Leinonen, Juho; Pirttinen, Nea; Hellas, Arto

# Crowdsourcing Content Creation for SQL Practice

*Published in:*
ITiCSE 2020 - Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education

*DOI:*
10.1145/3341525.3387385

Published: 15/06/2020

*Document Version*
Peer reviewed version

*Please cite the original version:*
Leinonen, J., Pirttinen, N., & Hellas, A. (2020). Crowdsourcing Content Creation for SQL Practice. In *ITiCSE 2020 - Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 349-355). (Annual Conference on Innovation and Technology in Computer Science Education). ACM. https://doi.org/10.1145/3341525.3387385

# Crowdsourcing Content Creation for SQL Practice

Juho Leinonen
juho.leinonen@helsinki.fi
University of Helsinki
Helsinki, Finland

Nea Pirttinen
nea.pirttinen@helsinki.fi
University of Helsinki
Helsinki, Finland

Arto Hellas
arto.hellas@aalto.fi
Aalto University
Espoo, Finland

## ABSTRACT

Crowdsourcing refers to the act of using the crowd to create content or to collect feedback on some particular tasks or ideas. Within computer science education, crowdsourcing has been used – for example – to create rehearsal questions and programming assignments. As a part of their computer science education, students often learn relational databases as well as working with the databases using SQL statements. In this article, we describe a system for practicing SQL statements. The system uses teacher-provided topics and assignments, augmented with crowdsourced assignments and reviews. We study how students use the system, what sort of feedback students provide to the teacher-generated and crowdsourced assignments, and how practice affects the feedback. Our results suggest that students rate assignments highly, and there are only minor differences between assignments generated by students and assignments generated by the instructor.

## CCS CONCEPTS

• **Information systems** → *Crowdsourcing*; • **Applied computing** → *Interactive learning environments*; • **Social and professional topics** → *Computing education.*

## KEYWORDS

crowdsourcing, SQL, introduction to databases, teacher generated content, student generated content, assignment creation, assignment quality

**ACM Reference Format:**
Juho Leinonen, Nea Pirttinen, and Arto Hellas. 2020. Crowdsourcing Content Creation for SQL Practice. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387385

## 1 INTRODUCTION

Crowdsourcing has been used in computing education, for example, to have students create multiple-choice questions [12] or programming assignments [15, 31], to have educators create a large assignment pool [35], and to have educators generate new learning material [36]. However, an aspect of introductory computing that has been largely overlooked by those creating crowdsourcing systems is SQL practice. While a versatile tool such as PeerWise [12] can be used for creating multiple-choice questions related to any topic, it would be useful to have a system in which students create SQL assignments that require others to write actual SQL queries.

In this article, we present a system in which students can design SQL assignments, peer review assignments created by other students, and complete assignments from the large crowdsourced assignment pool. We study students' use of the system by analyzing the feedback students give to assignments within the system during peer review. Additionally, we compare the feedback to assignments created by the instructor to those assignments created by the students.

Our hypothesis is that student-generated SQL assignments are close to the instructor generated assignments with regards to quality. This is because previous work on assessing the quality of student generated crowdsourced resources has found that students can create assignments that are of good quality [13, 19] and that in the case of crowdsourced programming assignments, there are no large differences between assignments created by more experienced programmers and novice programmers [32].

This article is structured as follows. In Section 2, we detail prior work related to crowdsourcing in computer science education and learning and teaching SQL. In Section 3, we present the system we have developed. Section 4 describes our research methodology, and Section 5 presents the results of the study, which are discussed in Section 6. Lastly, we conclude the work in Section 7.

## 2 RELATED WORK

### 2.1 Crowdsourcing in CS Education

Crowdsourcing is a method of obtaining content, ideas or goods from a large group of people, usually on the Internet. The term first appears in a 2006 article by Howe [21], born as a portmanteau between words *crowd* and *outsourcing*, the act of employing outside workforce to perform required tasks.

The rapid growth of the usage of the Internet and especially social media platforms has made both the advertising of and participating in crowdsourcing activities easier. A well-known example of crowdsourcing is Wikipedia[1], an online encyclopedia maintained in almost 300 languages through user-based open collaboration. Crowdsourcing is very suitable for small tasks that require human interaction, and cannot be automated. For example, Amazon Mechanical Turk[2] provides a platform for businesses to hire users to complete discrete tasks.

