Mining SQL Problem Solving Patterns using Advanced Sequence Processing Algorithms

Sophia Yang
University of Illinois
Urbana-Champaign
USA
sophiay2@illinois.edu

Geoffrey L. Herman
University of Illinois
Urbana-Champaign
USA
glherman@illinois.edu

Abdussalam Alawini
University of Illinois
Urbana-Champaign
USA
alawini@illinois.edu

ABSTRACT

SQL is a crucial language for managing relational database systems, and is an essential skill for individuals in roles such as researchers, developers, and business professionals who work with databases. However, learning SQL can be a challenge, presenting an opportunity to study the various methods students use to arrive at semantically equivalent SQL queries. In this study, we examined students' SQL submissions to homework assignments in the Database Systems course offered to upper-level undergraduate and graduate students at the University of Illinois Urbana-Champaign during the Fall 2022 semester. Our goal was to understand how students arrive at SQL solutions and overcome challenges in the learning process by building on prior research on line chart visualizations that instructors can use to increase visibility on students who are struggling. However, a major limitation of this approach was the difficulty for instructors to sift through a large number of visuals representing each student's performance on a SQL problem and generate action items at scale, especially when dealing with enrollments of over 700 students. To overcome this limitation, we developed a novel technique to generate textual representations of the student submission sequence using global sequence alignment scores and regular expression algorithms to further compact these submission sequences. This allows instructors to gain insights quickly, on an aggregate level, and in an automated manner, enabling them to identify students who may be struggling with SQL based on their submission sequence characteristics and take appropriate action to improve database education. Our study discovered common textual submission patterns and pattern elements, and we present our recommendations to instructors to improve database education based on these findings.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Social and professional topics** → **Computing education**.

KEYWORDS

SQL, database education, online assessment, pattern mining, sequence alignment, compaction, regular expression, algorithms

DataEd '23, June 23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0207-5/23/06.
https://doi.org/10.1145/3596673.3596973

ACM Reference Format:
Sophia Yang, Geoffrey L. Herman, and Abdussalam Alawini. 2023. Mining SQL Problem Solving Patterns using Advanced Sequence Processing Algorithms. In *2nd International Workshop on Data Systems Education: Bridging education practice with education research (DataEd '23), June 23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3596673.3596973

1 INTRODUCTION

SQL is the primary language for managing data that is widely supported by most Database Management Systems, making it an essential skill for users, developers, and researchers who interact with databases [11]. The language has a highly structured, English-like syntax that makes itself accessible to beginners, since it does not depend on expertise in other programming languages. Despite this, as noted in prior research, some students still encounter challenges in learning SQL, and the need to analyze how students come up with their SQL solutions and the difficulties they face is crucial [22].

To improve the efficiency and effectiveness of analyzing how students arrive at their SQL solution query, it would be beneficial for database instructors to examine student SQL submissions in an automated manner. In particular, instructors who use auto-graders to accept student submissions for SQL problems can gain valuable insights by tracing students' attempts and progress towards constructing the correct solution. These insights can assist instructors in identifying challenging SQL concepts that students commonly struggle with and adjusting their instructional strategies to address these challenges. By analyzing student submissions in an automated way, instructors can optimize their teaching and support students in acquiring SQL skills more efficiently.

Our research work builds upon and extends our previous research work on line chart visualization types that display a student's submission pattern based on the sequence alignment score between each of their submissions on a SQL problem and their final submission [32]. A major limitation of the work includes the scalability factor of the visuals - in a class of a few hundred students, it's not practical to have instructors sift through all of the line chart graphs to determine which students are struggling apart from ones who are succeeding, because:

$$\textit{Total \# of Visuals} = \textit{\# of Students} * \textit{\# of SQL Problems}$$

We address and overcome this limitation by generating textual representations of the student submission sequence based on the global sequence alignment score inputs, and then applying regular expression algorithms to compact and aggregate these textual representation submission sequences. Therefore, our research questions

include: 1) *what are the different types of SQL solution submission patterns students display?* and 2) *do certain submission patterns indicate students encountering obstacles in their learning path?*

