Experiences from Learnersourcing SQL Exercises: Do They Cover Course Topics and Do Students Use Them?

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

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

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

ABSTRACT

Learnersourcing is an emerging phenomenon in computing education research and practice. In learnersourcing, a crowd of students participates in the creation of course resources such as exercises, written materials, educational videos, and so on. In computing education research, learnersourcing has been studied especially for the creation of multiple-choice questions and programming exercises, where prior work has suggested that learnersourcing can have multiple benefits for teachers and students alike. One result in prior studies is that when students create learnersourced content, the created content covers much of the learning objectives of the course. The present work expands on this stream of work by studying the use of a learnersourcing system in the context of teaching SQL. We study to what extent learnersourced SQL exercises cover course topics, and to what extent students complete learnersourced exercises. Our results continue the parade of previous learnersourcing studies, empirically demonstrating that learnersourced content covers instructor-specified course topics and that students indeed actively work on the learnersourced exercises. We discuss the impact of these results on teaching with learnersourcing, highlight possible explanations for our observations, and outline directions for future research on learnersourcing.

CCS CONCEPTS

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

KEYWORDS

learnersourcing, crowdsourcing, SQL, exercise creation, SQL exercises, coverage of course topics, topic coverage, quality of learnersourced content

ACM Reference Format:
Nea Pirttinen, Arto Hellas, and Juho Leinonen. 2023. Experiences from Learnersourcing SQL Exercises: Do They Cover Course Topics and Do Students Use Them?. In *Australasian Computing Education Conference (ACE '23), January 30-February 3, 2023, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3576123.3576137

ACE '23, January 30-February 3, 2023, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-9941-8/23/01.
https://doi.org/10.1145/3576123.3576137

1 INTRODUCTION

Learnersourcing is increasingly used in computing classrooms as a part of the contributing student pedagogy approach, where students are encouraged to contribute to their peers' learning and value the contributions of their peers [17]. In learnersourcing, students participate in the creation of educational resources such as exercises [7, 27], videos [14, 16] and wiki-based course materials [1, 15]. One of the earliest systems that focused on exercise creation is PeerWise [7, 8] in which students can create multiple-choice questions (MCQs) and complete MCQs created by other students. While PeerWise originated in the field of computing, due to the generic nature of the learnersourced materials, it has seen use in other fields as well such as psychology [18] and biology [26]. More recently, systems for creating programming exercises [11, 27] and SQL exercises [24] have been developed.

One of the appeals of using learnersourcing is to reduce the time the instructor needs to spend on the creation of course materials, leaving more time for, for example, face-to-face support to students. However, a concern related to this approach is whether the resources created by students are of similar quality to those created by the instructor, and whether the created resources cover varying course topics. If the created resources do not cover all course topics, the value of learnersourcing is naturally not as high compared to if all course topics are covered. To further alleviate teachers' workload, automatic assessment of the materials created by students should be supported as much as is viable. Systems supporting assessment for SQL exercises are widely in use, utilising both automatic comparison checks [2, 23, 32], as well as peer review [4].

Prior work into the coverage of course topics of learnersourced materials has found that the materials seem to cover a wide varity of course topics [9]. However, prior work into the coverage of course topics in learnersourcing has focused mostly on multiple-choice questions, and it is not clear whether the same holds for SQL exercises. In another study on the use of learnersourcing in the context of SQL exercises has focused on the ratings learnersourced SQL exercises receive [24], but did not analyse the use of the system. The research gap this study seeks to fill is to examine (1) the coverage of course topics of learnersourced SQL exercises and (2) how students use the SQL learnersourcing system, including analysing when during the course do students use the system. Our research questions for this work are as follows:

RQ1. What topics do students create SQL exercises for and what keywords are included in created exercises?
RQ2. What topics and keywords are present in students' SQL exercise submissions?

2 RELATED WORK

2.1 Crowdsourcing, Contributing Student Pedagogy, and Learnersourcing

In computing education, crowdsourcing – that is, the practice of using a group of people to produce an artefact by completing small tasks – has typically been used to create course materials. Efforts such as the ITiCSE 2013 working group Canterbury QuestionBank [34] host a sizable set of instructor-created multiple-choice questions related to various CS1 and CS2 topics. However, in most recent cases, reported crowdsourcing efforts rely on students as the "crowd", as one key purpose of crowdsourcing is to alleviate teachers' workload and free resources that would have to be used for, for example, question creation.

Contributing Student Pedagogy (CSP) [17] "encourages students to contribute into the learning of other students, and to value the contributions of others". As a closely related idea, learnersourcing [22], a form of crowdsourcing where the crowd consists of students, guides students to create learning artefacts for each other, and encourages using the sourced artefacts either on the same or a future course iteration. In computing education, learnersourcing has been used in various ways, such as to have students create multiple-choice questions [7, 21], SQL exercises [24], programming exercises [11, 27], open-ended questions [25, 33], and wiki-type learning materials [1, 15], to mention a few.

2.2 Effects of Learnersourcing

Creating content, as opposed to only reading it, has been shown to lead to improved recall [3]. This phenomenon is referred to as the *generation effect*, which has been demonstrated in various domains, computing included [35]. While the generation effect was originally established and studied with simple memorisation tasks, it has also been shown to generalise to more complex learning situations, such as arithmetic problems [31], and multiple-choice questions [20].

The effects of learnersourcing have been studied in various contexts in computing education. Participating in question creation as a preparation and revision strategy for introductory programming exams led to significantly better performance in exams compared to students who practiced with exercises created by their peers but did not create their own [5]. The effects were most noticeable when students answered exam questions on topics they had created exercises for, highlighting the usefulness of creating content. Students who created programming exercises, in addition to solving them, achieved more than 10 % higher scores than the group of students who only solved practice questions [6]. However, the effects are not always as clear, and studies in both multiple-choice question [36] and programming exercise creation [19] contexts have reported finding no statistically significant difference in exam results.

