A Quantitative Study of the Relative Difficulty for Novices of Writing Seven Different Types of SQL Queries

Alireza Ahadi, Julia Prior, Vahid Behbood and Raymond Lister

University of Technology, Sydney, Australia

{Alireza.Ahadi, Julia.Prior, Vahid.Behbood, Raymond.Lister}@uts.edu.au

ABSTRACT

This paper presents a quantitative analysis of data collected by an online testing system for SQL "select" queries. The data was collected from almost one thousand students, over eight years. We examine which types of queries our students found harder to write. The seven types of SQL queries studied are: simple queries on one table; grouping, both with and without "having"; natural joins; simple and correlated sub-queries; and self-joins. The order of queries in the preceding sentence reflects the order of student difficulty we see in our data.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages – query languages.

General Terms

Management, Measurement, Human Factors.

Keywords

Online assessment; databases; SQL queries.

1. INTRODUCTION

The chief executive officer of edX, Anant Agarwal, has been quoted as saying that Massive Open Online Courses (MOOCs) will be the "*particle accelerator for learning*" (Stokes, 2013). A similar sentiment was expressed 30 years earlier. In her study of the usability of database query languages, Reisner (1981) asked rhetorically, "*Why experiment at all*?" before answering:

> *... should not just using one's own judgment, or the judgment of some expert, suffice? ... Unfortunately, the computer scientist is not necessarily a good judge of the abilities of* [students]*; there are no experts whose opinions would be generally accepted, and if there were, they might not agree. ... A more cogent reason to forego judgment by peers or by an expert is that such judgment is not quantitative.*

With the same spirit as Agarwal and Reisner, in this paper we use data collected over eight years, from 986 students, to study quantitatively the relative difficulty our students had with completing seven different types of SQL queries.

ITiCSE'15, July 4–8, 2015, Vilnius, Lithuania.
Copyright © 2015 ACM 978-1-4503-3440-2/15/07…$15.00.
http://dx.doi.org/10.1145/2729094.2742620

2. BACKGROUND

Many reports have been published about online SQL tutoring/assessment tools. However, most of these reports focus on the functionality of the tool itself, or on how the system supports a certain pedagogical model (e.g. Brusilovsky *et al*., 2008 & 2010; Mitrovic, 1998 & 2003; Prior *et al*., 2004 & 2014). Those reports do not analyze the data collected by those systems to determine the difficulties students have with SQL queries.

From the literature, we identified only Reisner (1981) as a published systematic study of student difficulties with SQL queries. She found that the subjects in her study had difficulty recognizing when they should use "group by", but her subjects could successfully use "group by" when explicitly told to do so. Reisner's focus was on comparing SQL with three other query tools that were then seen as competitors to SQL. Thus, the "group by" case was her only discussion of an aspect of SQL that was troublesome for novices. Over 30 years later, we believe a new study is warranted that focuses on SQL and looks at multiple aspects of SQL.

We also found some papers in which the authors mentioned their intuitions about student difficulties, based upon their teaching experiences. These intuitions were mentioned briefly in the introductory/motivation sections of the authors' papers, before those authors went on to describe the architecture of their tutoring/assessment system. Kearns *et al*. (1997) and also Mitrovic (1998) mentioned "group by" as a problem, especially aggregate functions and the use of "having". They also mentioned as a problem the complete specification of all necessary "where" conditions when joining multiple tables. Sadiq *et al*. (2004) nominated the declarative nature of SQL as a problem for students as it "*requires learners to think sets rather than steps*" (p. 224).

3. ASSESQL

We collected our data in a purpose-built online assessment system, AsseSQL. It is therefore necessary to describe some aspects of how AsseSQL works, so that readers can assess for themselves the validity of our data. In this paper, we provide the briefest possible description of the system. Further details of AsseSQL, and how we used it to test our students, can be found in prior publications (Prior *et al*., 2004 & 2014). The students in this study were all undergraduates at our university, studying Bachelor degrees in Information Technology or Software Engineering.