One of the main uses of crowdsourcing in computer science education is to alleviate the time spent on creating new exercises for each course iteration. This can be achieved by using material banks, usually consisting of exercises, visualizations or lecture materials from instructors, or by tools that both teach programming and

*ITiCSE '20, June 15–19, 2020, Trondheim, Norway*


[1] https://www.wikipedia.org/
[2] https://www.mturk.com/

collect content at the same time. Using the student population for crowdsourcing efforts produces a larger mass of results with less effort from the instructors, but requires more curating, either with systematically set out tasks during the crowdsourcing, or with manual or automated filtering afterwards.

For example, PeerWise [12] is a web-based tool that is used to crowdsource multiple-choice questions. While the format of multiple-choice questions makes them applicable for a wide range of educational fields, studies mostly investigate the usage of PeerWise in computer science courses. Students create questions and two to five answer alternatives, mark the correct answer and give an explanation for why the chosen option is the correct one. All of the multiple-choice questions entered into the system are available as exercises for other students. After completing an exercise, students can review and rate the question for its cohesion, difficulty, relevance and so forth. This way, PeerWise works as both a revision system through creating and answering multiple-choice questions, and curates itself through a peer-review system.

Similarly, CrowdSorcerer [31] is an embeddable tool that is used for programming assignment creation. Students come up with an assignment description according to some specifics given by an instructor. This can be, for example, creating an assignment that uses conditional clauses. Then, the student programs a model solution and a code template for the assignment. The model solution is a full, working answer to the assignment, while the code template only contains the basic structure of the program, like the main method declaration, without the actual implementation lines. The user also creates test cases for their program. The program code is automatically tested for compilation errors, and the user-given tests make sure that the program works as it is supposed to. Finished assignments are peer reviewed using both written feedback and Likert-like review statements.

As for the expert curated material banks, Canterbury QuestionBank [35] is a comprehensive set of multiple-choice questions suitable for first-year computer science courses. An ITiCSE 2013 working group of computer science educators produced over 650 questions and answers suitable for, for example, quizzes and exams. The question bank can be found and contributed to through its website[3]. OpenDSA [36] is a collection of online materials supporting wide range of computer science courses. The materials include visualizations, course books, exercises and modules that can be used as a part of any course material regarding, for example, programming, data structures and algorithms, or formal languages. The materials provided by the system are available on the webpage[4] of the project.

## 2.2 Quality of Crowdsourced Content

End-product quality and the trustworthiness of the crowd are valid concerns with crowdsourcing. The issuer cannot be entirely sure of the skill level, nor the ulterior motives, their crowd has. Low quality contributions are not necessarily only due to varying levels of competence, but also due to intentional maliciousness, either for the person's own amusement or to hurt the company they are working for.

[3]http://web-cat.org/questionbank/
[4]https://opendsa-server.cs.vt.edu/

The crowd can be pre-selected based on their applications or self-reported expertise on the area of the task [6], but this can be time and resource consuming, taking away from the advantages crowdsourcing has over outsourcing, mainly flexibility and swiftness. While carefully defining the scope of the task and the end-product criteria beforehand can work as a method of quality control, many companies accept the additional workload of reviewing the crowdsourced content and discarding low-quality contributions. Allahbakhsh et al. [6] list existing review methods for analysing end-product quality, such as expert review and comparison to a golden standard.

Besides the framing of the tasks, the format in which the tasks are given can also affect the quality of the crowdsourced outcome. According to Lukyanenko et al. [26], free-format tasks seem to produce better quality contributions, though this causes more workload for the issuer as the crowdsourced contents need to be carefully checked and formatted before they can be used.

Another way to alleviate quality issues is peer reviewing, as with the multiple-choice questions in PeerWise [12] and programming assignments in CrowdSorcerer [31]. Studies have shown that the quality of peer reviewed exercises is good [13, 19], even when both the creators and the reviewers are from the same crowd of students. Hamer et al. [19] did not find any significant differences between peer reviews and expert reviews of the same, student-created exercises. No significant differences have been found in reviews done by novices or experienced students [19, 32]. Pirttinen et al. [32] also report that at least with CrowdSorcerer, assignments created by students with more programming experience did not seem to differ from those created by novices quality-wise.

## 2.3 Learning and Teaching SQL

SQL as a language and the formulation of SQL queries can be complex for novices. Several studies have reported about the types of mistakes and errors students encounter when learning to write SQL [3, 4, 9, 37, 41, 42]. Performance in writing SQL is also linked with students' course performance [2]. Thus, instructional approaches for teaching and opportunities to learn SQL are called for.

