Novice Perceptions on Effective Elements of PostgreSQL Error Messages

Novice Perceptions on Effective Elements of PostgreSQL Error Messages

TONI TAIPALUS, Tampere University, Finland
HILKKA GRAHN, SAIMA RITONUMMI, and VALTTERI SIITONEN, University of Jyväskylä, Finland
TERO VARTIAINEN, University of Vaasa, Finland
DENIS ZHIDKIKH, University of Jyväskylä, Finland

SQL compiler error messages are the primary way users receive feedback when they encounter syntax errors or other issues in their SQL queries. Effective error messages can enhance the user experience by providing clear, informative, and actionable feedback. Despite the age of SQL compilers, it still remains largely unclear what contributes to an effective SQL error message. With 2,052 answers yielded by 165 participants for qualitative analysis, this study is an attempt to understand what novices perceive as effective elements in SQL error messages. The results uniformly indicate that communicating the precise error position, articulating what is wrong in the query with clear natural language, and showing hints on how to fix the error are perceived as the most effective elements for error recovery. These insights have potential to be utilized in providing more effective error messages in SQL compilers and SQL learning environments, and for guiding generative artificial intelligence for enhanced error messages in order to minimize frustration caused by cryptic error messages, improving learning and adoption, and reducing debugging time.

CCS Concepts: • **Applied computing** → **Education**; • **Information systems** → **Query languages**; • **Human-centered computing** → *Empirical studies in HCI*.

Additional Key Words and Phrases: SQL, error, error message, effective, relational database, novice, software development, PostgreSQL

1 INTRODUCTION

Software developers spend a significant amount of time interpreting error messages [4]. Several studies have shown the importance of the development environment's role in improving the user experience [14], reducing debugging time [31], facilitating student motivation [11], and increasing user confidence [42]. Despite this practical importance, error messages are still perceived as a source of much confusion and frustration [6, 26]. This state of error messages, especially regarding programming language compiler error messages, is arguably not due to a lack of scientific research. In fact, a recent literature review summarized hundreds of studies on programming language compiler error messages [6], concluding that the effects of enhancing error messages on error recovery are still inconclusive.

Despite the age of SQL and the compilers of different database management systems (DBMS), and the fact that SQL is still the *de facto* language for data management, taught in effectively all information technology curricula in higher education [22, 46, 47], the number of studies on SQL error messages is scarce. However, some studies have shown that

there are differences in the effectiveness of different SQL error message designs [42, 44], which arguably warrants further investigation into the elements that make some SQL error messages more effective than others.

A recent study [44] compared the effectiveness of SQL error messages of four popular relational DBMSs: MySQL, Oracle Database, PostgreSQL, and SQL Server by several factors. The results indicated that PostgreSQL error messages facilitated *somewhat* higher rates of error fixing success than those of MySQL, and that study participants felt that PostgreSQL error messages were more helpful in error fixing than those of MySQL and Oracle Database. Other differences in error message effectiveness between the DBMSs were not statistically significant or uniform. A recent mixed-methods study [41] analyzed the results of eight relational DBMSs collectively rather than individually. While this approach provides a broader understanding of general trends and patterns, examining a single DBMS in isolation allows for deeper insights into system-specific behaviors that might be obscured in an aggregate analysis. Such a focused approach can help refine database-specific recommendations and contribute to a more nuanced understanding of error message design and usability. For these reasons, we chose to investigate effective elements in PostgreSQL error messages with the following research question:

RQ: *What error message elements are perceived to contribute to an effective SQL error message?*

To address this research question, we collected data from 165 query formulation novices who had received formal training in SQL and the fundamentals of relational databases. Our primary reason behind studying query formulation novices is that beginners such as students arguably need more support in query writing than experts, and benefit more from effective error messages. The participants were shown sixteen different erroneous SQL queries and respective error messages, and were asked to both fix the errors, and describe which elements in the error messages facilitated error recovery. Therefore, our study focuses specifically to PostgreSQL error messages in a controlled setting where the participants do not engage in a feedback loop with the PostgreSQL query compiler, but rather fix pre-determined SQL errors and describe their experiences with the error messages.

The results indicate, among other findings, that novices value that the error message *(i)* communicates the position of the error, which is done in PostgreSQL via a line number as well as with a caret symbol pointing to the precise error position on the erroneous line, *(ii)* clearly communicates what is wrong, i.e., provides an explanation on what principle is violated, and *(iii)* contains a hint on how the error could be fixed or speculates what the user might want to accomplish. We detail the effective elements of each of the sixteen queries, as well as discuss which error message elements the participants deemed detrimental to efficient error recovery. These results are applicable in designing SQL error messages for DBMS compilers, for SQL learning environments which implement custom error messages, and for guiding language model enhanced error message generation.

