A Web-based Semi-Automatic Assessment Tool for Formulating Basic SQL Statements: Point-and-Click Interaction Method

Aisha AL-Salmi
Department of Computer Science, Loughborough University, Loughborough ,U.K.

Keywords: Semi-Automated, Learning, Assessment, Online, SQL.

Abstract: Learning the Structured Query Language (SQL) is an important step towards developing students' database skills. As such, the number of higher education students learning SQL is constantly increasing. In this context, most researches focus on marking and providing feedback on the final query output rather than on the formulation of the SQL statement clauses. Focusing on statements formulation can assist the examiners in diagnosing the strengths and weaknesses of students' answers and provide detailed feedback on SQL statements that have been submitted for marking. This paper proposes a new semi-automatic assessment tool called SQL Formulation Editor (SQL-*FE*) for higher levels of education. The tool allows students to formulate SQL statements using point-and-click interaction method. To ensure the effectiveness of the method; the research has conducted an experiment which compares SQL-*FE* with the SQL Management Studio (SSMS) tool. The results have provided reasonable evidence that using SQL-*FE* can have a beneficial effect on formulating SQL query on-time and demonstrated a significant improvement in students' performance.

1 INTRODUCTION

Paper-based assessment has shown a number of problems due its manual nature, especially when greater number of students are enrolled in one class (Carter et al., 2003). Manual assessment might affect examiners' time management as the marking workload is increased, therefore forcing the examiners to either set their students less assessment tasks (e.g. mid-terms, quizzes and assignments) or add additional marking time to their schedules (Carter et al., 2003). In addition, large class sizes, limited time for marking assessments and non-effective feedback have led educators to consider computerised assessments. Automated assessment is becoming more useful for both students and staff since computer networking technology can support teaching and learning in higher education. Peat and Franklin (2002) stated that online assessment has become more popular and supporting the improvement of teaching and learning. A study conducted by Woit and Mason (2003) shows that automated assessment may improve students' motivation and programming efficiencies when it is implemented securely and efficiently. In addition, online assessment provides students with appropriate feedback that can help them enhance their learning progress (Ihantola et al., 2010).

1.1 SQL Manual Assessment

The Structured Query Language (SQL) is a database language for querying and manipulating relational databases (Bobak, 1996). It is one of the essential topics in database modules taught in higher education. Formulating and executing SQL queries is an essential part of relational database courses. However, manual SQL formulation poses a great challenge for both examiners and students. A research by Renaud and van Biljon (2004) states that the difficulties of solving SQL questions are "…due to the nature of SQL, and the fact that it is fundamentally different from the other skills students master during their course of study".

1.2 Case Study

To confirm the challenges of manual SQL assessments, several SQL statements were retrieved from 150 exam scripts of two years (2013 and 2014). This data was collected from the Database module taught to undergraduate students at Loughborough University. Each question on the exam script was

AL-Salmi, A.
A Web-based Semi-Automatic Assessment Tool for Formulating Basic SQL Statements: Point-and-Click Interaction Method.
DOI: 10.5220/0006671501910198
In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018)*, pages 191-198
ISBN: 978-989-758-291-2

analysed individually to find common errors as well as the number of students who made the same error. After analysing all the SQL script answers, there were multiple common errors in SQL statements attempted in manual SQL assessments. This research initially categorised the students' common errors as synonyms, syntax errors, incorrect keywords/ functions and incomplete SQL statements. Table 1 illustrates several common errors made by students and their descriptions.

Table 1: Examples of common errors made by students in the Database exam of June 2013.

Question	Model Answer	Students' Answer	Common Error Description
Display the department number and total salary of employees in each department that employs five or more people.	`SELECT DEPTNO, SUM(SALARY) FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO) >= 5;`	`SELECT DEPTNO, SUM(SALARY) FROM EMP GROUP BY DEPTNO WHERE COUNT (DEPTNO)>=5;`	a) Use Where instead of having clause.
		`SELECT DEPTNO,? SALARY FROM EMP GROUP BY DEPTNO HAVING COUNT (EMPNO)>5;`	b) Missing SQL function SUM()
		`SELECT DEPTNO, TOTAL(SALARY) FROM EMP GROUP BY DEPTNO HAVING COUNT (DEPTNO)>=5);`	c) Use Total as synonym of SUM
		`SELECT DEPNO, COUNT(SALARY) FROM EMP GROUP BY DEPTNO HAVING COUNT(SALARY)>=5;`	d) Use COUNT instead of SUM

The number of common errors made by the students in both years suggests that students might have found understanding the queries a challenge, because most of them made the same errors. In common error "a", many students tried to solve the first question using the "WHERE" clause instead of the "HAVING" clause, when there cannot be an aggregate function in a WHERE clause.

