Comparison of Student Learning Outcomes Among SQL Problem-Solving Patterns

Sophia Yang
Department of Computer Science
University of Illinois Urbana-Champaign
sophiay2@illinois.edu

Geoffrey L. Herman
Department of Computer Science
University of Illinois Urbana-Champaign
glherman@illinois.edu

Abdussalam Alawini
Department of Computer Science
University of Illinois Urbana-Champaign
alawini@illinois.edu

2023 IEEE Frontiers in Education Conference (FIE) | 979-8-3503-3642-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/FIE58773.2023.10343395

***Abstract*—Structured Query Language (SQL) plays a pivotal role in the effective management of relational databases and is a key skill across domains that engage with database systems, including research, development, and business management. However, mastering SQL can be challenging. To comprehend the approaches employed by students when solving SQL problems and address the challenges they faced during the learning process, our study analyzes submissions from the Database Systems course at the University of Illinois Urbana-Champaign during the Fall 2022 semester. We extend prior research involving line chart visualizations that facilitate instructors in identifying struggling students and understanding their submission behaviors. Yet, we acknowledge the limitations of this approach in providing timely feedback and actionable insights due to the sheer volume of visualizations. To address this, we developed an innovative technique using global sequence alignment scores and regular expression algorithms to compress student submission sequences. Our approach reveals submission patterns and pattern elements, leading to recommendations for instructors to enhance database education. By integrating student performance data, such as the number of submission attempts on a particular SQL problem and whether the student arrived at a correct final solution query, we aim to empirically support these recommendations, thereby enabling instructors to more accurately differentiate between struggling and excelling students.**

***Index Terms*—SQL; Database Education; online assessment; pattern mining; sequence alignment**

I. Introduction

SQL is the primary language used for managing data and is widely supported by most Database Management Systems, making it an essential skill for those who interact with databases, such as users, developers, and researchers [1]. Its highly structured, English-like syntax makes it accessible for beginners to learn without requiring proficiency in other programming languages [2]. However, previous research has shown that despite its accessibility, some students encounter challenges in learning SQL [2]. Therefore, it is crucial to analyze the approach students take when coming up with SQL solutions and the difficulties they encounter [3].

Database instructors could improve the efficiency and effectiveness of analyzing student SQL query construction approaches by automating the examination of SQL submissions [4]. This is particularly relevant for instructors who use autograders to evaluate student submissions for SQL problems, as this approach can yield invaluable insights into students' problem-solving attempts and progress toward crafting an appropriate solution [5]. Through this process, instructors can identify the common areas of difficulty that students face in mastering SQL concepts, and tailor their teaching strategies to address these challenges [5]. Consequently, by automating the analysis of student submissions, instructors may more easily fine-tune their pedagogical approaches and provide students with the necessary support to acquire SQL proficiency.

Our research work is a continuation and expansion of previous studies on line chart visualizations that exhibit a student's submission pattern using the sequence alignment score between each submission on a SQL problem and their final submission, as described in [6]. However, a major drawback of this prior research is the limited scalability of the visuals. Given that the total number of visuals is equal to the number of students multiplied by the number of SQL problems, it may be difficult for an instructor to analyze the visuals even in a small class with approximately 20 students, which can easily result in over 100 visuals (over five problems). In the case of a class comprising several hundred students, it is not possible for instructors to review all of the line chart graphs to differentiate between struggling and successful students and provide feedback in a timely manner. To address this limitation, our prior research work generates textual representations of the student submission sequence using global sequence alignment score inputs, followed by applying regular expression algorithms to compact and aggregate these textual representation submission sequences [5]. These textual representation submission sequences are what we define as our textual *submission patterns*, and the elements making up the submission patterns are what we define as *pattern elements*. However, we did not evaluate the submission patterns and pattern elements with empirical evidence, such as the student's performance data. Therefore, our research aims to extend our prior work to answer the question: *what are the submission patterns that are associated with improved performance or increased obstacles faced by students during their learning journey?*

According to previous research [7], students have different learning paths, and their learning experience can be improved if instructors can identify their individual learning approaches. Building on this finding, our research aims to improve the educational quality of students learning SQL in database courses. Specifically, we examine the data from an upper-level undergraduate and graduate Database Systems course

at the University of Illinois Urbana-Champaign in the United States, which typically has over 400 enrollees but had over 700 enrollees in the Fall 2022 semester. We focus on students' submissions to SQL homework assignments during the Fall 2022 semester, which consists of 15 problems that are automatically graded, with immediate feedback provided after each submission. However, due to the large class size, it is challenging for instructors to quickly identify common areas of difficulty or individual students in need of additional support. Moreover, with students submitting an average of more than 20 attempts per problem, the number of submissions for each SQL problem is too numerous to be manually examined. Therefore, our prior research work presented a method for evaluating students' progression as they attempt to solve an SQL problem, which enables instructors to recognize the diverse approaches students use to write semantically equivalent SQL queries. We aim to extend this research work by linking students' performance data with their submission patterns and pattern elements to evaluate and differentiate the ones that may indicate student challenges from the ones that may indicate success. As a result, instructors can identify struggling students more quickly and offer early assistance, leading to proactive support rather than reactive support.

