tables 2 and 3 serves two important purposes. First, to assess the comparability of student cohorts across academic years, we examined their overall academic performance in all subjects during the second year (CSE) and third year (CSE/M). As shown in Table 2, both the average marks and performance rates (i.e., the percentage of passed courses) remain

Table 3. Student Performance Rates (%) in the Degree Years Preceding and Following the Databases Course

Degree	Year	2018–2019	2019–2020	2020–2021	2021–2022	2022–2023	2023–2024
CSE	First year	72.14	77.60	71.91	74.02	82.52	74.77
	Third year	73.47	76.04	71.67	71.09	76.35	78.79
	Overall (4 years)	71.51	75.35	72.24	74.01	77.37	76.64
CSE/M	Second year	90.54	90.12	86.59	86.39	85.20	90.40
	Fourth year	95.73	99.35	94.02	93.10	92.57	92.94
	Overall (5 years)	90.41	93.14	90.84	90.69	89.22	89.97

Table 4. Summary of Automatic Judge Usage and Exam Statistics by Year

	Single degree CSE				Double degree CSE/M			
Year	2019–2020	2021–2022	2022–2023	2023–2024	2019–2020	2021–2022	2022–2023	2023–2024
# students	66	41	45	39	26	24	21	28
Attended exam (%)	87.88	70.73	84.44	71.79	100	100	100	100
Passed exam (%)	58.62	89.66	94.74	71.43	100	100	100	100
Exam mark (10) μ	5.07	7.24	6.87	6.17	8.88	9.1	9.09	9.12
Exam mark (10) σ	1.92	1.19	1.33	1.86	0.99	0.51	0.93	0.83
SQL exercises mark (6) μ	2.69	4.19	3.81	3.49	5.43	5.43	5.42	5.67
SQL exercises mark (6) σ	1.42	0.92	0.93	1.33	0.66	0.34	0.61	0.37
# submissions		4198	7231	2263		1197	1228	2421
# problems tried μ		29.9	41.13	19.95		16.71	23.76	22.75
# problems tried σ		21.96	35.61	22.4		14.87	10.35	14.84
# abandoned problems μ		1.24	1.2	1.56		1.08	0.57	1.0
# abandoned problems σ		1.77	1.67	3.36		1.02	0.81	1.25

relatively stable over the four academic years analyzed. In CSE, the average mark ranges from 5.60 to 6.14, while the performance rate varies between 52.26% and 63.56%. Similarly, in CSE/M, the average marks fluctuate slightly between 7.27 and 7.70, and the performance rate consistently exceeds 86%, reaching up to 93.69%. Besides the year in which the databases course is taught, in Table 3 we show the absence of substantial increases in student's performance along the academic years, with the exception of first-year students in year 2022–2023. These observations suggest that the academic level of students has remained relatively consistent across cohorts, supporting the validity of our comparative analysis. Second, Table 2 highlights a consistent performance gap between the two degree programs. Students in the double-degree program (CSE/M) achieve average marks approximately 1.5 points higher than those in the single-degree program (CSE). Moreover, their performance rate indicates that they successfully complete nearly all the credits they enroll in, whereas CSE students pass between 52% and 64% of their enrolled credits, depending on the academic year. Another aspect where profile differences between the two groups of participants can be perceived is on the performance of their students in the database course. Table 4 in the next section displays some relevant performance indicators. For example, all the students from the double-degree CSE/M attend and pass the exam, whereas in the single-degree CSE several students drop the course and the percentage of students passing the exam ranges from 58.62% to 94.74%, depending on the academic year. Similarly, the average exam score is about 2.5 points (out of 10) higher in the double degree than in the single degree, and the mean score for the SQL exercises is also 2.5 points (out of 6) higher in the double degree. With an average exam mark close to 9 points

out of 10 and average SQL marks above 5 points out of 6, we can conclude that students from the double-degree CSE/M are top-performing students in the subject of databases.

3.3 Data Acquisition

In order to analyze the effect of the usage of the automatic judge LearnSQL in the student performance, we have collected data from three main sources:

- *Grade Books*: one for each group and academic year, they include the marks of every exercise of the final exam for each student registered in the course. In the analyses, we have considered the sum of the marks of the SQL exercises in the exam (6 points out of 10) and the final mark of the exam. Students that have not attended the exam have been omitted in the analyses.
- *Submission Logs from the Automatic Judge*: for each submission, they include an identifier of the student, an identifier of the problem, a timestamp, and the simplified verdict obtained from the automatic judge, among *accepted*, *wrong answer*, *run error*, and *validation error*.[3] This information is only available from the academic year 2021–2022 onward, since the automatic judge was not used until then. We have derived some descriptive statistics aggregated by student including the number of submissions, the number of problems that have been tried and solved, the mean number of attempts until a problem is solved or abandoned, and the mean time delay between the first and last attempts on a problem. Other global statistics have been obtained such as the number of submissions done during class hours.
- *Questionnaires*: to gain a better understanding of students' personal impressions of the automatic judge, at the conclusion of the 2021–2022, 2022–2023, and 2023–2024 academic years, questionnaires were distributed to students enrolled in the database course. Participation was entirely voluntary, and all responses were fully anonymous. The questionnaires included numerical rating questions addressing various aspects of the automatic judge, as well as open-ended questions that provided students with the opportunity to highlight both the tool's strengths and the areas for potential improvement.

All the data gathered for this analysis, both the grade books and submissions logs from the automatic judge, have been completely anonymized and the dataset is available at [3], along with the Python scripts used for automatizing the analyses.

For the data acquisition phase, we have followed the guidelines included in the "Code of Conduct" and the "Basic Guide to Data Protection in Research" of our institution. Questionnaires were completely anonymous by design, and regarding grade books and submission logs, they have been irreversibly anonymized to remove any trace of personal information. Moreover, at the beginning of each course, students were informed that the automatic judge was under evaluation and their usage and marks could be used, and we did not receive any requests to opt out of the study. On the other hand, all students are bound by the "Coexistence Guarantee System" of our institution, which explicitly forbids any form of academic dishonesty and outlines various levels of penalties for disciplinary offenses. In this context, and considering the in-person and supervised nature of the final exam, we believe that the data in grade books directly reflects the student's abilities. Regarding submission logs, the automatic judge was provided as a supplementary learning tool, its use was completely free, and submissions were not subject to grading. Therefore, we also consider submission logs reflect legitimate student interactions and the collected data is not affected by any kind of academic fraud (e.g., copying third-party solutions or using an AI system to solve the problems), as cheating on automatic judge submissions would not yield any benefit to the students.

[3]A submission receives this verdict when it does not fulfil the problem requirements, for example, containing too many or too few SQL statements.

3.4 Statistical Tools

For assessing the statistical significance of the findings, we rely on standard statistical instruments like correlations, hypothesis tests, and confidence intervals. First, we have used the Shapiro–Wilk test to evaluate whether the mark of the SQL exercises and the final mark of the exam are sampled from a normal distribution, and the results suggest rejecting normality (the null hypothesis) with a p-value of $5.69 \cdot 10^{-4}$ in CSE ($4.2 \cdot 10^{-11}$ in CSE/M) for the marks of the SQL exercises and $3.86 \cdot 10^{-6}$ in CSE ($2.33 \cdot 10^{-3}$ in CSE/M) for the marks in the final exam. Graphical confirmation of this fact can be obtained by observing the distribution of the marks, along the Y axis, in Figure 9. In order for the Shapiro–Wilk test not to reject the normality of the distribution, a significant amount of the sample (60%) needs to be discarded. Therefore, we have decided to limit ourselves to nonparametric methods. In particular, instead of Pearson's correlation and Student/Welch's t-test, we have used Spearman's and Kendall's τ correlation coefficients and Mann-Whitney U test. The p-values needed to evaluate the Spearman's correlation significance are calculated using permutation testing when the subsamples considered are small. Confidence intervals for mark improvement after the adoption of the automatic judge are calculated using Bayesian methods with Jeffreys' distribution as prior, since we do not have a more specific assumption about the student grades. The computed intervals are very similar to those that would have been built from the Welch's t-test in case we had assumed normality. As a complementary measure to characterize the effect size, we calculate Cohen's d coefficients [11], that is, the difference between means divided by a weighted combination of the standard deviations. Given that the data in which it is applied is not normally distributed, the classification of the effect into *small*, *medium*, and *large* should be taken with additional reserves.

Since the two groups included in this study have quite different participation and performance results, as shown evident in the summary statistics referred before, we have analyzed them separately. More specifically, the statistic technique used depends on each RQ:

- Confidence intervals are obtained for the difference of mean marks between the academic years when the automatic judge was used and those in which it was not. Hypothesis testing is also used to argue whether these improvements are statistically significant.
- Correlations are computed between the number of problems a student has tried as independent variable, and the mark obtained in the SQL exercises of the exam as dependent variable.

As mentioned in Section 3.3, students who did not attend the exam have not been considered in our analyses. Moreover, we discarded three outliers in the CSE/M group (1 in the academic year 2019–2020 and two in the academic year 2022–2023) with respect to the exam marks, namely those whose marks were more than three standard deviations away from the mean (3σ rule). No student has been removed from the CSE group under the same criterion. Outliers regarding the number of tried or solved problems were retained, as the analysis relies exclusively on order-based nonparametric methods. These methods are more robust against outliers, since the magnitude of the samples are immaterial to them except for their relative order.

4 Results

In this section, we first discuss how the automatic judge was used according to its activity logs. Combining this information with the marks of the SQL exercises of the final exam, we analyze whether there is quantitative evidence to support that the automatic judge has had a positive effect on the students' performance and learning process.

4.1 Usage of the Automatic Judge

While the net population consists of 293 students registered in the courses described in Section 3.1 along the four academic years, 102 of them from the double-degree group, we have limited our

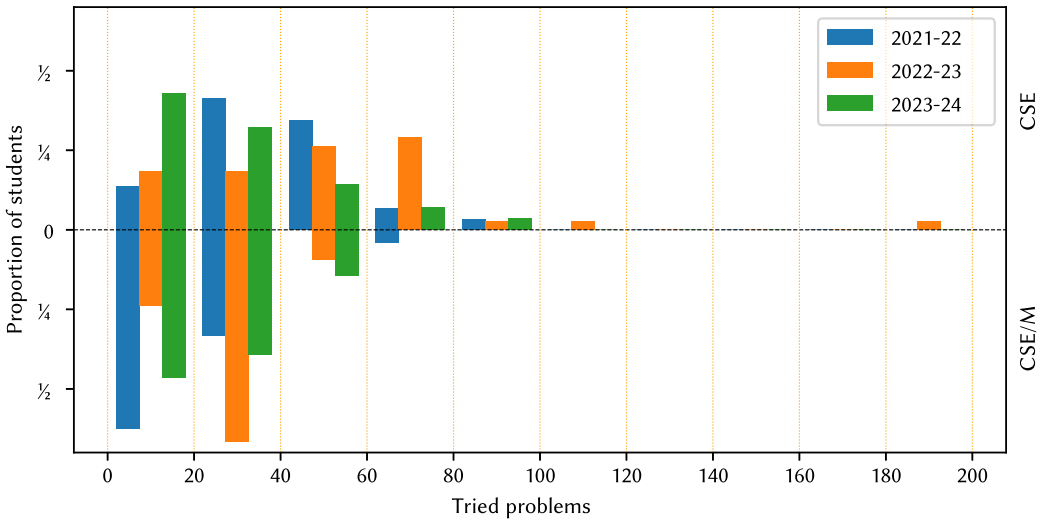

Fig. 3. Histogram: distribution of problems tried per student and academic year. Each bar between two consecutive marks a and b of the x-axis represents the ratio of CSE (upward) or CSE/M (downward) students who tried between a (included) and b (excluded) problems in one academic year.

analysis to the 255 who attended the final exam, since we need their marks for our analysis. As mentioned before, we have also discarded three students in the CSE/M group as outliers, because their SQL exercise marks were more than 3 standard deviations away from the mean mark. Two of them can be clearly identified in Figure 9, while the other one belongs to the 2019–2020 academic year. Among those who did not attend the exam since 2020, only 9 of 30 have practiced with LearnSQL, solving 19.78 problems in mean. The students who attended the final exam have tried an unequal quantity of exercises among the 200 available in the online automatic judge, as shown in Figure 3. Most students have tried to solve around 30 exercises ($\mu = 30.98$, $\sigma = 24.83$, median= 25), while only 17 students (5 of them in the CSE/M group) have not used LearnSQL at all in the three academic years analyzed. The students in the double-degree CSE/M have tried fewer problems ($\mu = 21.05$, $\sigma = 13.88$, and median = 21 vs. $\mu = 38.62$, $\sigma = 28.51$, and median = 38 in the single-degree CSE). Moreover, the students have effectively solved the majority of problems they have tried, as illustrated in Figure 4. We observe that just a few problems were left unsolved (most of the students abandoned between 0 and 2 problems), with an average of 5.87 ($\sigma = 7.43$) attempts before giving up. Several students retry the problems some days later with an average of 41.74 hours ($\sigma = 224.03$) between the first and last unsuccessful attempts. The delay between the first attempt and the first correct submission is around 10 hours ($\mu = 10.05$, $\sigma = 95.99$) in the whole population of students, although it decreases to 1.55 hours ($\sigma = 13.71$) in the double-degree CSE/M. In this same degree, the problem abandon rate is smaller with a mean of 0.9 abandoned problems instead of 1.66 for the single-degree CSE, while the numbers of attempts before solving ($\mu = 1.59$, $\sigma = 3.36$) and abandoning ($\mu = 5.68$, $\sigma = 5.73$) a problem are only slightly lower. The number of attempts for a student to solve a given problem is usually low ($\mu = 1.82$, $\sigma = 4.52$), with the first try being enough in 55.03% of the cases, and the second attempt in 18.25%. However, everyone has failed some problem and obtained feedback from the automatic judge.

