UNIVERSITY OF PITTSBURGH

INFSCI 2970 INDEPENDENT STUDY COURSE - SPRING 2018

SQL Practice Tool: Self-Assessment Problems for Learning SQL Programming

---

Author:
Kamil AKHUSEYINOGLU

Supervisor:
Dr. Peter BRUSILOVKSY

Contents

Chapter 1

Abstract

Structured query language (SQL) is most widely used query language. Students graduating from computer science, information science and information systems majors need to show mastery in SQL writing skills this knowledge as a building block for understanding other data manipulation techniques. However, learning SQL is a well-known challenge even for the students who successfully completed their introductory programming classes. To better assist students in SQL domain, researchers proposed different set of learning tools. In our work, we attempted to broaden the automatic assessment of SQL problems from traditional assessment orientation to practice-oriented self-assessment context by introducing SQL Practice Tool (SPT). SPT possess two important feature like parameterized problem generation capability through templates and visualization of query execution. We also presented the result from comprehensive evaluation of its usage in several database classes. Our analyses results point to the effectiveness of the SPT as a self-assessment learning tool.

Chapter 2

Introduction

Structured Query Language (SQL) is an important component of almost every introductory database course and has to be mastered by nearly every student graduating in computer science, information science, and information systems field. However, learning SQL is a known challenge. While SQL could be considered as a kind of programming language, its nature is quite different from languages like C++, Java, or Python, which are typically learned in introductory programming courses. As a result, mastering SQL could be hard even for students who succeeded in introductory programming classes.

To support teaching and learning SQL, computer science educators developed and evaluated a range of learning tools. In the context of our paper, two groups of tools are important. The first group attempts to uncover the internal work of SQL language by visualizing the results of query execution. The second group used different version of automatic assessment to evaluate the correctness of student queries. In our work, we mostly attempt to expand the second group of systems by taking the automatic assessment of SQL problems from traditional assessment context to practice-focused self-assessment.

Web-based practice systems with self-assessment problems become quite popular in the field of Computer Science Education over the last 10 years. These systems allowed students to practice and self-assess their knowledge of complex topics by solving large number of diverse programming problems. Practice systems allowed to remarkably expand both the amount of problem-solving and the amount of feedback that a student receives when learning programming. However practice systems for SQL are rare and no studies have been reported to assess their value. In

this paper we present a self-assessment technology for SQL, which is designed for a practice context. To better fit learning-oriented practice context the system includes several features like parameterized problem generation and visualization of query execution. We also present an extensive evaluation of this system in several database classes. Our data provides some insight on student use of the practice system and its value in the learning process.

Chapter 3

Related Work

3.1 Assessment and Self-Assessment Tools for Computer Science Education

Automatic assessment of program construction tasks has been one of the most popular topics in Computer Science education research. As early as 20 years ago, a number of pioneer programming assessment tools have been developed and evaluated in learning process Brusilovsky and Higgins, 2005. These tools allowed students to upload their solutions to programming assignments, evaluated them against a set of instructor-defined tests and quality metrics, and returned the results to the students much faster than it was done in a traditional manual grading process. The effectiveness and value of these tools made them very popular over the years and nowadays large number of programming courses use some automatic assessment tool. For example, one of the most popular systems of this kind, Web-CAT Edwards and Perez-Quinones, 2008 is used in hundreds of classes.

More recently the same approach has been used to create self-assessment tools for student practice. Practice tools usually allow students to submit their code through a Web form and offer "skeleton" code as a starter. Nick Parlante's CodingBat is one of the earliest examples of these tools Parlante, 2018. This model has been adopted by several tools, including CodeWrite Denny et al., 2011, CodeAssessor Zanden et al., 2012, PCRS Zingaro et al., 2013, and CloudCoder Hovemeyer et al., 2013. While the automatic assessment tools have been extensively studied in the context of mandatory assignments and their value has been reported in a number of papers

Brusilovsky and Higgins, 2005, the use of this technology in a self-assessment practice mode has received very little attention. Moreover, the majority of the research has focused on automatic assessment for programming languages such as C, Java, or Python. The role of similar technologies for different kind of languages, such as SQL, has received little attention especially in the self-assessment practice mode. In this paper, we attempt to bridge this gap by presenting a parameterized self-assessment system for SQL queries and reporting the results of its evaluation.