In common error "b", students attempted the query; however, they failed to add an important component of an SQL query into their solution – namely the "SUM" function. Common error "c" shows that some students could understand the requirement of the query, that is, that they needed to use a function. However, they used "TOTAL" instead of "SUM", which causes errors in the query. The last common error, "d", shows another way of changing the keyword, where students attempted the query using "COUNT" instead of the SUM function. As is clear from Table 1, the last three common errors are based on functions, which indicate that students might have had some confusion or lack of awareness of functions and their usage.

Therefore, one can conclude that manual assessment leads to a less efficient learning process and creates difficulty in assessing students' work; whereas automated assessment can achieve an improvement in learning and teaching processes, since it can encourage interactions between examiners and students and enhance the marking after submission.

This paper addresses the problems of manual formulation of SQL statements. It discusses the point-and-click method that aims to minimise or remove trivial errors of SQL statements. Furthermore, it describes an experiment that was conducted using the new implemented SQL formulation editor (SQL-*FE*) with an existing SQL tool and highlights its impact on time efficiency and students' performance.

2 METHOD

The point-and-click interaction method can be used with different input devices; for instance, computer mouse devices, touch pads, and touch screens. However, there are two questions to identify the selection of the point-and-click method, which are:

a) Why has this tool been chosen to use the point-and-click interaction method rather than the drag-and-drop interaction method or typing using the keyboard?

b) Does using this method lead to enhancing the performance of students in SQL assessment exercises?

Several researchers have examined the differences in speed and accuracy between the two methods — point-and-click and drag-and-drop — in various tasks (Boritz et al. 1991; Gillan et al. 1990; MacKenzie, 1992). However, the decision to select either drag-and-drop or point-and-click depends mostly on the task to be completed. For example, Adesina et al. (2013) used multi-touch drag-and-drop method to solve basic arithmetic problems. Such a method allows the student to drag numbers from the problem and drop them in the solution pad; then by using multiple gestures, the mathematical operation can be computed using the arithmetic operators. The study demonstrated improvements in the students' performance when solving mathematical problems and gave more functionality to the learning process. This means that the editor restricts students from writing SQL statements using the keyboard to avoid any trivial errors such as spelling errors, unnecessary words and synonyms. Furthermore, as it works based on the point-and-click interaction method, it is compatible with different touch screen technology devices (e.g. tablets). These technologies have improved the effectiveness of students' performance in various educational aspects (Bonastre et al., 2006;

Murray & Olcese, 2011; Moran et al., 2010; Adesina et al., 2015). As such, this means that students might find it easier to touch the screen and complete the syntax using tablet devices.

3 DESIGN OF SQL-*FE*

The new SQL formulation editor (SQL-*FE)* is critical to supporting students and improving their performance. It was designed to provide an effective avenue for testing students' SQL statements, as well as to provide quick feedback responses after marking students' SQL statements using the automated system. Figure 1 shows the use case diagram which displays the core functionalities of the SQL-*FE* tool. The use case identifies the primary actors (users) of the SQL-*FE* tool along with the key use cases. Two types of actors use the tool: examiners and students.

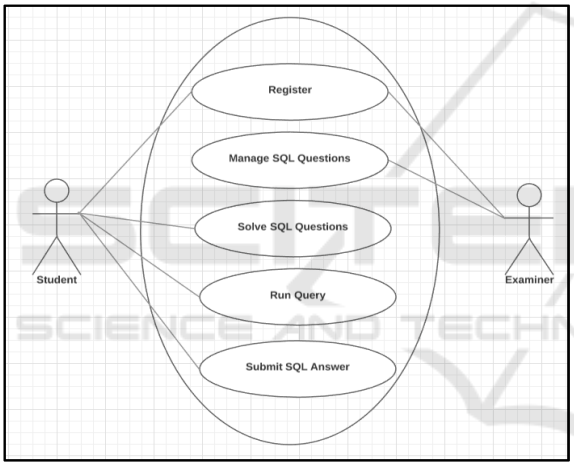

Figure 1: Use case diagram of the SQL-*FE* tool.

