Analyzing Student SQL Solutions via Hierarchical Clustering and Sequence Alignment Scores

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

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

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

ABSTRACT

Structured Query Language (SQL), the de facto standard language for relational database systems management, proves to be a vital skill for a wide array of users, developers, and researchers who interact with databases. Given that there are many diverse ways for people to acquire SQL as a skill set, and various methods to write semantically equivalent SQL queries, this presents to us both the challenge and opportunity of understanding how students learn SQL as they work on homework assignment questions. In this paper, we analyze students' SQL submissions to the homework assignment problems of the Database Systems course available to upper-level undergraduate and graduate students at the University of Illinois at Urbana-Champaign. For each student, we compute the sequence alignment scores between every submission and their final submission to understand how students reached their final solution, and whether there were any obstacles in their learning process. We also utilize hierarchical clustering techniques to create a class-wide aggregate view to determine the number of different approaches used by students in the course. We compute the resulting dendrogram visualization based upon students' final attempt to a homework problem. Our system enables instructors with more visibility to identify interesting learning patterns and approaches. These findings aim at supporting instructors to target their instruction in difficult SQL areas for the future so students may learn SQL more effectively.

CCS CONCEPTS

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

KEYWORDS

SQL, database education, online assessment, hierarchical clustering, sequence alignment

ACM Reference Format:
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 (DataEd'22), June 17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3531072.3535319

DataEd'22, June 17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9350-8/22/06.
https://doi.org/10.1145/3531072.3535319

1 INTRODUCTION

The Structural Query Language (SQL) is the defacto data management language that is supported by most Database Management Systems [DiFranza 2020]. Being the dominant database language, acquiring SQL skills is crucial for users, developers, and researchers alike who interact with databases [Mitrovic 1998]. With an English-like syntax, this language is highly structured and accessible for beginners, as it does not depend on expertise of other programming languages. However, many students still experience difficulties in learning SQL [Mitrovic 1998], and more research is needed in this area to closely analyze how students learn SQL and the challenges they face in their learning path.

In order to analyze how students are learning SQL more effectively and efficiently, database instructors could benefit from examining student SQL submissions in an automated fashion. This is especially the case for instructors who utilize auto-graders to accept student submission attempts to a given SQL problem, as tracing through their attempts will uncover valuable insights regarding how they progressed toward the correct solution. Such insights may then assist instructors in determining difficult SQL concept areas that students more often struggle with, and adjust instruction plans to mitigate these challenges. Our research questions include: 1) how do students learn SQL and come up with their final solution to a SQL problem? and 2) how can we empower database instructors to deliver SQL concepts more effectively, adapting to the diverse student population?

Our research questions aim at improving students' educational quality for learning SQL in the Database Systems course offered at the University of Illinois at Urbana-Champaign. The Database Systems course usually has an enrollment of more than 400 students. Students in this course are given multiple SQL in-class group exercises and an individual homework assignment containing 10-15 SQL problems to complete. All SQL questions are auto-graded, and students receive immediate feedback after each submission attempt. It is difficult for the instructor to quickly identify common class-wide areas of struggle in SQL or pinpoint students who are challenged to delegate more attention and support. Due to the class size, the number of submissions for the SQL problems is too large to manually examine as students can submit well over 20 submissions on a single problem, on average.

Previous research work shows that students may take different learning paths, and their learning experience may be significantly enhanced given the instructor's ability to identify the way students

learn [Jinkens 2009]. Based upon these findings, this paper introduces a technique for examining students' progress as they work to solve an SQL problem, and make it easier for instructors to identify different approaches students take to solving the same SQL problem. This technique builds upon our previous research work, which uses the Levenshtein Edit Distance to compute the distance between each students' submission with their final submission to understand their thought-processes for reaching their final solution [Yang et al. 2021]. We optimized our technique to use the global sequence alignment scores instead, due to the algorithm's customizability. For example, by using the global sequence alignment scores, we may use custom scoring matrices that helps to reduce noise in the dataset and increase accuracy in our analyses.