3.2 SQL Learning Tools

Learning query languages, especially SQL, is an important part of Computer Science Education. Researchers have tried to address this challenge by delivering different solutions to the challenge of SQL learning for over two decades. In general, SQL learning tools can be classified into two main groups: (1) visualizing query execution steps Danaparamita and Gatterbauer, 2011; Hardt and Gutzmer, 2017 and (2) automatic assessment tools. In our paper, we attempt to expand the second group of systems by incorporating the self-assessment practice mode as the main functionality. Since our learning tool mainly falls into the second category, we review the prior work in this group.

One of the earlier systems that provides support to students on formal query languages (including SQL) was introduced by Dietrich et al. Dietrich, Eckert, and Piscator, 1997. The aim of the introduced system was to improve the underlying grading system in institutions. Similarly, Sadiq et. al. proposed *SQLator* tool Sadiq et al., 2004, which was also created to be used in assignments for auto-grading purposes and had a pool of SQL problems with different levels of difficulty. Even if the authors noted the usage of the system due to practicing, having assessment on the same tool encourages the students to use the system which may not be considered as practice oriented self-assessment. Alternatively, Raadt and Lee introduced *SQLify* Raadt and Lee, 2006 as an auto-assessment tool, which adds peer-reviews into automated grading process. Kleiner et. al. presented *aSQLg* Kleiner, Tebbe, and Heine, 2013 as an online interactive tool for automated assessment of SQL queries, which

checks the syntax and cost of the answers provided by students. *SQL Tester* is a recent tool Kleerekoper and Schofield, 2018, which was designed for testing student on online tests as a part of their course grades. This tool was not designed to target the practicing through self-assessment but developed for testing purposes similar to *AsseSQL* Prior and Lister, 2004.

Most of the related work assessed the developed tools in a knowledge testing and grading context offering evidence of their values as well as some side effects Prior and Lister, 2004, however the value of automatic assessment in a practice context has been mentioned at most as a side benefit and has not been studied. Our paper bridges this gap by examining the impact of self-assessment practice on student learning outcome extensively.

Chapter 4

SQL Practice Tool

SQL Practice Tool (SPT) is an interactive web-based SQL learning tool which is designed to help students to self-assess their SQL problem solving skills in a practice context. As summarized in section 3.2, most of the available SQL tools are focusing on *automatic assessment* of the student queries from testing perspective. In contrast, SPT emphasizes the importance of SQL learning rather then testing and augments standard automatic assessment functionality with features focused on self-assessment. Most important of these features are: (1) parameterized problem generation using templates and (2) expanded feedback on students' queries by showing query execution results.

To allow students self-assess their SQL knowledge, a practice-oriented system should allow students to attempt the same type of problem multiple times until a desired level of mastery is achieved. To support this functionality, SPT needs a question generation process, which could produce different questions of the same

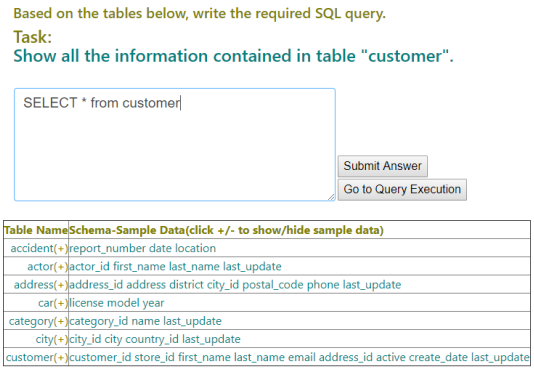

FIGURE 4.1: SQL Question Screen. It presents an SQL select problem to show all records stored in *customer* table. Students have access to see sample database schema (at the bottom) and can submit their answer or proceed to *Query Execution* mode.

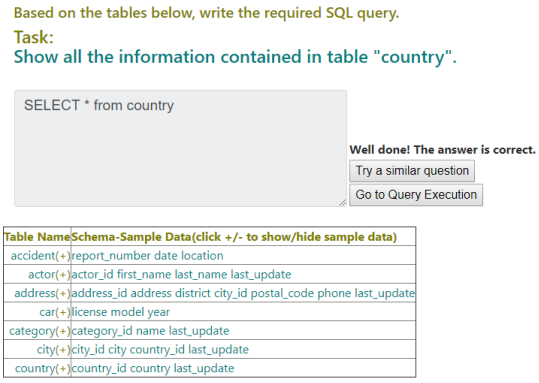

FIGURE 4.2: SQL Practice Tool shows the submission result of the student solution, in this case is successful. Students can select to try a similar problem to the one that solved or go to *Query Execution* mode to see the output of their query.

type. To fulfill this need, we used *template-based* parameterized question generation approach to create multiple sets of similar questions to be used by SPT. The template-based parameterized question generation also helped us to better track student learning in other studies [1]. Table 5.2 lists the topics and number of templates used for particular topic. This topic-template mapping was used for the classroom studies analyzed in this paper. [2]