II. Related Works

A substantial number of studies have been conducted to investigate the difficulties and misunderstandings that students face when learning procedural programming languages such as Java [8]–[10], C++ [9], [11]–[13], and Python [8], [9], [13]. In contrast, there has been a comparably smaller amount of research on declarative or database query languages, particularly SQL [14]–[19]. Although several existing studies have focused on the SQL problems and concepts that students commonly find challenging, such research has yet to attain the same breadth as procedural programming language research. One of the prior studies that analyzed SQL queries submitted by students was conducted by Taipalus et al., and the authors found that students made various syntax, semantic, and logic errors in their SQL queries [16]. Specifically, Taipalus et al. identified some of the syntax errors that had been reported previously in the literature, including undefined parameters, data type mismatch, and date time field overflow [14]. Additionally, Taipalus and Perälä [17] examined persistent error types that students frequently encounter while creating SQL queries, along with the SQL query concepts that are linked to such errors. Other studies have explored the various types of SQL queries that students find challenging to write [20], [21], the most frequent SQL semantic errors [22], and the categorizations of semantic errors [23].

Prior research has highlighted the difficult SQL concepts and challenges that students encounter, but the underlying reasons for these difficulties remain largely unexplored. To unravel the roots of SQL misconceptions, a qualitative research methodology is necessary. In a think-aloud study, Miedema et al. analyzed the problem-solving processes of 21 students 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 [18]. Furthermore, Miedema et al. examined factors that contribute to self-inflicted query complexity based on correctness, execution order, edit distance, and query intricacy [24]. Their findings suggest that the reasons for students' difficulties in SQL query writing are multi-faceted and may require targeted interventions tailored to the specific underlying causes.

A few studies have explored the difficulties students encounter while learning SQL and have also investigated various techniques for visualizing and identifying these obstacles and learning strategies [6], [25]–[27]. To illustrate, Miedema et al. created the SQLVis system, which employs visual query representation (VQR) to assist novices in learning to write SQL queries [25]. QueryViz, developed by Danaparamita and Gatterbauer, is another tool that helps comprehend SQL queries through visualization [27]. Moreover, the authors of this study have developed visualization techniques, including edit distance and clustering, to identify learning obstacles and approaches based on students' SQL submission sequences [6], [26]. By leveraging these techniques, researchers can gain a deeper understanding of the SQL learning process and provide effective learning support to students.

A limited number of studies have been conducted to uncover the causes of students' difficulties in writing SQL queries, as evidenced by the works of Miedema et al. [18], Poulsen et al. [15], Ahadi et al. [20], and Taipalus et al. [17]. However, there is a dearth of actionable recommendations for SQL educators to address these issues. One approach suggested in the literature is to improve the quality of SQL compilation error messages. It is noted that students often face obstacles and ultimately abandon the task when confronted with syntax errors [14], [15]. Another recommendation is to tackle the misconceptions that can arise from transferring prior knowledge in mathematics, natural language, and other programming languages [18]. This challenge is not unique to SQL instruction as it has been observed in other programming languages as well [28], [29]. Additionally, previous research has emphasized the utility of generating automated recommendations for instructors based on the various SQL solution submission patterns exhibited by students [26].

To the best of our knowledge, no prior studies have investigated the various SQL solution submission patterns that students exhibit with the aim of identifying problematic patterns and promoting automated instructor recommendations. To achieve this goal, we adopt regular expression techniques employed in earlier research studies [30], [31] to transform input data [6] into a string representation that is easier to analyze, aggregate, and understand, thereby ensuring transparency in the data processing. The input data comes from the global sequence alignment scores between each student's submission with their final submission on each SQL problem; the global sequence alignment score is a measure of similarity between two sequences, aligned optimally end-to-end against each other, based on dynamic programming [32]. A higher

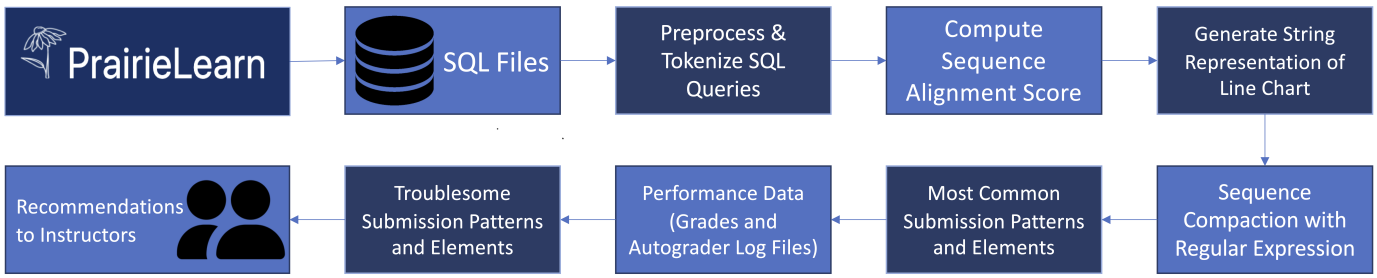

FIG. 1: System Overview Diagram.

alignment score indicates higher similarity and overlap, while a lower alignment score indicates lower similarity between two sequences. The commonly used edit distance algorithm [33] in genomic sequencing may be considered a variation of the global sequence alignment algorithm with a modification in the scoring matrix. Our research has two main objectives: first, we aim to analyze the different patterns of SQL submission sequences displayed by students, which has been explored in our previous study [5]. We have expanded upon our previous study and included the average submission attempts for each pattern and pattern element, and a case study examining the frequently occurring "_" pattern. Our second objective is to identify submission patterns that are linked with superior performance, as well as those associated with difficulties or challenges faced by students. Ultimately, we aim to help database instructors by alerting them on students who are struggling, based on our analysis of their submission sequence behaviors. This could assist instructors in identifying students who require assistance at an early stage.