The use of worked examples and sequencing course content can help students' learning [27]. Helping students reason about and structure patterns that occur in SQL problems may help with recalling appropriate statements needed to solve the problems and to identify (and solve) similar problems [5]. This can be approached through the use of checklists and patterns, or through more problem-specific templates accompanied with a strategy for using those templates [5, 33, 39, 40].

Learning SQL is typically not limited to the language itself, as students need to also learn to study and reason about the domain and the underlying schema [20, 33]. In other words, the way how the problems are phrased [7, 8, 10] and the database schema [25] both contribute towards the difficulty of the query formulation task. At the same time, once students learn SQL, it can be a useful tool in teaching: one can, for example, use SQL to teach other database concepts such as transaction management [18].

Multiple tools have been developed for teaching SQL. These include both traditional practice systems where students are given a set of automatically assessed SQL problems that they need to

solve [1, 17, 24, 38], and tutoring systems [22, 23, 28–30], which provide further scaffolding to the learning process by, for example, providing hints, partial solutions, and controlling when the student can proceed to the next type of problems. In both types of systems, instructors have traditionally constructed the problems that students need to solve.

# 3 SYSTEM DESCRIPTION

Here, we describe the system that is used to practice SQL and to crowdsource the content. The system is open source and available at https://github.com/rage/iticse2020-sqltrainer.

## 3.1 Creating Content

The system provides two views: one for the teacher and one for the student. The teacher has the opportunity of creating databases, topics, and assignments, while the student can create and complete assignments. Databases are created by writing a name for the database and the SQL statements needed for creating the database. Topics are essentially categories for assignments – when creating a topic, the system asks for the name of the topic as well as a short description and a rank, which defines the order in which the topics are shown to the student. When creating an assignment, the database and the topic are selected, and the name of the assignment, the assignment handout, and the SQL statements needed to complete the assignment are written.

When students use the system, they see a list of topics ordered by their rank. Each topic has the option of practicing it and creating content into it. If the student chooses to practice a particular topic, they will be given a randomly selected assignment that the student has not yet completed from the pool of assignments within that topic. If the student chooses to create content for a particular topic, the system will ask which database the student wants to use (from the pre-defined list of databases created by the teacher), as well as ask for the name, handout, and SQL statements for that assignment. Once the assignment is completed, it is added to the pool of assignments within that topic.

## 3.2 Practice and Feedback

When working on an assignment, in addition to the assignment name and handout, the student can view the database schema and the expected outcome of the query – if it is possible to show one. If a student is unable to complete the currently selected assignment, they can choose to select another assignment from the pool of assignments within that topic. All attempts towards solving an assignment are stored, and each student has access to their attempt history for each assignment. The number of completed and created assignments by the student is shown in the topic list.

Whenever a user completes an assignment, they are shown a list of feedback questions related to that assignment. The questions that the system shows are created by the teacher. By default, the questions are answered using a 7-point Likert-scale questions ranging from (1 = Not at all, 7 = Very much so). Answering the feedback questions is not mandatory, and skipping the questions leads the student to the next randomly selected assignment within the topic that the student is currently practicing.

## 3.3 Assessment of SQL Assignments

The system supports automatic assessment of SQL select statements as well as SQL statements that alter the database schema. This is realized through depending on the use of in-memory databases that are created and destroyed whenever a student submits a solution to an assignment. For each submission (i.e. assignment attempt), the system creates two in-memory databases: one for the expected outcome and one for the student's outcome. Both of these databases are initiated using the teacher provided SQL statements used for creating the database, and the subsequent queries are based on the model solution in the case of the database depicting the expected outcome, and the student's queries in the case of the database depicting the student's outcome.

First, during the assessment, the system compares the structure and content of the databases, including indices and key constraints. This step is used to ensure that any expected (and non-expected) modifications to the database schema and the content are verified. Then, the output of the last query is assessed; here, the structure of the output and the content of the output is compared for both the model solution and the student's solution to ensure that the output of the query is as expected. Finally, the student's query is heuristically compared with the output to verify that the student has not written a simple query that outputs the expected values without actually querying the database. If these verification steps pass, the student is awarded a point for completing the assignment.

## 3.4 Integrations

The system uses the OAuth 2.0 protocol[5] for user authentication, which means that the backend used for authentication can be practically any backend that supports the protocol. Moreover, the system provides a REST interface that allows teachers access to the points of all students in the system, and access to students for their own points. This interface can be used to e.g. create centralized scoreboards and to embed the points of the current student to the learning material, assuming that students use the same authentication mechanism in the learning materials.