The rest of this study is structured as follows. In the next section, we describe the error recovery process in general, prior works on compiler error messages as well as works specific to SQL error messages. In Section 3, we detail the participants, our data collection instrument and pilot study, and data analysis. Section 4 and Appendix A contain the results, and Section 5 the implications of our findings as well as threats to validity. Section 6 concludes the study.

2 BACKGROUND

2.1 Error Recovery Process

Error messages are one of the primary methods an end-user communicates with software. Error messages may be general system messages [39] displayed by operating systems, web pages, or applications, or error messages output by computer language compilers [6]. Both of these types of messages serve different end-users: some are intended for and

read by software professionals, while others serve all types of users. When an end-user encounters a system message, an error recovery feedback loop begins [50, 52]. This loop is different for compiler error messages and general system messages. In general system messages, the end-user has relatively limited options provided by the user interface, while with compiler error messages, it is typically up to the developer to recover from the error.

From a software developer perspective, error recovery consists of three phases [50, 52]. First, error *detection* refers to the act of understanding that the piece of software code contains an error. In terms of syntax errors, this is often communicated to the developer by the compiler via an error message. In relational DBMSs, the syntax of SQL statements is checked by the query parser [17], a software component responsible for determining aspects such as syntactical validity as well as access rights to different database objects. Second phase, *explaining* refers to the developer trying to interpret the error message and find the error. As SQL statements, especially transaction processing statements, are often relatively short compared to units of execution in programming languages, finding the erroneous part arguably plays a less significant role in SQL than in programming languages. In the final phase of error *fixing*, the developer attempts to modify the software code to fix the error. As this process of submitting a fix and receiving output from the compiler typically repeats until the error is fixed, this process is often called a feedback loop between the developer and the compiler. From a human perspective, the compiler is largely a *black box* that determines whether the feedback loop ends or continues. Consequently, it is only natural that especially novices view the compiler as an unerring authority [42] that has the final say whether the query is erroneous or not.

2.2 Error Messages

In contrast to programming language compiler error message research, SQL error messages have received little scientific attention. Programming language compiler error messages have been studied extensively [6] with languages such as Java, C# and Python. Scholars and educators have focused on topics such as understanding the most common logical [15] and syntactical errors [10], student difficulties with error messages [1, 30], proposing tools to facilitate error recovery [2, 27, 29, 33], and enhancing compiler error messages [3, 8, 9]. Especially enhancing programming language compiler error messages has been seen as a vital research goal, and there have been several proposals for error message redesign guidelines [48], as well as attempts to study if and how enhanced error messages facilitate error recovery. However, while some studies have shown success in enhancing error messages [e.g., 5, 6], other studies have implied inconclusive or negative results [e.g., 32, 53].

Several authors have proposed guidelines for more effective, user-centered error messages, many of which are based on expert opinion. For example, a study on programming language compilers published in 1974 proposed guidelines such as *be as specific as possible*, *be user-directed* and *be source-oriented*, the latter two meaning that the error message should focus on what the user has done instead of what the compiler has done, and this should be communicated in regards to the user's source code instead of the compiler's source code [18, p.543]. In 1982, a study on more general *system messages* proposed rather similar guidelines such as *be brief* and *be comprehensive* [39]. Guidelines along similar lines have also been proposed later [cf. e.g., 48]. More recently, scholars have also focused on more nuanced aspects such as error message readability [7, 12, 13]. Additionally, generative AI tools have been proposed in enhancing error messages of programming language compilers [28], yet the results in educational contexts have at least initially been minor or even detrimental [25].

2.3 SQL Error Messages

For a programming language, there is often a limited selection of compilers available. In contrast, the SQL language is defined by the SQL Standard [20, 21], which is in turn implemented by a myriad of DBMSs (such as PostgreSQL) or storage engines (such as InnoDB). Each implementation is different regarding how query syntax is processed and what is considered a syntax error [34], as well as which keywords are available and how they work [40]. Consequently, effectively all DBMSs have different SQL error messages.

In late 1970s and early 1980s, a set of studies proposed that the DBMS should provide at least some lexical support in query writing such as a synonym dictionary for database objects such as table and column names [35–37]. More recently, studies have shown that different SQL error messages seem to have an effect on syntax error fixing success and the user's error recovery confidence in both older relational DBMSs [44] as well more novel NewSQL systems [42].

Despite scientific efforts towards enhancing programming language compiler error messages, it still remains largely unclear what contributes towards making an SQL error message effective. One study on SQL error messages indicated that the general system message guidelines [39] poorly explain error fixing success in SQL query formulation [41]. The same study – in contrast to many guidelines based on expert opinion [e.g., 18, 39, 48] – formulated a framework for SQL error message design based on empirical data. The framework consists of nine guidelines such as *explain why the error occurs* and *provide working examples of similar query concepts*. Despite the fact that the framework was based on empirical, qualitative data, the effects of designing SQL error messages based on the framework have shown that some of the guidelines proposed by the framework facilitate more successful error recovery, while other guidelines might be detrimental to error recovery [43]. In other words, what query writers *think would help them* in error recovery seems to be partly separated from *what actually helps them* in fixing errors successfully. On the other hand, at least one study [42] has shown correlation between perceived error message usefulness and effective usefulness.