We examine students' submissions to SQL homework assignment problems offered in the Spring 2021 semester. Our system visualizes the global sequence alignment scores of each student's submissions with their final submission on a SQL problem. We also visualize the global sequence alignment scores of each student's final submission with the instructor's solution to the SQL problem, in order to cluster students' approaches for a class-wide aggregation visualization. These visualizations empowers instructors to identify interesting learning patterns and approaches, which will help target their instruction in difficult SQL areas for the future and support students to learn SQL more effectively.

2 RELATED WORKS

In our previous work-in-progress research, we examined student SQL submissions by computing the Levenshtein Edit Distance between each students' submission with their final submission [Yang et al. 2021]. The resulting visualization was a line chart with the Levenshtein Edit Distances between each of the submission numbers and the final submission [Yang et al. 2021]. The Levenshtein Edit Distance algorithm was a variation of the global sequence alignment algorithm, computing the global alignment score with a scoring matrix of +1 for matches and -1 for mismatches/gaps. However, it provides little opportunity for optimizations based on our use-case; we build upon this work but instead utilize the global sequence alignment scores for an improved accuracy. We explored variations of scoring matrices and opted for one (+3 matches;-1 mismatches/gaps) that worked well with reducing noise in our data, making it easier to extract insights from our dataset. We also opt to use SQL keywords to be used as keys for comparisons in the sequence alignment (instead of a letter-by-letter comparison); this optimizes the accuracy of our alignments by assigning an equivalent weight for each SQL keyword, instead of varying weights based on the length of the arbitrary SQL keyword. This helps to focus our analyses based on the structure of the SQL query, instead of the non-semantically related components of a SQL query.

We leverage hierarchical clustering techniques to generate a dendrogram visualization that shows distinctive approaches to solving a SQL problem from the class. The hierarchical clustering is performed via single-linkage which utilizes the least interconnecting dissimilarity between two elements (in this case is the SQL query submission, with the similarity being represented by the alignment score) [Murtagh and Contreras 2012; Nielsen 2016]. The dendrogram visualization is a graphical representation of the tree-structure that embodies all the data elements and their distances between the subsets that they belong to [Nielsen 2016]. We construct our dendrogram using the Scipy cluster library [Virtanen et al. 2020].

Park et al. utilize sequence pairwise alignment techniques to detect SQL injection attacks, which is a security phenomenon where malicious attackers retrieved access to the database behind an application, and can reveal or exploit confidential data [Park and Noh 2007]. Park et al. applied sequence alignment algorithms to the web application parameters to detect and prevent SQL injection attacks [Park and Noh 2007]. Our work differs such that sequence alignment techniques are applied directly to the SQL queries in order to gain insights regarding how the students progressed through their thought-processes for forming their final solution.

Argenta et al. combines tokenization, sequence alignment, and clustering techniques to events in the adaptive game-based training (AGBT) domain [Argenta and Hale 2015]. They define their alignment keys as the tokenized events, instead of the string DNA sequence alphabet (more commonly what the algorithm is used for) [Argenta and Hale 2015]. Similarly, our work tokenizes the SQL *Keyword* and *Name* elements, and constructs our alignment dictionary based on these elements instead of the DNA sequence alphabet. However, the applications of such techniques differ drastically between Argenta et al.'s work and our research work.

The SQL-Tutor is an intelligent tutoring system aimed at guiding students toward a correct SQL solution through on-demand feedback [Mitrovic 1998]. Feedback is divided into five levels, which gives students varying amounts of information to troubleshoot their query, or understand why their query was wrong (given the correct solution at the highest level)[Mitrovic 1998]. SQL-Tutor analyzes students' submissions by comparing them with a knowledge base of correct solutions based on constraints[Mitrovic 1998]. It supports *SELECT* statements, but does not support advanced SQL queries involving subqueries. This work focuses on assisting students to write correct SQL queries, while our research work focuses on assisting instructors to understand how students struggle in writing SQL queries.