# 4 METHODOLOGY

## 4.1 Context and Data

For the purposes of this study, we used the system described in Section 3 in three instances of an introductory-level databases course, which is typically taken right after the introductory programming course (CS1). The course is offered by University of Helsinki, a public research-oriented university in Finland, and the course can be attended for free by both affiliated and non-affiliated students.

The teacher of the course constructed 11 topics into the system, and added three to five assignments into each topic. The topics included simple SQL select statements on one or more tables, comparisons for limiting results, constructing aggregate queries, constructing nested queries, and altering the database schema (creating, removing, updating tables, working with indices). In total, the teacher generated 45 SQL assignments into the system.

The grade of the course is formed from weekly assignments (55% of the grade), two projects (30% of the grade), and an exam (15% of

[5] https://oauth.net/2/

the grade). Grading-wise, the students must receive at least half of the overall points and at least half of the exam points to pass the course. The highest grade can be achieved by collecting at least 90% of the overall course points. In the course, completing and creating SQL statements within the system contributed approximately 10% of the overall grade. To reach full points from SQL training, students were expected to complete at least 44 SQL assignments (4 from each topic) and to create at least 11 assignments (1 for each topic).

Furthermore, six feedback questions (FQ) with 7-point Likert-scale answer options ranging from (1 = Not at all, 7 = Very much so) were included. In this work, we refer to the questions with single-word labels. The labels and questions were as follows:

FQ1. Easiness: The exercise was too easy.
FQ2. Clarity: I think that the exercise handout was clear.
FQ3. Educational: I learned something when working on the exercise.
FQ4. Suitability: The difficulty of the exercise was suitable for me.
FQ5. Confidence: I think that my solution is good.
FQ6. Frustration: Working on the exercise was frustrating.

The analysis in the present work focuses on students' work within the system. Data from all course instances have been aggregated for the purposes of the analysis.

### 4.2 Research Questions and Approach

Our research questions for this study are as follows.

RQ1 How do students respond to the feedback questions within the system?
RQ2 How are the responses to the feedback questions related to each other?
RQ3 How does the feedback from the crowdsourced assignments compare to the feedback of the assignments created by the instructor?
RQ4 To what extent does working on the assignments influence feedback?

For correlations, we have used Spearman's Rank Order Correlation. For comparing populations, we have used the Mann-Whitney U test. In all analyses, when analyzing statistical significance, we have used a Bonferroni correction [16] to correct for multiple ($n = 61$) comparisons. We consider $p < 0.05$ as statistically significant, after correcting the p-value using the Bonferroni correction.

## 5 RESULTS

### 5.1 Descriptive Statistics

A total of 1,410 students entered the system and attempted at least one assignment. From these, 1,359 completed at least three assignments (96%), 1,184 created at least one assignment (84%), and 1,007 reviewed at least one assignment (71%). In total, students generated 12,059 assignments (on average, 10.1 per student who created at least one assignment). There were in total 198,165 attempts in completing the assignments, and from the 12,059 assignments, 1,163 were not solved by any of the students (9.6% of generated assignments). Furthermore, 5,325 of the assignments received at least 1 review (44.2% of generated assignments).

| Question | Mean, Median, (std) |
|---|---|
| FQ1. Easiness | 3.60, 4, (1.82) |
| FQ2. Clarity | 5.65, 6, (1.65) |
| FQ3. Educational | 5.35, 6, (1.64) |
| FQ4. Suitability | 5.24, 6, (1.62) |
| FQ5. Confidence | 5.94, 6, (1.37) |
| FQ6. Frustration | 2.33, 2, (1.78) |

**Table 1: Feedback averages. Likert answers between 1 (Not at all) and 7 (Very much so).**

| | FQ1. | FQ2. | FQ3. | FQ4. | FQ5. | FQ6. |
|---|---|---|---|---|---|---|
| FQ1. Easiness | 1.00 | | | | | |
| FQ2. Clarity | ns | 1.00 | | | | |
| FQ3. Educational | -0.46 | 0.46 | 1.00 | | | |
| FQ4. Suitability | -0.42 | 0.49 | 0.77 | 1.00 | | |
| FQ5. Confidence | ns | 0.56 | 0.35 | 0.41 | 1.00 | |
| FQ6. Frustration | ns | -0.60 | -0.28 | -0.36 | -0.43 | 1.00 |

**Table 2: Correlations between answers to different feedback questions. Statistically non-significant correlations are indicated with "ns".**

In the subsequent analyses, unless otherwise stated, we limit the analysis to those 1,359 students who have completed at least 3 assignments.