The percentages of problems solved on the first and second attempt are higher than those reported in similar studies on SQL exercises [8, 28, 39]. This difference can be attributed to the distinctive features of the automatic judge used. First, LearnSQL provides students not only with

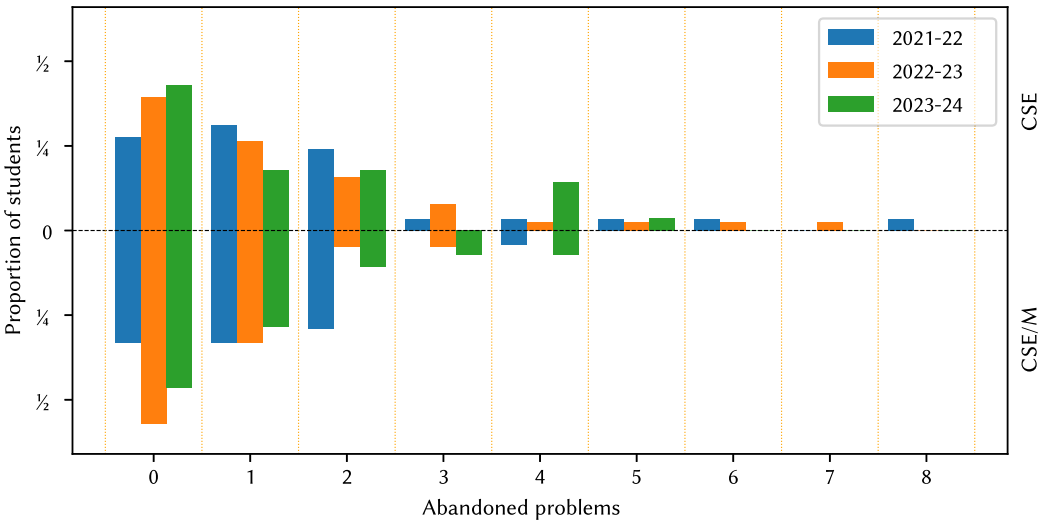

Fig. 4. Histogram: distribution of problems tried but not solved per student and academic year. Each bar over a mark a in the x-axis represents the ratio of CSE (upward) or CSE/M (downward) students who abandoned a problems in the corresponding academic year.

the schema of the tables but also with the values stored in the rows and the expected results (see Figure 1). Furthermore, LearnSQL includes a feature allowing students to download an SQL script to create and populate the database associated with the problem. This enables students to practice locally on their own database instances, submitting their solutions to the automatic judge only when they feel confident, thereby minimizing simple syntactic and semantic errors. Additionally, LearnSQL offers a set of approximately 200 problems, including a substantial number of easy and very easy problems. This is in contrast to the 13 problems analyzed in [8] and the 6 problems used in [28], some of which address advanced SQL concepts such as self-joins, `NOT EXISTS` clauses, and correlated subqueries. Finally, as previously noted, we rule out the possibility of academic dishonesty (e.g., using AI tools or copying third-party solutions) because LearnSQL was exclusively used for voluntary practice; submissions were not graded and did not impact students' final marks. The majority of the interactions (70.19%) with the online automatic judge have concentrated during the first to second month of the lessons, two months before the exam. This time window coincides with the time when SQL queries and procedural SQL are taught in class. However, shortly before the final exam in January, the number of submissions has slightly increased (15.15%). The submission concentration is more pronounced in the double-degree CSE/M, as shown in Figure 6, with a 99% of the submissions in 3 weeks. This concentration in the double-degree CSE/M is explained by the fact that students were assigned laboratory practices using LearnSQL in late October to early November. Additionally, throughout the rest of the course, they used other systems that were not employed by students in the single-degree CSE. It is also interesting to consider the information in Figure 5, which indicates that the automatic judge has been used extensively. Although some students have not used the automatic judge at all, those that have used it have submitted many times, and some of them sent more than 500 submissions (and all of them have passed and obtained good results in the exam).

In order to analyze which kind of problems is the most difficult for students, we have categorized our problems into **Data Query Language (DQL)** (for data retrieving queries, i.e., `SELECT`), DML, for data modification queries, such as `INSERT`), and **Embedded Procedural Language (EPL)**

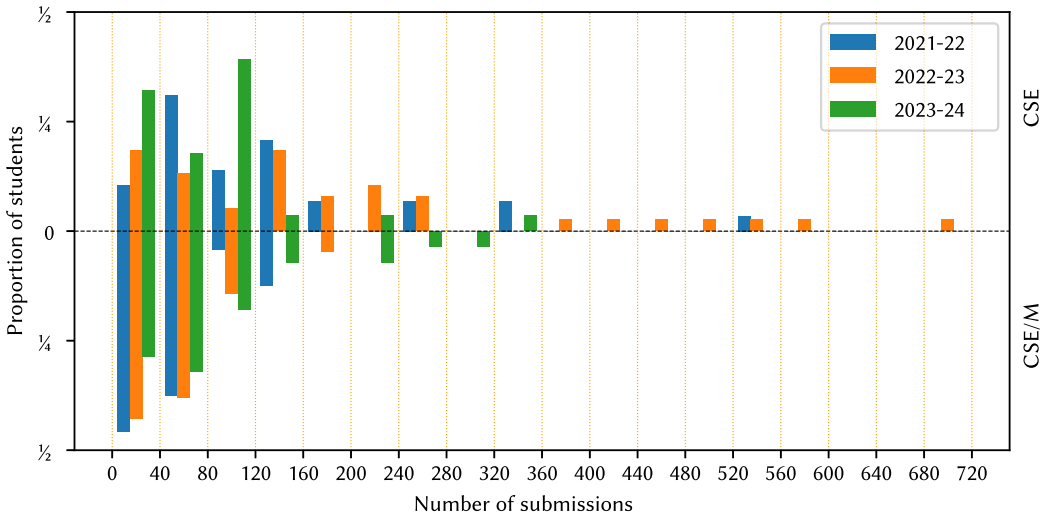

Fig. 5. Histogram: distribution of solutions submitted per student and academic year. Each bar between two consecutive marks a and b of the x-axis represents the ratio of CSE (upward) or CSE/M (downward) students who submit between a (included) and b (excluded) solutions in one academic year.

(a general-purpose programming language embedded in the DBMS, e.g., PL/SQL in Oracle for triggers, functions, and procedures). In turn, DQL problems have been classified using tags inferred from the SQL query capabilities employed in both the instructors' solutions and the students' submissions. These tags include: inner joins, outer joins, nested subqueries, grouping (using `GROUP BY`), aggregation functions (e.g., `SUM`, `MAX`), group filtering (via `HAVING`), management of `NULL` values, result sorting (using `ORDER BY`), row existence (with `EXISTS`), and set operations (such as `UNION`, `INTERSECT`, etc.). For the whole repertory of problems and some selected categories, Figure 7 shows the proportion of judge verdicts (accepted answers, syntactic errors, and semantic errors), the success rate defined as

$$\frac{|\{(s,p) \mid \text{student } s \text{ solved problem } p\}|}{|\{(s,p) \mid \text{student } s \text{ tried problem } p\}|},$$

and the mean number of submissions required to solve those problems. Exercises involving triggers stand out as they require more submissions to be solved (four tries in mean), while problems involving the `EXISTS` operator are those which are abandoned the most. All other subtypes (including all possible combinations of the aforementioned tags) exhibit figures similar to those of the whole repertory of problems. Moreover, we have also studied whether the performance of the students varies along the semester. Figure 6 shows the number of aggregated attempts (i.e., sequences of submissions of a student to the same problem with less than 10 days between them), their success rate as defined above, and the mean number of submissions required on them to solve the problem for each week of the semester and for both groups. While the outliers in the number of submissions and success rates can be explained by the small amount of attempts in those weeks, it is worth noticing that, in the CSE group, one of the lowest success rates and the greatest number of tries occur in the two weeks before the exam. This can be explained by the following factors: (i) The most frequently attempted problems during that period involve triggers, which appear to be the most challenging, as mentioned earlier, (ii) 85% of the attempts correspond to the first time the student tried that problem, even though the problems had been available throughout the course—indicating

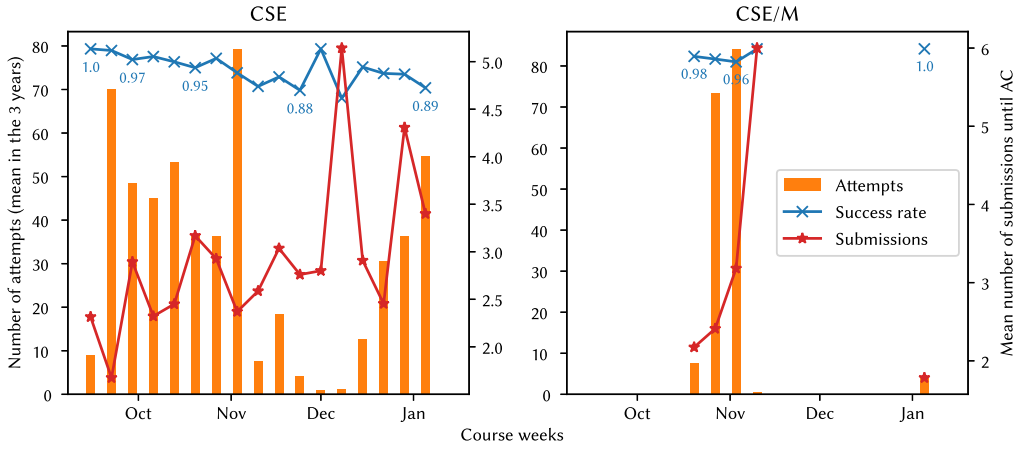

Fig. 6. Number of attempts of some problem by some student, success rate (proportion of successful attempts), and mean number of submissions in successful attempts per course week and by group. The scale for the number of attempts is on the left axis, the scale for the submissions is on the right, and the success rate is a proportion between 0 and 1 (which is attained in both graphs). Values are means among all years considered.

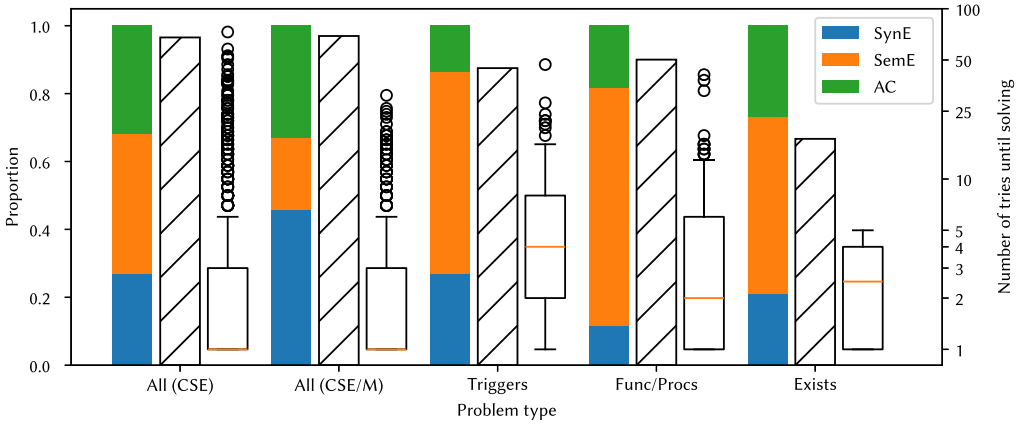

Fig. 7. Distribution of judge verdicts (syntactic error, semantic error, or accepted), success rate (hatched bar), and number of submissions until a problem is first solved (box plot) by category of problems. The scale for the bar plots in on the left, while the scale for the box plot is on the right. Except the first two columns, all others combine CSE and CSE/M data.

that students were not reviewing them for the exam, and (iii) The most active users during that period are among the least active users throughout the course.