III. Data Collection

We collected a large dataset from an elective Database Systems course in the Computer Science curriculum for upper-level undergraduate and graduate students offered at the University of Illinois Urbana-Champaign in the Fall 2022 semester. Enrollment in the course mandates prior completion of a data structures course, typically taken in the second year of study. This prerequisite ensures students possess the necessary programming background, given the inclusion of programming assignments in the course. The course employed a flipped-classroom model, which involved the dissemination of pre-recorded lectures to students, followed by short quizzes aimed at testing their understanding of the material presented, which were graded on completion. Classroom time was reserved for collaborative group activities designed to reinforce the concepts presented in the pre-recorded lectures, with groups comprising 4-5 students. Additionally, students were given a set of approximately 15 SQL programming questions for homework, to be completed individually over a week. A total of 730 students were enrolled in the course during the Fall 2022 semester.

After acquiring a large dataset that contained all of the SQL submissions and their orders, we safeguarded student anonymity and privacy by adhering to our institution's Institutional Review Board (IRB) data safety protocols. These protocols involved eliminating identifying details from SQL files and assigning each student a random identifier.

A. Description of Homework Assignments

We obtained our data from PrairieLearn - an online learning management system that autogrades students' code and provides immediate feedback on their submissions [34]. The autograder serves to validate the students' solutions by comparing their query's data result output with the expected solution's data result output, using a binary grading scale that does not offer partial credit, although students may see the passed and failed test cases. Students are unlikely to locally test their SQL queries before submitting on PrairieLearn due to random test case data generation and the convenience of PrairieLearn's environment, eliminating the need for setting up local environments. Homework questions can be answered in any order, and students can submit unlimited attempts until the deadline. Moreover, students can revisit previously answered questions, even if they were answered correctly. For an illustration of a SQL problem and its corresponding instructor solution, please refer to Figures 2 and 3.

> *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.*

FIG. 2: SQL Homework Problem Statement Example

IV. Methodology

The graphical representation of our system, illustrated in Figure 1, provides a comprehensive overview of the steps we take to analyze the SQL solution queries submitted by students in the database course via PrairieLearn. First, we gather the data from the system, which is stored in our dataset as .sql files. In the data cleansing phase, we remove all non-relevant information and identifiers from these files. Subsequently, we employ the Python sqlparse library [35] to parse the SQL queries based on their component types such as *Punctuation*, *Keyword*, *Comment*, *Name*, *Literal*, *Operator*, among others.

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 I: Example of global sequence alignment

```
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;
```

FIG. 3: SQL Homework Solution Example

Components of the SQL queries that do not add to the meaning of the query, such as white-spaces and comments, are excluded to minimize noise in our dataset. The remaining tokens are used to calculate global sequence alignment scores for each student's submissions compared to their best attempt, which is either the first correct attempt or the final attempt if the student never submitted a correct one. The sequence alignment scores are then used to generate string representations of line charts and their features; we compact the string representations using the regular expression algorithm [30], [31]. We thoroughly validate our textual representations by comparing them to their corresponding line chart graphs at each step of our process to maintain accuracy and consistency.

Our analysis aims to identify prevalent submission patterns that can indicate challenges and potential areas of difficulty faced by students. We use this information to provide recommendations to instructors to support their teaching and assist students in overcoming these challenges.

A. Global Alignment Score Computation

We employed the Needleman-Wunsch algorithm [32] as the basis for our global sequence alignment approach. This algorithm, which is based on dynamic programming, is well-suited for aligning two sequences end-to-end in an optimal manner, making it particularly sensitive to differences in the lengths of the sequences being aligned.

In order to adapt this algorithm to our specific needs, we made a few key modifications to the scoring matrix and the keys used in the alignment process. Instead of using the alphabetical letters commonly used in sequence alignment, we defined our alignment dictionary with the tokenized components of the SQL queries that we were analyzing. This allowed us to assign an equivalent weight to each token in the alignment scores, thereby eliminating any bias that might arise based on the length of the individual tokens.

For example, consider a scenario in which one query includes a *SELECT* token, while another query includes a *NATURAL JOIN* token. If we were to rely on character counts alone to evaluate the similarity between these two queries, the *SELECT* token would be counted as six matches, while the *NATURAL JOIN* token would be counted as twelve matches. Yet, by using a token-based approach to alignment with uniform token weighting, we are able to compare the two queries more accurately and impartially, regardless of the length or complexity of the individual tokens involved.