Research has indicated that students may follow diverse learning paths and that their learning experience can be significantly improved if instructors can identify how individual students learn [14]. Our research aims to enhance the educational quality of students learning SQL in database courses, building upon the aforementioned findings. Specifically, we analyze the data from the Database Systems course at the University of Illinois Urbana-Champaign, which usually has over 400 enrollees; the Fall 2022 semester has over 700 enrollees. Our focus is on students' submissions to SQL homework assignments during the Fall 2022 semester. Throughout the course, students participate in multiple SQL in-class group exercises and an individual homework assignment with 10-15 SQL problems. These problems are automatically graded, and students receive immediate feedback after each submission. However, due to the large class size, it can be challenging for instructors to quickly identify common areas of difficulty or individual students in need of additional support. Furthermore, the number of submissions for each SQL problem is too numerous to be manually examined, as students typically submit more than 20 attempts per problem on average. The present paper presents a method for evaluating students' advancement as they attempt to solve an SQL problem, facilitating instructors in recognizing the diverse methods students use to solve the same SQL problem.

2 RELATED WORKS

Extensive research has been conducted on the difficulties and misunderstandings that students face when learning procedural programming languages like Java [12, 15, 17], C++ [5, 6, 15, 30], and Python [5, 15, 17]. However, there has been comparatively less research on declarative or database query languages such as SQL [2, 16, 18, 24, 26, 27]. Most of the existing studies have focused on the SQL problems and concepts that students commonly find challenging. For example, Taipalus et al. analyzed over 33,000 SQL queries submitted by students and found that students make a variety of syntax, semantic, and logic errors [27]. Some of the syntax errors they identified were previously reported in the literature, including undefined parameters, data type mismatch, and date time field overflow [2]. Taipalus and Perälä [26] investigated persistent error types that students encounter when forming SQL queries, as well as the SQL query concepts that give rise to such errors. Other studies have examined different types of SQL queries that students find challenging to write [3, 21], the most common SQL semantic errors [1], and semantic error categorizations by Brass and Goldberg [7].

While earlier studies have identified problematic SQL concepts and problems for students, few have delved into the reasons why students struggle with them. To gain insight into the causes behind these SQL misconceptions, a qualitative approach is necessary. In a think-aloud study, Miedema et al. [18] analyzed 21 students' problem-solving processes and identified four reasons for their SQL errors: interference from prior course knowledge, incorrect generalization of answers, flawed mental models, and confusion between SQL and natural language. Miedema et al. [20] also examined factors contributing to self-inflicted query complexity based on correctness, execution order, edit distance, and query intricacy.

In addition to exploring the challenges students face when learning SQL, researchers have investigated methods for visualizing and detecting these obstacles and learning approaches [10, 19, 31, 32]. For example, Miedema and Fletcher [19] developed the SQLVis system, which uses visual query representation (VQR) to help beginners develop proficiency in writing SQL queries. Similarly, Danaparamita and Gatterbauer [10] created QueryViz to enhance understanding of SQL queries through visualization. Authors of this paper have also developed visualization techniques, such as by utilizing edit distance and clustering, to detect learning obstacles and approaches through students' SQL submission sequences [31, 32].

Although prior research has explored why students encounter difficulties while writing SQL queries [3, 18, 24, 26], few recommendations have been made for SQL educators to take actionable steps. One suggestion is to improve SQL compilation error messages since students often struggle and give up when encountering syntax errors [2, 24]. Another recommendation is to address the misconceptions that arise from transferring prior knowledge from mathematics, natural language, and other programming languages [18], although this is not unique to teaching SQL and has been noted in other programming languages as well [9, 25]. Previous studies have also highlighted the advantages of generating automated instructor recommendations based on the different types of SQL solution submission patterns that students exhibit [31].

As far as we know, no previous research has explored the various types of SQL solution submission patterns exhibited by students, with the goal of identifying problematic patterns and advocating for automated instructor recommendations. In order to achieve this, we employ regular expression methodologies used in previous research [8, 28] to transform input data for line chart visualizations [32] into a string representation that is easier to analyze, aggregate, and understand (for transparency in data processing). Our objectives are two-fold, based on previous work: First, we want to examine the different SQL solution submission patterns that students display. Secondly, we aim to identify patterns of submission that may indicate students' struggle in learning how to write SQL queries, in order to provide practical recommendations to database instructors, enabling them to support students early on as they require assistance.