In order to enforce proper security, each actor must first register into the editor before he/she can use any of the other functionalities. Registration ensures that a proper email address and password are created for each new user. The two actors —examiners and students — will have different functionalities using the editor. The first step for the examiner is to handle a given SQL assignment by creating and managing the SQL questions.

The examiner will then assign SQL answers for each question, providing alternative ways of solving the same question when applicable. Once the student logs in to the editor, the time count will start automatically for each submitted SQL answer. The student will then solve the SQL questions and try to run them before submitting them for marking.

4 IMPLEMENTATION

SQL-*FE* has been developed to enable students to formulate SQL statements, execute or run the queries and submit the statements for marking.

4.1 Components

The SQL-*FE* user interface contains eight main components illustrated in Figure 2. Component (A) represents the question pane that shows the SQL question scenario and identifies the query requirements needed to solve the SQL statements. The SQL question is placed in the same SQL-*FE* web page making it more convenient for students to solve the SQL statements. In addition, it saves on printed paper normally used for listing the SQL questions manually. Component (B) of the interface consists of the left navigation bar which is composed of two main parts, basic "`SELECT`" clauses and functions. The clauses list assists students while solving the SQL statements. In addition, functions have been added for performing calculations on data. These clauses and functions are placed on the left side of the interface where students can easily find and access them to solve the queries. Element (C) of the interface represents the right navigation bar which consists of reserved SQL keywords used for defining, manipulating and accessing a database. Furthermore, it contains a set of operators used in the "`WHERE`" clause to perform operations such as comparisons and arithmetic functions. Separating the navigation bars into left and right panels provides more vertical space for the main contents such as the SQL questions and the SQL statement answer bars. Component (D) of the interface represents the table schema that displays the table name, field names and their datatype to be used while solving the SQL questions. This means that there is no need for printed paper to display the table schema to the student, as it is ready to view on the web page.

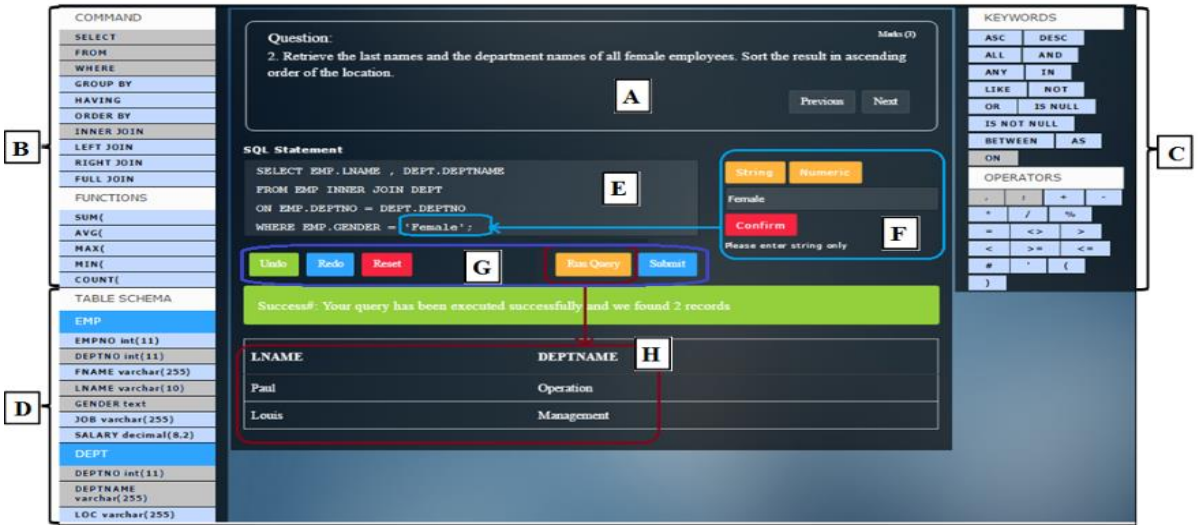

Figure 2: Description of SQL-*FE* user interface.