Templates are constructed to create similar questions by using same set of SQL constructs or concepts to make sure that the questions generated have similar difficulty level. For example, the template focused on simple *select-from-where* queries, generates questions which require querying single database query, listing all fields in that table and also having a simple *where clause* with a single constraint. This way, set of questions can be solved similarly as it has same query structure. In fact, to answer a particular question, students pay attention on other details as well such as table names, column names, column types etc. Practicing by self-assessment on similar questions after successful attempts may help them to improve their skills or trying similar questions after an incorrect attempt may help them to overcome their struggle. This feature also makes it easy for us to provide *model* solutions to generated questions.

Figure 4.1 illustrates the SPT's SQL question query screen. It shows the task of a

[1] Self-references are removed to the previous studies where SPT was used.

[2] SPT is designed and focused on SQL Select statements. It does not support Data Definition Language (DDL) and Data Manipulation Language (DML) type of queries.

given question (related to a template) and prompts students to write an SQL query based on the given Database schema. The sample schema information is shown as sample table names and related column names. The list of tables are dynamically determined based on the question. SPT also provides sample data for the tables in the schema (i.e. top 5 records are shown). The sample data is provided to assist students in understanding data types of each column and serves as a column description. Students can write their SQL queries in the provided text-area in a free format. SPT does not provide the structure of the SQL statement which decreases the scaffolding but increases the challenge and helps them to self-assess their SQL statement writing skills like in a database query system.

Once a student types the SQL statement and clicks *Submit Answer*, the student's answer is executed on the sample database at the server and SPT provides immediate correct/incorrect feedback. To improve the security, only select statements are allowed and only the tables shown to the student are allowed to be queried. The student query answer is compared to the model solution of that particular question. Student answer is evaluated as correct only if the query result contains exactly same data with the associated correct solution (model solution) and has the same column names in order. After the answer is checked, SPT displays a screen similar to Figure 4.2 (success case). Students can *try a similar question* or move to *Query Execution* mode as shown in Figure 4.3. If the student asks for a similar question, SPT displays new question from the same template with the previous question. This feature permits students to keep self-assess their knowledge if they need to practice more on similar questions.

We need to note that besides providing access to the similar questions related to the current presented question, SPT is not equipped with question sequencing or difficulty level adjustment. SPT is designed to be used as a self-assessment practice tool in a practice system context. Hence, in the classroom studies analyzed in this paper, SPT were a part of the *container* practice system where it filled the self-assessment needs of the students. Selection of the template (eventually the question) was dealt by the practice system. To practice with the questions to a particular template, a template identifier was provided to SPT by the practice system. This allowed us to

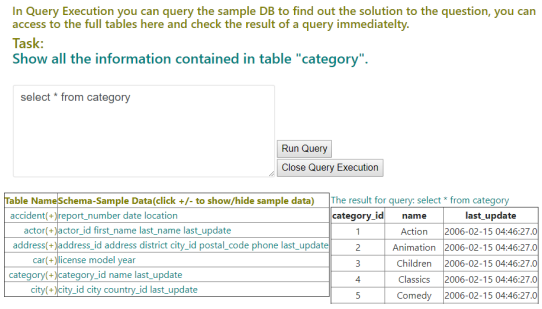

FIGURE 4.3: SQL Practice Tool Query Execution Mode. Students can execute SQl select queries and dynamically see the results of their query as shown as a result table on the right bottom of the figure. Students can elaborate their queries in this mode and later can return back to solution submission mode.

reduce the response time to present a question to the students and control the number of different questions used in the experiments. Thus, we used template-based question generation feature offline to generate the set of questions and randomly select one of the questions from the template-question sets based on the given template identifier.

Chapter 5

Classroom Studies

To better examine the effect of the SPT on learning SQL concepts, we performed three classroom studies at a graduate level Database Management course in Fall 2014, Spring 2015 and Fall 2015 semesters at a University in the USA. Both courses in fall semesters were thought by the same instructor while the course in the spring semester was thought by another instructor. The structure of the course remained same for the mentioned semesters including the syllabus and the grading policy.

The practice system was introduced before SQL concepts were introduced. As the studies conducted at the graduate level course, some students may already learned SQL concepts before enrolling to this course. To better evaluate the learning outcomes, a pretest and a post-test were administered to measure knowledge of the students at the beginning of the semester and at the end. Both tests had same 10 questions that requires writing SQL statements related to a database schema on a paper.