Given the prior body of knowledge, especially the mixed-method study on error messages in different DBMSs [41], our work makes the following contributions. First, we provide insights specific to PostgreSQL, as opposed to a more general approach (eight different DBMS). Second, our analysis is based on a large qualitative dataset of 165 participants, as opposed to fewer observations (approximately 40 observations for PostgreSQL). Third, this study provides scientific evidence potentially supporting or contesting the more general insights provided by earlier studies.

3 METHOD

3.1 Participants

We recruited the study participants from an undergraduate database and data management course given at the university of the second author. The students majored in computer science, software engineering, or information systems. The course outline consisted of conceptual modeling, database normalization, database distribution, data warehousing, and practical SQL exercises with SQLite. In addition to course material such as lectures and exercises, students could use any resourse at their disposal during the course. At the end of the course, the students were asked to answer our survey. Answering was voluntary, and carried no advantages or disadvantages. Prior to answering, the students were shown a full privacy statement detailing how the data they submit will be used in our analysis. A total of 165 students (response rate 81%) chose to participate. As we wanted to focus on the technical aspects of SQL error messages, we chose not to complicate or lengthen our data collection with demographic data.

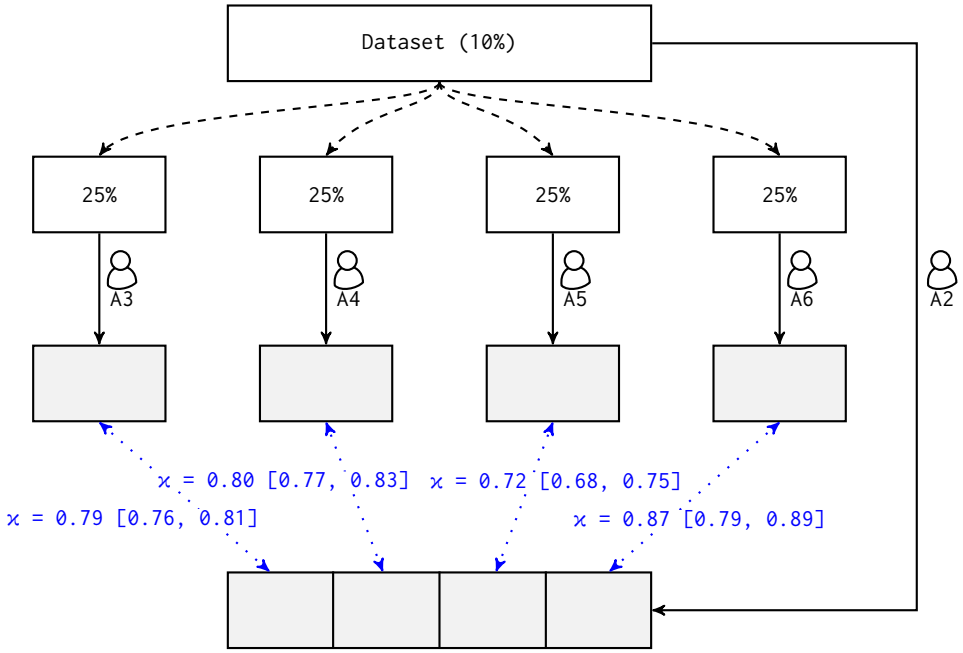

Fig. 1. Initial data analysis steps and authors (A) who participated in the analysis steps; a white rectangle represents raw data, a gray rectangle an analysed dataset, a solid line analysis, a dashed line division of data, and a blue dotted line comparison of analysed datasets to determine inter-rater agreement; Cohen's kappa (κ) and 95% confidence intervals are presented along with the dotted lines

3.2 Data Collection Instrument

A prior study reported several data collection instruments for assessing the SQL error message qualities [44]. Since the same study reported that PostgreSQL error messages are, with a small margin, the most effective by several metrics, we chose to utilize the instrument for PostgreSQL error messages to understand *what* makes them effective. The data collection instrument consists of sixteen SQL queries (or *tests)*, each of which contains a different syntax error, and the corresponding error message output by PostgreSQL. Each of the sixteen queries embody one of the sixteen most frequent SQL syntax errors identified by an earlier study [45].