Cagliero et al. similarly examined errors and challenging areas of SQL [Cagliero et al. 2018]; however, their analyses only includes data in an aggregate form which limits the opportunity for instructors to pinpoint specific students who require more attention and assistance [Cagliero et al. 2018]. Our research work empowers instructors with both an aggregate class-wide view with the dendrogram visualizations and an individual view of each student with line-chart visualizations showcasing how the student progressed through constructing their final solution query.

QueryViz is a novel visualization tool for helping pupils to read, understand, and comprehend existing SQL queries in a more efficient manner [Danaparamita and Gatterbauer 2011]. Our research work instead allows instructors to examine changes in students' solutions between submissions in order to identify new approaches or thought-processes that students utilized while constructing a SQL solution query.

Ahadi et al. conducted work [Ahadi et al. 2015] that evaluated the difficulty of constructing particular types of SQL queries through examining whether a student reached the correct solution. Our research work instead looks at how each student may have overcame

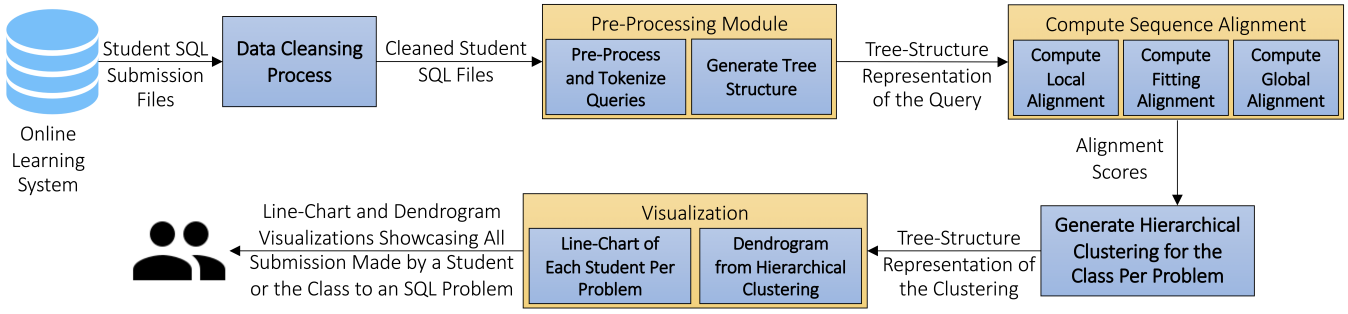

Figure 1: System Overview Diagram.

obstacles in their learning journey to progress toward the correct SQL solution query for a given problem.

3 DATA COLLECTION

We collected data from CS 411 Database Systems, a course available to upper-level undergraduate and graduate students at the University of Illinois at Urbana-Champaign. The data collected from the Spring 2021 semester was amidst the COVID-19 pandemic where instruction was delivered remotely following a flipped-classroom model; students are assigned pre-recorded lectures to review followed by a short quiz about the lecture video. The bulk of the class meeting time is dedicated for students to collaborate on group exercises aimed at solidifying their understanding of the concepts presented in the pre-recorded lectures. Students were also assigned a week-long homework assignment consisting of approximately 10-15 SQL questions. Course enrollment for the Spring 2021 semester consists of 417 students.

3.1 Description of Homework Assignments

We collected our data in PrairieLearn, an online learning management system that auto-grades code and provides immediate feedback to students regarding their submissions [West et al. 2015]. PrairieLearn showcases and compares the student query's data result output against the expected solution's data result output to validate the student's solutions on a binary grading scale (no partial credit is given, although students may see that their query passed some test cases). Students may answer the homework questions in any order with unlimited attempts up until the deadline. Students may also choose to move on and/or come back to homework questions, even if the question had already been answered correctly. An example of an SQL problem from the Spring 2021 semester and its instructor solution is shown below.