The practice system was provided to the students solely as a practice system thus the usage of it was not mandatory. However, in order to encourage the students to check the system and see the provided tools, one percent extra credit was provided to the students who solved at least 10 SQL problems from SPT. More details about the practice content provided at section 5.2

5.1 Participation Summary

Table 5.1 represents the distribution of the students participated to the classroom studies. For our analyses, we considered a student as a *practice system user* if the student solved at least one SQL question or viewed an example. In addition, our

analysis is limited to students who took both the pretest and post-test. As shown in the table, most of students who took pretest and post-test also logged into the practice system at least once. However, when we excluded the students who did not interacted with the content, 82 students left which is about 32% of the registered students.

TABLE 5.1: Classroom studies participation information.

	Fall 2014	Spring 2015	Fall 2015	Total
Registered students	103	46	108	257
Students logged into practice system	81	36	84	201
Students who took pretest&posttest	84	36	84	198
Practice system users	40	20	22	82

5.2 Practice Content

Students accessed the practice content through the practice system with their user accounts and the interactions were logged with time-stamp information. SPT had 396 SQL practice questions generated from 46 different templates. The questions were grouped under 10 topics ranged from select-from to sub-queries through templates (i.e. each topic had different number of templates). Table 5.2 shows the number of templates associated to each SQL question topic and average success rates by semesters. As the table shows, most challenging topics were *Group By and Having*, *Sub-Queries* and *Set-Operations*. The overall success rate conformed with the findings of Kleerekoper and Schofield Kleerekoper and Schofield (2018) and Ahadi et. al. (Ahadi et al., 2016) where they also reported lowest success rate in *Group By* and *Group By Having* queries in a testing context. Even though direct comparison of success rates is not a solid evidence, similar findings suggest that students who used our system as a practice tool had similar performance to students who were tested through SQL Tester and AsseSQL.

Students also had access to another interactive web-based tool which enables students to view annotated examples of SQL statements and explore explanations for each example statement line by line which we called it SQL Example Tool (SET).

TABLE 5.2: Template to topics distribution with success rates.

		Success Rates			
Topic	Templates	Fall 2014	Spring 2015	Fall 2015	Overall
Select-from	6	0.79	0.74	0.84	0.79
Arithmetic Expressions	5	0.74	0.70	0.69	0.71
Select-from-where	5	0.50	0.39	0.61	0.48
Pattern Matching	5	0.69	0.52	0.65	0.62
Multiple Table Queries	5	0.51	0.41	0.44	0.46
Order By	4	0.41	0.52	0.62	0.47
Set Operations	1	0.40	0.42	0.34	0.39
Aggregate Functions	5	0.64	0.61	0.69	0.64
Group By and Having	5	0.34	0.37	0.38	0.35
Sub-Queries	5	0.35	0.35	0.50	0.38

In total, interactive SET offered 64 SQL statement examples with 268 distinct explanation lines. The examples were grouped into 19 topics which also overlap with the topics of SPT. The set of practice examples and problems was same for all classroom studies.

5.3 Usage Statistics and Learning Metrics

User interaction data collected from different semesters combined together. Table 5.3 shows the overall usage statistics and learning metrics. We measured the usage activities of users in the practice system usage separately for SQL Practice Tool and SQL example tool. The practice systems were available to the students until the end of the database classes, however, the analyzed data has only the student interactions between the pretest and the post-test period.

For the SQL Practice Tool, we counted the number of question attempts, successful attempts, distinct templates attempted, distinct templates solved and SQL query attempts. For SET, we quantified the number of examples accessed and number of explanations viewed. We also measured the total time users spent on each tool. The total time spent in SQL Practice tool covers the time students spent on solving the questions and the time they spent in running SQL queries. For the examples tool, the total time measured combines the time students spent on viewing examples and explanations provided.

Students, on average, tried 66 times to solve SQL questions and successfully solved them in 35 attempts. By the design of our SQL Practice Tool, students' question attempts were recorded with template identifier rather than specific question identifier. So, the numbers noted previously covers the cases where students tried the same questions. To understand how students' trials covers unique questions, we calculated *distinct templates attempted* and *distinct templates solved*. On average, students tried solve questions from 34 (74%) distinct templates and solved questions from 32 (70%) distinct templates. 26 students (33%) solved questions from all templates. Students also ran SQL queries 59 times on average regardless of the given question. On average, they spent 61 minutes on SQL Practice tool.