2.3 Quality and Coverage of Learnersourced Content

Previous studies on the repository coverage of learnersourced content have provided promising results, regardless of how much instructor intervention there is in choosing topics. In their study on PeerWise, Denny et al. [9] concluded that the coverage of a repository consisting of student-created exercises, even without the teacher's explicit guidance on which topics to create content for, can result in a sizeable database covering all the major topics discussed on the course. In another study, Purchase et al. [30] investigated the quality of a PeerWise multiple-choice question repository, finding the general quality of student-created questions to be good, and that the peer review system the tool provides was useful for finding erroneous exercises, thus improving the quality of the repository through student-driven effort. Looking further into the quality of student-created exercises, Denny et al. [10] reported that students' questions are very commonly clearly formatted and correct, and that, similar to [30], students are able to detect and correct erroneous exercises. In learnersourcing programming exercises, a study on CodeWrite [11] reports that students using the tool create and practice a majority of the expected course topics.

3 METHODS

3.1 System

For the present study, we used SQL Trainer [24] which is a learnersourcing system for practicing SQL queries. The system uses PostgreSQL[1] as the database containing exercises, and the H2 database[2] for automatic assessment of the exercises.

Teachers who use the system can define topics and databases, while students who use the system create and complete exercises using the teacher-provided databases within topic-specific areas. Exercise creation is done by selecting a topic and a database, entering a name for the exercise, providing an exercise description, and writing a sample solution (one or more SQL queries) for the created exercise. Created exercises are added to the pool of exercises for the topic, which are then used for practicing SQL queries. An exercise can only be categorised under one topic.

If a student wants to practice writing SQL queries, they only need to pick a topic from the list of teacher-generated topics. The system then randomly selects an exercise that the student has not yet completed from the pool of exercises for the topic, and the exercise will then be shown to the student for practice. Students are allowed to make multiple submissions for the exercise. Once the student successfully completes the exercise, they can ask for a new exercise on the same topic. A new exercise can also be asked if the students choose not to attempt to complete the presently given exercise. This can happen, for example, if the given exercise is poorly tailored (which may happen in learnersourcing).

When an exercise is submitted, correctness is checked by creating two in-memory H2 database instances. For both instances, any queries used to create the database chosen by the creator of the exercise are executed. This is followed by running the sample solution on one of the database instances, and the student's solution on the other instance. After this, the structure and contents of the two databases, including primary keys etc., are compared, which is followed by a comparison of the outputs of the last database query to both instances. Simpler checks are also used to verify that the student's code does not simply output expected data without querying the database. If the structure, contents, and the outputs match, the exercise is deemed correct, while in other cases, the exercise is deemed incorrect. After assessment, the two in-memory databases are dismantled.

[1] https://www.postgresql.org/
[2] https://www.h2database.com

Table 1: Topics used in SQL Trainer listed based on their order of appearance.

#	Topic
1	Selecting data from a table
2	Filtering and ordering data selected from a table
3	Selecting data from multiple tables
4	Selecting data from even more tables
5	Other approaches for joining tables
6	Adding and removing tables
7	Adding data to a database
8	Updating and removing data
9	Aggregating data with functions
10	Aggregating data using group by
11	Advanced data aggregation: having, order, etc.

3.2 Context and Data

SQL Trainer was used in three iterations of a 7-week introductory databases course[3] in the University of Helsinki, Finland. The introductory databases course covers principles of SQL and working with database systems as well as the topics such as data modeling, normalisation techniques, non-relational databases, building applications that use databases, and so on.

SQL was primarily trained in the first two weeks of the course during which basics of SQL were trained to gain a hands-on understanding of working with a database (topics listed in Table 1). SQL Trainer was used as a drill-and-practice tool for learning SQL, as a basic understanding and basic skills to apply it was one of the learning objectives of the course.

The topics in the SQL trainer system were created so that they followed the course materials, and students were guided to use the system during the course. The eleven topics are outlined in Table 1. In the course, approximately 10 % of the grade was based on completing and creating SQL exercises (for full points, students were expected to complete four exercises and to create one exercise per course topic). Completing and creating more exercises than were required for full points was possible, although no additional incentives to do so were provided.

During the course iterations where SQL Trainer was included in the material, 1569 students launched the system. Out of these students, 1419 completed at least one exercise (90.4 %), while a total of 1187 created at least one exercise of their own (75.7 %). In total, 1085 students provided a review for at least one exercise (69.2 %). As we only have the data from SQL Trainer and not the course itself, we do not have demographic data about the students, or data about other activities students engaged with on the courses.

[3]One of the authors of this article was the responsible teacher for the course and also developed the system.

Table 2: Mean, median, standard deviation and maximum for the number of created and submitted exercises over the whole population.

	Created				Submissions			
Topic	Mean	Med	SD	Max	Mean	Med	SD	Max
1	0.83	1	0.61	5	8.77	7	8.81	79
2	0.76	1	0.55	5	10.87	9	11.28	101
3	0.71	1	0.56	4	16.42	12	19.39	203
4	0.67	1	0.55	4	12.03	8	19.84	499
5	0.42	0	0.56	5	10.25	7	13.80	189
6	0.67	1	0.55	4	7.53	6	8.28	78
7	0.67	1	0.53	4	6.02	5	6.94	87
8	0.66	1	0.55	4	7.09	6	8.32	105
9	0.65	1	0.54	4	7.93	6	9.45	111
10	0.60	1	0.53	3	14.98	10	19.85	321
11	0.56	1	0.54	4	16.06	9	24.12	357

3.3 Research Approach

Our approach to answering the research questions are as follows. To answer RQ1, *"What topics do students create SQL exercises for and what keywords are included in created exercises?"*, we analyse the learnersourced SQL exercises. We focus on how many exercises are created for each instructor-specified topic, how many students create exercises per topic, as well as on which SQL keywords are present in the created exercises and whether these relate to the instructor-specified topics.

To answer RQ2, *"What topics and keywords are present in students' SQL exercise submissions?"*, we analyse student submissions for learnersourced SQL exercises. We examine how many exercises students submit for each topic, and which SQL keywords are present in student submissions.