4.2 Performance before and after the Introduction of the Automatic Judge

Even though the usage of automatic judge and the performance of the students have been heterogeneous during the three academic years, as shown in Table 4, we have compared the marks obtained in the only year without automatic judge (2019–2020) with those of the rest of the years using the Mann-Whitney U test. Our null and alternative hypotheses can be stated as follows:

Table 5. Statistical Analyses by Academic Year

		Single degree CSE			Double degree CSE/M		
Year		2021–2022	2022–2023	2023–2024	2021–2022	2022–2023	2023–2024
Mann-Whitney test	U	329	566.5	537	380	284	315
	p	✔ 0.0000021	✔ 0.000031	✔ 0.0057	✘ 0.9083	✘ 0.5972	✘ 0.2005
Mean mark difference CI	[–	0.78	0.44	−0.08	−0.41	−0.56	−0.18
	–]	2.2	1.8	1.7	0.4	0.5	0.6
Cohen's d		1.18	0.9	0.58	-0.01	-0.02	0.44
Spearman tried/SQL mark	ρ	0.7906	0.4816	0.4127	0.3656	0.4205	0.3815
	p	✔ 0.0001	✔ 0.0016	✔ 0.0147	✔ 0.0389	✔ 0.0277	✔ 0.0254
Spearman solved/SQL mark	ρ	0.773	0.4884	0.3939	0.3839	0.453	0.3664
	p	✔ 0.0001	✔ 0.0014	✔ 0.0177	✔ 0.035	✔ 0.022	✔ 0.0296

Mann-Whitney U test, Bayesian confidence intervals for the difference of marks, and Cohen's d effect size with respect to the academic year 2019–2020; and Spearman's correlation coefficients (ρ) with corresponding p-values (calculated by permutation testing) for the amount of practice.

(*H*0) *The marks of the SQL exercises of the academic year 2019–2020 coincide with the marks on the rest of the years.*

(*Ha*) *The marks of the SQL exercises of 2019–2020 are lower than the marks of the rest of the years.*

For the single-degree CSE, the tests rejects (*H*0) in favor of (*Ha*) with a p-value of $3.31 \cdot 10^{-7}$ ($U = 1433$). On the contrary, for the double-degree CSE/M, the test yields a p-value 0.596 ($U = 979$) that does not allow us to reject the null hypothesis. In order to quantify the potential improvement after the introduction of the automatic judge, we have also calculated a confidence interval for the difference of mean marks using Bayesian methods with a Jeffrey's prior. With a significance level of 0.95, the confidence interval is covered by $[0.55, 1.74]$ for the single-degree CSE, and by $[-0.29, 0.46]$ for the double-degree CSE/M. Moreover, the Cohen's d effect sizes are 0.94 and 0.17, respectively, which could be interpreted as a large and a small effect. Comparisons by academic year are shown in Table 5, where the Mann-Whitney row tests the null hypothesis "the marks of the SQL exercises of the academic year 2019–2020 coincide with those marks on the academic year Y" for each academic year Y, and the *Confidence interval* rows indicate the lower and upper bounds of the confidence interval for the difference of means between the year 2019–2020 and Y calculated as explained before.

4.3 Performance with Respect to the Amount of Practice

In order to further evaluate the effect on learning of LearnSQL, we compare the usage profile of the students with the marks they have obtained in the SQL-related exercises of the final exam. Table 5 shows the Spearman's correlation coefficient between these metrics for each academic course and group. The lower row indicates the p-values (values lower than the significance level of 0.05 have a green tick, while p-values greater than 0.05 have a red cross) computed by permutation testing for the null hypothesis that the correlation is zero and the alternative hypothesis that it is positive. Every p-value falls below the significance level of 0.05, so the existence of a direct correlation between the magnitudes can be accepted, although the correlation coefficients are usually modest. Combining all years by degrees, we obtain Spearman's and Kendall's τ correlations of respectively 0.482 ($p = 3.72 \cdot 10^{-7}$) and 0.351 ($p = 3.37 \cdot 10^{-7}$) for the single-degree CSE, and 0.389 ($p = 3.3 \cdot 10^{-4}$) and 0.272 ($p = 4.33 \cdot 10^{-4}$) for the double-degree CSE/M. Since problems are rarely abandoned before solving them, similar results are obtained if solved (instead of tried) problems are considered: the

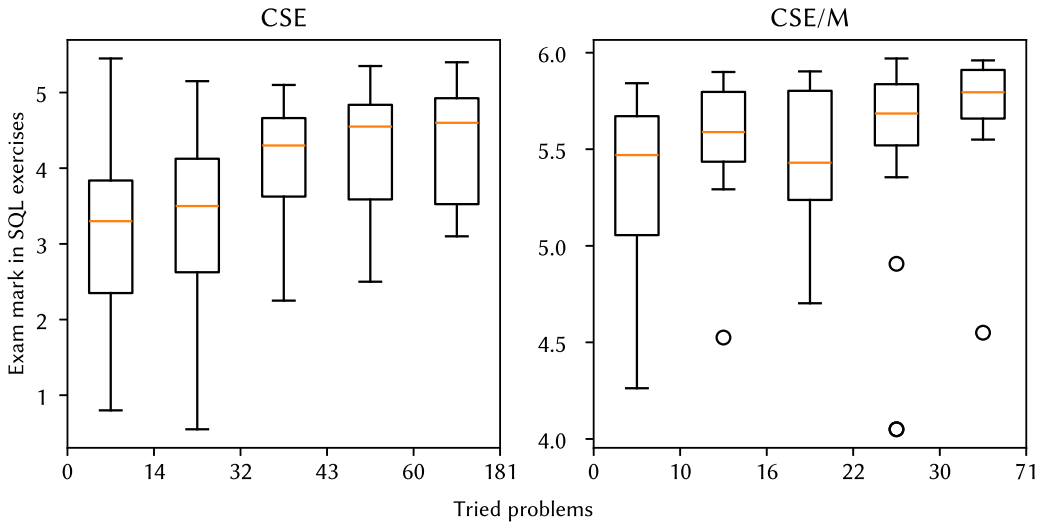

Fig. 8. Box plot of the SQL exercises' marks by number of tried problems, aggregated by quintiles.

Spearman's and Kendall's τ correlations become 0.486 ($p = 3.01 \cdot 10^{-7}$) and 0.352 ($p = 3.45 \cdot 10^{-7}$) for the single-degree CSE, and 0.401 ($p = 2.18 \cdot 10^{-4}$) and 0.279 ($p = 2.94 \cdot 10^{-4}$) for CSE/M. Thus, there is a statistically significant monotonic relationship between problems tried/solved and SQL exercise marks in the single-degree CSE, which is weaker in the double-degree CSE/M.

Figure 8 gives a more intuitive insight by showing the distribution of those marks aggregated into five equally numerous subsets by increasing number of tried problems. Its monotonicity suggests statistical evidence that students using the automatic judge to a greater extent improve their performance in the SQL exercises of the exam. Moreover, considering the quantiles of order 8 on the number of tried problems, we compare the marks of the students who have tried more than this number of problems against those who have tried less with the Mann-Whitney U test. In all cases, we obtain a p-value under the confidence level of 0.05 (the maximum is $p = 7.67 \cdot 10^{-3}$ for CSE, and 0.0167 for CSE/M), which can be interpreted as quantitative evidence for the conclusions that have been drawn from the box plot. Notice that the students who have not used the judge at all (12 for CSE and 5 for CSE/M, plus a discarded low-mark outlier in the latter group) are included in the lowest quintile of each group. Their mean mark is shortly below the mean of the whole group, although there is a student in each group that obtained more than 9 points out of 10 in the exam or 5.9 points out of 6 in the SQL exercises.

4.4 Impact of the Automatic Judge on Women Students

Women's underrepresentation in STEM careers is a well-known issue [4, 48], and the distribution of students shown in Table 1 confirms this underrepresentation in both CSE and CSE/M groups. Some studies focusing on Computer Science suggest that women students show differences in their learning styles [24] and their performance and satisfaction [29]. However, prior studies [16, 17] have found no significant gender differences in the use of automatic judges and their improvements for database learning. In light of these findings, we have examined whether the conclusions drawn for RQ1 and RQ2 hold when considering only the subset of women students.

We first observe that there are no significant differences between women students and the whole class in the SQL exercises marks (the mean is only 0.12 points out of 6 above for women in the whole population) and any other variable in the study. Regarding RQ1, the Mann-Whitney U test for

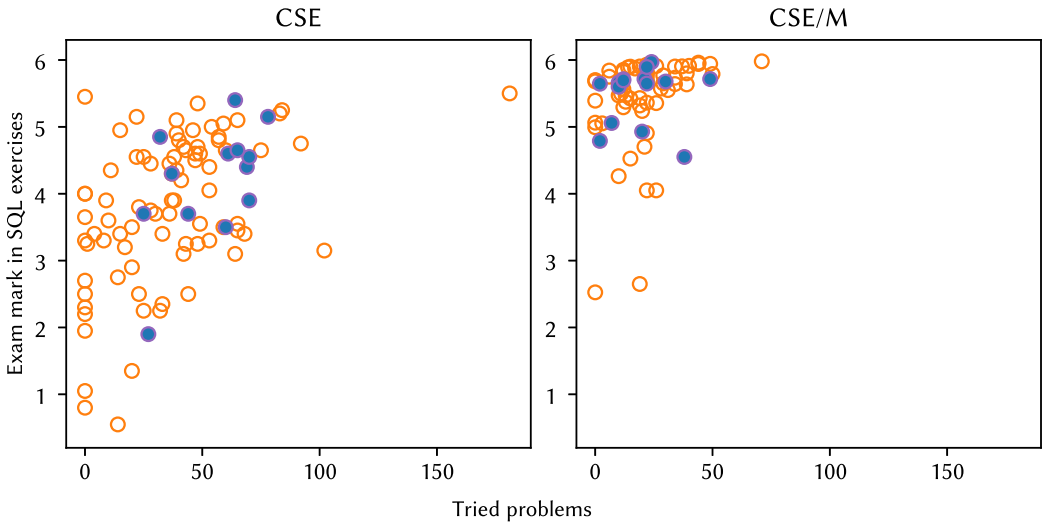

Fig. 9. Marks for SQL exercises vs. tried problems (women students shown as filled blue circles).

the mark improvement between the academic year 2019–20 and the rest of the academic years gives a p-value of $1.6 \cdot 10^{-3}$ in CSE and of 0.589 in CSE/M, similarly to the whole population. The Cohen's d measures are respectively 1.56 (large effect) and 0.2 (small effect). The confidence intervals are $[0.25, 3.41]$ and $[-0.72, 0.93]$, respectively, which are too wide due to the reduced sample size of women students. Regarding RQ2, Figure 9 shows the distribution of marks obtained by all students in the SQL exercises of the exam, arranged by number of tried exercises in the automatic judge, with women depicted as filled bluish circles. The hypothetical monotonic relation between the automatic judge usage and the grades is less clear but still observable in the single-degree CSE, and not clear at all for the double-degree CSE/M. This uncertainty is maintained by the quantitative measures, with a Spearman's correlation of 0.482 ($p = 0.0493$) and a Kendall's τ correlation of 0.312 ($p = 0.082$) for the single-degree CSE, which are respectively 0.411 ($p = 0.0641$) and 0.351 ($p = 0.0456$) for the double-degree CSE/M.

In summary, after repeating our analysis for RQ1 and RQ2 on the subset of women students, we observe that the results are largely consistent with those obtained for the overall student population, as reported in previous studies [16, 17]. Specifically, in the CSE degree, we identify a positive improvement in final exam scores and a mild correlation between the frequency of judge usage and performance in SQL exercises of the ordinary exam. However, in the CSE/M degree, no significant improvement is detected. These findings should be interpreted with caution, as the small sample size of female students considerably limits the reliability and generalizability of the results. Further research with larger and more representative samples is necessary to derive robust conclusions regarding the gender-specific impact of the LearnSQL automatic judge as a tool for self-directed practice.

5 Students' Perspectives: Insights from a Survey on LearnSQL

At the end of the academic years 2021–2022, 2022–2023, and 2023–2024, questionnaires were given to students enrolled in the database course. Participation in the survey was voluntary and responses were completely anonymous. This section focuses solely on responses from the CSE group selected for this study because the students from the double-degree CSE/M did not submit any response for these questionnaires. The questionnaires comprised numerical rating questions concerning

Table 6. Survey on LearnSQL: Average Numeric Ratings by Academic Year

	2021–2022 (5 responses)	**2022–2023** (16 responses)	**2023–2024** (9 responses)
Clarity of question statements	4.8	4.3	4.6
Browsing between different sections	4.8	4.0	4.6
Usability of SQL editor	4.8	4.6	4.6
Feedback given after each submission	4.0	3.9	4.0
Amount of exercises on SQL queries	4.8	4.5	4.7
Amount of exercises on PL/SQL	4.0	3.9	4.6
Learning curve of exercises	4.2	3.8	4.2
Overall assessment of LearnSQL	5.0	4.6	4.3

Ratings range from 1 (very negative) to 5 (very positive).

various aspects of LearnSQL, the overall system, and its efficacy in facilitating the acquisition of subject-specific competencies. Additionally, open-ended questions allowed students to point out both the positive aspects of the tool and areas for improvement.