Component (E) is the SQL answer pane used to enter the SQL answer using the left and right navigation bars. Component (F) represents the text-area pane that helps students to add different numeric or string values to limit the data retrieved, which cannot be done by using the available navigation bars. Component (G) consists of the control buttons which are divided into two categories; one is used to make any amendment in the SQL statements, for example, to redo, undo or reset the SQL statements. The second control buttons are used to deal with the functionality of the SQL statements and include the "`Run Query`" button which shows the SQL result output (indicated by letter (H)) and the "`Submit`" button which saves students' SQL answers for marking.

4.2 SQL-*FE* User Interface

Figure 2 depicts the SQL-*FE* user interface where an SQL answer has been attempted using a point-and-click interaction technique. The figure shows there are four steps to complete an SQL answer using the SQL-*FE* tool. Firstly, the student reads the SQL question and understands the requirements needed to write the SQL statements. Secondly, the student starts pointing and clicking on the SQL clauses and navigation bars to formulate the SQL statement (as illustrated in B, C and D). Thirdly, the student clicks on the "`String`" button to retrieve sting "`Female`" value as the question requested and then clicks on the "`Confirm`" button to insert the string into the SQL statement answer (as illustrated in F and G). The last step is to provide the student with the ability to check the correctness of their SQL statement syntax and query output by clicking on the "`Run Query`" button (as illustrated in G and H).

5 EXPERIMENT

The main objectives of the experiment were to measure the mean time spent and students' performance (grades) by comparing two query formulation tools, SQL-*FE* and SQL Management Studio tool (SSMS) illustrated in Figure 3.

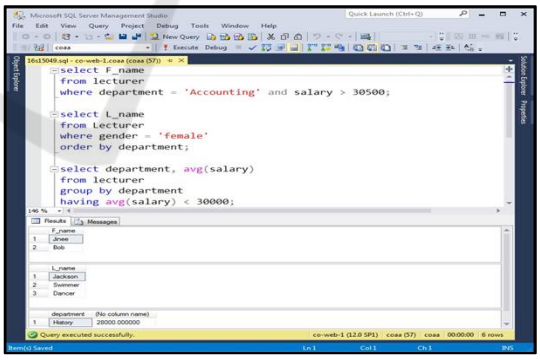

Figure 3: SQL Management Studio (SSMS) user interface.

The SSMS enables users to enter and execute SQL statements to perform calculations, store and retrieve query results.

It shows the execution of SQL statements submitted by participating students, which can run one statement at a time or several statements simultaneously. In order to provide a better understanding of the effect of using SQL-*FE* over

SSMS, the research has identified two questions for the query formulation experiment, which are:
RQ1: Does using SQL-FE during the experiment lead to spending more or less time on solving SQL questions? To investigate if students were spending more or less time to complete the SQL questions.
RQ2: Does using SQL-FE enhance students' performance (i.e. grades)? To investigate if students using SQL-*FE* were achieving higher marks in solving SQL questions than solving them using SQL formulation tools.

5.1 The Experimental Design

A crossover design (also called "change-over design") study is a special form of a controlled double randomised trial (Gardiner and Gettinby, 1998). The randomised nature of the study means that every student has an equal chance of being assigned to the experimental subject on a random basis. This design is more efficient in establishing the highest possible similarity among SQL questions exposed to different tools (Li, 1964). To attain the purpose of the study, a cross-over experimental design has been employed. Table 2 provides a full description of the cross-over experimental design implemented over a two-week time period. In one week, two different sessions took place.

Table 2: Cross-over experimental design.

Group	Tool	Question SET	No. of Participants	Period	Session No.
X	SQL-*FE*	SET A	15	Week I	Session 1.1
Y	SSMS	SET B	15		
W	SQL-*FE*	SET A	15		Session 2.1
Z	SSMS	SET B	15		
X	SSMS	SET A	15	Week II	Session 1.2
Y	SQL-*FE*	SET B	15		
W	SSMS	SET A	15		Session 2.2
Z	SQL-*FE*	SET B	15		

5.1.1 Participants

The experiment involved a total of 60 college undergraduate students in the 20 and 21 years age group.