3 DATA COLLECTION

We obtained data from the Fall 2022 semester of CS 411 Database Systems, a course offered to both upper-level undergraduate and graduate students at the University of Illinois Urbana-Champaign. The course was taught using a flipped-classroom approach where students were provided with pre-recorded lectures and were tested on the material through short quizzes. In-class time was primarily reserved for group activities that aimed to reinforce the concepts taught in the pre-recorded lectures. Additionally, students were assigned a homework assignment consisting of approximately 15 SQL questions to be completed over a period of one week. A total of 702 students were enrolled in the course during the Fall 2022 semester.

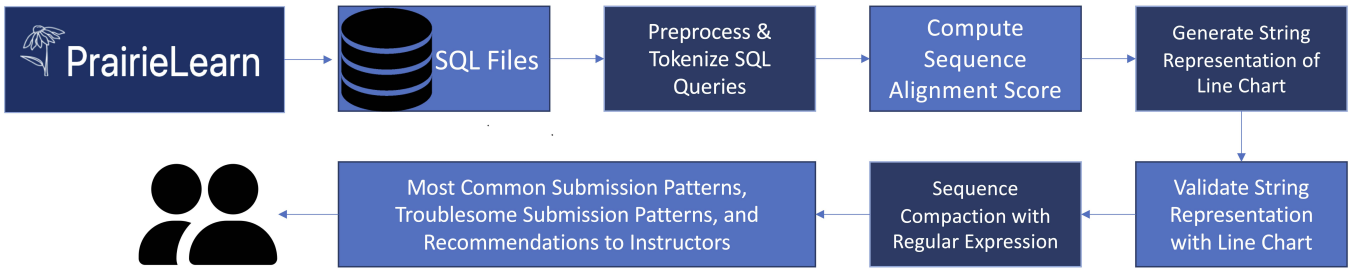

Figure 1: System Overview Diagram.

After acquiring a dataset that contains all students' SQL submission files and their submission order, we adhered to the data safety procedures outlined by the University of Illinois Urbana-Champaign (UIUC) Institutional Review Board (IRB) to ensure the anonymity of the students. To achieve this, we eliminated all identifying information from the SQL files and assigned a random number to each student as their identifier.

3.1 Description of Homework Assignments

We obtained our data from PrairieLearn, which is an online learning management system that autograde students' code and provides immediate feedback on their submissions [29]. To validate students' solutions, PrairieLearn compares their query's data result output with the expected solution's data result output using a binary grading scale (partial credit is not given, although students may see the passed and failed test cases). Homework questions may be answered in any order with unlimited attempts until the deadline, and students can revisit previously answered questions, even if they were answered correctly. An SQL problem and its corresponding instructor solution are presented below as an example.

> *Write an SQL query that returns the ProductName of each product made by the brand 'Samsung' and the number of customers who purchased that product. Only count customers who have purchased more than 1 Samsung product. Order the results in descending order of the number of customers and in descending order of ProductName.*

```
SELECT Pr1.ProductName, COUNT(C1.CustomerId) as numCustomers
FROM Products Pr1 NATURAL JOIN Purchases Pu1
  NATURAL JOIN Customers C1
WHERE Pr1.BrandName = 'Samsung'
  AND C1.CustomerId IN (
    SELECT C2.CustomerId
    FROM Customers C2 NATURAL JOIN Purchases Pu2
      NATURAL JOIN Products Pr2
    WHERE Pr2.BrandName = 'Samsung'
    GROUP BY C2.CustomerId
    HAVING COUNT(C2.CustomerId) > 1
  )
GROUP BY Pr1.ProductName
ORDER BY numCustomers DESC, Pr1.ProductName DESC;
```

4 SYSTEM OVERVIEW

The diagram in Figure 1 presents an overview of our system. To begin, we collect data from PrairieLearn [29], where students submit their SQL solution queries for the Database Systems course. Each submission attempt is represented by a .sql file in our dataset. The data cleansing process removes all irrelevant information and identifiers from the .sql files. We then use the Python sqlparse library [4] to tokenize the SQL queries based on their component type, including *Punctuation*, *Keyword*, *Comment*, *Name*, *Literal*, *Operator*, etc. To reduce noise in our dataset, we exclude components of SQL queries that do not contribute to the query's meaning, such as comments and white-spaces. The remaining tokens are used to calculate global sequence alignment scores between each student's submissions and their final attempt. The sequence alignment scores are used to generate string representations of line chart visualizations[32]; the string representations then undergo sequence compaction via the regular expression algorithm [8, 28]. We validate all textual representations alongside their corresponding line chart graphs, before and after each mentioned step, to ensure consistent accuracy. Ultimately, we examine the prevalent submission patterns displayed by students, identifying patterns that suggest challenges and potential areas of difficulty. We use this information as a basis to provide instructors with recommendations to support their teaching and to help students overcome these challenges.