In order to facilitate the understanding of how the global alignment scores are computed, we present an example alignment of two pre-processed SQL query submissions.

```
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%"']
```

The global alignment of the two pre-processed SQL query submissions can be visualized in Table I. The matches are represented by blue highlights, assigned an alignment score of +1, while the mismatches/gaps are indicated by orange highlights and assigned an alignment score of -1. The final alignment score of 2 is marked in green. The top row of numbers displays the current alignment score up until the specified sequence component.

B. String Representations of Line Charts

Textual line-chart visualizations that demonstrate the progress of students towards their final solutions are created using both the global alignment scores and the median length of all submissions for a particular problem, measured in SQL tokens. The median length of submissions for each homework problem is calculated using the percentile function of the NumPy library [36], which gives the number of SQL tokens that a 50th percentile submission contains. The mean length was found to be skewed due to some students having a high number of submissions or long queries. As a result, the median was deemed a better measure.

Subsequently, we determine the variation in the alignment scores between successive submissions made by a student for a SQL homework problem and compare that value with the number of tokens contained in a 50th percentile submission. Whenever the absolute difference is less than one-third of the tokens found in a median length submission, we mark that submission attempt with the character "_" since it contains only minor changes, resulting in an almost level line chart shape between the two submissions. When the difference falls between one-third and two-thirds of the tokens in a median length submission, we assign the submission attempt the characters " ` " or " ´ ", depending on the value sign, as it indicates moderate changes, resulting in a line chart shape that moderately slopes between the two submissions. In case the

difference is equal to or greater than two-thirds of the tokens in a median length submission, we assign the submission attempt the characters "/" or "\", depending on the value sign, as it indicates significant changes, resulting in a steeply sloping line chart shape between the two submissions. If only one submission attempt is found for the student on the problem, we indicate it with a " . " character, representing one data point on the line chart.

In our exploratory study, we selected one-third and two-thirds as our threshold values after evaluating several other options. The threshold values of one-thirds and two-thirds performed better at minimizing noise (i.e. not having smaller changes be considered steep changes and not having larger changes be considered a small change) in our textual sequences. The even division of threshold values among the three categories (flat, gradual, and steep) likely contributed to this improved performance. These values, therefore, resulted in improved accuracy and consistency between the textual representations and the corresponding line chart features. Further research with more extensive testing will be conducted to determine the optimal threshold value.

C. Sequence Compaction of the String Representation

To simplify the representation of student submission patterns and minimize noise, we adopt a compaction method that eliminates consecutive duplicate elements in a sequence. For example, if a sequence consists of successive occurrences of "_" characters, only one will be displayed unless followed by other elements later in the sequence. This approach leads to a more concise representation of submission patterns that facilitates aggregation across students.

To further streamline the sequence, we apply a string regular expression algorithm to match " ` " with " ´ " characters, which are represented as either a " ^ " or a "U" depending on the direction of the curve in the line chart. Similarly, we match "\" with "/" characters and also represent them as a " ^ " or a "U" depending on the direction of the curve.

We conduct a validity check on the sequence before and after each compaction step to ensure the resulting representation accurately reflects the corresponding line chart features.

V. Results

In this section, we will present an exploratory data analysis [37] of student submission patterns and sub-patterns (pattern elements), including an analysis of the average number of submission attempts associated with each pattern element. We will also analyze the distribution of incorrect submission pattern elements, and compare the submission elements between successful and challenged student groups. Finally, we will examine the relationship between student homework assignment grades and the pattern elements displayed in their submissions.

A. Most Common Student Submission Patterns

Table II summarizes the top five submission patterns among all SQL homework problems and students to address the different types of SQL solution submission patterns exhibited by students. The findings were unexpected, as the most recurrent pattern was "_", indicating that students made only slight modifications between submissions before achieving their best solution. The second most prevalent pattern was " . " which signifies that the majority of students submitted only once to the SQL problem, and were successful in doing so. The remaining three patterns imply that students initially made minor alterations to their SQL query before making moderate or significant changes and then reaching their best solution or made additional minor tweaks before reaching the best solution. Other patterns not shown occurred infrequently so they were excluded for the sake of brevity.

Pattern	# of Occurrences	% of All Occurrences	Average Submission Attempts
_	3724	35.21%	6.02
.	1355	12.81%	1.00
_´_	759	7.18%	14.25
_/_	609	5.76%	12.75
_´	324	3.06%	7.77

TABLE II: Most Common Student Submission Patterns

B. The "_" Submission Pattern and Element Case Study

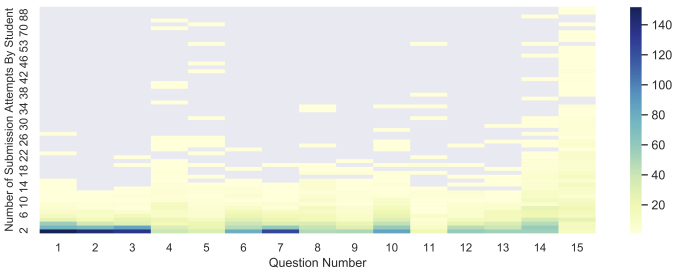

FIG. 4: Breakdown of Student Submissions on the "_" Pattern. *The x-axis shows homework question number, the y-axis displays pre-best attempt number of submission attempts, and color intensity indicates corresponding student count.*

With further analysis of the submission behaviors within the category of the "_" pattern, which constitutes 35.21% of all student submissions as reported in Table II, we aimed to gain a deeper understanding of the breakdown of the number of submission attempts made by students. The high occurrence of the "_" pattern, which requires all elements of a submission sequence to be "_", was a surprising finding. To visually depict our findings, we created Figure 4.