They were divided into two different experiment days, as each experiment involved 30 students and the number of available PCs in each computer lab was limited as illustrated in Table 2. The students were randomly assigned into two groups. An equal distribution of 15 students used SQL-*FE* and another 15 students used the SSMS tool. This means that there was one experiment in session 1.1 involving 30 students, with 15 students using SQL-*FE* and 15 students using SSMS. Then, a week after, session 1.2 was held and the participants swapped order. The same process was repeated in session 2.1 and 2.2, with a total of 30 students taking part over a two-week time period.

5.1.2 SQL Questions and Model Answers

Each tool used in the experiment was attached to a certain set of questions and alternative methods of solving the queries, as illustrated in Table 3 and Table 4 Basic SQL `SELECT` clauses were included in the experiment such as `SELECT, FROM, WHERE, GROUP BY, HAVING` and `ORDER BY.`

Table 3: SQL questions with model answer: SET A.

Question 1	**Find the first names of all lecturers who work in the accounting department with salaries greater than 30500.**

Model Answer 1	`SELECT F_NAME`
	`FROM LECTURER`
	`WHERE DEPARTMENT='Accounting'`
	`AND SALARY > 30500;`

Output 1	F_NAME Bob Jinee
Question 2	**Retrieve the last names and the course titles of all female lecturers. Sort the result in ascending order of the department.**

Model Answer 2.1	`SELECT L.L_NAME, C.COURSE_TITLE`
	`FROM LECTURER L INNER JOIN COURSE C`
	`ON L.LECT_ID = C.LECT_ID`
	`WHERE L.GENDER = 'Female'`
	`ORDER BY L.DEPARTMENT;`

Model Answer 2.2	`SELECT L.L_NAME, C.COURSE_TITLE`
	`FROM LECTURER L, COURSE C`
	`WHERE L.LECT_ID = C.LECT_ID`
	`AND L.GENDER = 'FEMALE'`
	`ORDER BY L.DEPARTMENT;`

Output 2	L_NAME | COURSE_TITLE Jackson | Principles Of Accounting II Dancer | Pascal Programming
Question 3	**Find the department and average salary of lecturers at each department where the average salary is greater than 35000.**

Model Answer 3	`SELECT DEPARTMENT, AVG(SALARY)`
	`FROM LECTURER`
	`GROUP BY DEPARTMENT`
	`HAVING AVG(SALARY)> 35000;`

Output 3	DEPARTMENT | AVG(SALARY) Computer Science | 39833.3333
Question 4	**Find the title of all courses taught by lecturers in the history department.**

Model Answer 4	`SELECT COURSE_TITLE`
	`FROM COURSE`
	`WHERE LECT_ID IN (SELECT LECT_ID`
	`FROM LECTURER`
	`WHERE DEPARTMENT = 'History');`

	`SELECT C.COURSE_TITLE`
	`FROM COURSE C INNER JOIN LECTURER L`
	`ON L.LECT_ID = C.LECT_ID`
	`WHERE L.DEPARTMENT = 'History';`

Output 4	COURSE_TITLE England History Europe History
Question 5	**Identify the department with the highest average salary.**

Model Answer 5	`SELECT DEPARTMENT, AVG(SALARY)`
	`FROM LECTURER`
	`GROUP BY DEPARTMENT`
	`HAVING AVG(SALARY) >= ALL (SELECT AVG(SALARY)`
	`FROM LECTURER`
	`GROUP BY DEPARTMENT);`

Output 5	DEPARTMENT Computer Science

Table 4: SQL questions with model answer: SET B.

Question 1	**Find the first names of all employees who work as a clerk and earn a salary of more than 2500**

Model Answer 1	`SELECT EMP.FNAME`
	`FROM EMP`
	`WHERE EMP.JOB= 'CLERK'`
	`AND EMP.SALARY > 2500;`

Output 1	FNAME: Jones
Question 2	**Retrieve the last names and the department names of all female employees. Sort the result in ascending order of the location.**

Model Answer 2.1	`SELECT EMP.LNAME, DEPT.DEPTNAME`
	`FROM EMP INNER JOIN DEPT`
	`ON EMP.DEPTNO = DEPT.DEPTNO`
	`WHERE EMP.GENDER='FEMALE'`
	`ORDER BY DEPT.LOC;`