In the online test, students were allowed 50 minutes to attempt seven SQL questions. On their first test page in AsseSQL, students see all seven questions, and they may attempt the questions in any order. All seven questions refer to a database that is familiar to the students prior to their test.

Each question is expressed in English, and might begin, "*Write an SQL query that* …" A student's answer is in the form of a complete SQL "select" statement.

When a student submits a "select" statement for any of the seven questions, the student is told immediately whether their answer is right or wrong. If the student's answer is wrong, the system presents both the desired table and the actual table produced by the student's query. The student is then free to provide another "select" statement, and may repeatedly do so until they either answer the question correctly, run out of time, or choose to move on to a different question. If a student moves to a new question, the student may return to this question later.

The grading of the answers is binary – the student's attempt is either correct or incorrect. As there may be more than one correct SQL statement for a specific question, a student's SQL statement is considered to be correct when the table produced by the student's "select" statement matches the desired table. (Some simple 'sanity checks' are made to ensure that a student doesn't fudge an answer with a brute force selection of all required rows.)

Prior to taking a test, the students are familiarized with both AsseSQL and the database scenario that will be used in the test. About a week before the test, students receive the Entity Relationship Diagram (ERD) and the "create" statements for the scenario database. Note, however, that students are not provided with the data that will fill those tables, nor are they provided with sample questions for that database scenario. Several weeks before the actual test, students are provided with access to a 'practice test' in AsseSQL, which has a different scenario database from the scenario database used in the actual test.

4. A SCENARIO DATABASE: BICYCLE

One of the scenario databases that we use in the online SQL test is called "Bicycle", based on a database from Post (2001). This is the database from which most of the results described in this paper are generated. The Bicycle database is composed of four tables, as shown in the Entity Relationship Diagram (ERD) of Figure 1. A bicycle is made up of many components; each component can be installed in many bicycles. Each installed component on a bicycle is called a `bikepart'. Just one manufacturer supplies each component, but one manufacturer may supply many components to the store, so there is a 1:M relationship between Manufacturer and Component. There is a self-referencing relationship in Component, as one component may be made up of many other components.

5. PRELIMINARIES TO RESULTS

5.1 A Wrong Answer versus No Attempt

Suppose for a given question, one third of students answered the question correctly, one third attempted the question but never answered it correctly, and one third never attempted the question – what should we consider the success rate to be? There are two options:

- 33%, if non-attempts are treated as incorrect answers.
- 50%, if non-attempts are ignored.

In the results presented below, we have supplied both options when reporting success rates for one of the seven types of questions. In cases where we are comparing two different types of questions, the issue arises as to whether a student did not attempt one of the two questions because the student knew they could not do it, or because they ran out of time. For that reason, when comparing performance between question types, we have applied the option where non-attempts are ignored.

5.2 Differences in a Pool of Variations

In the online test, students are presented with seven questions. Each question requires a different type of SQL "select" statement (e.g. a "group by", a natural join, etc). Each of the seven questions presented to a specific student is chosen at random from a small pool. In this paper, to avoid confusion between the actual questions presented to a student, and the questions in a pool, we henceforth refer to the questions in a pool as "variations".

In this section we discuss differences in success rates among variations within a single pool. To make the discussion more clear, we will focus on a concrete example – the four "group by" variations of the Bicycle Database. We chose this example as these four "group by" variations exhibited the greatest differences in success rate of any pool. The performance of each of these "group by" variations is shown in Figure 2.