Students also interacted with the SET and viewed 54 (84%) distinct examples and explored 126 (47%) distinct explanations. 46 students (59%) viewed all examples at least once but none of them viewed all the provided explanations in the tool as the maximum number of viewed distinct explanations was 185 (69%). SET users spent 58 minutes on average. Even if the total usage differences between SQL Practice Tool and Example Tool, time spent for each distinct item was significantly higher for SQL Practice Tool (M=1.62, SD=1.01) compared to Example Tool (M=1.02, SD=0.91), $t(77) = -5.4$, $p < .001$. However, this might be caused only by the nature of the tools provided as one of them asks students to solve questions while the other presents examples.

Pretest and post-test scores are reported as normalized to 0-1 range. By using pre and post-test scores, normalized learning gain is calculated and used as one of the learning metrics. We calculated normalized learning gain by (post-pre)/(max_possible_post-pre). We used post-test and normalized learning gain as performance outcomes.

TABLE 5.3: Summary statistics for usage of SQL Practice Tool and SQL example tool by practice system users (N=78)

	Mean	Median	Min.	Max.
SQL PRACTICE TOOL				
Question attempts	65.9	72	3	163
Questions solved	34.7	43	1	75
Distinct templates attempted	34.1	44.5	3	46
Distinct templates solved	32.3	41	1	46
Free SQL query attempts	59.4	42.5	0	391
Total time on questions (min)	61.1	53.4	0.45	208
Average time (min) per distinct item	1.62	1.29	0.15	4.72
SQL EXAMPLE TOOL				
Distinct Examples accessed	53.5	64	4	64
Explanations viewed	271	274	0	716
Distinct explanations viewed	126	174	0	185
Total time on examples (min)	57.6	38.9	0.12	297
Average time (min) per distinct item	1.02	0.80	0.03	4.64
LEARNING METRICS				
Normalized pretest	0.15	0.12	0	0.62
Normalized post-test	0.52	0.52	0.1	0.83
Normalized learning gain	0.43	0.43	0.1	0.75

Chapter 6

Results

As seen in Table 5.1, 82 students are considered as the practice system user. We further excluded 4 more students from our analyses who had equal pretest and post-test scores. Thus, in this section, we analyzed the data collected from 78 students.

Before using the raw logged data for our analyses, we performed outliers treatment by Winsorization. We replaced all outliers with the value at the 95th percentile instead of totally excluding those students' records from analyses. Further, we log-transformed time based features.

We started our analyses by checking the overall comparison of the semesters. Then, we continued by analyzing the practice usage effect of our SQL Practice Tool on SQL learning.

6.1 Semester Comparison

To figure out any possible differences between semesters based on learning metrics and student success rates on problem solving attempts on SPT, we conducted exploratory analyses. First, we examined the student prior knowledge differences and found out that there were no significant differences among semesters (Kruskal-Wallis test:$\chi^2(2) = 4.8, p = .09$), thus they started the semester with similar prior knowledge. Second, we checked the if the students finished their database course at the same level of knowledge which was evaluated by the post-test. We found out that there was no significant difference between students' scores on post-test, *F(2, 38.95)=0.52, p = .6*. Last, we concluded checking differences in learning metrics by examining discrepancies in normalized learning gain. We observed that following the no difference in pre and post-test scores, there was no significant difference in

normalized learning gain as well, *F(2, 37.347)=2.92, p = .06*. These findings suggest that students from different semesters were not significantly different by the measurements of learning metrics.

We examined the success rate differences among semesters using two-way ANOVA with topic difficulty level (i.e. difficult/easy) and semester as factors. We classified topics as difficult/easy with respect to the overall success rate shown in Table 5.2. ANOVA results show that only difficulty level is the main effect on success rate, *F(1, 26)=65.8, p < .01*. It is also worth to note that there is no significant correlation between success rate and pretest scores. Thus, students had similar success rate at SQL problems provided.

6.2 Effect of SQL Practice Tool on Learning Outcomes

We evaluated the effect of our tool on two learning outcomes: (1) Post-test scores and (2) Normalized learning gain. We also reported the effect of SET along with SPT to clarify the relative importance of each tools presented to the students.

6.2.1 Methodology

To analyze the effects, we fit multiple linear regression models to predict the learning outcomes from usage features. As mentioned in section 5.3, we have different set of usage statistics. However, we need to be sure about the independence of these measures to overcome multicollinearity problem in regressions. Most of the reported measures were highly correlated with each other. In specific, distinct templates solved in SPT was highly correlated with other template and question related features such as questions attempts ($\rho = 0.83$), questions solved ($\rho = 0.91$), distinct templates attempted ($\rho = 0.98$), total time spent ($\rho = 0.79$). If a student solve a question served from a particular template, that template was marked as solved. If the student keep solving similar questions from the same template, it was not counted as additional solved template. We selected to use distinct templates solved feature among these feature since it had the highest correlation with the learning outcomes ($\rho_{post} = 0.23, \rho_{lgain} = 0.34$). However, it did not have high correlations

with average-time-per-item ($\rho = -0.18$). Similarly, we also checked the strong correlation between the SET features. In particular, distinct examples accessed was highly correlated with explanations viewed ($\rho = 0.77$), distinct explanations viewed ($\rho = 0.71$) and average time per distinct item ($\rho = -0.77$). Thus, we selected distinct examples accessed as a representation of other highly correlated features since it had the highest correlation with the learning outcomes ($\rho_{post} = 0.17, \rho_{lgain} = 0.27$).