### 5.2 Overall Feedback on Assignments

To answer RQ1 *"How do students respond to the feedback questions within the system?"* we first calculated the means, medians, and standard deviations for the different feedback questions, which are shown in Table 1. The Likert-scale was between 1 (Not at all) and 7 (Very much so).

Overall, students rated the clarity and suitability of the assignments high, and considered that the assignments were educational and that their own solutions (to the assignments) were good. The assignments were not too easy, but also not too difficult, and on average the assignments were not considered frustrating.

### 5.3 Feedback Correlations

To answer RQ2 *"How are the responses to the feedback questions related to each other?"* we calculated the correlations between students' answers to different feedback questions. Analysis was limited to those assignments that received at least three reviews. Summary of the correlations are shown in Table 2. All of the correlations except for FQ1-FQ2, FQ1-FQ5, and FQ1-FQ6 were statistically significant. Non-significance is indicated with "ns" in the table.

Overall, the answers to the feedback questions correlate with each other mostly moderately. The only strong correlation between any two feedback questions is the correlation between FQ3 and FQ4 ($r = 0.77$), i.e. whether the student felt they learned something and whether the student considered the assignment suitably difficult.

| Question | Instructor mean (std) | Student mean (std) |
|---|---|---|
| FQ1. Easiness | 3.62 (1.77) | 3.58 (1.84) |
| FQ2. Clarity* | 5.89 (1.33) | 5.51 (1.79) |
| FQ3. Educational* | 5.59 (1.44) | 5.20 (1.73) |
| FQ4. Suitability* | 5.33 (1.53) | 5.18 (1.67) |
| FQ5. Confidence | 5.96 (1.28) | 5.93 (1.42) |
| FQ6. Frustration | 2.31 (1.68) | 2.34 (1.83) |

**Table 3: Feedback averages for instructor and student generated assignments. Likert answers between 1 (Not at all) and 7 (Very much so). For questions marked with an asterisk, the difference between averages was statistically significant (p-value < 0.05) after a Bonferroni correction. Medians in both groups followed the medians shown in Table 1.**

## 5.4 Comparing Feedback from Teacher's Assignments with Students' Assignments

To answer RQ3 *"How does the feedback from the crowdsourced assignments compare to the feedback of the assignments created by the instructor?"* we compared the means and standard deviations of the feedback questions separately for instructor generated assignments and student generated assignments. The feedback means and standard deviations are shown in Table 3, medians in both groups follow the overall median in the feedbacks as shown in Table 1.

Overall, the differences in feedback answer means are small between the assignments created by the instructor and the assignments created by students. The differences are only statistically significant for three feedback questions FQ2, FQ3, and FQ4, i.e. whether the handout was clear, whether the student felt that they learned something, and whether the student found the assignment suitably difficult.

## 5.5 Behavior and Feedback

To answer RQ4 *"To what extent does working on the assignments influence feedback?"* we calculated the correlations between answers to feedback questions and students' assignment completion behavior, i.e. the number of assignments attempted, completed, created and reviewed. Additionally, the correlation between the previous and the average distance to deadline was calculated. The average distance to deadline was calculated from students' attempted assignments. For the metric, a larger value represents a higher distance. The results are shown in Table 4.

Overall, assignment completion behavior only correlates weakly or statistically non-significantly with feedback question answers. Assignment completion behavior metrics correlate with each other moderately, except for the distance to deadline, which correlates either weakly or non-significantly with the other behavioral metrics.

# 6 DISCUSSION

Our results show that students rank crowdsourced assignments highly overall, which is in line with previous work [32]. On the other hand, other previous work found that students tend to undermark their peers' scores [19]. The relatively high overall scores of assignments in this work indicates that at least most of the crowdsourced assignments seem good to other students.

When looking at how the answers to feedback questions relate to one another, we found that the highest correlation (0.77) was between suitability and educational value, which is not surprising. After all, considering the zone of proximal development [11], assignments should be suitably difficult for maximal educational value. The non-significant correlations were between easiness and clarity, confidence, and frustration. Of these, it is somewhat surprising that easiness does not correlate with confidence or frustration statistically significantly. One would assume that the easier the assignment, the more confident a student would be in their own answer. One explanation for the non-significance between frustration and easiness could be that both too easy and too hard assignments cause frustration, and thus the correlation is not linear.