For extracting SQL keywords, we analyse model solutions to the learnersourced SQL exercises as well as the student submitted solutions using substring match for a set of SQL keywords that the instructor of the course expects students to learn through the material. We took into account that some keywords are substrings of other keywords, for example, join is a substring of inner join, and modified our matching criteria accordingly. The full list of SQL keywords used was the following: *select, from, order by, on, join, left join, right join, inner join, create, table, drop, insert, into, update, set, delete, group by, having, distinct, where, min, max, sum, and, or, not, is null, primary key, foreign key, in*. For further inspection, we included keywords that appeared at least 5 % of the time for any topic in either the created exercises or student submissions. This excluded *inner join*, which is a synonym for *join*, as well as *sum, and, or, not, is null, in* from the list.

4 RESULTS

The full list of short topic descriptions can be found in Table 1, used in all the following sections and while discussing the results. All the results for created and submitted exercises for each topic are reported per student.

4.1 Exercise Creation

We present results related to the first research question *"What topics do students create SQL exercises for and what keywords are included in created exercises?"* in this section.

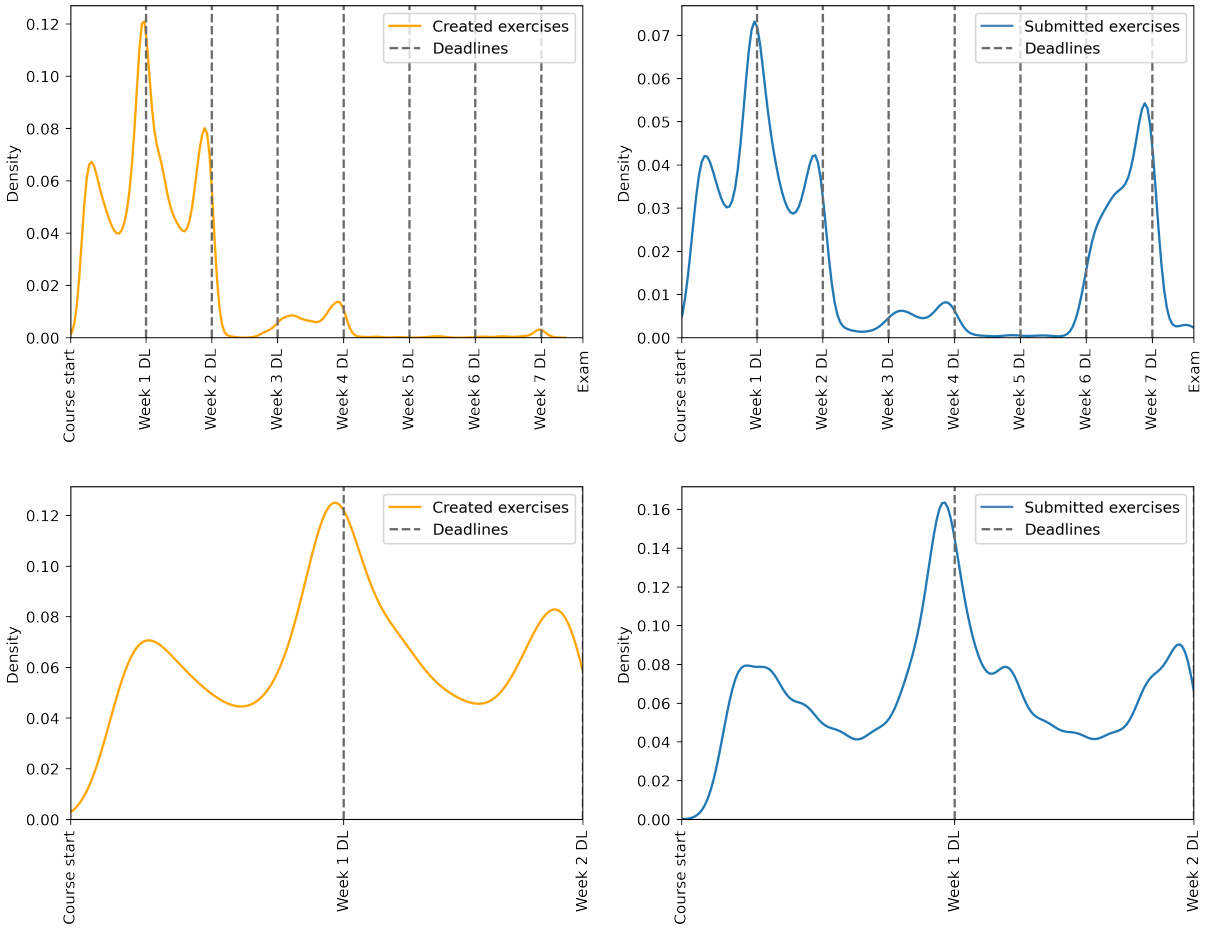

Figure 1: Distribution of created and submitted exercises over the entire duration of the first course where the system was used (top row) and the first two weeks (bottom row) of that course.

Table 3: Mean, median, standard deviation and maximum for the number created and submitted exercises over the active population. A student is considered active if they have at least one submission for the topic.

	Created				Submissions			
Topic	Mean	Med	SD	Max	Mean	Med	SD	Max
1	0.93	1	0.57	5	9.84	8	8.75	79
2	0.92	1	0.47	5	13.06	10	11.14	101
3	0.88	1	0.49	4	20.50	15	19.63	203
4	0.91	1	0.44	4	16.51	11	21.60	499
5	0.59	1	0.58	5	14.40	10	14.42	189
6	0.94	1	0.41	4	10.51	9	8.02	78
7	0.95	1	0.37	4	8.57	7	6.84	87
8	0.95	1	0.39	4	10.24	8	8.23	105
9	0.95	1	0.38	4	11.49	9	9.41	111
10	0.89	1	0.39	3	22.18	17	20.59	321
11	0.89	1	0.42	4	25.45	19	26.13	357

Table 4: The number of exercises created per topic, 11 335 in total. Arranged in the order the topics appear in the course material. The percentage in parentheses from the total created exercises.