All participants were shown each SQL query, one at the time, and asked to fix the error, and to describe *what elements make the error message effective*, given the purpose of the query. Both questions were answered in free-form input fields. In some cases, participants were verbose on what makes the error message effective, providing several considerations. In some cases, the participants left their answer blank. The data collection instrument is available as supplementary online material *S3*[1] of a prior study [44]. Each test in the data collection instrument consists of a database schema diagram, a task in natural language, a corresponding yet syntactically erroneous SQL query, a resulting error message, and questions for the participant to answer. We modified this instrument by removing the four control questions and three Likert questions, and adding the free-form field for describing the effective elements in the error message. The instrument has not been systematically validated.

[1]https://ars.els-cdn.com/content/image/1-s2.0-S016412122100131X-mmc3.pdf

Table 1. The number of mentions of elements in all 16 error messages which facilitated or complicated error recovery; the theoretical maximum number of occurrences of one element is 2,640 (16 queries × 165 participants), although not all error messages contain all the elements

	Facilitating element		Complicating element
813 (40%)	Shows error position.	180 (9%)	Does not tell what is wrong.
526 (26%)	Tells what is wrong.	179 (9%)	Is not written in clear English.
184 (9%)	Shows the line number.	169 (8%)	Does not provide an example.
130 (6%)	Is descriptive.	104 (5%)	Does not show exact error position.
129 (6%)	Contains a hint.	30 (1%)	Does not mention other errors in the query.
92 (4%)	Tells how to fix the error.	28 (1%)	Describes an error that is not relevant in this context.

3.3 Pilot Study

We conducted a pilot study with 25 participants. These 25 participants belonged to an earlier student cohort as the 165 participants in the study proper. The pilot study was conducted with the same protocol as the subsequent study proper. All the 25 participants answered the sixteen questions provided in the data collection instrument, yielding a total of 400 answers for qualitative analysis. The pilot study data were analysed by the first two authors by utilizing conventional content analysis [19], according to which the data are analysed without preconceived categories, and the categories and their names are conveyed from the data. The first two authors analyzed the data by discussing each of the 400 answers, solving discrepancies, formulating categories of explanations, and merging and splitting categories to achieve a certain level of abstraction. Nine explanations for what elements make PostgreSQL error messages effective were found: the error message *(i)* shows the position of the error, *(ii)* tells what is wrong with the query, *(iii)* speculates on what the user might want to accomplish, *(iv)* is written in common English, *(v)* contains an example, *(vi)* tells how to fix the query, *(vii)* contains a hint on what the user might want to accomplish, *(viii)* is descriptive, and *(ix)* has a line number that shows the position of the error. These explanations are not listed in any particular order. It is worth noting that there is some level of overlap in these explanations, e.g., in *(i)* and *(ix)*. This is due to different levels of abstraction different participants chose. The overlap of these particular explanations might be due to the fact that PostgreSQL indicates the error position with both line numbers as well as a caret symbol (cf. Figures in Appendix A).

3.4 Data Analysis

In addition for preliminary testing of the research setting, the nine explanations from the pilot study provided a starting point for the analysis of the answers yielded by the 165 study participants. We analysed the data by utilizing directed content analysis [19], using the nine explanations as initial codes. When an answer that did not fit any of the explanations was encountered, a new code and explanation was formulated. This was done in three phases. First, we selected 10% of the dataset. This subset was divided into four parts, and each part was independently coded by one author (authors A3..A6, cf. Fig. 1). Second, another author (A2) independently coded the same 10% of the dataset, and the first author (A1) compared the codes of the second author (A2) and the rest of the authors (A3..A6) to determine inter-rater agreement to assess the portion of subjective interpretations with Cohen's kappa ($\varkappa$). Inter-rater agreement was observed to be high ($\varkappa$ being between 0.72 and 0.87), possibly due to the brevity of the answers. Next, authors A3..A6 each proceeded to code the rest of the answers for error messages assigned to them. Finally, authors A1 and A2 convened to discuss whether new codes created independently by authors A3..A6 were similar enough to be merged. We coded a total of 2,052 responses.

Table 2. Selected (Q)uotations with (P)articipant and (E)rror (M)essage identifiers; the error messages are listed in Appendix A