> *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;
```

3.2 Data Cleansing

Given that students are allowed unlimited attempts on a given SQL problem, we only kept the submission attempts up till and including the first correct submission receiving full-credit (according to the auto-grader feedback in PrairieLearn). If the student did not receive credit for a homework problem, we assumed that the last submission is their best solution attempt and kept all of their submissions. We labeled the .sql files in our dataset such that they are sorted based on the semester the course was delivered, student, homework problem number, and submission number. The submission number is sorted using natsort [Morton 2021] to determine the chronological order of submission, since larger submission numbers indicate a later submission while smaller submission numbers indicate earlier submissions.

Upon obtaining a clean dataset, we followed all of the University of Illinois at Urbana-Champaign (UIUC) Institutional Review Board (IRB) specified data safety protocols to maintain maximum anonymity for the students whom we collected the SQL submissions from. We removed all identifiers from the SQL files and assigned a randomized number to represent each student.

4 SYSTEM OVERVIEW

Figure 1 is our system overview diagram. First, we collect our data from PrairieLearn [West et al. 2015], where students submit their SQL solution queries for the Database Systems course. Our dataset is a collection of .sql files, where each file represents one submission attempt of a student to a particular SQL homework assignment problem. After the data cleansing process, the .sql files undergo transformations in the pre-processing module which utilizes the Python sqlparse library [Albrecht 2020] to parse and tokenize the SQL queries based on their component type (*Punctuation*, *Keyword*, *Comment*, *Name*, *Literal*, *Operator*, etc.). Components of SQL

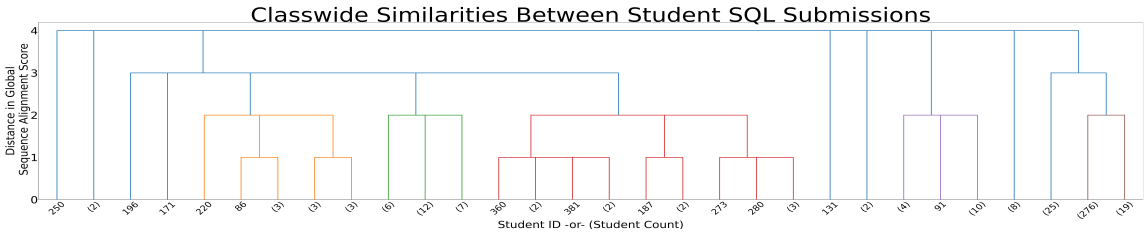

Figure 2: Dendrogram representation of the hierarchical clustering of students' initial SQL query submission attempt against the instructor's solution query. The "lastp" truncate_mode parameter for improved readability is used, which results in both the student ID and student count (when truncated) being displayed.

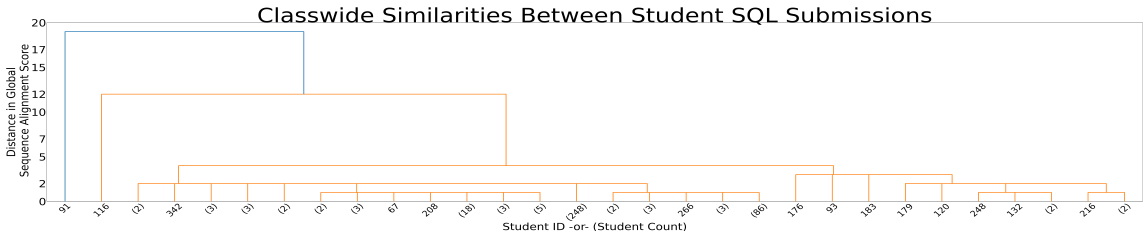

Figure 3: Dendrogram representation of the hierarchical clustering of students' final SQL solution query submission attempt against the instructor's solution query. The "lastp" truncate_mode parameter is used, resulting in both the student ID and student count (when truncated) being shown.