Topic	Exercises created
1	1301 (11.6 %)
2	1205 (10.7 %)
3	1113 (9.9 %)
4	1048 (9.3 %)
5	666 (5.9 %)
6	1063 (9.5 %)
7	1047 (9.3 %)
8	1037 (9.2 %)
9	1028 (9.1 %)
10	948 (8.4 %)
11	879 (7.8 %)

	1	2	3	4	5	6	7	8	9	10	11
SELECT	1285	1200	1109	1045	664	3	28	21	1016	947	876
FROM	1285	1197	1108	1044	645	5	28	184	1017	933	871
WHERE	152	926	917	907	468	5	31	917	472	175	80
ORDER BY	30	530	214	204	211	0	0	2	4	106	444
ON	4	13	1057	1021	619	0	3	6	59	803	766
JOIN	4	11	1060	1019	101	0	0	5	55	519	678
LEFT JOIN	0	2	7	2	476	0	0	0	5	280	83
RIGHT JOIN	0	0	2	1	27	0	0	1	0	7	5
CREATE	14	4	1	1	7	702	22	2	1	1	0
TABLE	13	4	2	1	9	1045	24	8	1	1	1
DROP	0	0	1	0	2	358	1	2	0	0	0
INSERT	9	1	1	4	8	9	1031	30	1	1	0
INTO	8	1	1	4	8	15	1032	29	1	1	0
UPDATE	0	1	0	0	5	1	4	843	2	0	0
SET	0	1	0	0	5	11	8	845	3	2	1
DELETE	0	0	0	0	0	3	0	168	0	0	0
GROUP BY	0	0	1	3	3	0	1	1	10	895	814
HAVING	0	0	0	0	0	0	0	0	0	14	518
DISTINCT	0	1	50	161	122	0	0	1	41	19	18
MIN	1	0	1	0	1	5	15	2	107	39	60
MAX	2	0	0	0	0	1	1	1	171	50	115
PRIMARY KEY	1	1	0	0	4	552	11	1	0	0	1
FOREIGN KEY	1	0	1	1	2	138	1	0	1	1	0

Figure 2: The number of SQL keywords used per topic for exercises created by students. Color legend: keyword used in over 80 % of the created exercises for that topic in dark blue, over 40 % light blue, over 10 % yellow, under 10 % gray, and 0 occurrences white.

The total number of exercises created for each topic can be found in Table 4, arranged in the order in which the topics appear in the course material. In total, 11 335 exercises were created across eleven topics, ranging from the maximum of 1301 exercises for topic 1, *selecting data from a table*, to minimum of 666 exercises for topic 5, *other approaches for joining tables*. The exercises are fairly equally distributed across all the topics, with topic 5 as the only notable outlier. The distribution of created exercises can be seen on the left-hand side of Figure 1, over the duration of the first 7-week course where the system was used on the top and for the first two weeks of that course on the bottom. Students were prompted to use the system on weeks 1, 2, 4, and 7 – these can be seen as increase in created exercises near the deadlines of those weeks. Creation of exercises is mostly focused on the first two weeks of the course, peaking at the very beginning of the course and before weekly deadlines.

The numbers of occurrences of the different SQL keywords on each topic for the created exercises are summarised in Figure 2. Overall, students were able to recognise which keywords to use for the task at hand and use them suitably to create functioning SQL query exercises. There are some outliers, such as *distinct*, which, while not strictly related to any given topic, occurs more often in exercises created for topics 3-5.

For the overall user population, accounting for students who used the system at some point but did not necessarily do so for each topic, the mean number of created exercises per student per topic ranges from 0.83 to 0.42 (Table 2), showing a general downward trend with topic 5 as an outlier (mean 0.42). This indicates that generally, students created fewer exercises as the course went on (Table 4 and Figure 1). The standard deviation varies between 0.61 and 0.53, again showing a downward trend as the course progressed.

Table 5: The number of exercise submissions, 185 050 in total. Arranged in the order the topics appear in the course material. Correctness is the percentage of submissions for the topic that were correct.

Topic	Exercise submissions	Correctness
1	13 753 (7.4 %)	59%
2	17 061 (9.2 %)	42%
3	25 763 (13.9 %)	23%
4	18 869 (10.2 %)	25%
5	16 090 (8.7 %)	28%
6	11 812 (6.4 %)	47%
7	9442 (5.1 %)	51%
8	11 119 (6.0 %)	43%
9	12 440 (6.7 %)	40%
10	23 507 (12.7 %)	19%
11	25 194 (13.6 %)	15%

The median for created exercises per students was 1 on all topics except topic 5, *other approaches for joining tables*, where the median was 0. The maximum of created exercises ranged from 3 to 5 per topic.

Similar trends occur when inspecting the active student populations, as seen in Table 3, though not as clearly as with the whole student population. A student is considered active for a topic if they have made at least one submission to at least one exercise related to that topic or if they created at least one exercise for the topic. Across the topics, the mean ranges from 0.95 to 0.59, again with topic 5 as the clearest outlier with a mean of 0.59. For the active students, the standard deviation of created exercises varies between 0.57 and 0.37, with slightly lower variations in the latter topics. At most, students create five exercises for one topic (maximum of topics 1, 2, and 5). The median for created exercises is 1 per student for all topics, and the maximum ranges again from 3 to 5.

4.2 Exercise Submissions

This section outlines results for RQ2, *"What topics and keywords are present in students' SQL exercise submissions?"*

The number of exercises submitted for each topic is outlined in Table 5, arranged in the order the topics appear in the course material. A total of 185 050 exercise submissions were made across the eleven topics, ranging from the maximum of 25 763 exercises for topic 3, *selecting data from multiple tables*, to minimum of 9442 exercises for topic 7, *adding data to a database*. Again, the exercise submissions are fairly equally distributed across all the topics. Table 5 also includes the average correctness for each topic, that is, the percentage of correct submissions out of all submissions. Typically, a student makes more than one submission attempt before answering the exercise correctly, as the only way to get feedback for the answer is through submitting the exercise. This means that several incorrect submissions will lower the average correctness.