The results of our analysis shed light on the reason behind the lower average submission attempt number observed in the "_" pattern, as presented in Table II; a considerable portion of students who exhibited the "_" pattern had a limited number of submissions (a majority only had 2-5 attempts per problem), as evidenced by the concentration of cases in the lower range of the submission attempts on the y-axis in Figure 4. The less common instances of students submitting over 10 attempts per problem are scattered along the upper range of the y-axis in light yellow.

Element	# of Occurrences	% of All Occurrences	Average Submission Attempts
_	8975	47.51%	13.16
´	3083	16.32%	18.49
/	2774	14.69%	18.14
U	1591	8.42%	28.02
.	1355	7.17%	1.00
`	535	2.83%	30.20
^	291	1.54%	39.99
\	285	1.51%	35.48

TABLE III: Average Submission Attempts for Each Pattern Element

C. Pattern Elements and Average Submission Attempts

To gain a better understanding of the most common sub-patterns within student submission sequences, we analyzed the frequency of occurrence of each pattern element across all students and SQL homework problems. The "Average Submission Attempts" column shows the average number of submissions made by students for each pattern element, and the occurrences columns display the frequency that each element is seen across all submission patterns. The results, displayed in descending order by frequency in Table III, reveal that the element "_" was the most frequently occurring. This is consistent with our previous findings that students mostly made minor changes throughout their submission sequence. The next two most common patterns, " ´ " and "/", suggest that students were getting closer to their best submission query.

The appearance of the "U" element in our sequence analysis suggests that the student initially submitted a query, modified it, and then returned to a version that was similarly distant from the best solution. Our speculation is that this pattern is indicative of query testing, where the student temporarily comments out a section of their query to identify the source of the error, and then resubmits a query that closely resembles their initial submission. Our analysis of the data supports this hypothesis, as we found that testing occurred in 77.55% of the cases where the "U" pattern was observed. To reach this conclusion, we evaluated the two submissions at the ends of the "U" pattern; we computed their edit distance [33] and validated whether their difference was less than our previously used threshold value to be considered a "_" or minor change.

The presence of the " ` " and " \ " elements indicates that the student moved away from their best submission, while the presence of the " ^ " element suggests that the student started moving in the right direction but then deviated from it by making changes to their query. Fortunately, these patterns are less frequent. Moreover, we believe that the elements "U" and " ^ " can be classified similarly - both elements exhibit query testing behavior, but with changes made in opposite directions relative to the best submission. This hypothesis is also supported by our analysis of the data, as we found that testing occurred in 84.46% of the cases where the " ^ " pattern was observed; we believe testing to have occurred if the two submissions at the ends of the " ^ " pattern have an edit distance that's less than the threshold value used for the "_" pattern (therefore within the range of a minor change). To strengthen this conclusion, we recommend performing a qualitative evaluation with students to analyze their cognitive processes during the debugging process.

The highest average submission attempts were observed for the pattern elements " ` " and " ^ " with 30.20 and 39.99 submissions, respectively. The next highest average submission attempts were made for the pattern element "U" with 28.02 submissions. The lowest average submission attempts were made for the pattern element " . " with only one submission, as it corresponds to the submission pattern " . " with only one attempt. The other pattern elements had an average of 13.16 to 18.49 submissions. These findings offer valuable insights into student submission behavior and the difficulty level of various pattern elements in SQL problem submission sequences.

D. Pattern Element Occurrences for Incorrect Submissions

Element	# of Occurrences	% of All Occurrences	Average Submission Attempts
_	164	46.07%	18.05
´	64	17.98%	21.08
/	42	11.80%	20.10
U	41	11.52%	33.46
.	23	6.46%	1.00
`	12	3.37%	24.58
^	5	1.40%	27.80
\	5	1.40%	58.00

TABLE IV: Pattern Elements Found in Incorrect Submissions

In order to obtain a deeper view on the sub-patterns of students who are struggling (fail to correctly solve the problem in the end), we exclude all student submissions to homework problems that eventually resulted in a correct submission attempt - results are displayed in Table IV. The element "_" has the highest number of occurrences (164) and the highest percentage of all occurrences (46.07%). The element "\" has the highest average number of submission attempts (58.00), followed by the element "U" (with an average of 33.46 submission attempts) and element " ^ " (with an average of 27.80 submission attempts).

While the order of number of occurrences for the pattern elements are consistent with Table III, we observe a slightly decreased percentage of occurrence for the " / " element (14.69% vs. 11.80%) and a slightly increased percentage of occurrence for the "U" element (8.42% vs 11.52%).