Q#	P#	EM#	Quotation
1	12	T01	The error message specifically stated what type of error it was and from which line it could be found. There is not *too* much data in error messages in this scale... so it probably wouldn't hurt to list found redundancies, e.g., "column 'name' in table Product, 'name' in table Supplier..." etc.
2	101	T01	For a novice SQL user, error messages don't really tell anything. Every error message is checked using the famous Google. The situation would surely be different if I had more experience with SQL.
3	153	T02	Useful was the reference to the occurrence of the error, a general description of the nature of the error, and detailed information about the column causing the error in this case. Admittedly, the error message did not direct the user straight to the correct solution because, in this case, the user was not supposed to compare the values of two columns but one column value to a fixed string. However, since both use cases are possible, the compiler cannot, in my opinion, infer which use case the user has in mind in a general case. Therefore, I don't think there's a reason to try to improve the error message in this regard. The compiler seems to stop processing when encountering the first error, so the equivalent error hidden in the second operand of the `OR` operator may go unnoticed by the user, so the error message could possibly be improved by searching for and printing other similar errors.
4	99	T03	Useful was that I knew where the error was and that the statement was correct up to that point. However, this leads me to act in a way that I check up to that point, try again, and move on to the next error... so the end might be incorrect even after fixing the error, which confuses me.
5	120	T04	Pointed to the right place but misled about the cause of the error. In my opinion, the error message should not assume the cause of the error if it cannot be entirely certain about it.
6	43	T05	Again, the same as before – the error message points out the wrong position, which may be misleading at first glance. However, the crucial aspect is that the error message indicates on which *line* the error is, as it is easy to start investigating from there.
7	12	T07	A quite perfect error message when the description of the error was accurate, and a hint was provided on an appropriate way to fix the error.
8	117	T07	In my opinion, this is an exemplary error message that provides both necessary general and detailed information for fixing the error, along with clear instructions for correction.
9	162	T07	By far the most useful error message so far. The error message was so helpful that it would solve the problem even for someone who is not very familiar with SQL.
10	76	T08	Maybe it pointed to the right line, but placing the error at the `LIKE` part made finding the actual error unnecessarily difficult.
11	3	T08	The error message indicates the location of the error. However, since the error message reads that the error is near the `LIKE` part, and line 3 presents the entire statement, one might not necessarily pay attention to the incorrectly written `WHERE` keyword. In my opinion, it would be better if the error directly referred to the `WHERE` keyword.
12	102	T09	The error message led me completely to wrong tracks... I started looking into why the system would treat email as a boolean. However, this time the location of the error was correct, which was a plus.
13	66	T11	The 'Error' statement precisely indicates where the issue is, but on the other hand, the caret now slightly misleadingly points to the wrong line. As an improvement suggestion, another arrow pointing to the `GROUP BY` line.
14	90	T13	The error message does not tell what's wrong in the syntax, as the `WHERE` clause was written correctly. The message could suggest using `AND` because there were multiple `WHERE`s in a row.
15	22	T15	The error was easily found, but there was no hint for correction. For example, "Did you mean p.price_usd?" might not be entirely impossible as an error message.

4 RESULTS

During the analysis, we observed that many participants, in addition to describing the elements contributing to effective error messages, also pointed out elements that complicated error recovery. Due to the large number of these mentions in the data, we have also listed elements complicating error recovery in Table 1, although several of them are simply a facilitating element phrased differently. The numbers of occurrences of the elements are closely related to the sixteen queries described in Section 3.2. For example, the number of occurrences of the facilitating element *error message contains a hint* would arguably be greater if more than three of the queries contained a hint. Therefore, Table 1 merely provides an overview of the facilitating and complicating elements of error messages, and for more in-depth analysis, please refer to Appendix A. For example, simply providing a hint is not intrinsically helpful, but the hint must also be phrased correctly. Only the six most mentioned elements are listed in Table 1, as the numbers of occurrences of other elements were less than one percent.

Query-by-query insights on the facilitating elements are detailed in Appendix A due to their length. Most of the effective elements in the sixteen error messages were uniform. The participants appreciated, above all, that the error message shows the position of the error (with a line number or more accurately, with a caret symbol), tells what is wrong in the query, is descriptive, and helps the query writer fix the error by providing hints or more straightforward suggestions. In contrast, if the error message failed to indicate what is erroneous or was worded ambiguously, the error message was perceived to hinder error recovery.

The responses of the participants were overall short, typically consisting of one or two short sentences. Table 2 contains selected quotations based on the insights gained during the coding process. We deemed these quotations particularly representative, insightful, or detailed enough to include. For example, quotations #1, #7, #8, #9, and #15 commended the error message for giving a hint, or suggested adding a hint if one was missing. Quotations #4 and #5 commended the error message for pointing specifically to the erroneous position, while quotations #6, #10, #11, and #13 criticized the error message for being misleading about the error position. Quotation #3 goes into detail on several positive and negative aspects of the error message, as well as discusses how the compiler perhaps worked in this error recovery situation. Quotation #2 expresses frustration about the generally obscure nature of the error messages, and quotation #14 criticizes the message for not communicating that the problem was a duplicate clause. The quotations in Table 2 also show the dual nature of the responses, as some participants discussed elements that were helpful, some elements that were detrimental to error recovery, and others discussed both.