In Figure 2, the line beginning "Baseline" provides the aggregate statistics when the data from all four variations are combined. That baseline row shows that a total of 986 students were assigned one of these four variations. Of those 986 students, 742 students successfully provided a correct answer, which is 75% of the total number of students. While 986 students is a large sample, if we collected data from another set of students, of similar ability, the success rate for that new sample is likely to be at least a little different. Equally, Figure 2 shows that the success rate for Variation 1 is 87% and 70% for Variation 2, but if we collected data from more students, how likely is it that the success rate for Variation 1 would drop below the success rate for Variation 2? To address those sorts of issues, we assumed a normal distribution of student abilities and estimated a 95% confidence interval for the success rate of the baseline and each of the four variations, using a well known statistical technique – the offset to the upper and lower bounds of the confidence interval is 1.65 × Standard Eror (SE), where the SE is estimated as:

$$SE = \sqrt{\frac{Success\ Rate(1 - Success\ Rate)}{Total}}$$

For instance, the lower and upper bounds of the Success Rate for the baseline shown in Figure 2 are 72% and 78% respectively (i.e. the offset was estimated as being 3%). That confidence interval and the confidence intervals for all four variations are also represented graphically in Figure 2. The colors in the confidence intervals of the variations show the proportion of those confidence intervals that are outside the baseline confidence interval.

In Figure 2, the column headed "p-value" indicates the probability that the difference in the success rate of a variation from the baseline is due to the existing sample being atypical. Those p-values indicate that the success rates of variations 1 and 3 are likely to remain higher than the baseline, and variation 4 remain lower, if another sample of data was collected from an equivalent group of students. The column headed "improvement" shows the difference in success rate between each variation and the baseline. The 95% confidence interval shown in that "improvement" column was calculated the same way as above.

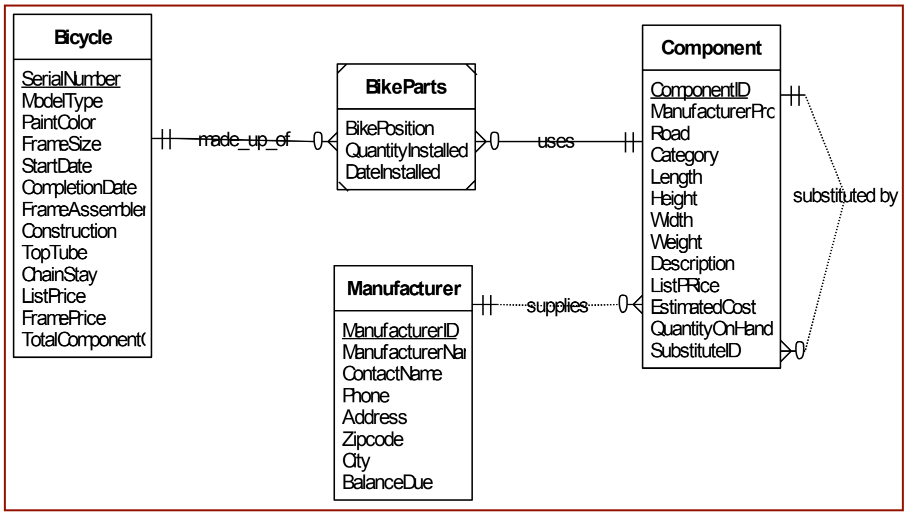
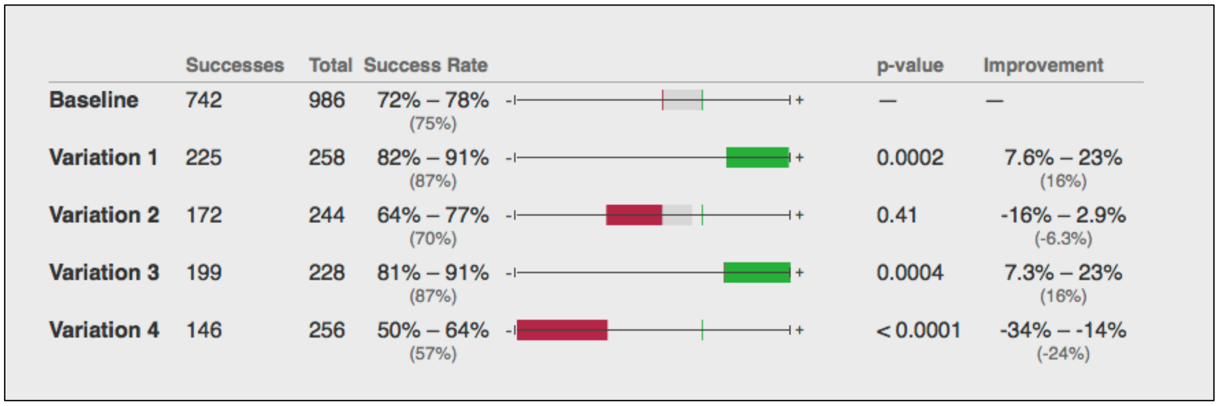


Figure 1. Bicycle Database ERD.