To better interpret the effect of practice tools, we introduced a new set of features. For SPT, we calculated success rate of each student on questions as *questions solved/questions attempts*, which had moderate negative correlation with distinct templates solved ($\rho = -0.35$). We also measured the median time spent by student in each question attempt as a measure of their struggle in solving questions. Median question time had weak negative correlation with distinct templates solved ($\rho = -0.18$). For the Example Tool, we calculated median time spent on examples by students as a measure of how carefully they were paying attention to the examples and explanations. This new feature negatively correlated with distinct examples accessed ($\rho = -0.17$). To better incorporate the viewed explanations within each example, we calculated average explanation coverage by (distinct explanations viewed)/(available explanations). Explanation coverage was weakly correlated with distinct examples viewed ($\rho = 0.11$).

To make sure that we did not include any highly correlated features in our regression models, we further check the collinearity of regression features by calculating the *Variance Inflation Factors (VIF)*. In all mentioned regression models, none of the features had $\sqrt{VIF} > 2$.

6.2.2 Effect on Post-Test Scores

To predict post-test scores, we fitted multiple regression models and controlling the effect of prior knowledge by adding normalized pretest score as a predictor. First, we wanted to see the main effect of question solving on post-test scores by fitting a regression model with pretest score and distinct templates solved (dist_temp_solved). The results of this model shown in Table 6.1-Model 1 column. The regression results showed that *dist_temp_solved* is a significant independent factor after controlling by the pretest. As expected, students with high prior knowledge had better post-test

scores. One unit increase in normalized pretest score increases the normalized post-test score by 0.6 unit. More importantly, each additional solved template increases normalized post-test score by 0.003. This means that, if a student solve 10 questions (22%) from different templates, the student would have increased the post-test score by 3%.

To improve the model performance and to detect other important usage features' on post-test score, we ran backward stepwise regression model with all SPT related features plus pretest to control prior knowledge. Stepwise regression model revealed that after pretest score, *dist_temp_solved* and average time (seconds) spent per question (*sql_time_per_item*) were significant features (Table 6.1- Model 2 column). The regression coefficients stays similar in *Model 2* compared to *Model 1* with minor improvement in the favor of *dist_temp_solved* after the inclusion of *sql_time_per_item* into the model. If a student spent one minute more in average on all questions she attempted, the post-test score decreased by 4% after controlling by pretest and distinct solved templates. This result hints us that students who spent more time in average on the same number of distinct solved templates had less post-test scores. Overall, the Adjusted R^2 increased to 0.435.

Further, we ran separate stepwise regression model for all the features related to SET controlling by the pretest. This model shows that median time spent on examples (*ex_median*) and distinct examples accessed (*ex_dist*) are the significant independent factors. To see the relative importance of the SET features with SPT features, we ran another multiple regression model with all significantly important features. Table 6.1- Model 3 column shows the model results. This model's Adjusted R^2 is improved slightly compared to Model 2, thus this model explains more variability in post-test scores however none of the features related to example tool are found to be significant. However, *dist_temp_solved* feature stays significant as in other two models.

As a summary, in all three regression model to predict post-test score, SPT features are constantly found to be positive significant factors. These findings highlights the importance of self-assessment with SPT in a practice context on student learning outcomes.

TABLE 6.1: Dependent Variable: Post-test

Variable	Model 1	Model 2	Model 3
pre	0.63*** (0.09)	0.605*** (0.089)	0.616*** (0.089)
dist_temp_solved	0.003** (9.8e-4)	0.004*** (0.001)	0.0046** (0.001)
sql_time_per_item		-6.4e-4* (2.7e-4)	-4.9e-4 (2.8e-4)
ex_dist			-8.3e-3 (5.1e-3)
ex_median			-8.9e-4 (0.001)
R^2	0.399	0.435	0.442
p-value	1.94e-9	7.53e-10	3.91e-9

(Standard errors in parentheses)
$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