queries that do not contribute to the meaning of the query (such as comments, white-spaces, new-line characters, etc.) were excluded to reduce noise in our dataset. We also only decided to include SQL *Keyword* and*Name* elements in the sequence alignment score computations for the sake of better understanding changes in the students' semantic approaches for tackling the SQL problem. These elements include strings such as *SELECT*, *FROM*, *NATURAL JOIN*, and table name aliases. We generate our tree-like structure based on these elements, and compute the global sequence alignment scores between a student's submission to their final submission attempt. These scores are utilized to generate the line-chart visualization for depicting how the student progressed to their final solution. We also compute the global sequence alignment scores between each students' final submission and the instructor's solution, so that we may cluster similar approaches that were used in the class by students while also examining if certain students had overwhelmingly similar solutions.

4.1 Computing the Global Alignment Scores

A global alignment is an optimal end-to-end alignment of the keys within two given sequences. By using the global alignment scores instead of the fitting or local alignment scores, our computations will be the most sensitive to differences in the lengths of the two sequences. We implemented the global alignment algorithm based on the Needleman-Wunsch[Needleman and Wunsch 1970] algorithm using dynamic programming with modifications to the scoring matrix and keys. We define our alignment dictionary with the tokenized *Keyword* and *Name* components of the SQL queries, instead of using the simple alphabet of letters. This allows us to give an equivalent weight in the alignment scores for each component, instead of having varying scores being biased based upon the length of each token. To demonstrate this concept, a match with a *SELECT* would give the same value as an alignment match with a *NATURAL JOIN*, instead of having the *SELECT* count as 6 matches and the *NATURAL JOIN* count as 12 matches (by characters).

We experimented with various scoring matrices to find a desirable balance between penalties (for gaps and mismatches) and additions (for matches) to highlight findings in our line-chart and dendrogram visualizations. From our experimentation, we decided to continue using the scoring matrix +3 for matches and -1 for mismatches/gaps.

To help visualize the global alignment score computation, we present an example from two submissions of student number 0 to

Alignment Score	3	6	9	8	11	10	13	12	15	14	17	20	23	26	29	28	27	26	25	24	23
Submission 2	select	product name	count	-	as	num customers	from	products	natural join	purchases	where	brandname	group by	product name	having	num customers	-	-	-	-	-
Submission 3	select	product name	count	customerid	as	customers	from	purchases	natural join	products	where	brandname	group by	product name	having	customers	order by	customers	desc	product name	desc

Figure 4: Global sequence alignment for submission 2 and 3 of student number 0 for homework question 12.

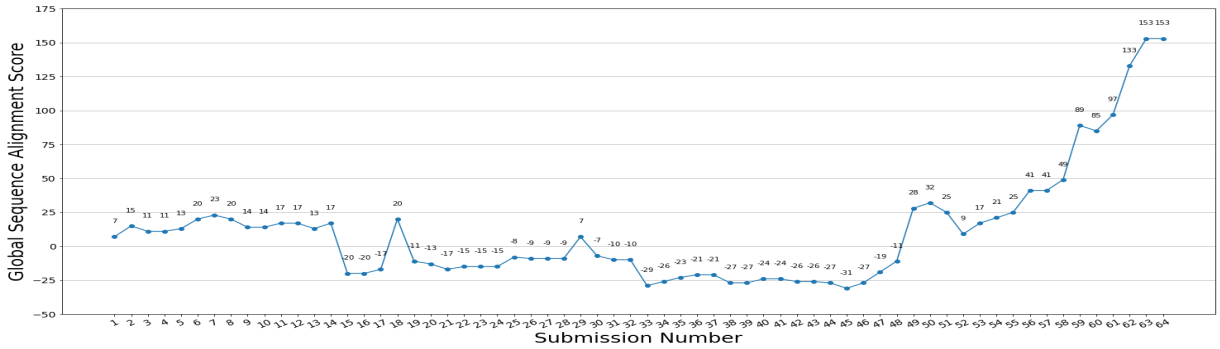

Figure 5: Student 277's visualization of SQL submissions for question 12

question 12 on the SQL homework assignment. Submission number 2 and 3 are shown, after they have been pre-processed.