Although the LearnSQL automatic judge interface allows for usage on both personal computers and mobile devices, nearly all students reported using it solely on desktop or laptop computers. Over 80% of the students who responded to the survey reported completing at least 20 exercises on LearnSQL. When considering those who completed 30 exercises or more, the percentages were 60% (year 2021–2022), 75% (year 2022–2023), and 55% (year 2023–2024), respectively. Therefore, we believe that the students who responded to the survey have used the automatic judge enough to provide informed responses.

5.1 Responses to Numeric Rating Questions

The first part of the questionnaire consists of a set of questions related to the functionality of LearnSQL and its usability. Each of the questions could obtain a rating between 1 (very negative) and 5 (very positive). The average ratings are shown in Table 6. Given that the amount of responses received in years 2021–2022 and 2023–2024 is rather limited, we can only draw significant conclusions from year 2022 to 2023. The most valued aspects of LearnSQL are the user-friendliness of the SQL editor and the large number of available exercises about SQL queries. On the other hand, among the aspects with a more modest but still very good rating, we can mention the feedback provided by the system in response to students' failed attempts. Overall, students value the learning tool with very positive ratings (5 out of 5 in year 2021–2022, 4.6 in year 2022–2023, and 4.3 in year 2023–2024). In the 2021–2022 and 2022–2023 academic years, the overall evaluation was equal to or higher than that of all the previous sections. One factor that may explain this is the *halo effect*, whereby students may have had a generally positive impression of the automatic judge, influencing their overall assessment more favorably than the sum of the individual characteristics, even if some of these characteristics have weaknesses. This bias might also be due to students' perception of the importance of maintaining the tool as a support in subsequent courses. However, in these cases, the deviation of the overall evaluation from the maximum rating obtained in the previous sections is relatively minor, so we believe that it does not pose a significant threat to the validity of the questionnaire.

The second part of the questionnaire (Table 7) focuses on the students' perception of the usefulness of LearnSQL in acquiring the course's skills. Again, the rating ranges from 1 (not useful) to 5 (very useful). In this sense, students consider that the tool is useful (4.5 in year 2022–2023) to achieve the

Table 7. Survey on LearnSQL: Students' Perception of Learning by Academic Year

To what extent has LearnSQL helped you to...?	**2021–2022** (5 responses)	**2022–2023** (16 responses)	**2023–2024** (9 responses)
Learn SQL queries	5.0	4.6	4.6
Learn DML sentences	3.2	4.1	3.8
Learn PL/SQL procedures and functions	3.2	4.0	4.2
Learn PL/SQL triggers	4.0	3.9	3.9
Acquire learning outcomes	5.0	4.5	4.1

Ratings range from 1 (very negative) to 5 (very positive).

Table 8. Some of the Students' Responses to Open-Ended Questions on LearnSQL

What positive aspects of LearnSQL would you highlight?
"*The clarity of feedback when entering an incorrect query.*"
"*Easy to use and the feedback is very helpful.*"
"*The number of exercises [and] the progressive difficulty of those exercises.*"
"*I would emphasize the feedback, since it is very useful to know where you have made a mistake in order to identify where you have made that mistake and not have to review the entire code.*"
What negative aspects do you find in LearnSQL?
"*In the basic exercises, errors should be made more explicit.*"
"*In my opinion there should be a section with answers or after a certain number of attempts there should be the possibility of seeing the complete or partial answer as a clue.*"

aims of the database course. The learning outcomes are notably stronger in the area of SQL queries compared to other topics covered in the course (specifically, DML and EPL). This can be explained by the fact that the vast majority of problems included in the automatic judge are related to SQL queries, accounting for 86% of approximately 200 problems in total.

5.2 Responses to Open-Ended Questions

The second part of the questionnaire contained optional open-ended questions about the positive and negative aspects of LearnSQL. The first of these asked about the positive aspects of LearnSQL and the features that students found useful. Over the three courses in which the survey was conducted, 15 students (out of a total of 30 responses received) answered this question (in Table 8 we show some of them). Nine of the responses received explicitly mentioned the tool's provision of immediate feedback on submitted solutions. It is noteworthy that these students were also enrolled in other programming courses that make use of automated judging systems (e.g., DOMjudge [20]), which only provide information on whether the submitted solution is correct or not, without further elaboration. Additionally, other aspects that received positive feedback include the user interface of the automatic judge (mentioned in four responses) and the wide variety of exercises organized by difficulty level (three responses).

Another optional question in the questionnaire involved the negative aspects of LearnSQL and the features that should be changed or removed. Six students answered this question. Two of them pointed out that the feedback messages are somewhat unclear, particularly in cases where the student's response deviates from the expected answer beyond simply yielding more or fewer rows

than the latter. Another student suggested allowing access to solutions after a certain number of attempts, along with the inclusion of hints to aid in problem-solving.

6 Discussion

In this section we will summarize the main insights obtained during the evaluation of the automatic judge LearnSQL, giving answer to the RQs introduced previously and presenting also the detected threats to validity as well as some practical implications for educators and other professionals.

6.1 Impact of the Automatic Judge in Database Learning

As a short summary, from the analyses performed in Sections 4 and 5, the main insight we draw is that free practice with LearnSQL positively impacts database learning. This effect becomes more noticeable as the amount of practice increases, but it is also observable with relatively little training. Moreover, the availability of an automatic judge is highly appreciated by the students, who are engaged by the problems and persist until they are solved, which increases their motivation toward learning.

The first RQ ("*RQ1: Does providing an automatic judge for practicing database problems improve final marks in the ordinary exam?*") is the more direct question related to measure LearnSQL effectiveness on improving database learning. The global U test, its p-values, and the confidence intervals in Table 5 allow us to conclude that, for the single-degree CSE, there is statistical evidence that the marks of the SQL exercises are higher in the academic years 2020–2021, 2021–2022, and 2023–2024 (years in which the automatic judge was available) compared to the academic year 2019–2020 (when the automatic judge was not available). Moreover, the confidence intervals quantify this improvement in marks between 0.55 and 1.74 points out of 6. As the only difference in the teaching methodology between the academic year 2019–2020 and the rest of years is the availability of the automatic judge, this improvement of the marks is, with high probability, due to the facilities provided by the automatic judge. For the double-degree CSE/M, the U test does not provide enough evidence supporting an improvement on the marks due to the automatic judge.

The second RQ ("*RQ2: Do students who use the automatic judge more achieve higher marks in the ordinary exam?*") focuses on detecting a statistical relation between automatic judge usage and improvement in marks. According to the data grouped in quintiles in Figure 8 and the Spearman's and Kendall's correlation coefficients presented in Section 4, we can conclude that there is a correlation between the number of problems tried and solved and the marks obtained in the SQL exercises in the final exam for the single-degree CSE. Figure 8 shows a general but less acute increasing trend on the marks in SQL exercises with respect to the number of tried problems in the double-degree CSE/M, so the correlation coefficients found in this case are smaller but still statistically significant. It is important to note that the relationships identified represent correlations, and therefore we cannot categorically assert that increased usage of the automatic judge directly causes the improvement in students' marks, as discussed in Section 6.2.3. However, based on the data and analysis presented, we can provide recommendations for students regarding the use of the automatic judge. Specifically, to develop a solid understanding of SQL (as measured by performance on the final exam) we recommend completing at least 45 problems in the single-degree CSE program and 25 problems in the double-degree CSE/M program, ensuring coverage of the full course content and a range of difficulty levels.

Finally, the third RQ ("*RQ3: Is the impact of the automatic judge dependent on students' profiles?*") focuses on the possible differences on learning caused by the automatic judge discriminating on the student profile. As previously detected in Section 4, the use of the automatic judge in the single-degree CSE produces a statistically significant increment on the marks obtained in the SQL exercises, but we cannot find similar evidences of this improvement in the double-degree CSE/M.

This lack of evidence is not unexpected, as students enrolled in the double-degree CSE/M program consistently demonstrate high performance, with mean scores on SQL exercises exceeding 5 out of 6 points in each academic year, as shown in Table 4. As a result, their margin for further measurable improvement is limited. It is also worth noting that all students had access to the same set of SQL problems, which covered a range of difficulties within the scope of the final exam. While this design ensures fairness and alignment with the course objectives, it did not include a tailored set of highly challenging problems specifically aimed at top-performing students such as those in the CSE/M program. It is therefore possible that the automatic judge could have had a stronger impact on their learning had they been exposed to more complex or open-ended problems. However, given that the final exam was common to all students and focused on standard course content, any additional learning gains at the high end of the performance spectrum may not have been captured in the exam results. On the other hand, we did find statistically significant evidence that increased use of the automatic judge is associated with improved student performance. This correlation appears in both the single-degree CSE and the double-degree CSE/M, although the coefficients are milder in the double-degree CSE/M. Our hypothesis is that having an automatic judge for free practice is a factor of extrinsic motivation for those students with a low level of self-motivation toward database learning. In the double-degree CSE/M, where the majority of the students are initially very self-motivated toward learning, the impact on exam marks does not increase clearly with automatic judge usage. However, in the single-degree CSE program, the automatic judge increases engagement by offering a wide range of achievable short-term goals that are automatically and instantly evaluated. This external help encourages students to focus on learning, ultimately leading to higher grades. As an additional support for this hypothesis, we stress the great reception of the automatic judge by the students (measured in the number of students that used it even though it was not mandatory, the high number of problems tried and solved, and the positive opinions gathered from students' questionnaires) which positively supports its motivational value. Therefore, for *RQ3* we conclude that the impact on database learning is more pronounced on average students, and that the integration of an automatic judge in this scenario improves both students' motivation and learning outcomes. Integrating an automatic judge is less effective when improving learning in top-performing, highly self-motivated students in a database course, although the general good reception of the automatic judge by students suggests that it is a positive addition to any database course for providing free practice.

6.2 Threats to Validity

In this section, we briefly discuss those aspects of the analysis that might compromise the validity of the findings.

6.2.1 Experimental Design. For evaluating the impact on learning outcomes of the automatic judge LearnSQL we have not followed a purely randomized experimental design where the participants are taken from a uniform population and assigned randomly to a control group (without using the automatic judge) or an experimental group (where the automatic judge is available). Although this experimental design is the safest to extract conclusions from learning interventions, we have decided not to follow it because of some reasons. The main concern is purely practical: it is not technically possible to avoid the use of the automatic judge by the students in the control group once it is online and publicly available. Automatic judge access is controlled by usernames and passwords, which are unique and personal, but as the students in the same group are acquaintances from the previous year or even friends, they would easily share their passwords if they considered the automatic judge is useful. This password sharing could not be detected, which would imply a contamination on the data gathered from the automatic judge usage (mainly considering some

students that have indeed used the automatic judge inside the control group, but also having artificially inflated submissions and problems tried) that would invalidate any conclusion for our RQs. Another concern about applying a purely experimental design is ethical: we believe that having an automatic judge is positive for learning, or at least will not cause any harm, so explicitly excluding some students from its usage was perceived as a questionable decision. Finally, another concern of the experimental design is group sizes. In the single-degree CSE there are around 40 students per academic year, whereas in the double-degree CSE/M there are around 25 students. If we are required to split them into to equal groups, the sizes will be so small that it would be difficult to extract any statistically relevant evidence from them.

Considering the previous concerns, for evaluating the impact on learning outcomes of the automatic judge LearnSQL we have followed a quasi-experimental design [44] in which the control group were the students in the academic year 2019–2020 (where the automatic judge was not available at all because it was not even under development) and the experimental group were the students from the following academic years 2021–2022, 2022–2023, and 2023–2024 (all of them with full access to the automatic judge, excluding from the experiment the academic year 2020–2021 affected by the COVID-19 pandemic). With this design we mitigate the previous concerns because all the students have their own username to access the automatic judge, which is available to every student in every group, and combining several academic years we can obtain a sufficiently big population. However, in a quasi-experimental design like this one there are some threats to validity that must be considered.

First, it may be the case that students from different academic years do not have a uniform initial learning level. One possibility to validate this uniformity is performing level pre-tests to assess the initial level of the students, but instead we have considered another piece of objective information: we accessed the data gathered by the institutional intelligence center of the university to obtain the students' performance in all the subjects of the same academic year. As presented in Table 2 (Section 3.1), the average marks for the academic years 2021–2022, 2022–2023, and 2023–2024 are comparable to those of the control group from 2019 to 2020 in both the CSE and CSE/M degree programs, with a slight decline observed in the most recent years. Regarding performance rates, the double-degree CSE/M shows similar values for 2019–2020 and 2021–2022, but a decrease of approximately 7% is noted in 2022–2023 and 2023–2024. For the single-degree CSE, there is an 11% decline in 2021–2022 compared to 2019–2020, followed by a recovery in 2022–2023 and a new decrease of about 6% in 2023–2024. These trends suggest that the improvements observed in students' performance in database courses cannot be attributed solely to an increase in their initial knowledge levels. In fact, the data indicate a slight decline in general performance when compared to the academic year 2019–2020.