The distribution of submitted exercises can be seen on the right-hand side of Figure 1, over the duration of a 7-week course on the top and for the first two weeks on the bottom. Similarly to the creation of exercises, submitting exercises is focused on the first two weeks of the course, peaking at the very beginning and

	1	2	3	4	5	6	7	8	9	10	11
SELECT	13486	17012	25681	18821	15928	66	360	354	12244	23366	25082
FROM	13377	16912	25459	18698	15808	224	354	3694	12156	22950	24787
WHERE	1463	13415	17902	12401	9529	31	270	7826	6098	3067	5020
ORDER BY	240	5125	3800	3437	3161	6	3	28	48	1629	8825
ON	520	440	21988	17393	14840	22	136	132	963	19284	22324
JOIN	447	434	22396	17333	5617	8	63	116	921	13169	19912
LEFT JOIN	92	15	280	190	8322	3	2	22	42	6009	2229
RIGHT JOIN	5	0	22	37	881	6	3	3	0	307	165
CREATE	104	1	1	2	19	7668	110	38	127	6	13
TABLE	163	10	27	3	99	11277	161	364	212	19	52
DROP	1	0	9	0	77	3773	17	199	1	1	46
INSERT	96	2	1	10	61	30	9116	224	127	2	6
INTO	96	2	17	12	61	120	9171	224	127	2	10
UPDATE	0	2	0	0	27	5	31	7194	46	2	16
SET	13	4	0	0	27	252	87	7172	66	7	19
DELETE	5	9	1	6	9	228	7	3448	4	1	11
GROUP BY	229	145	353	180	327	14	18	27	399	17466	19395
HAVING	105	23	46	43	113	3	2	16	23	416	10633
DISTINCT	69	22	777	3259	2557	0	1	0	500	889	1164
MIN	38	48	27	4	3	30	187	9	2158	412	3044
MAX	38	1	5	47	6	12	18	1	1462	767	1897
PRIMARY KEY	22	0	0	0	11	5753	67	17	1	1	2
FOREIGN KEY	9	0	0	0	0	1355	2	5	2	0	1

Figure 3: The number of SQL keywords used per topic for exercises submitted by students. Color legend: keyword used in over 80 % of the exercise submissions for that topic in dark blue, over 40 % light blue, over 10 % yellow, under 10 % gray, and 0 occurrences white.

before weekly deadlines, as well as smaller increases when students were reminded of the system in the material (weeks 1, 2, 4, and 7). However, exercise submissions also have a notable spike at the end of the course, just before the last deadline on week 7.

The numbers of occurrences for SQL keywords on each topic for submitted exercises are summarised in Figure 3. Similar to the created exercises, students were able to recognise which keywords to use to complete the exercise. However, students used a much wider range of keywords in their submissions, as almost all of the inspected keywords appeared in every single topic at least once (only 1 to 3 keywords missing in 7 topics total).

From the overall user population, the mean for the number of submissions ranges from 6.02 to 16.42 (Table 2) for submitted exercises with no obvious upward or downward trends. The standard deviation varies between 6.94 and 24.12, again with no clear upward or downward trends. The median for submissions per student ranges from 6 to 12.

When inspecting the active student populations for each topic (Table 3), the mean for the number of submissions ranges from 8.57 to 25.45, and the standard deviation between 6.84 and 26.13, with slightly lower variations in the latter topics. The median for submissions per student ranges from 7 to 19.

Curiously, the maximum number of attempts for a student for exercise submissions was 499 on topic 4, *selecting data from even more tables*. These attempts were made across six different exercises on the topic, with one exercise amassing 267 submissions from one student. Thus, the maximum number of submissions per student ranges from 78 to 499.

5 DISCUSSION

In this section, we discuss how our results can be used to reflect on the learnersourcing process, course topic coverage and course structures. Generally, we can see from our results that students use the keywords we would expect them to based on the topics and the course material, and that these keywords match between exercise creation and submission.

5.1 Course Structure and Topic Coverage

The topics in the data we used were created by the teacher and followed the structure in the course. As there are a number of students who drop out during the course, or skip some parts of the course, there are students who may not practice some topics at all. Dynamic online learning environment could allow changes in the structure and order of course topics in the material, and seek to have students practice all topics regardless of students omitting some parts. One possibility would be utilising spaced repetition, that is, reviewing newer and more difficult topics more often, and older and more practiced topics less frequently. Similar topics could be introduced at the same time and taught fully over a longer period of time, as opposed to focusing on one topic at a time.

While in general the created exercises used the keywords that were discussed in the course and highlighted by the topic, we also observed that some students may not have used the material in order. This was visible e.g. in the use of the SQL keyword *distinct*, for which we saw some occurrences already in the third topic, which students are directed to before they learn about *distinct* in the course. Thus, it seems that some students are prone to using new keywords, even if they are not required or necessarily expected for the topic; this also highlights the possibility of students having some prior experience on the topic.

Such behaviour can lead to learnersourced exercises that are more advanced than the topic requires and may, in some cases, be too complex for the topic's expected level of difficulty. If the difficulty of the topic is evaluated, for example, by peer review, more difficult exercises could be offered as more advanced practice questions for students with prior knowledge. An interesting point for future work could be studying whether the creation of too difficult exercises could be automatically assessed with keyword parsing or with other methods. Students could be then either noted through the system that the keywords they are using are not necessarily relevant or expected for the topic at hand, or the system could automatically classify and move the exercises to another topic.

We also observed that combining concepts under a single topic can lead to one keyword being used more than others in learnersourced exercises. For example, in topic 6, *adding and removing tables*, the keyword *create* was used in 702 exercises, while the keyword *drop* was used in 358 exercises. This suggests that the teacher can, unintentionally, affect the contents and coverage of the exercise repository by having multiple concepts under one topic.