	Successes	Total	Success Rate	p-value	Improvement
Baseline	742	986	72% – 78% (75%)	—	—
Variation 1	225	258	82% – 91% (87%)	0.0002	7.6% – 23% (16%)
Variation 2	172	244	64% – 77% (70%)	0.41	-16% – 2.9% (-6.3%)
Variation 3	199	228	81% – 91% (87%)	0.0004	7.3% – 23% (16%)
Variation 4	146	256	50% – 64% (57%)	< 0.0001	-34% – -14% (-24%)

Figure 2. Variation in Success Rate for the "group by" pool.

The analysis illustrated in Figure 2 indicates that, even among questions considered to be variations that test the same fundamental SQL concept (i.e. "group by"), the success rates of the variations can be quite different. (Recall, however, that the "group by" example illustrated here manifested the greatest differences of any set of pool variations.) The reason for those differences is an issue we will return to in the "Results" section.

6. RESULTS

Table 1 summarizes the performance of students on the Bicycle database. In AsseSQL, the order in which the seven questions are presented on the computer screen to the students is fixed. (Recall, however, that students may attempt the questions in any order.) That same ordering is used in Table 1. Our choice of this ordering reflects our *a priori* intuition (i.e. when AsseSQL was built), of the relative difficulty of the seven query types.

As a general rule of thumb, based upon the estimation of the 95% confidence interval described in the previous section, the success rate percentages in Table 1 that differ by more than 5% can be considered to be significantly different. Table 1 shows that, in general, our intuition was correct – the success rate tends to fall from question type 1 to 7. Our intuition proved to be incorrect at the bottom of the table, as students were substantially less successful on self-joins then they were on correlated sub-queries.

Table 1. Query Success rates for various types of queries on the Bicycle database.

Type of "Select" statement required in answer		No. of question variations	Success Rate: non-attempts wrong	Success Rate: non-attempts ignored	Non-attempts
1	Simple, one table	6	89%	90%	1%
2	"group by"	4	74%	75%	2%
3	"group by" with "Having"	4	58%	61%	5%
4	Natural Join	3	57%	61%	6%
5	Simple subquery	4	53%	58%	9%
6	Self-join	3	18%	24%	23%
7	Correlated subquery	6	39%	46%	16%

6.1 Simple Queries on a Single Table

We begin a more detailed analysis by considering the simplest queries in the system: queries made on a single table, using only the reserved words “select”, “from”, “where” and “and”. Table 1 summarizes student performance on these queries for the Bicycle database (see row 1). For that particular database, there are six variations. AsseSQL randomly assigned one of these six queries to each student.

As Table 1 shows, 90% of the students were able to provide a correct query. That students did so well is not surprising, given the simplicity of this type of query, but it does establish that at least 90% of the students were able to understand the English-language instructions given by AsseSQL (which could not be taken for granted, since many of our students have English as a second language) and that 90% of the students are competent users of AsseSQL.

6.2 “Group By”

Table 1 shows that around 75% of students were able to answer a question that required “group by”. However, as described in the method section, the “group by” pool exhibited the greatest differences in success rate of any set of variations.

One possible explanation for the difference in the success rate is the linguistic complexity of the four variations, since English is the second language of many of our students. However, Table 2 shows that there is no clear relationship between success rate and linguistic complexity, when linguistic complexity is estimated by the number of words in each variation. In fact, the variation with the lowest success rate also has the lowest word count.

Table 2. Success rates for the “group by” variations.