Comparing student- and instructor-generated assignments, we found that the feedback to student-generated assignments had slightly larger standard deviations. This is not surprising, since only a single instructor created assignments, and thus those assignment are likely to be of similar quality. Overall, there are no major differences between the feedback to student- and instructor generated-assignments, which is in line with previous work [14]. The statistically significant differences, albeit minor, were in clarity of the handout, educational value of the assignment, and suitability. For all of these three, the instructor-generated assignments were rated slightly higher, which makes sense, since the instructor probably has more experience in generating assignments and thus can generate assignments that are slightly clearer, more educational, and more suitable.

Considering students' behavior with regards to attempting, completing, creating, and reviewing assignments and their correlations with answers to feedback questions, we found that the number of attempted, completed, and created assignments did not correlate much with feedback questions, while the number of reviewed assignments correlated weakly with feedback questions. We believe that most students attempt and complete assignments, since that is one of the main methods for students to learn SQL on the course, while a smaller sub-population reviews assignments, which was optional and does not contribute towards points. Interestingly, distance to deadline did not have a significant correlation with almost anything, with the only exception being the number of attempted and completed assignments, where the correlation was weak and negative. This means that the students who attempted assignments close to the deadline, attempted and completed slightly more assignments on average. We are not sure of the reasons for this phenomenon, but one possible explanation could be that students who have performed well – who also attempt and complete more assignments – in the course are more confident in their skills and thus have no need to start work early, thus working closer to the deadline. Alternatively, another possible explanation could be that students who perform poorly start work later and need more attempts to feel comfortable with the studied topic.

| | Attempted | Completed | Created | Reviewed | Distance to DL |
|---|---|---|---|---|---|
| Attempted | 1.00 | | | | |
| Completed | 0.77 | 1.00 | | | |
| Created | 0.55 | 0.69 | 1.00 | | |
| Reviewed | 0.29 | 0.36 | 0.36 | 1.00 | |
| Distance to DL | -0.22 | -0.30 | ns | ns | 1.00 |
| FQ1. Easiness | -0.13 | -0.12 | -0.13 | ns | ns |
| FQ2. Clarity | ns | ns | ns | 0.28 | ns |
| FQ3. Educational | ns | ns | ns | 0.18 | ns |
| FQ4. Suitability | ns | ns | ns | 0.22 | ns |
| FQ5. Confidence | ns | ns | ns | ns | ns |
| FQ6. Frustration | ns | ns | ns | -0.12 | ns |

**Table 4: Correlations between feedback question answers and assignment completion behavior. Non-significant correlations are replaced with "ns".**

## 6.1 Limitations

There are multiple limitations to this work, which we will explain here. Firstly, we have calculated averages from Likert-scale questions, where the distance between different options is not necessarily the same. For example, it is possible that the distance perceived by students between e.g. 5 and 6 is not as great as the perceived distance between 6 and 7. Thus, using averages instead of strictly using medians might not represent the data accurately. This is also influenced by the cultural differences in answering Likert-scale questions [34]. In the culture where the study was conducted, people tend to avoid choosing the extreme ends of the Likert-scale, which might affect the results. To combat this problem, we chose to use a 7-point Likert-scale instead of a 5-point scale.

Another limitation is that the instructor disabled some student generated assignments during the course, when it was noticed that none of the students could solve them. It is possible that out of the 1,163 assignments (9.6% of all student-created assignments) some would have been solvable. At the same time, due to feedback being collected once the assignment was completed, these excluded assignments were not reviewed at all. Hence, it is possible that this has influenced the results as some of the student-generated assignments were excluded.

Most of the analysis relies on feedback questions that students answered while reviewing assignments. There were no official guidelines to reviewing except for a prompt asking the student to *review the exercise and your solution using the following form*. Responses to the Likert-scale questions are subjective and it is also possible that students have perceived the feedback questions differently – this may have influenced the reviews that students have given.

Lastly, there is a selection bias when considering the feedback students gave. Since giving feedback on assignments was optional, it is likely that the population that decided to give feedback differs from the overall course population. For example, it is possible that more active students gave more feedback, which could affect the results.

# 7 CONCLUSIONS

In this work, we presented a system for crowdsourcing SQL assignments from students. In the system, students can design their own SQL assignments, review assignments created by their peers, and complete assignments from the crowdsourced assignment pool. We studied how students use the system by analyzing the feedback given by students and students' behavior within the system.