E. Average Submission Attempts for Pattern Elements in Successful Student Groups

Element Detected	# of Occurrences	% of All Occurrences	Average Submission Attempts
_	1003	45.38%	3.07
.	869	39.32%	1.00
/	192	8.69%	3.60
´	129	5.84%	3.74
`	8	0.36%	5.00
U	7	0.32%	5.57
^	1	0.05%	7.00
\	1	0.05%	7.00

TABLE V: Average Submission Attempts for Pattern Elements in Lower 25th Percentile Submissions

To analyze submission pattern elements in high-achieving or successful students that are able to complete a SQL problem

relatively quickly, we excluded all student submissions where their submission count to the particular homework problem exceeded the class-wide 25th percentile number of submissions. We believe this may help validate pattern elements thought to be linked to successful student behaviors and shed light on the frequency of unfavorable sequences among successful students. Table V displays the frequency and average submission attempts of pattern elements occurring in the lower 25th percentile submissions (by submission count).

We observe that the element "_" has a similar level of presence compared to Tables III and IV; however, we have a noticeably elevated percentage of " . " submissions, understandably since they only consist of one submission attempt which counts towards the lower 25th percentile. We saw a reduced presence for both elements " ´ " and " / ", and very scarcely observed " ` ", "U", " ^ ", and " \ " elements.

F. Average Submission Attempts for Pattern Elements in Challenged Student Groups

Element Detected	# of Occurrences	% of All Occurrences	Average Submission Attempts
_	2446	37.52%	28.97
´	1275	19.59%	31.28
/	1037	15.91%	32.82
U	984	15.09%	36.98
`	336	5.15%	40.25
^	229	3.51%	46.82
\	212	3.25%	42.43
.	0	0.00%	-

TABLE VI: Average Submission Attempts for Pattern Elements in Top 75th Percentile Submissions

To analyze submission pattern elements in challenged students that require more time or submission attempts to complete an SQL problem, we excluded all student submissions where their submission count to the particular homework problem was lesser than the class-wide 75th percentile number of submissions. We believe doing so may help validate pattern elements that we thought to be associated with challenged student behaviors. This may also offer perspectives on the frequency of unfavorable sequences in challenged student groups. Table VI displays the frequency and average submission attempts of pattern elements occurring in the upper 75th percentile submissions (by submission count).

We observe that the element "_" has a much lower level of presence compared to Tables III, IV, and V; instead, we have almost double the proportion of "U" elements compared to Table III (8.42% vs. 15.09%). There are also almost double or more than double the percentages of occurrence for elements " ` ", " ^ ", and " \ ", compared to Table III. Understandably, there were no occurrences of the " . " element given the top 75th percentile submission attempts.

G. Student Performance Grades and Pattern Elements

In order to explore the relationship between pattern elements and students' performance grades, we showcase the students' grades data associated with submission patterns containing the corresponding pattern elements, in an aggregated (averaged) format in Table VII. However, PrairieLearn's autograder feedback, along with unlimited attempts before the deadline, resulted in a high average grade for the SQL assignment. As a result, the majority of students received full credit for the SQL problems. This is reflected in the data shown in the column "% Correct" in the table, which demonstrates that there were no substantial differences in the grades received for the various pattern elements.

Element	# of Correct Submissions	# of Incorrect Submissions	# of Total Submissions	% Correct
/	2732	42	2774	98.49%
.	1332	23	1355	98.30%
^	286	5	291	98.28%
\	280	5	285	98.25%
_	8811	164	8975	98.17%
´	3019	64	3083	97.92%
`	523	12	535	97.76%
U	1550	41	1591	97.42%

TABLE VII: Relationship Between Pattern Elements and Student Grades

VI. INTERPRETATIONS

In this section, we will provide an in-depth examination of our findings and delve into the patterns or sub-patterns to answer our research question: *what are the submission patterns that are associated with better performance or more obstacles encountered by students in their learning journey?*

First, we look at patterns that seem to indicate difficulty among students, thereby requiring additional attention from the instructor. Specifically, elements such as " ` ", " \ ", and " ^ " are worth paying attention to as they may indicate a flawed mental model with misconceptions for SQL syntax or semantics. For instructors teaching advanced SQL problems such as ones containing correlated subqueries, they should be especially mindful of students with the " . " pattern (who received full credit), as this indicates that they only made one submission attempt, which could be a sign of plagiarism or cheating and should be investigated further. Otherwise, element "\" is of particular concern in a student's submission sequence, which suggests that they are rapidly moving further away from their final submission. The " ^ " element may indicate uncertainty with SQL concepts, as the student makes progress but then reverses those changes to arrive at a solution that is further from the final answer.

Our analysis with submissions of students who failed to correctly solve a SQL problem showed that elements " ^ ", "\", and "U" had the highest average submission attempts associated with patterns containing them (27.80, 58.00, and 33.46, respectively). These values appear to be much higher than the rest of the elements which ranged from 1.00 to 24.58 (refer to Table IV) attempts. While the "U" and the " ^ " elements may be indicative of testing behaviors, the testing behaviors presented in this particular group of students (who either abandoned the question or ran out of time) are likely

ineffective testing behaviors; therefore, instructors who see student patterns with the "U" or " ^ " elements coupled with high submission attempts may want to provide clarification or support to improve debugging and testing skills. However, it is unclear if the pattern elements (and misconceptions) caused the students to not develop a correct solution, or if the high submission attempts led to abandonment.