Variation	**Success Rate**: non-attempts considered wrong	Word count	Signal words
1	87%	17	“average”
2	70%	24	“average total”
3	87%	19	“average”
4	57%	16	“number of”

On inspection of the text of the four questions represented in Figure 2, another explanation suggests itself as to why variation 4 was substantially harder. This explanation is consistent with Reisner’s (1981) observation that the subjects in her study had difficulty knowing when to use “group by”, but they could successfully use “group by” when explicitly told to do so. With the exception of variation 4, all the variations use the word “average”, which is a clear signal to the student that the aggregate function “avg” is required, and hence “group by” is probably required. In contrast, the use of “number of” in variation 4 does not transparently signal that a specific aggregate function is required. In variation 2 the use of “average total” may have confused students, as those words signal two possible aggregate functions, which perhaps explains that variation’s middling success rate.

This analysis of differences in the success rates of the “group by” variations demonstrates the pedagogical value of looking at the data collected by our “particle accelerator” (i.e. AsseSQL). Because of this analysis, our teaching team was led to discuss exactly what it is that we are looking to test when we ask a “group by” question. Is our goal to (1) test whether a student recognizes that “group by” is required, or (2) merely test whether a student can actually write such a query when they know that “group by” is required? If our goal includes the former, then variations 1-3 need to be reworded or replaced. If the latter is our goal, then variation 4 needs to be replaced.

6.3 “Group By” Queries with “Having”

Table 3 compares the performance of students on two types of “group by” queries – queries with “having” and queries without “having” (i.e. the queries in rows 2 and 3 of Table 1). Table 3 shows that these two types of queries are significantly correlated ($p < 0.0001$), but only moderately so (phi = 0.49). The 19% in the top right of Table 3 may understate how much difficulty students have with “having”, as that is a percentage of all 986 students represented by the entire table. When the 186 in the top right cell is expressed as a percentage of the 742 in the top row, the figure is 25%. That is, a quarter of all students who could provide a correct “group by” without a “having” could not provide a correct “group by” that required a “having”.

Table 3. Comparison of “group by” with and without “having”. (N =986; phi correlation 0.49; $\chi 2$ test $p < 0.0001$)

	with “having” right	with “having” wrong
“group by” right	556 (56%)	186 (19%)
“group by” wrong	49 (5%)	197 (20%)

6.4 Natural Join and Self-join

Table 4 shows that the correlation between natural joins and self-joins is a moderate 0.41. The 38% in the top right of Table 4 is a percentage of all 986 students represented by that table. When the 371 in that top right cell is expressed as a percentage of the 599 students in the top row, the figure is 62%. That is, 62% of all students who could answer a natural join could not provide a correct self-join.

Table 4. Comparison of natural join and self-join. (N = 599; phi correlation 0.41; $\chi 2$ test $p < 0.0001$)

	self-join right	self-join wrong
natural join right	228 (23%)	371 (38%)
natural join wrong	9 (1%)	380 (38%)

Table 5. Comparison of simple and correlated sub-queries. (N =986; phi correlation 0.49; $\chi 2$ test $p < 0.0001$)

	Correlated right	Correlated wrong
Simple right	387 (39%)	189 (19%)
Simple wrong	71 (7%)	341 (35%)

6.5 Simple and Correlated Sub-queries

Table 5 shows, perhaps unsurprisingly, that the correlation between simple and correlated sub-queries is a moderate 0.49. The 189 (19%) in the top right of Table 5 is a percentage of all 986 students represented by that table. Expressed as a percentage of the 576 students in the top row, this figure is 33%. That is, one third of all students who could provide a correct simple sub-query could not provide a correct correlated sub-query.

6.6 Generalizing to Other Databases

We also collected data from students for two other databases, of similar complexity to the Bicycle database. Figure 3 compares the success rates for each query type in the Bicycle database with the success rates for those other two databases. (The numbers on the horizontal axis refer to the row numbers in Table 1.) All three databases show approximately the same pattern. The success rate for self-joins (point 6 on the horizontal axis) is the lowest success rate for all three databases. The only major difference between the three databases is that the "having" questions (point 3 on the horizontal axis) were especially difficult in "Bicycle2".