Our results indicate that overall, students rate assignments highly. On a 7-point Likert-scale, ratings given by students tend to be higher than four for positive questions such as whether the assignment was of suitable difficulty and lower than four for negative questions such as whether students were frustrated whilst working on the assignment. The feedback questions (see Section 4.1), which analyzed different aspects of the assignment, were mostly weakly or moderately correlated with one another. Additionally, differences between student- and instructor-generated assignments were either statistically non-significant or modest, which is in line with previous work [14]. Lastly, looking into students' behavior in the system, we found that the numbers of attempted, completed, created, and reviewed assignments were moderately correlated with one another. Additionally, the number of attempted, completed, and created assignments did not correlate significantly with feedback question answers, while the number of reviewed assignments only weakly correlated with answers to feedback questions.

Overall, our results support earlier results [13, 19, 32] which show that crowdsourced student-generated assignments can be of good quality. Thus, such assignments – or at least a curated pool of the best ones – could be used on courses, which can lessen the burden of the instructor with regards to creating a large assignment pool, although we acknowledge that our study does not take into consideration the initial setup overhead of the system. Large assignment pools are necessary, for example, in intelligent tutoring systems, in which students can practice certain skills over and over until the system judges the student to have learned the skill. In our future work, we are interested in studying whether there are differences when considering different types of assignments, for example, assignments related to different topics of the course. Additionally, we are interested in studying the effects of crowdsourcing on student performance and whether students improve as assignment creators as the course progresses, as well as whether their motivation can be positively influenced.

# REFERENCES


[1] Alberto Abelló, M Elena Rodríguez, Toni Urpí, Xavier Burgués, M José Casany, Carme Martín, and Carme Quer. 2008. LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*. IEEE, 592–593.

[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, 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*. ACM, 201–206.

[4] Alireza Ahadi, Julia Prior, Vahid Behbood, 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*. ACM, 272–277.

[5] Huda Al-Shuaily and Karen Renaud. 2010. SQL Patterns-A New Approach for Teaching SQL. In *8th HEA Workshop on Teaching, Learning and Assessment of Databases, Abertay-Dundee*. 29–40.

[6] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. 2013. Quality Control in Crowdsourcing Systems: Issues and Directions. *IEEE Internet Computing* 17, 2 (March 2013), 76–81. https://doi.org/10.1109/MIC.2013.20

[7] Micheal Axelsen, A Faye Borthick, and Paul Bowen. 2001. A model for and the effects of information request ambiguity on end-user query performance. *ICIS 2001 Proceedings* (2001), 68.

[8] A Faye Borthick, Paul L Bowen, Donald R Jones, and Michael Hung Kam Tse. 2001. The effects of information request ambiguity and construct incongruence on query development. *Decision Support Systems* 32, 1 (2001), 3–25.

[9] 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.

[10] Gretchen I Casterella and Leo Vijayasarathy. 2013. An experimental investigation of complexity in database query formulation tasks. *Journal of Information Systems Education* 24, 3 (2013), 6.

[11] Seth Chaiklin. 2003. The zone of proximal development in Vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context* 1 (2003), 39–64.

[12] Paul Denny, Andrew Luxton-Reilly, and John Hamer. 2008. The PeerWise System of Student Contributed Assessment Questions. In *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78 (ACE '08)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 69–74. http://dl.acm.org/citation.cfm?id=1379249.1379255

[13] Paul Denny, Andrew Luxton-Reilly, John Hamer, and Helen Purchase. 2009. Coverage of Course Topics in a Student Generated MCQ Repository. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*. ACM, New York, NY, USA, 11–15. https://doi.org/10.1145/1562877.1562888

[14] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2009. Quality of student contributed questions using PeerWise. In *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95*. Australian Computer Society, Inc., 55–63.

[15] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. CodeWrite: supporting student-driven practice of java. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 471–476.

[16] Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American statistical association* 56, 293 (1961), 52–64.

[17] Masoud I El Agha, Abdallah M Jarghon, and Samy S Abu-Naser. 2018. SQL Tutor for Novice Students. (2018).

[18] Alan Fekete. 2005. Teaching transaction management with SQL examples. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 163–167.

[19] John Hamer, Helen C. Purchase, Paul Denny, and Andrew Luxton-Reilly. 2009. Quality of Peer Assessment in CS1. In *Proc. of the 5th International Workshop on Computing Education Research Workshop (ICER '09)*. ACM, New York, NY, USA, 27–36.