Model Answer 2.2	`SELECT EMP.LNAME, DEPT.DEPTNAME`
	`FROM EMP , DEPT`
	`WHERE EMP.DEPTNO = DEPT.DEPTNO`
	`AND EMP.GENDER = 'FEMALE'`
	`ORDER BY DEPT.LOC;`

Output 2	LNAME: Paul, Louis; DEPTNAME: Operation, Management
Question 3	**Display the various jobs and the average salary of employees in each job where the average salary is greater than 2000.**

Model Answer 3 (Group by & Having Commands)	`SELECT EMP.JOB, AVG(EMP.SALARY)`
	`FROM EMP`
	`GROUP BY EMP.JOB`
	`HAVING AVG(EMP.SALARY)> 2000;`

Output 3	JOB: Manager; AVG(SALARY): 2650
Question 4	**List all department names of all employees who work as a manager.**

Model Answer 4.1	`SELECT DEPT.DEPTNAME`
	`FROM DEPT`
	`WHERE DEPT.DEPTNO IN (SELECT EMP.DEPTNO`
	`FROM EMP`
	`WHERE JOB = 'MANAGER');`

Model Answer 4.2	`SELECT DEPT.DEPTNAME`
	`FROM DEPT INNER JOIN EMP`
	`ON DEPT.DEPTNO = EMP.DEPTNO`
	`WHERE EMP.JOB = 'MANAGER';`

Output 4	DEPTNAME: Accounting, Operation
Question 5	**Identify the job with the lowest average salary.**

Model Answer 5 (Sub-Query >= ALL)	`SELECT EMP.JOB, AVG(SALARY)`
	`FROM EMP`
	`GROUP BY EMP.JOB`
	`HAVING AVG(EMP.SALARY) <= ALL (SELECT AVG(EMP.SALARY)`
	`FROM EMP`
	`GROUP BY EMP.JOB);`

Output 5	JOB: Salesman

6 DATA COLLECTION

The data collected from both tools was saved and dated in different folders to be analysed and evaluated. Once the participants finished solving the SQL questions and made sure they were satisfied by their answers, they were asked to log off (if using SQL-*FE*) to save all their answers. In addition, the examiner and assistants created a shared folder to save all the created files retrieved from the SSMS tool. All participants were asked in the instructions to save the file with their college email address to keep it anonymous.

7 RESULTS AND DISCUSSIONS

The main objective of the evaluation was to measure the participants' performance when using the SQL-*FE* tool over the SSMS tool.

The descriptive statistics of the time taken to complete the test using the two tools of the experiment can be summarised as follows: the SQL-*FE* tool reported an average of M = 20.4 minutes (SD = 7.8) while SSMS reported an average of M = 24.7 minutes (SD = 7.3). As such, the SQL-*FE* tool reported less mean times to complete the test. Figure 4 depicts a box plot of the distribution of time taken to complete the test using each of the two tools. The box plot reports a difference in the distribution of the time taken. However, for both tools, it does not report any abnormal outlier observation indicating that the distribution does not report a large departure from normality, which is an assumption for the validity of the results of the *t*-test.

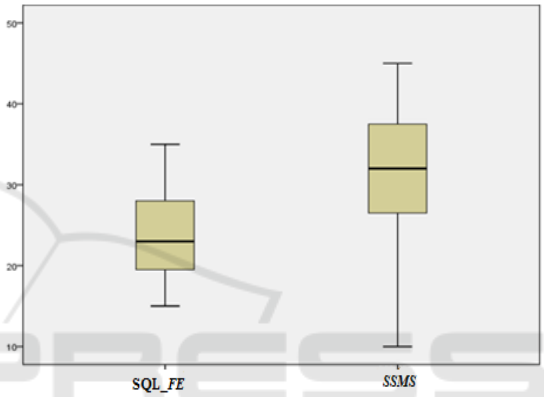

Figure 4: Box plot of the time taken to complete the test using each tool.

This is also supported by histograms of the distribution of time taken to complete the test (Figure 5 and Figure 6) using the SQL-*FE* tool and SSMS tool of test administration.