There were also some keywords that we expected to be more prominent, such as *right join, inner join*, and *delete*. If some keywords are found lacking in a repository, they could be enforced by introducing them as topics, or using them as examples in the instructions. All the keywords shown in Figures 2 and 3 appeared at least 5 % of the time for at least one topic in either the created exercises or student submissions. Many keywords were used some tens of times, but not enough to appear in our analysis.

5.2 Students' Use of the System

Overall, students used SQL Trainer for practice quite nicely, as outlined in Figure 1 and Tables 2, 3, and 5. For the most part, the behaviour was in line with the grading criteria, where students were expected to create at least one exercise per topic and to complete at least four exercises. We did notice some students being more active though, where some created multiple exercises, and some practiced with the system considerably more than expected.

Students were prompted to use the system on weeks 1, 2, 4, and 7, and these can be seen as notable increases in both created exercises as well as submissions in Figure 1. Additionally, it seems that students have used the system as a revision method before the exam, as the submissions also increase notably at the end of the course, just before the course exam. This is in line with student behaviour patterns observed by Denny [5] with the learnersourcing tool PeerWise. Denny et al. [12] have also found that in their case, practice using student-created questions is strongly predictive of subsequent test performance. This is something we would be interested in investigating in the future in the context of SQL exercises.

Correctness, reported per topic in Table 5, varies quite a bit between topics. As correctness is reported as the percentage of correct submissions out of all submissions, and the percentage decreases the more submissions are made before correct submission, we can discern between topics that seem easier or more difficult to students. Topics do not seem to become linearly more difficult as the course progresses – while the highest correctness percentage if for topic 1 (59 %) and the lowest for topic 11 (15 %), there are some notable outliers, such as topic 7, *adding data to a database* with higher correctness at 51 %. We propose some future work on the details of this in Section 6.1.

5.3 Keywords Used by Students

We also took a cursory look at the data related to the number of distinct keywords present in the created exercises and student submissions using an SQL parser instead of substring matching of predefined keywords. The advantage to this is to not miss rarely occurring keywords that are potentially missing from a predefined keyword list. We found that the created exercises had fewer used keywords per topic – for example, around 15 keywords were used in the created exercises for topic 1, while in the submissions for the same topic, approximately 65 unique keywords appeared in the data. Most of the occurrences are very rare, and no new keywords that would have passed the 5% appearance threshold we used in the substring matching were found. This does, however, show that students are aware of keywords that are not strictly related to the topic at hand – or keywords that are not even mentioned in the course –, and are able to use these to solve exercises, even if it is not necessarily required. This may also indicate that the title and the wording of the topic may limit students' creation process to include only keywords that are strictly relevant. This is not necessarily unfavourable, as ideally, the created exercises should be as clear as possible and such that students without prior knowledge are able to complete them. In contrast, having keywords not taught in the course (or keywords from latter course parts) in students' submissions does not involve similar issues.

Inspecting some of the topics more closely, we can see that the submissions for even the simplest selection exercises (topic 1, *selecting data from a table*) have some occurrences of late-course keywords that are clearly unnecessary for the topic, such as *update*. One possibility here is that students who are already familiar with the subject are trying out more advanced topics for fun, or some students could be trying to hack or break the system. The system in use did have protective measures in place to catch students trying to trick (or hack) the system; these caused the student to be rickrolled – that is, involuntarily redirected to YouTube page for Rick Astley's "Never Gonna Give You Up". It is also possible that, upon finding this, some students may have been encouraged to explore the system further than necessary, discussed next.

During data analysis, we also identified a student who had excessive submissions to topic 4, *selecting data from even more tables*, totaling 499 submissions. These submissions were made on six different exercises related to this topic, with the maximum of 267 submissions for one exercise. While this could indicate tinkering, i.e. doing repeated minor modifications to the query without a clear plan in the hope that these modifications will eventually make the query correct, it is also possible that the student was trying to push the boundaries of the system; the course discussed some basic security concepts including SQL injections.

5.4 Topics and Use of Learnersourcing in Context

We acknowledge that all learnersourcing systems are used in context, which influences how and when students use the system. Further, learnersourcing systems have design decisions, which influence their use. As an example, the system that we used had teacher pregenerated topics that students would then utilise. We do not see that this is the only possible approach and envision three variations when it comes to guiding the learnersourcing process with topics, all with benefits and possible issues.

The first variation would be full freedom of topic selection, which has been typical with PeerWise [7]. This can lead into unbalanced repositories if students tend to lean towards creating exercises for one topic more than others, typically at the beginning of the course when topics are simpler. However, Denny et al. [9] have reported that even with full freedom of choice, student-created exercises formed a repository that covered all the major topics of the course.

The second variation would be to have major topics outlined (e.g. by the instructor), but provide students freedom to create the content of the exercises. This is the case presented in our study. Based on our results, we report that the chosen topics receive almost equal numbers of created exercises, even if there is a slight decrease for topics that are introduced later in the course. This may be partially because of student dropouts – there are simply fewer students on the course at the time of topic 11. However, the instructor is able to better enforce the practice of certain keywords or topics through the use of the system, and possibly affect the coverage of the repository if it seems to be lacking in some areas. At the same time, this may lead to unwanted content in topics, although based on our data the risk is not substantial.

The third variation would be to restrict keywords that students can use when creating exercises. We are currently not aware of a

learnersourcing system that would not only outline the topic, but also prevent students from using keywords that have been deemed irrelevant for the topic. While a system like this would most likely help create exercises that are easily classifiable and relevant to the course content, it could be too restricting for the learnersourcing process, hindering students' creativity when they are creating an exercise, or their engagement with the system.

More scaffolding might be beneficial for systems that create more complex exercises, such as programming or SQL exercises. Multiple-choice questions can be fairly easy to create, even with full freedom of choice when it comes to the topic and the contents of the exercise, and this freedom may allow for more creativity for the students. For more complex creation tasks, some level of scaffolding may be necessary, as just the act of creating an exercise is most likely new (and somewhat daunting) to the students. Determining the correct level of support students need for the creation task is important, as the lack of proper instruction may lead to lower participation rates and lower quality of created exercises.