Although we cannot ensure that every student involved in the study group is also enrolled in all the courses of the second year (resp. third year) of CSE (resp. CSE/M), the low percentage of repeating students in the Databases course suggests that those enrolled in this subject also take a relatively high proportion of courses from the same year, since the curriculum does not yet include elective courses in the year in which Databases is taught. Nevertheless, it is possible that some of them have pending courses from the previous year or are already taking courses from the following year. However, if we look at the evolution of the aggregated results for these years (Table 3), we observe that in the first year of the CSE degree, the performance rate drops from 77.60% in 2019–2020 to 74.02% in 2021–2022. It then rises in 2022–2023 (82.52%), only to return to its previous level in 2023–2024 (74.77%). In the third year of CSE, the performance rates range from 76% to 79%, except for a decline to 71.09% in 2021–2022. As for CSE/M, the overall performance rates for the second and fourth years in 2019–2020 are better, or very similar, to those in 2021–2022. Therefore,

these rates do not support the possibility that the students had a higher initial level of knowledge than those who took the databases course in 2019–2020.

Similarly, designing an experiment that involves different student cohorts across multiple academic years has an inherent bias, as instructors' growing experience over time may affect their teaching effectiveness and, consequently, students' performance. This could potentially confound the effects of the automatic judge on databases learning, as obtained by the Mann-Whitney U tests for RQ1. However, we argue that the impact of this inherent bias is likely to be limited. By the first year of the study (2019–2020), the involved instructors had gained substantial experience on full-time university teaching. Specifically, the instructor of the CSE group had 6 years of experience teaching data management subjects at both undergraduate and master's levels, while the instructor of the CSE/M group had 21 years of experience teaching database subjects, 7 of which were dedicated to the same database course under analysis. Although the instructor's development over the years represents a potential bias when interpreting RQ1, their extensive prior experience provides sufficient grounds to attribute a substantial part of the improvement in marks to the availability of the automatic judge for voluntary practice.

6.2.2 Potential Confounding Effects of Large Language Model (LLM) Use on Study Validity. The improvements observed in students' performance could be influenced by external factors coinciding with the introduction of the automatic judge. In particular, the emergence of powerful LLMs such as ChatGPT, Gemini, Perplexity, Claude, Mistral, or Llama, which have demonstrated strong potential as teaching and programming assistants for SQL [9, 10, 38]. It is important to highlight that ChatGPT, the first widely available LLM, was launched on 30 November, 2022,[4] while other LLMs followed later. Given that our study spans the academic years 2019–2020, 2021–2022, 2022–2023, and 2023–2024, and that the database course runs from September to December (with the final ordinary exam around January 10), LLMs could only have influenced data from 2023 to 2024 academic year and, to a lesser extent, the latter part of 2022–2023 (for early adopters of ChatGPT). Although we cannot objectively measure the level of LLM usage during 2022–2023 and 2023–2024, if students relied heavily on LLMs as a database teaching assistant, we would expect a decrease in submissions and the number of problems attempted, as students might shift their interactions from the automatic judge to LLMs. Indeed, this trend is observable in the CSE degree, where the 2023–2024 academic year shows approximately a 30% decrease in submissions and a 50% reduction in attempted problems compared to 2022–2023, despite having a similar number of enrolled students. However, this pattern is not evident in the CSE/M double degree. Similarly, if LLMs were extensively used from academic year 2023–2024 and could justify a learning improvement on database learning, we would expect a mean mark increase in year 2023–2024 compared to the previous year 2022–2023, which is not the case for the CSE degree, but it is observed in the double-degree CSE/M. Regardless of the discrepancies in the evidence on LLM usage in the 2023–2024 academic year, we have repeated the analysis of RQ1 and RQ2 while excluding the data from this period to control for potential interference. For the single-degree CSE, we observe that the scores of the SQL exercises are significantly higher in the academic years 2020–2021 and 2022–2023 compared to 2019–2020 ($U = 896$, p-value $= 1.09 \cdot 10^{-7}$, with a confidence interval $[0.85, 1.72]$ for the mean difference). Additionally, we find a correlation between the number of problems attempted in the automatic judge and the final exam scores (Spearman coefficient: 0.515, p-value $= 4.07 \cdot 10^{-6}$; Kendall coefficient: 0.381, p-value $= 3.61 \cdot 10^{-6}$). For the double-degree CSE/M, no statistical evidence supports an improvement in the SQL exercise scores ($U = 718$, p-value $=$ 0.827, confidence interval $[-0.42, 0.339]$). However, we do observe a correlation between the number of problems attempted in the automatic judge and the SQL exercise scores in the final exam

[4] https://openai.com/index/chatgpt/ (accessed January 25, 2025).

(Spearman coefficient: 0.443, p-value = $9.16 \cdot 10^{-4}$; Kendall coefficient: 0.313, p-value = $1.20 \cdot 10^{-3}$). Similar results are obtained if the academic year 2022–2023 is also excluded, that is, if only the academic year 2021–2022 is considered, as shown in Table 5. Thus, while the emergence of LLMs may have influenced the most recent data, the consistent findings from academic years prior to their introduction provide robust evidence for the impact of the automatic judge, independent of LLM interference.

6.2.3 Self-Selection Bias. Regarding RQ2, we identified a statistically significant positive correlation between the number of problems attempted and solved by students and their performance in the SQL section of the ordinary exam, a trend observed consistently in both the CSE and CSE/M degrees. However, it is important to emphasize that this relationship represents a correlation and does not necessarily imply causation. Student characteristics related to self-motivation (e.g., intrinsic drive, self-regulation, and curiosity) or prior SQL knowledge may significantly influence the observed correlation. Highly motivated or more experienced students are naturally predisposed to engage more actively with the automatic judge, which could confound the relationship between problem-solving and exam performance.

To better assess the potential impact of self-selection bias, additional data would be necessary, such as metrics on student motivation (e.g., prior academic performance, class participation, and attendance) or objectively measured prior experience with SQL. In the absence of such data, we hypothesize that self-selection bias accounts for a portion of the observed correlation. However, it is important to emphasize that this potential self-selection bias would not impact the findings related to RQ1, as the evidence demonstrates an improvement in the performance of the entire class, rather than its effects on specific students.

6.2.4 Sample Size. Considering the application context of the evaluation (database learning focused on the programming languages SQL and PL/SQL), a net population of 293 students is generally considered a sufficient sample size for a performance evaluation like the one presented in this article. So are the aggregated sizes of the CSE and CSE/M groups, with 191 and 102 students, respectively. However, the sizes of the yearly cohorts, between 39 and 66 students for the CSE group, and between 23 and 28 students for CSE/M (see Table 1), pose some validity threats that need to be discussed. On the one hand, RQ2 is not affected by the smaller amount of the single-year cohorts, because the correlations that support it are calculated from the aggregated sample. Moreover, when correlations are calculated year by year in Table 5, similar results are obtained. On the other hand, RQ1 and RQ3 strongly rely on the reference groups of the academic year 2019–2020, with 66 and 27 students for the CSE and CSE/M group, respectively. While the size of the CSE cohort is widely acceptable, that of the CSE/M group could have limited our possibilities to obtain statistically significant results for RQ1 and RQ3.

Nevertheless, further research is needed on different databases groups and universities with varied and larger samples in order to verify the results before making broader generalizations.

6.3 Implications for Practice

In this section, we discuss several practical implications based on the evidence gathered and the lessons learned during the development and evaluation of the LearnSQL automatic judge.

6.3.1 Implications for Instructors.

— Instructors should introduce the automatic judge early in the course, ideally alongside the first SQL sessions. Although usage was voluntary in our setting, students responded positively and made extensive use of the tool when given early access (see Section 4.1 and Figure 6).

—Automatic judges can serve as an effective supplementary tool without being part of the formal assessment. Instructors can motivate students to use it by highlighting its value for self-assessment and learning, rather than through mandatory assignments. Additionally, automatic judges may be especially useful for students who benefit from structured and guided practice, so instructors should consider targeting such tools to students with weaker backgrounds or less developed self-regulation skills, offering them a scaffolded learning path.
—A broad set of problems of varying difficulty is essential to accommodate students with different prior knowledge and motivation levels. Instructors should include both basic and challenging problems, and ensure that easier problems are available early on to build confidence. This variety not only reinforces conceptual understanding but also maintains student motivation. Indeed, in our study, the richness of available problems was one of the most positively rated aspects of the tool (see Table 6).
—Special attention should be given to the clarity and self-sufficiency of problem statements. Since automatic judges typically operate in unsupervised settings, students must be able to understand and engage with each problem independently. As recommended in the mapping study on SQL education [54], effective problems should clearly present the database schema, include the expected output, and articulate the data requirements with minimal ambiguity. These design principles in problem statements help reduce students' frustration, thereby enhancing their autonomous learning.
—The use of LLMs, such as ChatGPT, can be an effective way to streamline the creation of SQL problems for automatic judges. In our experience, LLMs can be effectively leveraged to generate SQL queries targeting specific topics, provided that a manually designed, shared database schema with sufficient complexity is prepared in advance. LLMs can also assist in creating mock data to populate tables; however, this step often requires manual oversight. We have found that LLMs may produce data with too few rows or unrealistic content, making it more practical to generate data tailored to each individual problem rather than attempting to create a universal dataset for all exercises.
—Reusing a shared database schema and authentic problem scenarios across multiple exercises provides both pedagogical and cognitive benefits. On the one hand, schema consistency reduces the cognitive load on students by allowing them to become familiar with table structures and column semantics, enabling them to focus more effectively on formulating correct SQL queries. On the other hand, working with realistic and coherent problem contexts promotes integrated assessment tasks that better reflect real-world database usage, supporting deeper learning and the transfer of skills to professional practice.

6.3.2 *Implications for Tool Developers.*

—Deployment procedures for automatic judges should be carefully designed and thoroughly documented. In general, automatic judges can require setting up multiple components, including a web server, a WSGI server, and different database systems. Reducing the technical complexity of this process through containerization technologies like Docker Compose can significantly facilitate adoption by instructors with limited technical expertise.
—Ensuring acceptable performance under peak load is essential for the practical usability of an automatic judge. In our deployment, a modest server was able to handle over 800 submissions per day without noticeable performance degradation. However, system performance is influenced by variables such as the computational cost of the submitted queries and the number of students working simultaneously, particularly during class hours when submission traffic tends to concentrate.

— Once installed, an automatic judge should be designed for low maintenance during regular operation. In our experience, LearnSQL required very little upkeep after the initial setup, which is a significant advantage for long-term sustainability. Tool developers should aim to minimize routine maintenance tasks through robust defaults, logging, and automatic recovery from common failures.
— User management functionality should support both batch operations and individual updates to reduce administrative workload. Administrative tasks such as deleting users from previous academic years or configuring new classes should be supported through efficient user interfaces. This type of feature helps reduce friction during course transitions and contribute to broader adoption in diverse educational contexts.
— Tools should provide straightforward mechanisms to export usage data, including submissions, verdicts, timestamps, and access logs, in structured and easily analyzable formats. The ability to filter this information by criteria such as student, problem, submission date, or activity type is essential for instructors and researchers seeking to analyze learning patterns, monitor engagement, or evaluate tool effectiveness.

6.3.3 *Broader Impacts on Assessment Practices.*

— The design of automatic judges naturally promotes the use of feedback as an integral part of the assessment process. Rather than functioning purely as grading tools, these systems provide students with guidance that helps them identify and correct misunderstandings. This reinforces a formative perspective in which assessment is used to enhance learning, not just to measure it.
— The ability to track student activity over time opens new possibilities for continuous and process-based assessment. By analyzing submission logs and problem-solving trajectories, instructors and researchers can gain insights into student progress and learning strategies. This data can be used to inform personalized support or to identify at-risk students early in the course.
— The structured nature of problems and associated test cases in automatic judges aligns well with competency-based education models. Students can demonstrate mastery of specific skills through successful completion of targeted exercises, allowing instructors to map problem completion to specific learning outcomes. This supports flexible and outcomes-driven assessment strategies.

7 Related Work

This section reviews related work by first listing tools for database learning. Then, we summarize the findings from the literature concerning the evaluation of these tools. Finally, we compare the error coverage in the syntax and semantic analysis embodied in the automatic judge with known categorizations of errors found in the literature, and summarize the strengths and weaknesses of the LearnSQL automatic judge when compared to the rest of the tools.

7.1 Tools for Database Learning

Several tools have been proposed to improve database learning over the years. In this section, we present a comparison between the most established tools.