Our analysis of students who faced challenges and submitted more frequently (higher submission percentiles) showed that as the percentile increased for the # of submissions, there seems to also be a rise in the occurrence of unfavorable patterns such as " `", " ^ ", "\", where students deviated from their solution. We analyzed both the submissions in the 75th percentile (as shown in Table VI) and those in the 90th percentile (excluded for the sake of conciseness).

In contrast, when examining the submissions of successful students who made fewer attempts (lower submission percentiles), we predominantly found favorable patterns, such as " / ", "_", and " ´ ", which indicated that the students were making progress towards their best solution. Conversely, unfavorable patterns appeared only infrequently, with occurrences ranging from 0.05% to 0.36% (as shown in Table V).

Therefore, while there was no relationship between a particular pattern element and a student's grade, as most students performed exceptionally well due to the unlimited submissions and relaxed deadline, elements such as " / ", "_", and " ´ " were linked to fewer attempts before arriving at the best solution. On the other hand, elements like " `", " ^ ", "\" were more frequently observed in longer submission sequences and had the highest number of attempts in incorrect submissions.

VII. Limitations & Future Work

The results of our study are based on data from the University of Illinois Urbana-Champaign, with a highly ranked Computer Science department, which may not be representative of the broader population. To enhance the generalizability of our findings, it is recommended to gather and analyze data from a diverse range of institutions and universities.

While a strong association was established between the "U" and " ^ " elements and testing behavior, it is unclear if such testing behavior is productive to learning or leads to further frustration. The elevated average number of submission attempts linked to these elements introduces complexity and poses a challenge in determining whether students' struggles arise from the elements (and their behaviors) themselves or from the substantial number of attempts, which may be influenced by confounding factors. To gain a deeper understanding, a qualitative study through interviews with students would be valuable to comprehend their cognitive processes and the sources of frustration when exhibiting these pattern elements.

This highlights another limitation of our study, which is that the number of submission attempts may not accurately reflect a student's understanding or learning of the SQL problem. While a high number of submissions may indicate frustration or a sense of challenge, and conversely, a low number of submissions may indicate ease or success, there is no clear connection between these factors and actual learning. For instance, there may be students who are process-oriented and prefer to solidify their understanding at each step of their query, leading to a higher number of submissions, and there may also be students who are result-oriented and only seek to receive full credit on the autograder, possibly resorting to blind experimentation or even cheating (as possibly indicated by the " . " pattern). These two groups of students would likely have vastly different levels of learning, contrary to our current findings. We have conducted analyses to determine if students with higher submissions on earlier problems in the assignment performed better on later problems, and vice versa, but we did not find any strong associations to debunk our current findings. However, we still believe that an interview study to understand students' cognitive processes and learning would greatly benefit our technique and help make it more accurate and adaptive to different student behaviors.

Next, we plan to integrate study findings into an instructor-friendly dashboard system and validate the effectiveness of this technique through an empirical study with database instructors.

VIII. Conclusion

In this study, we expand on our previous research work by conducting an exploratory data analysis on student submission patterns and pattern elements, examining average submission attempts, distribution of incorrect submission elements, and disparities between successful and challenged students. The relationship between homework grades and submission elements is also investigated.

We discovered that although no pattern element was found to be directly linked to higher grades because students generally performed well on the assignment due to unlimited submissions and a lenient deadline, certain pattern elements, such as " / " (steep change getting closer to the best submission), "_" (minor change), and " ´ " (moderate change getting closer to best submission), were linked to fewer submissions before reaching the best solution. Elements such as " `" (moderate change getting farther from best submission), " ^ " (getting closer then farther from best submission), and "\" (steep change getting farther from best submission) appeared more frequently in longer submission sequences and were associated with higher submission attempts in incorrect submissions.

These findings, which help to validate submission patterns and pattern elements with empirical evidence, allow us to establish more robust associations between pattern elements and student performance, whether improved or challenged. Consequently, we can equip database instructors with data-driven techniques to swiftly differentiate struggling students from successful ones. By adopting a proactive approach, instructors can provide timely support and guidance, preventing the entrenchment of student misconceptions that may be difficult to address through a reactive approach.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 2021499.

REFERENCES

[1] A. DiFranza, "5 reasons sql is the need-to-know skill for data analysts," 2020.

[2] HANLijun, "Sql: A supposed english-like language," Jan. 2018. [Online]. Available: https://www.datasciencecentral.com/sql-a-supposed-english-like-language/

[3] A. Mitrovic, "Learning sql with a computerized tutor," in *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '98. New York, NY, USA: ACM, 1998, p. 307–311.

[4] M. Weston, H. Sun, G. L. Herman, H. Benotman, and A. Alawini, "Echelon: An ai tool for clustering student-written sql queries," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–8.

[5] S. Yang, G. L. Herman, and A. Alawini, "Mining sql problem solving patterns using advanced sequence processing algorithms," in *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research*, ser. DataEd '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 37–43. [Online]. Available: https://doi.org/10.1145/3596673.3596973

[6] S. Yang, Z. Wei, G. L. Herman, and A. Alawini, "Analyzing patterns in student sql solutions via levenshtein edit distance," in *Proceedings of the Eighth ACM Conference on Learning @ Scale*, ser. L@S '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 323–326. [Online]. Available: https://doi.org/10.1145/3430895.3460979

[7] R. C. Jinkens, "Nontraditional students: Who are they?" *SIGCSE Bull.*, vol. 43, no. 4, pp. 979–987, Dec. 2009.