Overall, the participants valued error messages which clearly communicated the position of the error. Additionally, hints in error messages were appreciated *when they were useful*. For example, the error message for query *T07* (Fig. 8) seems to be the archetype of an effective PostgreSQL error message. The hint in this error message ("HINT: Perhaps you meant to reference the column "employee.fname" or the column "employee.sname"".) was deemed effective by 34 participants (over 20%), while the hint in the error message of query *T04* (Fig. 5, "HINT: No operator matches the given name and argument types. You might need to add explicit type casts.") was not deemed effective by any of the participants. On the other hand, many participants were disappointed or even frustrated when the error message incorrectly identified the error position or provided misleading suggestions.

5 DISCUSSION

5.1 General Discussion

It has been hypothesized that especially self-reliant students merely read the line number from error messages [38]. If students can fix the errors without further consulting the error message, choosing not to read the error message further should not be problematic. It has also been proposed to encourage programming students to deliberately formulate erroneous code in order to engage more with error messages [49]. Although this approach appears productive in terms of learning, we believe it is justified to speculate whether novices should learn both the computer language *and* the language of the compiler, i.e., how a compiler communicates and what the messages mean. An earlier study on SQL error messages showed a quote from a participant, reading "Perhaps the message is technically the correct way to describe the error, but from a human perspective, this seems incomprehensible." [41]. In this regard, some error messages can make sense only if the reader understands how the compiler processes information. Arguably, this approach is easier for the developer designing the error messages, but unnecessary, since the compiler does not need to understand the error message, only output it. The following cynical quotation from one of our participants, "The best thing about error messages is that you can copy-paste them into Google," somewhat highlights the fact that some students simply skip trying to interpret the error message themselves, as some error messages are only useful when complemented with online search engines. Teaching students how a DBMS compiler interprets queries does not seem feasible, since all DBMS compilers work differently, and output different error messages for the same syntax error. For several DBMS vendors, the detailed documentation of DBMS internals (compiler included) is not publicly available.

Our interpretation is that the participants acknowledged that error messages are often problematic in their wordings and content. Possibly for this reason, many participants valued that the error messages accurately pinpointed the erroneous part of the query. In PostgreSQL error messages, the caret symbol seems to be the single most useful element in the error message, although at least one participant did not understand what the caret is supposed to represent. Given the simple and short nature of the erroneous queries in the test suite, and the fact that pinpointing the error position was deemed the single most effective element, it seems reasonable to argue that long SQL statements which can span dozens or even hundreds of lines of code will benefit even more when the error position is clearly shown in the error message.

Several participants pointed out in their responses (e.g., quotations #5, #6, #10, #11, #12 and #13 in Table 2) that the error message provided an incorrect hint or confused them in some other way. As can be seen in the quotations, such discrepancies in what the respondents know is incorrect and what the SQL compiler deems incorrect is a source of frustration, at least among novices. These findings are in line with programming language error message studies [48, 51]. It remains unclear whether misleading hints caused lower success rates in query fixing. While our research setting has confounding factors (e.g., differences in query complexity) that make determining this relationship unreliable, this remains an interesting future research topic.

Overall, our findings are, with some exceptions, in line with what was suggested in a closely related, earlier study [41]. However, due to the fact that our study was only concerned with PostgreSQL, and this previous study [41] with the error messages of eight DBMSs, the quantities of mentions of different effective or detrimental elements cannot be compared. That is, some error message problems identified in the earlier study are not applicable to PostgreSQL error messages. Despite the different research questions and focus of these two studies, the results presented in our study partly validate the results presented earlier [41], as many of the suggestions for error message improvements are in line with the elements deemed effective in our study.

Table 3. A summary of practical implications of the results of this study divided into education and software industry

	Practical implication
Education	Teaching students how to read error messages can now focus on the most effective elements of the message.
	Novices may now focus on the most effective elements when debugging queries.
	By focusing on systems with clear error messages, student frustration can potentially be reduced.
	Error messages with the identified elements may be implemented in SQL learning environments.
Industry	SQL compiler feedback can be redesigned to focus on the effective elements identified.
	SQL error message wording and structure can be iterated based on the findings.
	Error messages that are effective potentially reduce time spent on debugging.
	By implementing user-centered error messages, DBMS vendors can differentiate their products in the market.

5.2 Practical Implications

There are several potential implications of redesigning error messages, summarized in Table 3. SQL compiler error messages are the primary way users receive feedback when there is a syntax error in their SQL queries. Thus, as the results of this study showed, it is important that effective messages include clear ("shows error position"), informative ("tells what is wrong"), and actionable ("tells how to fix the error") feedback to enhance the user experience and reduce user frustration. Effective error messages can potentially reduce the time it takes for users, including database administrators and developers, to locate and fix SQL errors. This can, in turn, lead to more efficient troubleshooting and software development. Especially for individuals new to SQL, well-crafted error messages can be valuable learning tools, as they can provide insights into SQL syntax and help users understand the language better, potentially promoting the adoption of SQL or a particular DBMS. Furthermore, database support and maintenance can be costly. More effective error messages might help users resolve common issues independently, which could reduce the need for extensive technical support, and help more casual SQL users write queries easier.