Table 9 summarizes the features of these tools, where the first column presents the tool name, the second one is the part of SQL which is supported (DQL, DML, and EPL). The third column includes the list of DBMSs supported by each tool. Each cell in the column *Open Source* is checked if the tool has been made available in such a way. The fifth column contains the kind of feedback issued by each tool: from a basic one (Correct/Incorrect) to a rich one; in this column there are

Table 9. Tool Comparison

Tool	SQL coverage	DBMSs	Open source	Feedback	Evaluation	Asse-ssed
Active SQL	DQL	Oracle	✗	Rich feedback (accuracy)	Execution	✓
Aplicación BD	DQL DML	MySQL, Oracle, SQL Server	✗	Rich feedback (hints)	Execution + heuristics	✓
AsseSQL	DQL	Oracle	✗	Correct/Incorrect	Execution	✗
aSQLg	DQL	Oracle	✗	Numerical	Comparison of queries	✗
LearnSQL	DQL DML EPL	Oracle	✓	Rich feedback (DES)	Execution	✓
Query Competition	DQL	PostgreSQL	✗	Correct/Incorrect	Execution	✓
SQL-ACME	DQL DML	Oracle, SQL Server, PostgreSQL	✗	Correct/Incorrect	Execution	✗
SQLator	DQL	SQL Server 2000	✗	Correct/Incorrect	Comparison of queries	✗
SQLify	DQL	Oracle	✗	Numerical	Comparison of queries	✗
SQLTester	DQL	Oracle	✓	Correct/Incorrect	Execution	✓
SQL-Tutor	DQL	Ingres	✗	Rich feedback (CBM-based)	Execution	✓
SQLZoo	DQL	MySQL, SQL Server PostgreSQL	✓	Correct/Incorrect	Execution	✗

also tools that provide a numerical feedback. The next column, *Evaluation*, indicates which method is used to detect valid and invalid answers, where 'Execution' means than the outcome of the query is compared to a given valid instance (set of rows), 'Comparison of queries' means that the syntactical form of the queries are evaluated, and **constraint-based student modelling (CBM)** [33] is the constraint-based method employed by SQL-Tutor to evaluate queries. Finally, the last column checks tools whose effectiveness have been assessed in a field study by their own authors. Included tools in this table are:

— *Active SQL* [42] is a web system (although an SQLPlus interface with limited features is also available) for learning SQL queries. It computes a numerical value based in both syntax and logical errors; the latter is obtained by comparing the computed table and the expected one, and checking which rows/columns are correct. It also implements mechanisms for checking plagiarism. The researchers detected a significant improvement in students' learning experience by implementing an interactive online environment for teaching SQL, with a focus on automatic feedback and student behavior monitoring. Active SQL is no longer active,[5] and SQLZoo (last row in Table 9) is recommended now by the authors.
— *Aplicación BD* [16] is a web system that supports SQL and DML problems for several DBMSs, provides extra feedback when a submission is not valid, and includes plagiarism checking.

[5] https://db.grussell.org/sql/.

The authors verified that students using the tool obtained better results and with reduced plagiarism.

— *AsseSQL* [41] is a web system mainly designed for evaluation. It checks the correctness, based in the obtained results, of SELECT queries in a fixed time. Only students' feedback has been analyzed.

— *aSQLg* [23] provides a numerical mark for each submission (restricted to SQL query problems) based on inspection of the student code to measure the equivalence to the expected solution, coding style, efficiency, and so on. Its benefits were based on informal evaluations, mainly from student questionnaires.

— *LearnSQL* is the automatic judge presented in this article.

— *QueryCompetition* [32] is a web system that allows students to practice SQL queries (only DQL) in a competitive environment. Their goal is to obtain empirical evidence on how gamifying elements, such as challenges, points, and leader boards, integrated into QueryCompetition, affect student performance, motivation, and user experience. Metrics for student ranks include query efficiency (as execution time), student response time, and correctness.

— *SQL-ACME* [47] is a web system that supports different DBMSs. Besides SQL query problems, it includes a limited number of DML problems, but not EPL problems involving functions, procedures, or triggers. The authors report an increase in the students' motivation, and the subjective analysis of instructors was positive.

— *SQLator* [43] is a learning workbench for SQL that contains a multimedia tutorial as well as practice questions. It supports only SQL query problems, judged primarily by a set of equivalence heuristics instead of execution; extra feedback from the instructors is available via messages that can be sent directly from SQLator. It has a web interface and plagiarism checking, but its source code is not available. It was informally evaluated, showing an improvement in students' performance compared to other years when SQLator was not used, high student engagement, and a decrease in plagiarism.

— *SQLify* [15] is a web system very similar to *aSQLg*: it computes a numerical score for SQL query problems based on the similarities with the expected solution. Only informal evaluation has been conducted.

— *SQLTester* [22] is a web system based on Active SQL. It executes the student's solution and compares the obtained and expected results, but it is restricted to SQL query problems. Unlike the rest of the tools, it is open source software with an MIT license, although its development seems to have stopped in 2018. It provides a deep evaluation based both on student questionnaires and also statistical evidence that the students improved by using SQLTester. Concretely, they found a correlation between the number of practice tests taken by the students in the system and the final score, based on 79 students. It is worth noticing that this system is focused on evaluation rather than in learning, and hence the exercises have to be completed in a limited time.

— *SQL-Tutor* [31] is a system that personalizes the learning experience by choosing the questions to show based on the student's previous results. It supports only SQL query problems that are judged by executing them in Ingres Database, although it can provide rich feedback with messages obtained by CBM. SQL-Tutor has both a web and local interface, however, its code is not publicly available. Its empirical evaluation showed that students who used the system obtained better marks than those who did not. They also analyzed the effect of gamification [51], confirming some positive effects.

— *SQLZoo* [13] was initially developed at Edinburgh Napier University as a tool for students to learn SQL at their own pace by solving SQL exercises. It features instant feedback on the success of the student's attempts for several sample databases, and supports several SQL

engines (MySQL, PostgreSQL, and SQL Server). Being a Wiki-based web tool, teachers can edit and add practices and exercises, typically grouped by different difficulty levels.

7.2 Reported Assessments in the Literature

This section summarizes the various types of analyses (carried out by the tools' authors) using the tools presented in the previous section.

7.2.1 Objective Analysis. In *Active SQL* [42], the authors present the results of two consecutive courses: the first without using their tool and the second using it. Their statistics measure the number of students who worked at a regular pace through the material, concluding that there appears to be a 20% improvement in the second course compared to the first in terms of the average student target. They also highlight a slight negative impact on good students in the second course, meaning fewer students worked hard enough to be considered as following a "good student" time plan. In a follow-up article [14], they include a third course, where the student assessment, instead of containing two questions with a difficulty level related to the tutorial just completed, contained one question of that difficulty level and one question of the previous difficulty level. The effect of this change is that marks increased (from 51% to 64%) and the number of questions attempted also increased (from 4.6 to 5.8, out of a total of 8 questions).

Aplicación BD [16] poses four RQs in its analysis: *Academic Performance and Code Copies*, *Gender Analysis*, *Influence of Reviewing on Academic Performance and Copying*, and *Satisfaction with the Learning Tool*. Only two groups for an academic year were tested: the first one had to use the learning tool, and the second one might use it for self-learning. The findings revealed better grading (a difference from 0.41 to 1.89 in consecutive exams), less code copying (particularly for women, with a reduction of 27% compared to men), and more satisfaction in the second group (3.51 vs. 2.81 in the first group). Additionally, students who copied less and invested more effort obtained better results (with a maximum difference of marks in the last exam of 6.2 vs. 3.37). Moreover, students who did more voluntary exercises but copied exercises from other students obtained worse results than students who copied fewer exercises (4.71 vs. 6.2 in the last exam).

In *QueryCompetition* [32], the authors analyze one RQ: *How do challenges, points, and leaderboards impact student performance, motivation, and user experience in students using QC?* They conducted the analysis with two groups in a course: one using the tool with gamification (G) and another without using it (N). The analysis reveals that the grades obtained by students in G were higher than those in N, showing a better differential in performance for G ($\mu = 4.31$, median = 4) compared to N ($\mu = 1.70$, median = 2).

SQLTester [22] presents an assessment of the impact of their tool on a single group (there is no control group of students). First, it shows student engagement, indicating that students took an average of 10 practice tests each, spending over 4 hours actively engaged in those tests. Second, it demonstrates student performance, revealing that the more practice tests a student took, the higher their marks on the final test. A rising plot of the number of tests vs. average marks is provided, ranging from marks 4 to 8, and 1 to 20 tests.

In [30], the authors of *SQL-Tutor* include an analysis of the tool for a group of students divided into those who chose to use the tool and those who did not. The score for the students who used the system was 82.7, while the corresponding score for those who did not was 71.2 (an average difference of 11.5 marks). They also present a couple of objective analyses of the learning curve, considering the number of errors (constraint violations) per solution attempt. The first analysis shows that the probability of violating a constraint decreased in a negatively accelerated fashion with an increasing number of opportunities to acquire the knowledge embedded in that constraint. In the second analysis, the proportion of subjects who violated no constraints increased smoothly

and rapidly across occasions of applicability. In [51], the causal effects of gamification on learning in SQL-Tutor are analyzed, with three RQs: *RQ1: What are the effects of gamification on learning?*, *RQ2: Do students with different levels of prior knowledge react differently to gamification?* and *RQ3: What is the effect of gamification on student motivation?* The experimental procedure considered two groups: Experimental and Control (with and without using the tool). Results for *RQ1* show that, in the experimental group, time-on-task is positively correlated with learning outcomes, and badges have an indirect effect on learning outcomes by influencing time-on-task. However, there is no evidence to support that students receiving badges are more engaged with the tool by spending more time-on-task. Regarding *RQ2*, results show that prior knowledge affects the time students spend in the tools and that badges moderate this relationship. *RQ3* is addressed through the analysis of questionnaire data, as described in the following section.

7.2.2 Subjective Analysis. In addition to the objective analysis, several of these tools present a subjective analysis based on questionnaires filled out by students (*Active SQL* [42], *SQL-ACME* [47], *SQLator* [43], *SQLify* [15], and *SQLZoo* [13] do not include such a subjective analysis).

In *Aplicación BD* [16], the response to the satisfaction with the learning tool in general for the two groups was 2.81 in the mean (0.82 standard deviation) and 3.51 (0.61) using a four-point Likert scale.

AsseSQL [41], without providing data, claims that student feedback is still overwhelmingly positive, viewing AsseSQL as a fair, consistent assessment approach.

aSQLg [23] only includes this subjective analysis and concludes that the improvement in student learning through the tool's immediate feedback (40%) and allowing for refined student responses is confirmed by the questionnaire responses. However, the most negative aspect is that the tool does not value the way of solving problems (30%).

QueryCompetition [32] surveys (i) motivation, where students in the G group felt more interested (79.4% vs. 59.2%), more focused (64.7% vs. 49.0%), and more satisfied (52.9% vs. 44.9%) than the students in the N group; (ii) better user experience, where 5 of 14 aspects of this sub-component were better perceived in the G group than in the N group. As a negative effect, participants in the G group felt more anxious (64.7% vs. 55.1%) and stressed (64.7% vs. 42.9%); and (iii) learning, both for short- and long-term learning, were better perceived in the G group.

SQLTester [22] includes the results of an anonymous questionnaire consisting of eight statements with a five-point Likert scale. Most students (from 73.5% to 97.1%) agree or strongly agree with the statements, overall supporting the quantitative results that SQLTester motivated students to practice SQL and practice longer than without the tool.

In *SQL-Tutor* [51], the authors use two surveys (1 and 2) to answer research question *RQ3*. Results revealed a significant positive relation between badges and time-on-task, and that topic interest did not directly motivate students to spend more time on SQL-Tutor, indicating needed interventions to raise this motivation. Survey 3 was used to assess students' self-testing behavior. The responses of the experimental group indicated that students did not find badges very motivating, though 39% of them wanted to see the badges.

7.3 Error Coverage

Several works, including [1, 6, 27, 55], have detailed categorizations of SQL errors, mostly focusing on syntax and semantic errors. While a *syntax error* represents a compilation error, a semantic error shows an unexpected use of SQL features which does not always produce the intended results [6]. In [55], semantic errors are further classified: queries that are incorrect regardless of the *data demand* (i.e., intended interpretation of the query as expressed in natural language) are said to expose a *semantic error*, while queries that are incorrect for a particular data demand contain a *logical error*.

In addition, they include *complications* in their categorization, that is, queries containing features not required for a correct outcome (such as adding an unnecessary table to a join). While a judge system based on test sets can correctly identify true negatives, it can also produce false positives (e.g., when a student intentionally crafts a solution that matches the expected output without genuinely solving the problem). These cases, known as *deceptive errors*, have been categorized into several types [56]. To the best of our knowledge, no existing tool currently detects such errors, thus representing a promising line for future work.

Next, the error coverage of the automatic judge LearnSQL is assessed in relation to the most recent and comprehensive error classification [55]. We highlight what DES adds to informative error messages with respect to a typical DBMS such as the host Oracle used in this automatic judge.