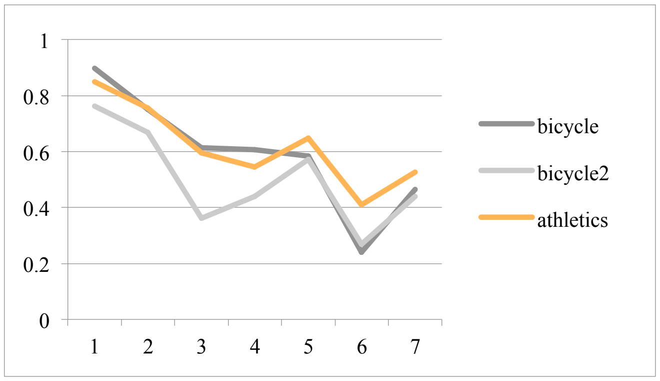


Figure 3. Success rates of the Bicycle database compared with two other databases.

6.7 Number of Attempts

In the "real world", nobody knows the desired result table before they write their query. In that sense, AsseSQL and most other online SQL testing systems are unnatural environments, as these systems provide the student with the desired table. We have therefore wondered whether a large portion of students may have answered questions correctly, particularly the easier questions, by brute force – that is, by making many attempts at the question, with quasi-random changes.

Analysis shows that this does not appear to be the case. For all three databases, the correlation (Pearson) between the median number of attempts at a question and that question's success rate is between -0.6 and -0.7, which is a strong negative correlation. The average number of attempts also exhibited a strong negative correlation between -0.6 and -0.7 for all three databases.

For the Bicycle database, this negative correlation is illustrated in Figure 4. The middle plot in that figure indicates the median number of attempts by the students who eventually provide a correct answer. A surprising aspect of Figure 4 is that students required relatively few attempts for the correlated sub-query. Note, however, that in this figure (and throughout this section of the paper) we are only considering students who eventually provide a correct answer. Thus, the figure merely shows that 39% of students (as shown in Table 1) who did provide a correct correlated sub-query were able to do so in relatively few attempts. The speed of those students on correlated sub-queries further emphasizes the difficulty of self-joins, since only 18% of the students could provide a correct self-join, half of whom needed 9 or more attempts, with a quarter requiring 16 or more attempts.

Figure 5 compares the median number of attempts needed to answer questions correctly in each of the three databases. All three databases show approximately the same pattern. There are two major differences across the three databases: (1) the number of attempts for simple sub-queries (i.e. point 5 on the horizontal axis) is relatively low in the "Bicycle2" and "athletics" databases, and (2) the number of attempts for self-joins (i.e. point 6 on the horizontal axis) is relatively low in the "athletics" database.

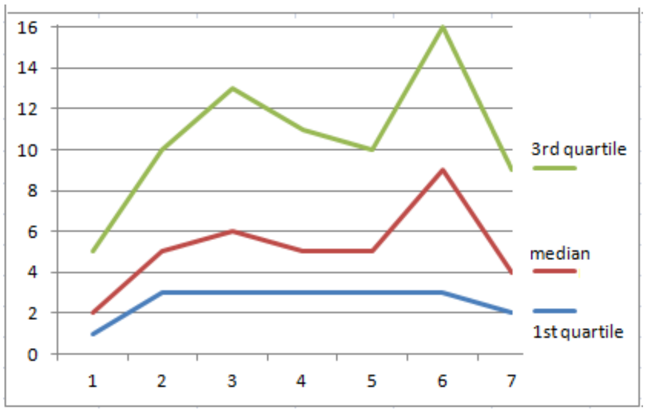


Figure 4. The distribution of number of attempts needed to answer each question type correctly in the Bicycle database.

In AsseSQL, we do not distinguish between attempts that are wrong because of a syntactic error, and attempts that are syntactically correct but return an incorrect table. It would be interesting to study data where that distinction is made.