An earlier study [41] analyzed the SQL error messages of eight relational DBMSs and participant answers on how to *improve* the error messages. The findings of that study suggest that the error message should *(i)* not contain unnecessary information such as error codes or environmental variables (PostgreSQL does not report either), *(ii)* mention all syntax errors in the query (PostgreSQL mentions only the first error encountered, yet SQL Server tries to mention several), *(iv)* specify the position of the error (PostgreSQL does this with both the line number as well as the caret symbol), and *(v)* describe what causes the error and how the error can be fixed.

An earlier study remarks that many SQL compilers have qualities that others do not [41]. These qualities can either facilitate or hinder error recovery, and the presence or absence of these qualities does not seem to be tied to the age of the compiler [41]. That is, it seems that the implementation of human-computer interaction practices are not necessarily taken into account in compilers developed in the 1980s, nor in the 2010s. In summary, many DBMSs have effective error messages, but none of the eight DBMSs studied before [41] seem to implement all beneficial qualities. If we examine the error message qualities of eight DBMSs in light of the findings yielded by this study, we can observe that line numbers are provided by SQL Server and PostgreSQL, and sometimes by MySQL (InnoDB), SingleStore (InnoDB), and NuoDB, while Oracle Database, CockroachDB and VoltDB provide no line numbers in their error messages. However, CockroachDB sometimes implements the caret symbol pointing to the erroneous position, along with PostgreSQL and NuoDB. Providing merely a line number can be problematic if the line of SQL code is long, as there are no technical limitations on the length of lines of code. On the other hand, providing merely the caret symbol may be problematic, if

there are similar lines of code in the same, lengthy SQL statement. Only PostgreSQL and VoltDB seem to provide hints on how to fix the error, but only for some syntax errors.

DBMS vendors have been seemingly reluctant in revising and redesigning SQL compiler error messages [40]. Consequently, we are not particularly hopeful that the results yielded by this study will change that. However, as different language models have shown much promise in both natural language to SQL translation [16, 23], interpreting and correcting strange and cryptic error messages seems arguably a goal at least partially achievable in the near future. We are currently investigating how to incorporate small, domain-specific language models into learning environments and as plugins to DBMSs to enrich and restructure DBMS error messages, similarly to tools such as *pg4n*[2]. Rather than perhaps using language models to formulate yet another set of cryptic error messages, we hope that the results attained here can guide the instructions on which elements the refactored error messages should and should not contain. An example of a practical application would be to send an input to a generative artificial intelligence service which uses a language model, and use its output as an iterated error message [cf. e.g., 24]. The input could consist of the original, erroneous query, the original syntax error output by the DBMS, database structure which can be queried from DBMS metadata, and natural language instructions following the insights uncovered in this study, e.g., "show the position of the error in the query, tell why the query is erroneous, and show hints on how the error could be fixed, but do not fix the error".

5.3 Threats to Validity

The dataset analysed in this study was collected from one student cohort, from one university, which poses a threat to validity regarding the generalizability of the results. Despite this one-cohort limitation, the coding of the dataset yielded by the pilot study (i.e., a different cohort) indicates that largely similar observations were made on the effectiveness of the error messages regardless of student cohort. However, the potential effects of e.g., culture and teaching style related differences remain open questions due to the one-university aspect of the dataset.

In this study, we considered only error messages output by PostgreSQL. Therefore, it is intrinsic that the participants focused on the elements that they perceived effective in the error messages *of PostgreSQL*. It is possible, even likely, that there are other DBMSs that have effective elements in their respective error messages, which are not present in PostgreSQL's error messages. However, we wanted to limit the scope of this study to PostgreSQL, which has been shown to have relatively effective error messages in terms of error fixing success rate [44]. A potential future research avenue would be to find out whether other DBMSs have different effective elements. One earlier study has briefly discussed the elements in different DBMS error messages without investigating which of these elements are effective [41].

The data collection instrument used in this study introduces an unnatural environment. While it is common that software developers read and interpret program code (SQL included) written by others, it is also common that learners write their program code from scratch. The data collection instrument had erroneous SQL queries which the participants were instructed to fix. These erroneous queries were not written by the participants, but given as-is by the data collection instrument. Furthermore, the data collection instrument did not initiate an error recovery feedback loop, because the participants were not informed whether they fixed the erroneous query correctly. We considered using a more free-form data collection environment instead of the survey instrument, but deemed the potential confounding variables larger threats to validity. For example, if we did not provide the erroneous queries, but let the participants commit the syntax

[2]https://pypi.org/project/pg4n

errors themselves, we would have risked that the participants would not have committed syntax errors, or committed syntax errors which would not have invoked different PostgreSQL error messages for the participants to describe. In summary, our research setting focused our participant responses to technical aspects we wanted to study, while introducing an unnatural error recovery environment.