[20] Joseph E. Hollingsworth. 2008. Teaching Query Writing: An Informed Instruction Approach. *SIGCSE Bull.* 40, 3 (June 2008), 351. https://doi.org/10.1145/1597849.1384393

[21] Jeff Howe. 2006. The Rise of Crowdsourcing. *Wired magazine* 14, 6 (2006), 1–4.

[22] Alison Hull and Benedict du Boulay. 2015. Motivational and metacognitive feedback in SQL-Tutor. *Computer Science Education* 25, 2 (2015), 238–256.

[23] Claire Kenny and Claus Pahl. 2005. *Automated tutoring for a database skills training environment*. Vol. 37. ACM.

[24] H Laine. 2001. SQL-trainer. In *Proceedings of Kolin Kolistelut/Koli Calling–First Annual Baltic Conference on Computer Science Education, Report A-2002*, Vol. 1. 13–17.

[25] Robert L Leitheiser and Salvatore T March. 1996. The influence of database structure representation on database system learning and use. *Journal of Management Information Systems* 12, 4 (1996), 187–213.

[26] Roman Lukyanenko, Jeffrey Parsons, and Yolanda F. Wiersma. 2014. The IQ of the Crowd: Understanding and Improving Information Quality in Structured User-Generated Content. *Information Systems Research* 25, 4 (2014), 669–689. https://doi.org/10.1287/isre.2014.0537

[27] Raina Mason, Carolyn Seton, and Graham Cooper. 2016. Applying cognitive load theory to the redesign of a conventional database systems course. *Computer Science Education* 26, 1 (2016), 68–87.

[28] Antonija Mitrovic. 1998. A knowledge-based teaching system for SQL. In *Proceedings of ED-MEDIA*, Vol. 98. 1027–1032.

[29] Antonija Mitrovic. 2003. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education* 13, 2-4 (2003), 173–197.

[30] Claus Pahl, Ronan Barrett, and Claire Kenny. 2004. Supporting active database learning and training through interactive multimedia. *ACM SIGCSE Bulletin* 36, 3 (2004), 27–31.

[31] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing Programming Assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 326–331. https://doi.org/10.1145/3197091.3197117

[32] Nea Pirttinen, Vilma Kangas, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Analysis of Students' Peer Reviews to Crowdsourced Programming Assignments. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling 2018)*. ACM, New York, NY, USA, 5.

[33] Gang Qian. 2018. Teaching SQL: a divide-and-conquer method for writing queries. *Journal of Computing Sciences in Colleges* 33, 4 (2018), 37–44.

[34] Catherine Roster, Gerald Albaum, and Robert Rogers. 2006. Can cross-national/cultural studies presume etic equivalency in respondents' use of extreme categories of Likert rating scales? *International Journal of Market Research* 48, 6 (2006), 741–759.

[35] Kate Sanders, Marzieh Ahmadzadeh, Tony Clear, Stephen H. Edwards, Mikey Goldweber, Chris Johnson, Raymond Lister, Robert McCartney, Elizabeth Patitsas, and Jaime Spacco. 2013. The Canterbury QuestionBank: Building a Repository of Multiple-choice CS1 and CS2 Questions. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports (ITiCSE -WGR '13)*. ACM, New York, NY, USA, 33–52. https://doi.org/10.1145/2543882.2543885

[36] Clifford A. Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L. Naps. 2011. OpenDSA: Beginning a Community active-eBook Project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling '11)*. ACM, New York, NY, USA, 112–117. https://doi.org/10.1145/2094131.2094154

[37] John B Smelcer. 1995. User errors in database query composition. *International Journal of Human-Computer Studies* 42, 4 (1995), 353–381.

[38] Josep Soler, Ferran Prados, Imma Boada, and Jordi Poch. 2006. A Web-based tool for teaching and learning SQL. In *International Conference on Information Technology Based Higher Education and Training, ITHET*.

[39] Lovisa Sundin and Quintin Cutts. 2019. Is it feasible to teach query programming in three different languages in a single session? A study on a pattern-oriented tutorial and cheat sheets. In *Proceedings of the 1st UK & Ireland Computing Education Research Conference*. 1–7.

[40] Toni Taipalus. 2019. A Notation for Planning SQL Queries. *Journal of Information Systems Education* 30, 3 (2019), 2.

[41] 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*. ACM, 198–203.

[42] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and complications in SQL query formulation. *ACM Transactions on Computing Education (TOCE)* 18, 3 (2018), 15.