5.5 Limitations

We acknowledge that our study comes with some limitations, which we outline here.

The system was used in our specific context of an introductory SQL course at a single university. Thus, it is possible that some of the results would not generalise to other contexts. For example, it is possible that there would be differences between creating SQL exercises and other types of artefacts.

Our study did not inspect the difficulty of the student-created exercises. Varying levels of difficulty is important for a comprehensive repository, and it is possible that students are mostly creating exercises that are on the simpler side, even for the more advanced topics [28]. However, a previous study in the programming context has found that students are able to create exercises with varying levels of difficulty, even in an introductory programming course [29].

Creating and submitting answers to exercises awarded course points that have an effect towards the final grade, which may encourage students to use the system more. It is possible that participation rates would be lower if no points were awarded.

As we used an author-created list of SQL keywords for the substring matching, we acknowledge that our list of keywords is not complete. However, based on the additional – although cursory – check with an SQL parser, we did not find any additional commonly used keywords that would have been missing from the list of keywords created by the authors. Depending on the exact contents of the course, this list of keyword could look very different in another context.

The topic granularity was chosen by the teacher when the system was first taken into use on the courses. However, the system supports all levels of granularity, thus, making it possible to utilise both coarser and finer topics if necessary.

6 CONCLUSION

In this paper, we studied the topic coverage of student-created SQL exercises. To summarise, the answers to our research questions are as follows:

RQ1. What topics do students create SQL exercises for and what keywords are included in created exercises? **Answer:** Students create exercises for all the major course topics. Slightly more exercises are created for topics that occur near the beginning of the course.

RQ2. What topics and keywords are present in students' SQL exercise submissions? **Answer:** Similar to the created exercises, students submit answers to exercises of all of the major course topics. Submissions do not have a clear downward trend – instead, students seem to focus on a few specific topics at the beginning and at the end of the course.

Based on our results, students are able to create a database of SQL exercises with great course topic coverage with only a little guidance from the lecturer. While there are still many angles to consider, outlined in Section 6.1, learnersourcing SQL exercises seems to show great potential of functioning as a study support mechanism that can also alleviate teachers' workload.

6.1 Future work

In the future, we are interested in studying the difficulty of student-created SQL exercises – whether students create exercises with varying difficulty levels, and if it is possible to classify the exercises manually or automatically by their difficulty. Based on our results on the correctness of the exercise submissions, we would also like to investigate what makes some topics more difficult for students than others.

Another important piece of future work would be evaluating the learning effects of the system: whether creating and answering to learnersourced SQL exercises affects students' performance on the course, as it has done in other learnersourcing contexts [5, 12, 13]. We are interested in examining the effect of giving course credit on student behaviour and tool usage. One example could be testing how students create and submit answers to exercises when credit is given based on the number of created or submitted exercises overall, not per topic.

Lastly, we would like to extend our current study to include analysis on the quality of student questions. Additionally, we are interested in students' opinions of the system, and would like to collect feedback to further improve system where necessary.

ACKNOWLEDGMENTS

We are grateful for the doctoral research grant awarded by Jenny and Antti Wihuri Foundation to the first author.

REFERENCES

[1] Chris Bennett. 2009. Student-authored wiki textbook in CS1. *Journal of Computing Sciences in Colleges* 24, 6 (2009), 50–56.

[2] Amol Bhangdiya, Bikash Chandra, Biplab Kar, Bharath Radhakrishnan, K. V. Maheshwara Reddy, Shetal Shah, and S. Sudarshan. 2015. The XDa-TA system for automated grading of SQL query assignments. In *2015 IEEE 31st International Conference on Data Engineering*. 1468–1471. https://doi.org/10.1109/ICDE.2015.7113403

[3] Robert J Crutcher and Alice F Healy. 1989. Cognitive operations and the generation effect. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15, 4 (1989), 669.

[4] Michael de Raadt, Stijn Dekeyser, and Tien Yu Lee. 2006. Do Students SQLify? Improving Learning Outcomes with Peer Review and Enhanced Computer Assisted Assessment of Querying Skills *(Baltic Sea '06)*. Association for Computing Machinery, New York, NY, USA, 101–108. https://doi.org/10.1145/1315803.1315821

[5] Paul Denny. 2015. Generating Practice Questions as a Preparation Strategy for Introductory Programming Exams. In *Proceedings of the 46th ACM Technical*

Symposium on Computer Science Education (Kansas City, Missouri, USA) *(SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 278–283.

[6] Paul Denny, Diana Cukierman, and Jonathan Bhaskar. 2015. Measuring the Effect of Inventing Practice Exercises on Learning in an Introductory Programming Course. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '15)*. Association for Computing Machinery, New York, NY, USA, 13–22.

[7] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise: students sharing their multiple choice questions. In *Proceedings of the fourth international workshop on computing education research*. 51–58.

[8] 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*. Citeseer, 69–74.

[9] Paul Denny, Andrew Luxton-Reilly, John Hamer, and Helen Purchase. 2009. Coverage of Course Topics in a Student Generated MCQ Repository *(ITiCSE '09)*. Association for Computing Machinery, New York, NY, USA, 11–15. https://doi.org/10.1145/1562877.1562888

[10] 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* (Wellington, New Zealand) *(ACE '09)*. Australian Computer Society, Inc., AUS, 55–63.

[11] 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* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 471–476. https://doi.org/10.1145/1953163.1953299

[12] Paul Denny, Fiona McDonald, Ruth Empson, Philip Kelly, and Andrew Petersen. 2018. Empirical Support for a Causal Relationship Between Gamification and Learning Outcomes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13.