4.1 Computing the Global Alignment Scores

We utilized the Needleman-Wunsch algorithm [23] based on dynamic programming to implement a global sequence alignment that optimally aligns the keys of two given sequences end-to-end, making it highly sensitive to differences in the lengths of the sequences. Modifications were made to the scoring matrix and keys during implementation. Instead of using the alphabetical letters, we defined our alignment dictionary with the tokenized components of the SQL queries. This approach allowed us to assign an equivalent weight to each token in the alignment scores, eliminating any bias based on token length. For instance, a match with a *SELECT* token would carry the same weight as an alignment match with a *NATURAL JOIN* token, instead of having the *SELECT* token count as six matches and the *NATURAL JOIN* token count as 12 matches based on character count.

We provide an example alignment of two pre-processed SQL query submissions to aid in the visualization of the computation of global alignment scores.

```
Submission x:
['select','customerid','firstname','phonenumber','from',
'customers','where','firstname','=','"%a"']
Submission y:
['select','customerid','firstname','phonenumber','from',
'customers','as','c','where','c','firstname','=','"%a"',
'or','c','lastname','=','"b%"']
```

Alignment Score	1	2	3	4	5	6	5	4	5	4	5	6	7	6	5	4	3	2
Submission x	select	customerid	firstname	phonenumber	from	customers	-	-	where	-	firstname	=	"%a"	-	-	-	-	-
Submission y	select	customerid	firstname	phonenumber	from	customers	as	c	where	c	firstname	=	"%a"	or	c	lastname	=	"b%"

Table 1: Example of global sequence alignment

Table 1 displays the global alignment that was obtained, where the green highlights indicate matches (with an alignment score of +1) and the red highlights represent mismatches/gaps (with an alignment score of -1). The final alignment score of 2 is marked in blue. The top row of numbers denotes the current alignment score up until the specified sequence component.

4.2 Textual Representation of Line Charts

We use the global alignment scores and the median length (defined by the number of SQL tokens) of all students' submissions for a particular problem to generate textual line-chart visualizations that depict how students progressed to their final solution. To determine the median length of all submissions for each homework problem, we leverage the NumPy [13] library's percentile function to obtain the number of SQL tokens that a 50th percentile submission contains. We found the mean length to be rather skewed, since there were students who had either an extremely high number of submissions or very long queries; therefore, the median was a better measure.

Next, we calculate the difference in alignment scores between each consecutive submission for a student on a SQL homework problem and compare that value with the number of tokens a 50th percentile submission contains. If the absolute difference is less than one-third of the tokens in a median length submission, we denote that submission attempt with the "_" character as it only has minor changes, resulting in a nearly flat line chart shape between the two submissions. If the difference is between one-third and two-thirds of the tokens in a median length submission, we label that submission attempt with the " ` " or " ´ " characters, depending on the value sign, as it has moderate changes, resulting in a moderately sloping line chart shape between the two submissions. If the difference is greater than or equal to two-thirds of the tokens in a median length submission, we label that submission attempt with the "/" or "\" characters, depending on the value sign, as it has significant changes, resulting in a steeply sloping line chart shape between the two submissions. Finally, if there was only one submission attempt found for the student on the problem, we denote it with a " . " (representing one data point on the line chart).

We chose one-third and two-thirds as our thresholds in this exploratory study among other tested values since these values made our resulting textual representations more accurate and consistent with their corresponding line chart features; we plan to run further studies with more robust testing for the optimal threshold number.