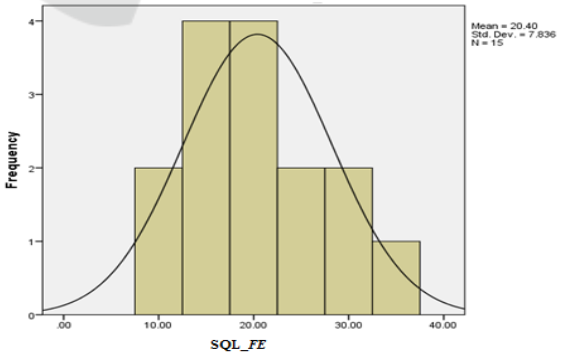

Figure 5: Histogram of the distribution of time taken to complete the test using the SQL-*FE* tool.

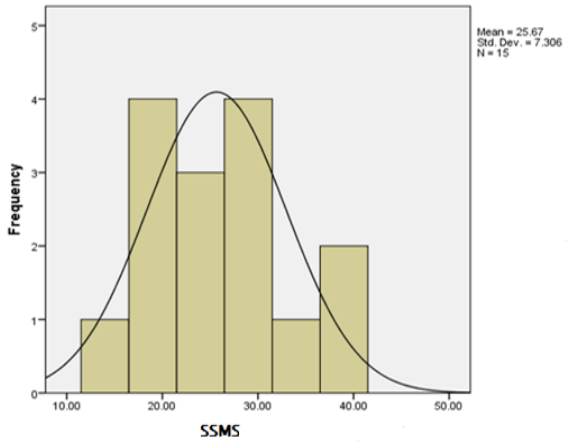

Figure 6: Histogram of the distribution of time taken to complete the test using the SSMS tool.

The SQL-*FE* tool reported less mean time compared to the SSMS tool of test administration. Results of the paired *t*-test indicate that the null hypothesis of no significant difference must be rejected at (0.05) level of significance. This indicates that there is a significant difference in the mean-time taken to complete the test or equivalently, that there is a significant difference in efficiency. Even for one-sided hypothesis (H_1: $\mu_{SQL\text{-}FE} < \mu_{SSMS)}$, the results indicate a significant difference. These results clearly provide strong evidence for the statistical significance of difference (reduction) in the time taken to complete the test between the SQL-*FE* tool and SSMS tool of test administration. That is, the SQL-*FE* test reported significantly higher efficiency compared to the SSMS tool. The descriptive statistics of the mean performance marks using the two tools of the experiment can be summarised as follows: the SQL-*FE* tool reported an average of M = 10.5 marks (SD = 3.1) while SSMS reported an average of M = 8.8 marks (SD = 3.7). As such, SQL-*FE* reported higher marks to complete the test. The null hypothesis is rejected, since $p < 0.05$. In light of this, there is strong evidence ($t= 2.41$, $p= .030$) that formulating the SQL statements using SQL-*FE* improves the participants' marks. In this data set, it improved marks by an average of approximately 2 marks. If the experiment takes other samples of marks, it could get a 'mean paired difference' in marks different from 1.76. This is why it is important to look at the 95% Confidence Interval (95% CI). In this case, the 95% CI ranges from 0.2 to 3.3. This confirms that, although the difference in marks is statistically significant, it is actually relatively small.

Figure 7 presents a box plot of the distribution of performance marks obtained from completing the test using the two tools. The box plot reports a difference in the distribution of performance marks which shows an increase in marks achieved using SQL-*FE*. For the SSMS tool, the figure depicts low marks since the participants had to write all SQL statements, which often led to making more errors.

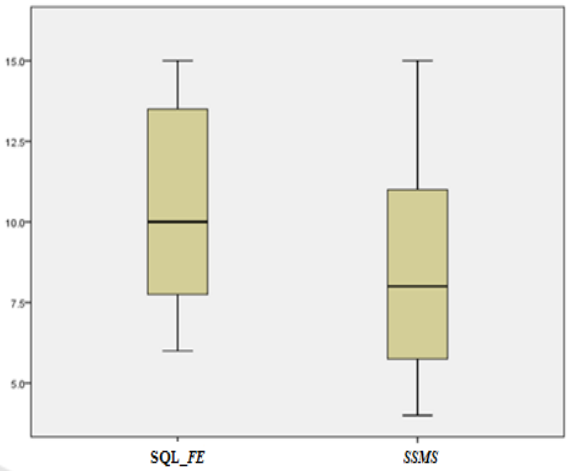

Figure 7: Boxplot of the performance marks obtained using the SQL-*FE* and SSMS tools.