```
Submission 2:
['select', 'productname', 'count', 'as', 'numcustomers', 'from',
'products', 'natural_join', 'purchases', 'where', 'brandname',
'group_by', 'productname', 'having', 'numcustomers']
Submission 3:
['select', 'productname', 'count', 'customerid', 'as', 'customers',
'from', 'purchases', 'natural_join', 'products', 'where',
'brandname', 'group_by', 'productname', 'having', 'customers',
'order_by', 'customers', 'desc', 'productname', 'desc']
```

The resulting global alignment is shown in figure 4. The matches (+3 in alignment score) are shown in green, and the mismatches/-gaps (-1 in alignment score) are shown in red. The final alignment score is 23 as highlighted in blue. The numbers in the top row represent the current alignment score up till the indicated sequence component.

4.2 Visualization

We create our visualizations with Python's matplotlib library [Hunter 2007] and Scipy cluster framework [Virtanen et al. 2020]. We visualize the global alignment scores between each student's submission with their final submission for a particular SQL problem. This list of global alignment scores become the basis for generating the line-chart visualizations depicting how the student made changes to their query and progressed towards their final solution query.

We also compute the global alignment scores between all the final submission queries of students in the Database Systems course and the instructor's solution query. We utilize the Scipy cluster library [Virtanen et al. 2020] to generate the dendrogram visualization representing the hierarchical clustering of these final submission attempts. The dendrogram easily depicts the number of dissimilar approaches taken by the class to solve the given SQL problem, as well as if any students had extremely similar submissions.

5 RESULTS

We present our insights from the visualizations generated by the matplotlib line-charts and Scipy dendrograms as part of our system. In figure 2, we present the dendrogram representation resulting from hierarchical clustering of students' first SQL submission attempt (based on its global alignment score with the instructor's solution query). In figure 3, we present the dendrogram resulting from the hierarchical clustering of students' final SQL submission attempt. We can see that based on the disparities of the two figures, students in the course start out with different starting points and approaches to solving the same SQL question, but tend to converge to only a few different approaches (approximately 2-3) in their final attempt. Instructors can identify the number of approaches and their similarity level by tracing the number of branches extending from the root of the dendrogram. For each type of approach, an instructor may observe students displaying similar behaviors, and the visualization may thus help instructors divide students into categories that may need varying types of support. Although most students share a generally similar approach, they have slight variations in their SQL solutions, resulting in a more densely packed lower level and a more sparse upper level in the diagram. For the sake of readability, the "lastp" truncate_mode parameter was used in the display of figures 2 and 3. We are able to identify students who had extremely similar solution queries at the leaf node level where the distance in global alignment scores between them are 0. Similarly, we are enabled with the feature to identify the most common approaches being utilized to solve the SQL problem by looking at the branch with the highest number of leaf nodes.

By identifying the students who utilize similar approaches and the approach they take (using a non-truncated dendrogram visualization), instructors may target instruction to include such types of approaches in lectures to strengthen students' understanding. Students may have a tendency to write SQL queries based on the structure of queries demonstrated in the lecture; these approaches may contain novel ways of solving SQL queries previously not taught in the lectures, and by exposing students to various approaches, it may potentially strengthen students' problem-solving capabilities. Instructors may also group students who utilize differing approaches to work on SQL problems together on the course collaborative learning assignments, so students can have the opportunity to familiarize themselves with different approaches to SQL problems.

In figure 5, we present the line-chart that showcases all of a sample student's submission being compared with their final submission using the global sequence alignment score. We can observe from the visual that the student was struggling during the earlier submissions. This observation is validated with the auto-grader results indicating that the student had syntactical errors with these submissions. Between submissions 15 and 17, while the student resolved their syntax error, a semantic error was present, implying that the submitted queries were logically incorrect. The dip in figure 5 here is supported by the fact that the student submitted a