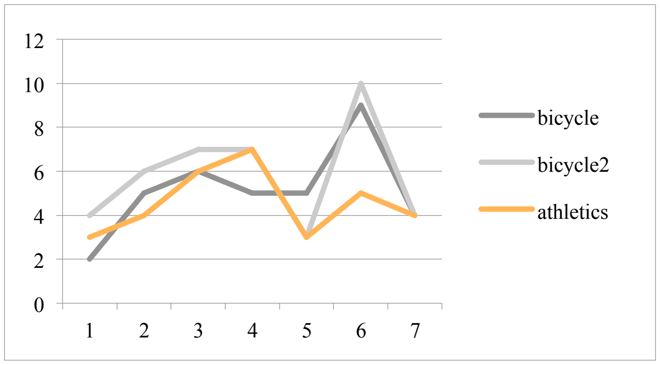


Figure 5. The median number of attempts needed to answer questions correctly in each of the three databases.

7. DISCUSSION

Many of our quantitative results are probably consistent with the intuitions of experienced database educators. It is important, however, that intuitions are tested. (After all, to return to Agarwal's metaphor quoted in the introduction, the Large Hadron Collider was expected to find the Higgs' Boson).

Perhaps our most surprising result – it was certainly a surprise to the authors of this paper – is that students found self-joins to be the most difficult of the query types we tested. Our intuition from many years of teaching databases was that correlated sub-queries were the most difficult type of query. Furthermore, we were surprised that our students found self-joins to be far harder than simple sub-queries, when these two types of queries can often be used interchangeably. A slightly surprising result was the extent of the difficulty of "having". While we suspected from experience that "having" troubled many students, we were surprised that a quarter of our students who could answer a "group by" that did not require "having" could not also correctly use "having".

Why are self-joins and (to a lesser extent) "having" troublesome for so many students? Our students have an equal opportunity to practice all the query types, so we are less inclined to believe that the trouble lies with insufficient practice. (But the prevalence in the database education literature of online tutoring systems may imply that the many educators believe that the problem is a lack of practice.) We are more inclined to believe that self-joins and “having” expose a conceptual problem for students. In the traditional teaching of SQL queries, the early emphasis is on the concept of a table. This leads to students thinking of operations on database tables as being much like the “real world” tables, such as spreadsheets. But self-joins and “having” have no common “real world” analog, so those operations are troublesome for many students. Reisner (1981) hypothesized a similar explanation, specifically about "group by". She wrote "*We suspect that* ... [students] ... *adopt an "operations-on-tables" strategy* ... [which] ... *does not work for the "group by" function, which requires users to think in terms of partitioning a table into subgroups*". As many database educators know, the foundational data structure for databases is the row, not the table, and database educators need to communicate that knowledge explicitly to their students. Furthermore, while we agree with Sadiq *et al.* (2004) that the declarative nature of SQL is a problem for some students, we suspect that it is especially a problem in the very early stages of learning SQL, and after students have begun to acquire that declarative understanding on simpler SQL queries, they are then taught “having”, self-joins and correlated sub-queries, which do require a procedural grasp of SQL.

Part of the contribution of our work is the establishment of a method for studying the difficulties students have with SQL queries. Our principal methodological contribution is our focus on the relative difficulties students have with SQL queries. For example, of our students who could provide a correct simple sub-query, one third could not provide a correct correlated sub-query, but it would be absurd to claim that this same ratio (i.e. one third) is universal. Clearly, that ratio will vary from database to database, from institution to institution, and even semester to semester within an institution. What we are claiming is that, when a cohort of students manifests significantly different success rates at (for example) simple and correlated sub-queries, then it will be the correlated sub-query that exhibits the lower success rate. (And likewise for other pairs of query types.) Note that our claim is not negated by cohorts that do not display any difference in success rates on simple and correlated queries. Clearly, an immature or low achieving cohort will find both simple and correlated sub-queries to be difficult, while a mature or high achieving cohort will find both simple and correlated sub-queries to be easy. What our quantitative results suggest are developmental stages in learning SQL – for example, competence at simple sub-queries precedes competence at correlated sub-queries. (And likewise for other pairs of query types.)