6.2.3 Effect on Normalized Learning Gain

In addition to the extensive analyses on post-test scores, we also conducted similar analyses to see the effect of self-practice usage on normalized learning gain (NLG). To predict NLG, we do not need to include pretest score as a control variable since the NLG already takes into account the differences in prior knowledge. As a first step, we fitted a regression model to see the importance of problem solving interactions in SPT on NLG. We again used distinct templates solved (*dist_temp_solved*) feature as a representation of problem solving interactions. Regression results show that dist_temp_solved is a significant factor that predict NLG (Table 6.2-Model 1). Each additional solved template increases NLG score by 0.003. Thus, if a student does practice and solve 10 problems, the NLG increases by 3%. However, it should be noted that the Adjusted R^2 for the model 1 is very low (0.09) which means that this model only explains 9% variance in the NLG.

To improve the regression model performance and to see if there are other significant SPT usage features' on NLG, we ran backward stepwise regression model with all SPT features. Stepwise regression model revealed that only *dist_temp_solved* and average time (seconds) spent per problem (*sql_time_per_item*) are significant features (Table 6.2- Model 2 column). *dist_temp_solved* has a coefficient of 0.004 which means

TABLE 6.2: Dependent Variable: Normalized Learning Gain

Variable	Model 1	Model 2	Model 3
dist_temp_solved	0.0032**	0.0042***	0.0046**
	(0.001)	(0.001)	(0.002)
sql_time_per_item		-6.6e-4*	-5.7e-4
		(2.8e-4)	(3.0e-4)
ex_dist			-7.7e-4
			(1.4e-3)
ex_median			-5.9e-3
			(5.5e-3)
R^2	0.098	0.148	0.14
p-value	0.003	9.2e-4	0.004

(Standard errors in parentheses)
$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

that if a student solve 10 distinct problems (from different templates) by practicing on SPT, the NLG score increases by 4%. Inclusion of average time spent per problem as a predictor improves the model's Adjusted R^2 to 0.15. Similar to the post-test predictions, *sql_time_per_item* has negative effect on NLG as well: students who spent one more minute in average on solving same number of distinct problems had 4% less NLG.

As a last step, we ran backward stepwise regression to check the importance of only SET interaction features (i.e. ex_dist and ex_median) on NLG. Apart from the regression results for the post-test score, none of the SQL Example tool feature found to be significant (i.e. there is no significant regression model which significantly predicts NLG). Nonetheless, we included them to the variables used in model 2 to see if it improves the prediction power of the regression model. As shown in Table 6.2-Model 3 column, the performance of the model degrades after including SQL Example tool features. However, as shown in prior regression models, *dist_temp_solved* feature stays as the only significant predictor.

To summarize, in all three regression model which predicts NLG, SPT features are constantly found to be positive significant factors. These findings emphasizes the positive effect of practicing through self-assessment on student learning outcomes.

Chapter 7

Conclusion and Future Work

Even though the SQL has quite different nature than other programming languages and causes unique learning challenges, there is a gap in the existing Computer Science Education research on SQL practice systems which encourage self-evaluation . We attempt to fill this gap by introducing an web-based SQL Practice Tool (SPT) in this paper. Different than earlier SQL learning systems which are based mainly on automatic assessment, SPT is practice-focused and supports self-assessment. SPT has two unique feature: template-based problem generation and visualization of query execution for detailed feedback on self-evaluation.

In contrast to the prior research, we examined the impact of self-assessment practice on student learning in a practice environment. SPT was not designed or not used to collect student assignment submissions to ease the auto grading rather it was offered to students as a non-mandatory self-assessment tool. To better assess the value of our tool, we presented extensive evaluation of the system in graduate level database courses. Our results showed that problem solving activities on SPT was significant predictor of the learning outcomes. Solving distinct problems which are served through templates was significant factor in predicting post-test scores after controlling the prior knowledge variances. Similar importance also reported in our analyses for the normalized learning gain.

As we reported the percentage of the users of the practice system among the registered students, most of students decided not to use the provided practice system in general. Thus, the analyses conducted in this paper are subject to the self-selection bias. Students who chose to use the provided systems already showed the passion in practicing and learning through educational technologies. Other students who

decided not to use our systems might have other means of learning strategies.

In future work, we plan to improve the web-technology of the SPT and make it fully Javascript compatible standalone application to overcome the need in a server-side application. We also want to improve the user interface design and include data-driven adaptive support to the students to improve the effects of self-assessment. Further, we want to conduct more classroom studies to explore the impact of SPT on student learning behavior in conjunction with other types of learning tools.