much shorter, simpler query. The rise at submission 18 is explained by the student readopting the earlier approach, while only minor changes were made to the submitted attempts between submissions 19-32. At submissions 33 and 47, the student utilizes new approaches to build upon for solving this SQL problem, in which we can see that submissions 48-64 continuously builds upon itself by wrapping earlier submissions as sub-queries to the final solution. We have analyzed other case studies of students' line-chart visualization, and were presented with similar and consistent findings and patterns.

By validating our visualizations against the student submissions, we can see that there are a few approaches students take to tackling the homework problem, empowering instructors with a more well-rounded understanding of how their students learn.

6 CONCLUSION AND FUTURE WORK

We have presented a novel system for analyzing students' SQL submissions by visualizing the global sequence alignment scores between their submission attempts and their final submission. The dendrogram generated from the hierarchical clustering helps database instructors to quickly identify 1) how many different approaches did the students of the course utilize for solving the same SQL problem? and 2) which students had submitted a SQL solution query that highly resembled others in the class, and whom? Answering such questions for instructors who manually sift through SQL submissions in a large-class setting is highly unrealistic and time-consuming; however, with our system, the analyses process is more straightforward and done in a much more automated fashion, empowering instructors with insights on a class-wide level. Essentially, this enables instructors the opportunity to familiarize themselves with how their students are picking up SQL.

For our future work, we propose to construct an interactive tool that encompasses the mentioned types of visualizations in this paper, in a more user-friendly, user-facing application. Based on the visualizations generated by the tool back-end system, action items may be automatically proposed to the instructor which may include 1) a list of student names that require more attention and support, and with which SQL concepts and 2) a list of set of students who had SQL submission attempts that shared an extremely high resemblance for further investigation of possible plagiarism.

ACKNOWLEDGMENTS

We give a huge thank you to Professor Mohammed El-Kebir for his continuous support, feedback, and guidance for giving our data analyses stage direction. His expertise in Computational Biology was a very crucial contribution in introducing tools and algorithms that directly impacted our research work, allowing us to more effectively extract insights from student SQL submissions. We thank Ziyuan Wei with her support and contribution with the pre-processing stage. We also thank all the students in the Database Systems course for providing us the dataset upon which our research work is based off of, and to allow us to contribute to the field of Computing Education.

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

REFERENCES

A. Ahadi, J. Prior, V. Behbood, and R. 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.

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

Chris Argenta and Christopher R Hale. 2015. Analyzing variation of adaptive game-based training with event sequence alignment and clustering. In *Proceedings of the third annual conference on advances in cognitive systems poster collection*. 26.

L. Cagliero, L. De Russis, L. Farinetti, and T. Montanaro. 2018. Improving the Effectiveness of SQL Learning Practice: A Data-Driven Approach. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 01. 980–989.

J. Danaparamita and W. 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)*. ACM, New York, NY, USA.

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

J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

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

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.

Seth M. Morton. 2021. 8.0.2 (2021). https://pypi.org/project/natsort/

Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2 (2012).

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

Frank Nielsen. 2016. *Hierarchical Clustering*. Springer International Publishing, Cham, 195–211. https://doi.org/10.1007/978-3-319-21903-5_8

Jae-Chul Park and Bong-Nam Noh. 2007. SQL Injection Attack Detection: Profiling of Web Application Parameter Using the Sequence Pairwise Alignment. In *Information Security Applications*, Jae Kwang Lee, Okyeon Yi, and Moti Yung (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 74–82.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

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.

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