[8] L. Mannila, M. Peltomäki, and T. Salakoski, "What about a simple language? analyzing the difficulties in learning to program," *Computer science education*, vol. 16, no. 3, pp. 211–227, 2006.

[9] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *Acm sigcse bulletin*, vol. 37, no. 3, pp. 14–18, 2005.

[10] A. E. Fleury, "Programming in java: Student-constructed rules," in *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, 2000, pp. 197–201.

[11] H.-C. Woon and Y.-T. Bau, "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*. New York, NY, USA: ACM, 2017, pp. 15–19.

[12] J. Bergin, "Java as a better c++," *ACM SIGPLAN Notices*, vol. 31, no. 11, pp. 21–27, 1996.

[13] N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen, and R. Lysecky, "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*. New York, NY, USA: ACM, 2018, pp. 86–91.

[14] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "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*. New York, NY, USA: ACM, 2016, pp. 401–406.

[15] S. Poulsen, L. Butler, A. Alawini, and G. L. Herman, "Insights from student solutions to sql homework problems," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2020, pp. 404–410.

[16] T. Taipalus, M. Siponen, and T. Vartiainen, "Errors and complications in sql query formulation," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 3, pp. 1–29, 2018.

[17] T. Taipalus and P. Perälä, "What to expect and what to focus on in sql query teaching," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 198–203. [Online]. Available: https://doi.org/10.1145/3287324.3287359

[18] D. Miedema, E. Aivaloglou, and G. Fletcher, "Identifying sql misconceptions of novices: Findings from a think-aloud study," in *Proceedings of the 17th ACM Conference on International Computing Education Research*, ser. ICER 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 355–367. [Online]. Available: https://doi.org/10.1145/3446871.3469759

[19] H. Lu, H. C. Chan, and K. K. Wei, "A survey on usage of sql," *SIGMOD Rec.*, vol. 22, no. 4, p. 60–65, dec 1993. [Online]. Available: https://doi.org/10.1145/166635.166656

[20] A. Ahadi, J. Prior, V. Behbood, and R. Lister, "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*, ser. ITiCSE '15. New York, NY, USA: ACM, 2015, p. 201–206.

[21] A. Migler and A. Dekhtyar, "Mapping the sql learning process in introductory database courses," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 619–625. [Online]. Available: https://doi.org/10.1145/3328778.3366869

[22] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "Students' semantic mistakes in writing seven different types of sql queries," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 272–277. [Online]. Available: https://doi.org/10.1145/2899415.2899464

[23] S. Brass and C. Goldberg, "Semantic errors in sql queries: A quite complete list," *Journal of Systems and Software*, vol. 79, no. 5, pp. 630–644, 2006, quality Software.

[24] D. Miedema, G. Fletcher, and E. Aivaloglou, "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*, ser. ICPC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 122–132. [Online]. Available: https://doi.org/10.1145/3524610.3529158

[25] D. Miedema and G. Fletcher, "Sqlvis: Visual query representations for supporting sql learners," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2021, pp. 1–9.

[26] S. Yang, G. L. Herman, and A. Alawini, "Analyzing student sql solutions via hierarchical clustering and sequence alignment scores," in *1st International Workshop on Data Systems Education*, ser. DataEd '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 10–15. [Online]. Available: https://doi.org/10.1145/3531072.3535319

[27] J. Danaparamita and W. Gatterbauer, "Queryviz: Helping users understand sql queries and their patterns," in *Proceedings of the 14th International Conference on Extending Database Technology*, ser. EDBT/ICDT '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 558–561. [Online]. Available: https://doi.org/10.1145/1951365.1951440

[28] M. Clancy, "Misconceptions and attitudes that interfere with learning to program," in *Computer science education research*. Taylor & Francis, 2005, pp. 95–110.

[29] Y. Qian and J. Lehman, "Students' misconceptions and other difficulties in introductory programming: A literature review," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 1, pp. 1–24, 2017.

[30] K. Thompson, "Programming techniques: Regular expression search algorithm," *Commun. ACM*, vol. 11, no. 6, p. 419–422, jun 1968. [Online]. Available: https://doi.org/10.1145/363347.363387

[31] C. Chapman and K. T. Stolee, "Exploring regular expression usage and context in python," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ser. ISSTA 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 282–293. [Online]. Available: https://doi.org/10.1145/2931037.2931073

[32] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970. [Online]. Available: https://doi.org/10.1016/0022-2836(70)90057-4

[33] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926–932, 1993.

[34] M. West, G. L. Herman, and C. B. Zilles, "Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning," 2015.

[35] A. Albrecht, vol. 0.4.1, 2020. [Online]. Available: https://pypi.org/project/sqlparse/

[36] C. R. Harris, K. J. Millman, S. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with numpy," *CoRR*, vol. abs/2006.10256, 2020. [Online]. Available: https://arxiv.org/abs/2006.10256

[37] "What is exploratory data analysis?" [Online]. Available: https://www.ibm.com/topics/exploratory-data-analysis