As another methodological issue, we advocate that prior to studying the relative difficulty of two different query types, a third query type be used as a screening test. In our study, a simple select on a single table (i.e. row 1 of Table 1) served that purpose. The purpose of the screening test is to establish that students have an interesting minimum level of knowledge. In our case, since 90% of our students met the minimum requirement, and since students in AsseSQL can answer questions in any order, we elected not to remove that 10% from our analysis. However, in future studies, it may be useful to conduct the screening test as a pre-test, and remove from the data those students who fail the screening test, especially when the percentage of students who fail the screening test is much greater than 10%.

Our analysis of the four variations in the “group by” pool points to one interesting future research direction. In that pool, three of the four variations clearly signaled that a “group by” was required, while the fourth variation did not. It would be interesting to verify and extend upon Reisner’s (1981) observation, that her subjects had difficulty recognizing when they should use “group by”, but they could successfully use “group by” when explicitly told to do so – does that observation hold for any other query types?

8. CONCLUSION

At the back end of many internet applications there is a database. However, the prevalence and importance of databases is not reflected in the computing education literature. There certainly is education literature on databases, but nothing like the literature on (for example) learning to program. Furthermore, most of the existing database education literature focuses on the architecture of online tutorial and assessment systems. There is very little literature on what students find difficult about writing database queries. This paper is the first published quantitative study of the relative difficulty for novices of different types of SQL queries. This paper provides a quantitative and methodological foundation upon which further studies may be built.

9. REFERENCES

Brusilovsky, P., Sosnovsky, S., Lee, D., Yudelson, M., Zadorozhny, V., and Zhou, X. (2008) *An open integrated exploratorium for database courses*. ITiCSE '08. pp. 22-26. http://doi.acm.org/10.1145/1384271.1384280

Brusilovsky, P., Sosnovsky, S., Yudelson, M. V., Lee, D. H., Zadorozhny, V., and Zhou, X. (2010) Learning SQL Programming with Interactive Tools: From Integration to Personalization. *Trans. Comput. Educ.* 9, 4, Article 19 (January 2010). http://doi.acm.org/10.1145.1656255.1656257

Kearns, R., Shead, S. and Fekete, A. (1997) *A teaching system for SQL*. ACSE '97. pp. 224-231. http://doi.acm.org/10.1145/299359.299391

Mitrovic, A. (1998). *Learning SQL with a computerized tutor*. SIGCSE '98, pp. 307-311. http://doi.acm.org/10.1145/273133.274318

Mitrovic, A. (2003) *An Intelligent SQL Tutor on the Web*. Int. J. Artif. Intell. Ed. 13, 2-4 (April 2003), pp. 173-197.

Post, G.V. (2001) *Database management systems: designing and building business applications*. McGraw-Hill.

Prior, J., and Lister, R. (2004) *The Backwash Effect on SQL Skills Grading*. ITiCSE 2004, Leeds, UK. pp. 32-36. http://doi.acm.org/10.1145/1007996.1008008

Prior, J. (2014) *AsseSQL: an online, browser-based SQL skills assessment tool*. ITiCSE 2014. pp. 327-327. http://doi.acm.org/10.1145/2591708.2602682

Reisner, P. (1981) *Human Factors Studies of Database Query Languages: A Survey and Assessment*. ACM Comput. Surv. 13, 1 (March), pp. 13-31. doi.acm.org/10.1145/356835.356837

Sadiq, S., Orlowska, M., Sadiq, W., and Lin, J. (2004) *SQLator: an online SQL learning workbench*. ITiCSE '04. pp. 223-227. http://doi.acm.org/10.1145/1007996.1008055

Stokes, P. (2013) *The Particle Accelerator of Learning*. Inside Higher Ed. https://www.insidehighered.com/views/2013/02/22/look-inside-edxs-learning-laboratory-essay