4.3 Sequence Compaction of the Textual Representation

In order to simplify the textual representations of student submission patterns and reduce noise, we employ a compaction method to eliminate duplicate elements that appear consecutively in a sequence. For instance, if a sequence has consecutive "_" characters, only one will be displayed unless following other elements later in the sequence. This results in a more streamlined representation of submission patterns, which facilitates aggregation across all students.

To further compact the sequence, we use the string regular expression algorithm to match " ` " with " ´ " characters, denoting this representation as a " ^ " or a "U" depending on the direction of the curve on the line chart. Similarly, we match "\" with "/" characters and also denote this representation as a " ^ " or a "U" depending on the direction of the curve on the line chart.

We validate the accuracy of the sequence both before and after each compaction step to ensure that the resulting representation is still representative of the corresponding line chart features.

5 RESULTS

Pattern	# of Occurrences	% of All Occurrences
_	3748	35.44%
.	1201	11.36%
_´_	767	7.25%
_/_	641	6.06%
_´	312	2.95%

Table 2: Top 5 Student Submission Patterns Across All SQL Homework Problems

We will present our findings on the different types of SQL solution submission patterns exhibited by students, which result from our regular expression and sequence compaction techniques. Table 2 displays the top five submission patterns observed across all SQL homework problems and students for the sake of brevity, in response to our first research question, "what are the different types of SQL solution submission patterns students display?" The results were surprising since the most frequent submission pattern was "_" indicating that the students made only minor changes between each submission before reaching their final solution. The second most common pattern was " . " showing that the majority of students submitted only once to the SQL problem, and most of them got it right. The other three patterns suggested that the students first made minor modifications to their SQL query before making moderate or significant changes and then arrived at their final solution or made some more minor adjustments before reaching their final solution.

We also looked at the frequency of occurrence of each representing sequence element across all students and SQL homework problems in order to get a better idea of which sub-patterns or "snippets" within student submission patterns are most common. The results are shown in table 3 in descending order by frequency. Not surprisingly, based on our earlier findings, the "_" element appeared most of the time, indicating that students mostly made

Element	# of Occurrences	% of All Occurrences
_	16276	57.46%
´	4001	14.12%
/	3080	10.87%
U	2426	8.56%
.	1201	4.24%
`	684	2.41%
^	347	1.22%
\	312	1.10%

Table 3: Frequency of Occurrence of Each Textual Sequence Element Across All SQL Homework Problems

minor changes throughout their submission sequence. The next two patterns, " ´ " and "/", indicates that students are getting increasingly closer to their final submission query. The occurrence of the "U" element in our sequence compaction process suggests that the student initially submitted a query, deviated from it and then returned to a query that was equally dissimilar from the final solution as their earlier attempt. We speculate that this pattern may indicate query testing, where the student comments out a portion of their query to identify the source of the error and then submits a revised query that is highly similar to their initial attempt. Our analysis of the data supports this hypothesis, as we found that testing occurred in 77.55% of cases where the "U" pattern was observed (with submissions at both ends of the "U" being similar within our thresholds for minor changes). The " ` " and "\" elements signify that the student drifted away from their final submission, while "^" elements indicate that the student initially made progress toward the final solution but then moved away from it again by modifying their query. Fortunately, these patterns are less prevalent.

5.1 Patterns That Require Instructor Attention

We would like to present the elements within submission sequences that we believe require extra instructor attention to address our second research question, "do certain submission patterns indicate students encountering obstacles in their learning path?"

We recommend that instructors review certain elements within student submission sequences, " ` ", "\", " ^ ", and " . ", due to the following reasoning: students making progress away from their final SQL submission query may indicate struggling or a flawed mental model with misconceptions towards SQL syntax or semantics. In particular, instructors should pay attention to students who have the "\" element in their submission sequence. Students with the " ^ " in their submission sequences may represent uncertainty with SQL concepts, as they make valid progress in their query but then revert those edits to arrive at a query further away from their final solution. For instructors who assign advanced SQL problems to their students, they should be especially mindful of the students with the " . " submission pattern, as these students only had one submission attempt; this is particularly concerning if the student arrived at a correct solution on advanced SQL problems prior to receiving any feedback from the autograder, or if they did not spend much time on the problem. These behaviors, which may indicate plagiarism or cheating, should be further investigated by the instructor.