7.3.1 Syntax Errors. These errors are handled by any DBMS, but the degree of detail may vary between them [53]. From a student's point of view, a detailed error message providing as many clues as possible is desirable. There are several syntax errors covered by LearnSQL by means of the underlying DBMS host Oracle and the connected tool DES which are issued to the student via error messages. While Oracle displays the error message, DES in addition displays the location of the error in the query and provides in general a more detailed message. Syntax errors in [55] are entitled as "*SYN-i Description*" and are discussed below:

— *SYN-1 Ambiguous Database Object*: DES displays which object is ambiguous and its class (e.g., *Error: Ambiguous column name "a"*), while Oracle does not[6] (e.g., *ORA-00918: column ambiguously defined*).
— *SYN-2 Undefined Database Object*: While Oracle simply returns the message "*table or view does not exist*" without even indicating which object it refers to, DES does provide this information and also follows a DWIM approach by looking for close matches and displaying them, such as in the message "*Error: Unknown table or view 'tacas'. Info: Possible relations (respect case): [tapas,taras].*"
— *SYN-3 Data Type Mismatch*: Oracle simply displays "*ORA-00902: invalid datatype,*" whereas DES issues a more detailed message, for example, *Error: (SQL) Expected argument type number(A) after: select sin(cast(a as string)).*
— *SYN-4 Illegal Aggregate Function Placement*: While Oracle simply displays "*ORA-00934: group function is not allowed here*" (where "*here*" refers to the whole query), DES adds context information: "*Error: This aggregate is not allowed in the WHERE clause: MIN(a).*"
— *SYN-5 Illegal or Insufficient Grouping*: Oracle displays "*ORA-00937: not a single-group group function,*" while DES indicates the source of the error, for example: *Error: This statement has a non-grouped attribute "t.a" in the SELECT clause.*
— *SYN-6 Common Syntax Error*: This category includes errors such as "misspellings of SQL keywords, missing semicolon, brackets or commas, projection in a wrong clause, incorrect clause ordering and missing clauses" [55], where LearnSQL also provides in general more details and context than Oracle alone.

7.3.2 Semantic Errors. To the best of our knowledge, no RDBMS supports reporting this kind of errors. However, the `sqllint`[7] analysis tool in [6] includes a limited number of their semantic error categorizations (identified by natural numbers 1…39). Comparing with DES (as used by LearnSQL) both systems deal with error numbers 1, 2, 3, 4, 5, 8, and 27; in addition to this, `sqllint` deals with

[6]These syntax error messages have been obtained from Oracle Database 19c, which are almost the same as in previous versions.
[7]https://dbs.informatik.uni-halle.de/sqllint/.

errors 34 and 39, while DES deals with errors 6, 7, 9, 11, 12, 13, 16, 17, 32, and 33. The automatic judge issues warning messages to the student for this kind of errors. Those errors present in a query which in addition leads to an unexpected data output are warned as non-valid. With respect to the coverage of the SQL language itself, DES analyzes full subqueries, while `sqllint` only supports `EXISTS` (no `IN`, `> = ALL`, ...). `sqllint` neither supports aggregates, nor `UNION`, nor `LIKE`, and nor `IS [NOT] NULL`. As types, it includes only strings and integers, and expressions are not allowed. Finally, it does not support `CHECK` constraints in table definitions. [50] lists the supported semantic errors in DES, and describes its constraint-solving-based approach to determine both inconsistent and tautological conditions (errors 1 and 8, respectively). There, solvers for integers, floats and strings are included, and domain cooperation between numeric constraints allows for refining the analysis. With respect to [55], the following semantic errors are listed:

— *SEM-1 Inconsistent Expression (an Expression Making the Result Table to Be Empty or to Contain All Rows)*: Covered by errors 1 and 8.
— *SEM-2 Inconsistent Join (a Join on Wrong Columns)*: Correctness check (no semantic warning).
— *SEM-3 Missing Join*: Covered by error 27.
— *SEM-5 Redundant Column Output*: Covered by error 4.
— *SEM-4 Duplicate Rows*: Correctness check. A semantic warning would be a straightforward addition to DES.

7.3.3 Logical Errors. A logical error is caught in LearnSQL via the correctness check, that is, whether the result of the submitted query does not match the expected answer for the data demand. This does not ensure that any logical error is caught because a query returning the rows in the expected answer for a given input database instance may return incorrect rows for another instance. Further, one can explicitly build a correct answer with a union of `FROM`-less `SELECT` statements (i.e., extensionally instead of intensionally), which would be considered as cheating the system (but not detected as such for now). For example, if the expected answer is simply a numeric value *N* possibly resulting of an aggregation, the query `SELECT N FROM dual;` would be accepted by the system. Though this tool is targeted to learning and it makes little sense for students to cheat, it is clearly a subject for future work. Ref. [55] identifies 6 categories for logical errors from *LOG-1 Operator error* to *LOG-6 Function error*, including 30 errors (from 52 to 81).

7.3.4 Complications. A complication in a query does not affect its result, but it may affect performance and obscures readability. Ref. [55] refers to this error classification in [6, Section 2]. The first error in this classification is the inconsistent condition (error 1) which would return an empty row set. Thus, we understand this as a semantic error. DES in this case is able to detect inconsistent conditions considering `CHECK` predicates in table definitions in selected cases. For example, consider a table creation query with `CHECK grade BETWEEN 0 AND 10`, and later a condition `grade > 10`, which the system detects as inconsistent and warns the user about. Other complications in this category that would lead a semantic error are: *3: Constant output column*, *4: Duplicate output column*, *8: Implied, tautological, or inconsistent subcondition*, *9: Comparison with* `NULL`, *10:* `NULL` *value in* `IN/ANY/ALL` *subquery*. Complications other than the former are: *2: Unnecessary* `DISTINCT`, *5: Unused tuple variable*, *6: Unnecessary join*, *7: Tuple variables are always identical*, *11: Unnecessarily general comparison operator*, *12:* `LIKE` *without wildcards*, *13: Unnecessarily complicated* `SELECT` *in* `EXISTS`*-subquery*, *14:* `IN/EXISTS` *condition can be replaced by comparison*, and errors in aggregation functions (15 to 17), in the `GROUP BY` clause (18 to 22), in the `HAVING` clause (23) and in the `ORDER BY` clause (24). Coverage of these errors by DES was already reported in Section 7.3.2, which shows room for improvement by developing additional warnings.

7.4 Summary of Strengths and Weaknesses of LearnSQL

As shown in the table and described in Section 7.1, all the tools are available as web-based systems, which is logical for studies in Computer Science. Moreover, together with the error coverage described in Section 7.3, this allows for identifying the following strengths in the automatic judge LearnSQL:

- —It is open source. This is possibly the most important strength of LearnSQL because it gives a number of important advantages: (i) it does not fully depend on an institution for being maintained, largely extending its lifespan; (ii) the software is much more transparent and flexible, allowing other users to add the features they require without depending on the original developers; and (iii) adding up the previous item, it makes it possible to create a community of developers that collaborate to fix and improve the tool.
- —It supports more types of problems than most of the tools. In particular, DML is not supported by most of the tools and it provides added value to LearnSQL, making it useful for more advanced exercises. In addition, no other tool supports EPL, which allows for including exercises based on procedures, functions, and triggers.
- —Unlike other tools, it typically provides more detailed feedback on syntax errors than native SQL engines, thereby assisting students in identifying and correcting their mistakes.
- —It provides semantic feedback about syntactically correct queries geared toward providing hints to better formulate a query, and ways to fix a suspicious one by indicating bad uses of SQL (this includes warnings about semantic errors and complications). Moreover, given a correct answer to an exercise, it can warn the student about unnecessary complex uses of SQL (e.g., unneeded DISTINCT, extra tables…). No other tool provides such hints.
- —Summarizing, it brings together different types of feedback and exercises in a single tool. As shown in Section 7.2, using a learning tool that supports students in different ways is confirmed to be useful in both objective and subjective analyses.

As a weakness, LearnSQL supports only one DBMS, Oracle, and should consider supporting more engines as future work, making it more appropriate for other courses. However, the open source nature of LearnSQL makes it possible for other institutions to adapt it to their necessities, as explained before.

Summarizing, we consider the features already implemented (in particular, the support of DML and EPL) and its versatility makes LearnSQL a very good option for use in the classroom. Moreover, the results presented in the previous sections illustrate its positive impact on learning.

8 Conclusion and Future Work

In this article, we have evaluated the real impact on learning when using the online automatic judge LearnSQL for free practice in a database course over four academic years. LearnSQL is an online automatic judge for automatically assessing the correctness of database exercises, including SQL queries, functions, procedures, triggers, and data manipulation, among others. LearnSQL has been designed to provide verbose explanations to students in order to help them detect their errors and fix them. For the evaluation we have considered students of the Faculty of Computer Science of the Complutense University of Madrid in the academic year 2019–2020, when the automatic judge was not used, and the academic years 2021–2022, 2022–2023, and 2023–2024, when the automatic judge was available. Using the automatic judge has shown statistically significant improvements on final marks when applied in a group of the single degree on CSE, which can be quantified in mean between 0.55 and 1.74 points out of 6. Moreover, we have also detected a direct correlation between the number of exercises tried in the automatic judge and the final mark, with an improvement

of more than one point out of six between the 20% of students who have practiced the least and the 40% who have practiced the most. We have not detected statistically significant positive or negative effect on final marks when the automatic judge is applied in the double-degree program on CSM/M, where students are top-performing and highly motivated. However, we have observed a direct correlation between the number of exercises tried in the automatic judge and the final mark for these students, although the effect is milder compared to that observed in single-degree students. Additionally, the analysis of student questionnaires have shown a great acceptance of the automatic judge, which is subjectively perceived as a very useful tool for database learning.

As future work, we plan to extend the automatic judge LearnSQL in three directions. First, we want to support more DBMSs (e.g., PostgreSQL or MySQL), as done in other similar tools. As the SQL executor engine is defined as an isolated component of the system, this task should not imply deep changes in the codebase. Second, we also want to investigate how to improve the feedback provided to the student and generate even further information. In this context, we are interested on integrating LLM. This approach has been investigated in a previous work [40], and our findings confirm that the hints generated by LLMs for SQL are highly informative, even without extensive post-processing or refinement. However, caution must be exercised with such LMMs, which are not based on formal methods, as they may produce inconsistent output. Finally, we plan to integrate the automatic judge into Moodle, the learning platform used in our virtual learning environment. Following an approach similar to that in [7], LearnSQL could be used to validate student assignments and directly assign marks based on the feedback generated by the automatic judge, thereby providing students with a unified interface for the entire subject. Once the automatic judge is extended, we would like to replicate the evaluation to statistically validate if the new enhancements have a positive improvement on learning outcomes.

Additionally, another line for future work would be to revisit the evaluation of the LearnSQL automatic judge, with a focus on subjective learning outcome measures, such as students' confidence in SQL, their self-assessment of skills, and their ability to solve novel problems. To accomplish this, the study design would need to incorporate specific questions into the questionnaires to assess these subjective aspects, while also implementing incentive mechanisms to encourage meaningful participation. By integrating these subjective measures with the objective assessment based on exam performance, as considered in this study, we could achieve a more comprehensive understanding of the automatic judge's impact on learning outcomes.

References

[1] Alireza Ahadi, Vahid Behbood, Arto Vihavainen, Julia Prior, and Raymond Lister. 2016. Students' syntactic mistakes in writing seven different types of SQL queries and its application to predicting students' success. In *47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, 401–406. DOI: https://doi.org/10.1145/2839509.2844640

[2] Lorin W. Anderson and David R. Krathwohl. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives: Complete Edition*. Addison Wesley Longman, Inc.

[3] Enrique Martin-Martin, Manuel Montenegro, Adrián Riesco, Rubén Rubio, and Fernando Sáenz-Pérez. 2024. LearnSQL: Impact of an Automatic Judge in Database Learning (Dataset). DOI: https://doi.org/10.5281/zenodo.11181272

[4] Brittany Bloodhart, Meena M. Balgopal, Anne Marie A. Casper, Laura B. Sample McMeeking, and Emily V. Fischer. 2020. Outperforming yet undervalued: Undergraduate women in STEM. *PloS One* 15, 6 (2020), e0234685. DOI: https://doi.org/10.1371/journal.pone.0234685

[5] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. David McKay Company.