These results clearly indicate that there is a significant difference in the mean performance marks obtained by completing the test using the two tools or equivalently, that there is a significant difference in the participants' performance after formulating the SQL statements using SQL-*FE*. That is the SQL-*FE* test reported significantly higher marks compared to the SSMS tool.

8 LIMITATIONS

There are two main limitations related to the newly implemented SQL-*FE* editor.

Firstly, the student is not allowed to use a keyboard. This made some students confused during their first use of the editor. However, restricting the use of the keyboard fulfils the main objective of avoiding unnecessary elements to the SQL statement syntax. The second drawback is the list of rows of table schema is small, restrictive and too difficult to view. This is because the design of the site has kept the table schema very limited, with lists of columns and data types only.

9 CONCLUSIONS

This paper has investigated the use of a point-and-click method to solve basic SQL statements. The experimental study has demonstrated that students were able to use the newly implemented SQL-*FE* tool.

Furthermore, the tool has minimised the unnecessarily elements that students often add while formulating SQL statements. This resulted in removing the ambiguity in the SQL answers which should support the examiners in understanding the students' level of SQL learning and enable them to provide accurate feedback. The SQL-*FE* editor has answered the two questions of this experiment and confirmed that by using the newly implemented tool, less time is spent formulating SQL statements and students' performance improves, leading to fewer errors and higher grades.

The newly implemented editor has provided students with an easy method of solving SQL statements. However, it should be noted that the experimental study was conducted under two limitations, which can be solved to accommodate the students' requirements.

10 FUTURE WORK

Further implementations will take place utilising a semi-automated assessment of SQL statements to provide partial marking for the submitted statements from the SQL-*FE* tool. This would be considered as second stage of the research, which means the examiners' role will start once students submit their SQL answers, thus ensuring that the answers are ready for marking and commenting by examiners.

ACKNOWLEDGEMENTS

I would like to thank my sponsor the Ministry of Manpower, Sultanate of Oman for their continuing support and motivation.

REFERENCES

Adesina, A. et al., 2013. Use of multi-touch gestures for capturing solution steps in arithmetic word problems. , pp.6–8.

Bobak, A.R., 1996. Distributed and Multi-Database Systems 2nd ed., Artech House,INC.

Bonastre, O., Benavent, A. & Belmonte, F., 2006. Pedagogical Use of Tablet PC for Active and Collaborative Learning. In *2006 IEEE International Professional Communication Conference*. IEEE, pp. 214–218.

Boritz, J., Booth, K.S. & Cowan, W.B., 1991. Fitts's Law Studies of Directional Mouse Movement. , pp.216–223.

Carter, J. et al., 2003. How shall we assess this? In *ACM SIGCSE Bulletin*. ACM, pp. 107–123.

Gardiner, W.P. & Gettinby, G., 1998. *Experimental Design Techniques in Statistical Practice: A Practical Software-Based Approach*, Elsevier Science.

Gillan, D.J. et al., 1990. How does Fitts' law fit pointing and dragging? In *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. New York, New York, USA: ACM Press, pp. 227–234.

Ihantola, P. et al., 2010. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. ACM, pp. 86–93.

Li, C.C., 1964. *Introduction to experimental statistics*, McGraw-Hill.

MacKenzie, I.S., 1992. Fitts' law as a research and design tool in human-computer interaction. , 7, pp.91–139.

Moran, M., Hawkes, M. & El Gayar, O., 2010. Tablet Personal Computer Integration in Higher Education: Applying the Unified Theory of Acceptance and Use Technology Model to Understand Supporting Factors. *Journal of Educational Computing Research*, 42(1), pp.79–101.

Murray, O.T. & Olcese, N.R., 2011. Teaching and Learning with iPads, Ready or Not? *TechTrends*, 55(6), pp.42–48.

Peat, M. & Franklin, S., 2002. Use of online and offline formative and summative assessment opportunities: have they had any impact on student learning? In *ASCILITE*. pp. 505–513.

Renaud, K. & van Biljon, J., 2004. Teaching SQL—Which Pedagogical Horse for This Course? In Key Technologies for Data Management. Springer, pp. 244–256.

Woit, D. & Mason, D., 2003. Effectiveness of online assessment. *ACM SIGCSE Bulletin*, 35(1), p.137.