6 LIMITATIONS AND FUTURE WORK

Given that our study is centered around data gathered exclusively from the University of Illinois Urbana-Champaign, whose Computer Science department ranks among the top in the field, the generalizability of our results could be impacted by the characteristics of the students and their data. To ensure greater generalizability, it is advisable to collect and analyze data from other institutions and universities.

Our work assumes that each student's final submission is their best attempt at solving the SQL problem, however, this is not always accurate. In future work, we plan to improve accuracy by using each student's best submission (first correct attempt or final attempt if no correct attempts) instead of just the final submission attempt, by combining the submission grades and query data.

In our future work, we aim to enhance our analysis by incorporating student performance data, allowing us to establish stronger correlations between student submission sequences and their academic success or challenges. As the "_" pattern constitutes 35.44% of all student submissions (table 2), we intend to conduct further analysis of the submission behaviors within this category to identify any underlying trends based on data such as the number of submission attempts and the changes made to the query between each submission. The 35.44% is unexpectedly high to us as the "_" pattern requires all elements of a submission sequence to be "_"; furthermore, the "_" element had occurred 57.46% out of all elements within patterns (table 3) - a much higher percentage than the anticipated one-thirds threshold value.

Additionally, we plan to investigate student submission sequences containing the "/" element to explore what prompts students to make significant changes in their queries. We speculate that students might have an "ah-ha!" moment when making such changes and moving closer to their final query. To explore this further, we propose a qualitative approach involving an interview study to investigate students' cognitive processes. Finally, we aim to integrate the methodologies presented into an automated dashboard system that can be easily utilized by instructors, and to then evaluate the effectiveness of this tool through an empirical study.

7 CONCLUSION

In this study, we have introduced a novel technique for efficiently and automatically evaluating and consolidating student SQL submissions by creating textual representations of the submission sequences through analyzing the global sequence alignment scores between submissions. To eliminate ambiguity and identify common patterns in the student submission sequences, we applied sequence compaction and regular expression procedures. The resulting sequence patterns and the element frequencies may then be utilized to generate suggestions for database instructors to improve their students' learning experience and rectify misconceptions with SQL query concepts before they become too challenging to correct. This approach enables proactive support, as opposed to reactive intervention when misconceptions have already taken root.

ACKNOWLEDGMENTS

This work is funded by the National Science Foundation (NSF) award number 2021499.

REFERENCES

[1] Alireza Ahadi, Vahid Behbood, Arto Vihavainen, Julia Prior, and Raymond Lister. 2016. Students' Semantic Mistakes in Writing Seven Different Types of SQL Queries. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (Arequipa, Peru) *(ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 272–277. https://doi.org/10.1145/2899415.2899464

[2] 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 *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, New York, NY, USA, 401–406.

[3] Alireza Ahadi, Julia Prior, Vahid Behbood, and Raymond Lister. 2015. A Quantitative Study of the Relative Difficulty for Novices of Writing Seven Different Types of SQL Queries. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. ACM, New York, NY, USA, 201–206.

[4] Andi Albrecht. 2020. 0.4.1 (2020). https://pypi.org/project/sqlparse/

[5] Nabeel Alzahrani, Frank Vahid, Alex Edgcomb, Kevin Nguyen, and Roman Lysecky. 2018. Python Versus C++ An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 86–91.

[6] Joseph Bergin. 1996. Java as a better C++. *ACM SIGPLAN Notices* 31, 11 (1996), 21–27.

[7] 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. https://doi.org/10.1016/j.jss.2005.06.028 Quality Software.

[8] Carl Chapman and Kathryn T. Stolee. 2016. Exploring Regular Expression Usage and Context in Python. In *Proceedings of the 25th International Symposium on Software Testing and Analysis* (Saarbrücken, Germany) *(ISSTA 2016)*. Association for Computing Machinery, New York, NY, USA, 282–293. https://doi.org/10.1145/2931037.2931073

[9] Michael Clancy. 2005. Misconceptions and attitudes that interfere with learning to program. In *Computer science education research*. Taylor & Francis, 95–110.

[10] Jonathan Danaparamita and Wolfgang Gatterbauer. 2011. QueryViz: Helping Users Understand SQL Queries and Their Patterns. In *Proceedings of the 14th International Conference on Extending Database Technology* (Uppsala, Sweden) *(EDBT/ICDT '11)*. Association for Computing Machinery, New York, NY, USA, 558–561. https://doi.org/10.1145/1951365.1951440

[11] Ashley DiFranza. 2020. 5 Reasons SQL is the Need-to-Know Skill for Data Analysts. (2020).