[6] Stefan Brass and Christian Goldberg. 2006. Semantic errors in SQL queries: A quite complete list. *Journal of Systems and Software* 79, 5 (2006), 630–644. DOI: https://doi.org/10.1016/j.jss.2005.06.028

[7] José L. Brita-Paja, Carlos Gregorio, Luis Llana, Cristóbal Pareja, and Adrián Riesco. 2019. Introducing MOOC-like methodologies in a face-to-face undergraduate course: A detailed case study. *Interactive Learning Environments* 27, 1 (2019), 15–32. DOI: https://doi.org/10.1080/10494820.2018.1451345

[8] Luca Cagliero, Luigi De Russis, Laura Farinetti, and Teodoro Montanaro. 2018. Improving the effectiveness of SQL learning practice: A data-driven approach. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 01, IEEE, 980–989. DOI: https://doi.org/10.1109/COMPSAC.2018.00174

[9] Luca Cagliero, Laura Farinetti, Jacopo Fior, and Andrea Ignazio Manenti. 2024. ChatGPT, be my teaching assistant! Automatic correction of SQL exercises. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 81–87. DOI: https://doi.org/10.1109/COMPSAC61105.2024.00021

[10] Nancy Carr, Farjahan R. Shawon, and Hasan M. Jamil. 2023. An experiment on leveraging ChatGPT for online teaching and assessment of database students. In *2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 1–8. DOI: https://doi.org/10.1109/TALE56641.2023.10398239

[11] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences* (2nd. ed.). Lawrence Erlbaum Associates. DOI: https://doi.org/10.4324/9780203771587

[12] William E. Copeland, Ellen McGinnis, Yang Bai, Zoe Adams, Hilary Nardone, Vinay Devadanam, Jeffrey Rettew, and Jim J. Hudziak. 2021. Impact of COVID-19 pandemic on college student mental health and wellness. *Journal of the American Academy of Child and Adolescent Psychiatry* 60, 1 (2021), 134–141.e2. DOI: https://doi.org/10.1016/j.jaac.2020.08.466

[13] Andrew Cumming. 2020. SQL Zoo. Intelligent Growth Solutions. Retrieved from https://sqlzoo.net/

[14] Andrew Cumming and Gordon Russell. 2005. Automatic checking of SQL: Computerised grading. *International Journal of Learning: Annual Review* 12 (2005), 127–134. DOI: https://doi.org/10.18848/1447-9494/CGP/v12i03/46714

[15] Michael de Raadt, Stijn Dekeyser, and Tien Yu Lee. 2006. Do students *SQLify*? Improving learning outcomes with peer review and enhanced computer assisted assessment of querying skills. In *6th Baltic Sea Conference on Computing Education Research, Koli Calling (Baltic Sea '06)*. Anders Berglund and Mattias Wiggberg (Eds.), ACM, 101–108. DOI: https://doi.org/10.1145/1315803.1315821

[16] César Domínguez, Arturo Jaime, Jónathan Heras, and Francisco J. García-Izquierdo. 2019. The effects of adding non-compulsory exercises to an online learning tool on student performance and code copying. *ACM Transactions on Computing Education* 19, 3 (2019), 22 pages. DOI: https://doi.org/10.1145/3264507

[17] Akın Efendioğlu and T. Yelken. 2010. Programmed instruction versus meaningful learning theory in teaching basic structured query language (SQL) in computer lesson. *Computers & Education* 55 (2010), 1287–1299. DOI: https://doi.org/10.1016/j.compedu.2010.05.026

[18] Ramez Elmasri and Shamkant B. Navathe. 2017. *Fundamentals of Database Systems* (7th. ed.). Pearson Harlow.

[19] CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education*. ACM, New York, NY.

[20] Nicky Gerritsen, Thijs Kinkhorst, and Tobias Werth. 2025. *DOMjudge 8.2 Manual*. DOMjudge.

[21] Musti K. S. Sastry. 2015. An effective approach for teaching database course. *International Journal of Learning, Teaching and Educational Research* 12, 1 (2015), 53–63.

[22] Anthony Kleerekoper and Andrew Schofield. 2018. SQL tester: An online SQL assessment tool and its impact. In *23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '18)*. Irene Polycarpou, Janet C. Read, Panayiotis Andreou, and Michal Armoni (Eds.). ACM, 87–92. DOI: https://doi.org/10.1145/3197091.3197124

[23] Carsten Kleiner, Christopher Tebbe, and Felix Heine. 2013. Automated grading and tutoring of SQL statements to improve student learning. In *13th Koli Calling International Conference on Computing Education Research (Koli Calling '13)*. Mikko-Jussi Laakso and Simon (Eds.). ACM, 161–168. DOI: https://doi.org/10.1145/2526968.2526986

[24] Wilfred Wing Fat Lau and Allan Hoi Kau Yuen. 2010. Gender differences in learning styles: Nurturing a gender and style sensitive computer science classroom. *Australasian Journal of Educational Technology* 26 (2010), 1090–1103. DOI: https://doi.org/10.14742/AJET.1036

[25] Yang Li, Aiwen Wang, Yalin Wu, Nana Han, and Huiming Huang. 2021. Impact of the COVID-19 pandemic on the mental health of college students: A systematic review and Meta-Analysis. *Frontiers in Psychology* 12 (2021), 669119. DOI: https://doi.org/10.3389/fpsyg.2021.669119

[26] Enrique Martin-Martin, Manuel Montenegro, Adrián Riesco, and Rubén Rubio. 2022. Improving database learning with an automatic judge. In *The 34th International Conference on Software Engineering and Knowledge Engineering (SEKE '22)*. Rong Peng, Carlos Eduardo Pantoja, and Pankaj Kamthan (Eds.). KSI Research Inc., 499–502. DOI: https://doi.org/10.18293/SEKE2022-025

[27] Daphne Miedema, Efthimia Aivaloglou, and George Fletcher. 2021. Identifying SQL misconceptions of novices: Findings from a think-aloud study. In *17th ACM Conference on International Computing Education Research (ICER '21)*. ACM, New York, NY, 355–367. DOI: https://doi.org/10.1145/3446871.3469759

[28] Daphne Miedema, George Fletcher, and Efthimia Aivaloglou. 2022. So many brackets! An analysis of how SQL learners (mis)manage complexity during query formulation. In *2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC)*. IEEE, 122–132. DOI: https://doi.org/10.1145/3524610.3529158

[29] Ivanović Mirjana, Putnik Zoran, Sisarica Anja, and Budimac Zoran. 2010. A note on performance and satisfaction of female students studying computer science. *Innovation in Teaching and Learning in Information and Computer Sciences* 9 (2010), 32–41. DOI: https://doi.org/10.11120/ital.2010.09010032

[30] Antonija Mitrovic and Stellan Ohlsson. 1999. Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education* 10 (1999), 238–256.
[31] Antonija Mitrovic and Stellan Ohlsson. 2016. Implementing CBM: SQL-Tutor after fifteen years. *International Journal of Artificial Intelligence in Education* 26, 1 (2016), 150–159. DOI: https://doi.org/10.1007/S40593-015-0049-9
[32] Miguel Ehécatl Morales-Trujillo and Gabriel Alberto García-Mireles. 2021. Gamification and SQL: An empirical study on student performance in a database course. *ACM Transactions on Computing Education* 21, 1 (2021), 29 Pages. DOI: https://doi.org/10.1145/3427597
[33] Stellan Ohlsson. 1992. Constraint-based student modelling. *Journal of Interactive Learning Research* 3, 4 (1992), 429. DOI: https://doi.org/10.1007/978-3-662-03037-0_7
[34] Claus Pahl, Ronan Barrett, and Claire Kenny. 2004. Supporting active database learning and training through interactive multimedia. In *9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*. ACM, New York, NY, 27–31. DOI: https://doi.org/10.1145/1007996.1008007
[35] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education* 22, 3 (2022), 40 pages. DOI: https://doi.org/10.1145/3513140
[36] Filipe D. Pereira, Elaine H. T. Oliveira, David B. F. Oliveira, Alexandra I. Cristea, Leandro S. G. Carvalho, Samuel C. Fonseca, Armando Toda, and Seiji Isotani. 2020. Using learning analytics in the Amazonas: Understanding students' behaviour in introductory programming. *British Journal of Educational Technology* 51 (2020), 955–972. DOI: https://doi.org/10.1111/bjet.12953
[37] Raymond Scott Pettit, John D. Homer, Kayla Michelle McMurry, Nevan Simone, and Susan A. Mengel. 2015. Are automated assessment tools helpful in programming courses? In *2015 ASEE Annual Conference & Exposition*, 26–230.
[38] Putsadee Pornphol and Suphamit Chittayasothorn. 2024. Using LLM artificial intelligence systems as complex SQL programming assistants. In *2024 12th International Conference on Information and Education Technology (ICIET)*. IEEE, 477–481. DOI: https://doi.org/10.1109/ICIET60671.2024.10542806
[39] Seth Poulsen, Liia Butler, Abdussalam Alawini, and Geoffrey L. Herman. 2020. Insights from student solutions to SQL homework problems. In *2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20)*. ACM, New York, NY, 404–410. DOI: https://doi.org/10.1145/3341525.3387391
[40] Kishore Prakash, Shashwat Rao, Rayan Hamza, Jack Lukich, Vatsal Chaudhari, and Arnab Nandi. 2024. Integrating LLMs into database systems education. In *3rd International Workshop on Data Systems Education: Bridging Education Practice with Education Research (DataEd '24)*. ACM, New York, NY, 33–39. DOI: https://doi.org/10.1145/3663649.3664371
[41] Julia R. Prior. 2014. AsseSQL: An online, browser-based SQL skills assessment tool. In *Innovation and Technology in Computer Science Education Conference 2014 (ITiCSE '14)*. Åsa Cajander, Mats Daniels, Tony Clear, and Arnold Pears (Eds.), ACM, 327. DOI: https://doi.org/10.1145/2591708.2602682
[42] Gordon Russell and Andrew Cumming. 2004. Improving the student learning experience for SQL using automatic marking. In *Cognition and Exploratory Learning in Digital Age (CELDA '04), IADIS International Conference*. Kinshuk, Demetrios G. Sampson, and Pedro T. Isaías (Eds.), IADIS, 281–288.
[43] Shazia Wasim Sadiq, Maria E. Orlowska, Wasim Sadiq, and Joe Y.-C. Lin. 2004. SQLator: An online SQL learning workbench. In *9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*. Roger D. Boyle, Martyn Clark, and Amruth N. Kumar (Eds.), ACM, 223–227. DOI: https://doi.org/10.1145/1007996.1008055
[44] William R. Shadish and Jason K. Luellen. 2012. Quasi-experimental design. In *Handbook of Complementary Methods in Education Research*. Routledge, 539–550.
[45] D. Shell and Leen-Kiat Soh. 2013. Profiles of motivated self-regulation in college computer science courses: Differences in major versus required non-major courses. *Journal of Science Education and Technology* 22 (2013), 899–913. DOI: https://doi.org/10.1007/s10956-013-9437-9
[46] Abraham Silberschatz, Henry F. Korth, and Shashank Sudarshan. 2019. *Database System Concepts* (7th. ed.). McGraw-Hill.
[47] Josep Soler, Ferran Prados, Imma Boada, and Jordi Poch. 2006. A web-based tool for teaching and learning SQL. In *International Conference on Information Technology Based Higher Education and Training, ITHET*.
[48] Gijsbert Stoet and David C. Geary. 2018. The gender-equality paradox in science, technology, engineering, and mathematics education. *Psychological Science* 29, 4 (2018), 581–593. DOI: https://doi.org/10.1177/0956797617741719
[49] Fernando Sáenz-Pérez. 2011. DES: A deductive database system. *Electronic Notes in Theoretical Computer Science* 271 (2011), 63–78. DOI: https://doi.org/10.1016/j.entcs.2011.02.011
[50] Fernando Sáenz-Pérez. 2019. Applying constraint logic programming to SQL semantic analysis. *Theory and Practice of Logic Programming* 19, 5–6 (2019), 808–825. DOI: https://doi.org/10.1017/S1471068419000206

[51] Faiza Tahir, Antonija Mitrovic, and Valerie Sotardi. 2022. Investigating the causal relationships between badges and learning outcomes in SQL-Tutor. *Research and Practice in Technology Enhanced Learning* 17, 1 (2022), 7. DOI: https://doi.org/10.1186/s41039-022-00180-4

[52] Toni Taipalus. 2023. SQL: A Trojan horse hiding a decathlon of complexities. In *2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research (DataEd '23)*. ACM, New York, NY, 9–13. DOI: https://doi.org/10.1145/3596673.3603142

[53] Toni Taipalus, Hilkka Grahn, and Hadi Ghanbari. 2021. Error messages in relational database management systems: A comparison of effectiveness, usefulness, and user confidence. *Journal of Systems and Software* 181 (2021), 111034. DOI: https://doi.org/10.1016/j.jss.2021.111034

[54] Toni Taipalus and Ville Seppänen. 2020. SQL education: A systematic mapping study and future research agenda. *ACM Transactions on Computing Education* 20, 3 (2020), 33 pages. DOI: https://doi.org/10.1145/3398377

[55] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and complications in SQL query formulation. *ACM Transactions on Computing Education* 18, 3 (2018), 29 pages. DOI: https://doi.org/10.1145/3231712

[56] Jinshui Wang, Shuguang Chen, Zhengyi Tang, Pengchen Lin, and Yupeng Wang. 2024. False positives and deceptive errors in SQL assessment: A large-scale analysis of online judge systems. *ACM Transactions on Computing Education* 24, 3 (2024), 23 pages. DOI: https://doi.org/10.1145/3654677

[57] Xiaomei Wang, Sudeep Hegde, Changwon Son, Bruce Keller, Alec Smith, and Farzan Sasangohar. 2020. Investigating mental health of US college students during the COVID-19 pandemic: Cross-Sectional survey study. *Journal of Medical Internet Research* 22, 9 (2020), e22817. DOI: https://doi.org/10.2196/22817

Received 24 May 2024; revised 17 September 2025; accepted 18 September 2025