[13] Paul Denny, Ewan Tempero, Dawn Garbett, and Andrew Petersen. 2017. Examining a Student-Generated Question Activity Using Random Topic Assignment. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 146–151. https://doi.org/10.1145/3059009.3059033

[14] Pablo Frank-Bolton and Rahul Simha. 2018. Docendo discimus: Students learn by teaching peers through video. In *Proceedings of the 49th ACM technical symposium on computer science education*. 473–478.

[15] Edward F Gehringer, Lillian Cassel, Katherine Deibel, and William Joel. 2008. Wikis: collaborative learning for cs education. *ACM SIGCSE Bulletin* 40, 1 (2008), 379–380.

[16] Ana I González-Tablas and Pablo Martín-González. 2019. Student-Generated Videos for Promoting Better Attitudes Towards Cryptography. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 372–378.

[17] John Hamer, Quintin Cutts, Jana Jackova, Andrew Luxton-Reilly, Robert McCartney, Helen Purchase, Charles Riedesel, Mara Saeli, Kate Sanders, and Judithe Sheard. 2008. Contributing student pedagogy. *ACM SIGCSE Bulletin* 40, 4 (2008), 194–212.

[18] Piers DL Howe, Meredith McKague, Jason M Lodge, Anthea G Blunden, and Geoffrey Saw. 2018. PeerWise: Evaluating the effectiveness of a web-based learning aid in a second-year psychology subject. *Psychology Learning & Teaching* 17, 2 (2018), 166–176.

[19] Vilma Kangas, Nea Pirttinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Does Creating Programming Assignments with Tests Lead to Improved Performance in Writing Unit Tests?. In *Proceedings of the ACM Conference on Global Computing Education* (Chengdu,Sichuan, China) *(CompEd '19)*. Association for Computing Machinery, New York, NY, USA, 106–112.

[20] Matthew R. Kelley, Elizabeth K. Chapman-Orr, Susanna Calkins, and Robert J. Lemke. 2019. Generation and Retrieval Practice Effects in the Classroom Using PeerWise. *Teaching of Psychology* 46, 2 (2019), 121–126.

[21] Hassan Khosravi, Kirsty Kitto, and Joseph Jay Williams. 2019. RiPPLE: A Crowdsourced Adaptive Platform for Recommendation of Learning Activities. *Journal of Learning Analytics* 6, 3 (2019), 91–105. https://doi.org/10.48550/arXiv.1910.05522

[22] Juho Kim. 2015. *Learnersourcing: improving learning with collective learner activity*. Ph. D. Dissertation. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/101464

[23] Carsten Kleiner, Christopher Tebbe, and Felix Heine. 2013. Automated Grading and Tutoring of SQL Statements to Improve Student Learning. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '13)*. Association for Computing Machinery, New York, NY, USA, 161–168. https://doi.org/10.1145/2526968.2526986

[24] 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*. 349–355.

[25] Andrew Luxton-Reilly, Beryl Plimmer, and Robert Sheehan. 2010. StudySieve: A Tool That Supports Constructive Evaluation for Free-Response Questions. In *Proceedings of the 11th International Conference of the NZ Chapter of the ACM Special Interest Group on Human-Computer Interaction* (Auckland, New Zealand) *(CHINZ '10)*. Association for Computing Machinery, New York, NY, USA, 65–68. https://doi.org/10.1145/1832838.1832849

[26] Heather A McQueen, Cathy Shields, DJ Finnegan, J Higham, and MW Simmen. 2014. PeerWise provides significant academic benefits to biological science students across diverse learning tasks, but with minimal instructor intervention. *Biochemistry and Molecular Biology Education* 42, 5 (2014), 371–381.

[27] 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*. 326–331.

[28] Nea Pirttinen and Juho Leinonen. 2021. Exploring the Complexity of Crowdsourced Programming Assignments. In *Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice"*. https://cssplice.github.io/SIGCSE21Workshop.html SPLICE@SIGCSE'21 Workshop CS Education Infrastructure for All III: From Ideas to Practice, SPLICE'21 ; Conference date: 15-03-2021 Through 16-03-2021.

[29] Nea Pirttinen and Juho Leinonen. 2022. Can Students Review Their Peers? Comparison of Peer and Instructor Reviews. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 12–18. https://doi.org/10.1145/3502718.3524762

[30] Helen Purchase, John Hamer, Paul Denny, and Andrew Luxton-Reilly. 2010. The Quality of a PeerWise MCQ Repository. In *Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103* (Brisbane, Australia) *(ACE '10)*. Australian Computer Society, Inc., AUS, 137–146.

[31] Bethany Rittle-Johnson and Alexander Kmicikewycz. 2008. When generating answers benefits arithmetic skill: the importance of prior knowledge. *Journal of Experimental Child Psychology* 101, 1 (2008), 75–81.

[32] Uwe Röhm, Lexi Brent, Tim Dawborn, and Bryn Jeffries. 2020. SQL for Data Scientists: Designing SQL Tutorials for Scalable Online Teaching. *Proc. VLDB Endow.* 13, 12 (sep 2020), 2989–2992. https://doi.org/10.14778/3415478.3415526

[33] Sam Saarinen, Shriram Krishnamurthi, Kathi Fisler, and Preston Tunnell Wilson. 2019. Harnessing the Wisdom of the Classes: Classsourcing and Machine Learning for Assessment Instrument Generation. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 606–612. https://doi.org/10.1145/3287324.3287504

[34] 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* (Canterbury, England, United Kingdom) *(ITiCSE -WGR '13)*. Association for Computing Machinery, New York, NY, USA, 33–52. https://doi.org/10.1145/2543882.2543885

[35] Dominique L Scapin. 1982. Generation effect, structuring and computer commands. *Behaviour & Information Technology* 1, 4 (1982), 401–410.

[36] Anjali Singh, Christopher Brooks, Yiwen Lin, and Warren Li. 2021. What's In It for the Learners? Evidence from a Randomized Field Experiment on Learnersourcing Questions in a MOOC. In *Proceedings of the Eighth ACM Conference on Learning @ Scale* (Virtual Event, Germany) *(L@S '21)*. Association for Computing Machinery, New York, NY, USA, 221–233.