[12] Ann E Fleury. 2000. Programming in Java: Student-constructed rules. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*. 197–201.

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[14] Robert C. Jinkens. 2009. Nontraditional Students: Who Are They? *SIGCSE Bull.* 43, 4 (Dec. 2009), 979–987.

[15] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *Acm sigcse bulletin* 37, 3 (2005), 14–18.

[16] Hongjun Lu, Hock Chuan Chan, and Kwok Kee Wei. 1993. A Survey on Usage of SQL. *SIGMOD Rec.* 22, 4 (dec 1993), 60–65. https://doi.org/10.1145/166635.166656

[17] Linda Mannila, Mia Peltomäki, and Tapio Salakoski. 2006. What about a simple language? Analyzing the difficulties in learning to program. *Computer science education* 16, 3 (2006), 211–227.

[18] Daphne Miedema, Efthimia Aivaloglou, and George Fletcher. 2021. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) *(ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 355–367. https://doi.org/10.1145/3446871.3469759

[19] Daphne Miedema and George Fletcher. 2021. SQLVis: Visual Query Representations for Supporting SQL Learners. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VL/HCC51201.2021.9576431

[20] Daphne Miedema, George Fletcher, and Efthimia Aivaloglou. 2022. So Many Brackets! An Analysis of How SQL Learners (Mis)Manage Complexity during Query Formulation. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension* (Virtual Event) *(ICPC '22)*. Association for Computing Machinery, New York, NY, USA, 122–132. https://doi.org/10.1145/3524610.3529158

[21] Andrew Migler and Alex Dekhtyar. 2020. Mapping the SQL Learning Process in Introductory Database Courses. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 619–625. https://doi.org/10.1145/3328778.3366869

[22] A. Mitrovic. 1998. Learning SQL with a Computerized Tutor. In *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '98)*. ACM, New York, NY, USA, 307–311.

[23] Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48, 3 (1970), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

[24] Seth Poulsen, Liia Butler, Abdussalam Alawini, and Geoffrey L Herman. 2020. Insights from student solutions to sql homework problems. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, New York, NY, USA, 404–410.

[25] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–24.

[26] Toni Taipalus and Piia Perälä. 2019. What to Expect and What to Focus on in SQL Query Teaching. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 198–203. https://doi.org/10.1145/3287324.3287359

[27] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and complications in SQL query formulation. *ACM Transactions on Computing Education (TOCE)* 18, 3 (2018), 1–29.

[28] Ken Thompson. 1968. Programming Techniques: Regular Expression Search Algorithm. *Commun. ACM* 11, 6 (jun 1968), 419–422. https://doi.org/10.1145/363347.363387

[29] Matthew West, Geoffrey L. Herman, and Craig B. Zilles. 2015. PrairieLearn: Mastery-based Online Problem Solving with Adaptive Scoring and Recommendations Driven by Machine Learning.

[30] Hoe-Chun Woon and Yoon-Teck Bau. 2017. Difficulties in Learning C++ and GUI Programming with QT Platform: View of Students. In *Proceedings of the 2017 International Conference on E-commerce, E-Business and E-Government*. ACM, New York, NY, USA, 15–19.

[31] Sophia Yang, Geoffrey L. Herman, and Abdussalam Alawini. 2022. Analyzing Student SQL Solutions via Hierarchical Clustering and Sequence Alignment Scores. In *1st International Workshop on Data Systems Education* (Philadelphia, PA, USA) *(DataEd '22)*. Association for Computing Machinery, New York, NY, USA, 10–15. https://doi.org/10.1145/3531072.3535319

[32] Sophia Yang, Ziyuan Wei, Geoffrey L. Herman, and Abdussalam Alawini. 2021. Analyzing Patterns in Student SQL Solutions via Levenshtein Edit Distance. In *Proceedings of the Eighth ACM Conference on Learning @ Scale* (Virtual Event, Germany) *(L@S '21)*. Association for Computing Machinery, New York, NY, USA, 323–326. https://doi.org/10.1145/3430895.3460979