Bibliography

Ahadi, Alireza et al. (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 - SIGCSE '16*, pp. 401–406. ISBN: 9781450336857. DOI: 10.1145/2839509.2844640. URL: http://dl.acm.org/citation.cfm?doid=2839509.2844640.

Brusilovsky, Peter and Colin Higgins (2005). "Preface to the special issue on automated assessment of programming assignments". In: *J Educ Resour Comput* 5.3, p. 1. ISSN: 15314278. DOI: 10.1145/1163405.1163406. URL: http://portal.acm.org/citation.cfm?id=1163406&dl=GUIDE&coll=GUIDE&CFID=7873848&CFTOKEN=75085508.

Danaparamita, Jonathan and Wolfgang Gatterbauer (2011). "QueryViz: Helping Users Understand SQL Queries and their Patterns". In: *Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11 (EDBT)*, p. 558. ISBN: 9781450305280. DOI: 10.1145/1951365.1951440. URL: http://portal.acm.org/citation.cfm?doid=1951365.1951440.

Denny, Paul et al. (2011). "CodeWrite: Supporting student-driven practice of Java". In: *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, pp. 471–476. ISBN: 9781450305006. DOI: 10.1145/1953163.1953299. URL: http://doi.acm.org/10.1145/1953163.1953299.

Dietrich, Suzanne W, Eric Eckert, and Kevin Piscator (1997). "WinRDBI: a Windows-based relational database educational tool". In: *ACM SIGCSE Bulletin*. Vol. 29. 1. ACM, pp. 126–130.

Edwards, Stephen and Manuel Perez-Quinones (2008). "Web-CAT: automatically grading programming assignments". In: *ITiCSE 08 Proceedings of the 13th annual conference on Innovation and technology in computer science education* 40.3, pp. 328–

328. ISSN: 00978418. DOI: 10.1145/1597849.1384371. URL: http://portal.acm.org/citation.cfm?doid=1597849.1384371.

Hardt, Ryan and Esther Gutzmer (2017). "Database Query Analyzer (DBQA) - A Data-Oriented SQL Clause Visualization Tool". In: *Proceedings of the 18th Annual Conference on Information Technology Education*. Rochester, New York, USA: ACM, pp. 147–152. ISBN: 9781450351003.

Hovemeyer, David et al. (2013). "CloudCoder: Building a Community for Creating, Assigning, Evaluating and Sharing Programming Exercises (Abstract Only)". In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, p. 742. ISBN: 978-1-4503-1868-6. DOI: doi:10.1145/2445196.2445451. URL: http://dx.doi.org/10.1145/2445196.2445451%0Aciteulike-article-id:13241936.

Kleerekoper, Anthony and Andrew Schofield (2018). "SQL Tester : An Online SQL Assessment Tool and Its Impact". In: *Proceedings of the 2018 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 87–92. ISBN: 9781450357074.

Kleiner, Carsten, 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 Calling '13*, pp. 161–168. ISBN: 9781450324823. DOI: 10.1145/2526968.2526986. URL: http://dl.acm.org/citation.cfm?doid=2526968.2526986.

Parlante, Nick (2018). *codingbat.com*. URL: http://codingbat.com/about.html.

Prior, Julia Coleman and Raymond Lister (2004). "The backwash effect on SQL skills grading". In: *ACM SIGCSE Bulletin*. Vol. 36. 3, p. 32. ISBN: 1581138369. DOI: 10.1145/1026487.1008008. URL: http://portal.acm.org/citation.cfm?doid=1026487.1008008.

Raadt, Michael De and Tien Yu Lee (2006). "Do students SQLify ? Improving Learning Outcomes with Peer Review and Enhanced Computer Assisted Assessment of Querying Skills". In: *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*. ISSN: 14043203. DOI: 10.1145/1315803.1315821.

Sadiq, S.W. et al. (2004). "SQLator: An Online SQL Learning Workbench". In: *Proceedings of the Ninth Annual SIGCSE Conference on Innovation and Technology in*

Computer Science Education, pp. 223–227. ISSN: 00978418. DOI: 10.1145/1007996.1008055. URL: http://dl.acm.org/citation.cfm?id=1008055.

Zanden, Brad Vander et al. (2012). "CodeAssessor: An Interactive, Web-Based Tool for Introductory Programming". In: *Journal of Computing Sciences in Colleges* 28.2, pp. 73–80.

Zingaro, Daniel et al. (2013). "Facilitating code-writing in PI classes". In: *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13*, pp. 585–590. ISBN: 9781450318686. DOI: 10.1145/2445196.2445369. URL: http://dl.acm.org/citation.cfm?doid=2445196.2445369.