Finally, it is worth noting that the results of this study, while empirical findings, are still subjective opinions of the study participants. We did not study if the elements found effective by the participants indeed contribute to, e.g., higher success rates in error recovery or less time taken to fix SQL syntax errors. However, it has been shown that subjective perceptions on what is useful in SQL error messages correlate with objective metrics for success in SQL error recovery [42]. An interesting future research topic would be to divide study participants into groups, where one group is shown an error message with a selected element, the other one without this element, and then compare success rates or fix time in error recovery to understand whether a particular element is indeed effective.

The interpretation of Cohen's $\varkappa$ depends on the context of the data and the field of study. In this study, the observed values of $\varkappa$ were high. As inter-rater agreement is context-dependent, even low values would arguably not have posed significant threats to validity, as the goal of our study was to *qualitatively* assess error message qualities, and the number of occurrences of specific mentions of each effective elements play a smaller role in our analysis.

6 CONCLUSION

SQL error messages are the primary way of communication between software developers and SQL compilers in error recovery situations. Despite the prevalence of SQL, many DBMSs have cryptic and unhelpful error messages. Some studies have shown that PostgreSQL has some of the more helpful error messages. In this study, we set out to investigate which elements of PostgreSQL's error messages are perceived most helpful by novices in error recovery situations. The results indicate that novices value, above all, that the error message communicates the location of the error, uses clear descriptive language, and suggest how the error could be fixed. These findings are applicable in redesigning SQL compiler error messages in both DBMS compilers as well as in different learning environments and DBMS extensions.

ACKNOWLEDGMENTS

This research has been partially supported by the *Foundation for Economic Education*. The authors thank all who participated in the study, and the participants of the *Human Factors Impact of Programming Error Messages (22052)* Dagstuhl Workshop for fruitful discussions inspiring this article.

REFERENCES

[1] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An Analysis of Patterns of Debugging among Novice Computer Science Students. *SIGCSE Bull.* 37, 3 (jun 2005), 84–88. https://doi.org/10.1145/1151954.1067472

[2] Umair Z. Ahmed, Pawan Kumar, Amey Karkare, Purushottam Kar, and Sumit Gulwani. 2018. Compilation Error Repair: For the Student Programs, from the Student Programs. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training* (Gothenburg, Sweden) *(ICSE-SEET '18)*. Association for Computing Machinery, New York, NY, USA, 78–87. https://doi.org/10.1145/3183377.3183383

[3] Titus Barik, Denae Ford, Emerson Murphy-Hill, and Chris Parnin. 2018. How Should Compilers Explain Problems to Developers?. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 633–643. https://doi.org/10.1145/3236024.3236040

[4] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. 2017. Do Developers Read Compiler Error Messages?. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 575–585. https://doi.org/10.1109/ICSE.2017.59

[5] Brett A. Becker. 2016. An Effective Approach to Enhancing Compiler Error Messages. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 126–131. https://doi.org/10.1145/2839509.2844584

[6] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE)*. ACM. https://doi.org/10.1145/3344429.3372508

[7] Brett A. Becker, Paul Denny, James Prather, Raymond Pettit, Robert Nix, and Catherine Mooney. 2021. Towards Assessing the Readability of Programming Error Messages. In *Proceedings of the 23rd Australasian Computing Education Conference* (Virtual, SA, Australia) *(ACE '21)*. Association for Computing Machinery, New York, NY, USA, 181–188. https://doi.org/10.1145/3441636.3442320

[8] Brett A. Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, and Catherine Mooney. 2016. Effective compiler error message enhancement for novice programming students. *Computer Science Education* 26, 2-3 (2016), 148–175. https://doi.org/10.1080/08993408.2016.1225464

[9] Brett A. Becker, Kyle Goslin, and Graham Glanville. 2018. The Effects of Enhanced Compiler Error Messages on a Syntax Error Debugging Test *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 640–645. https://doi.org/10.1145/3159450.3159461

[10] Brett A. Becker, Cormac Murray, Tianyi Tao, Changheng Song, Robert McCartney, and Kate Sanders. 2018. Fix the First, Ignore the Rest: Dealing with Multiple Compiler Error Messages. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 634–639. https://doi.org/10.1145/3159450.3159453

[11] Kathleen M. Cauley and James H. McMillan. 2010. Formative Assessment Techniques to Support Student Motivation and Achievement. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* 83, 1 (2010), 1–6. https://doi.org/10.1080/00098650903267784

[12] Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 480–486. https://doi.org/10.1145/3341525.3387384

[13] Paul Denny, James Prather, Brett A. Becker, Catherine Mooney, John Homer, Zachary C Albrecht, and Garrett B. Powell. 2021. On Designing Programming Error Messages for Novices: Readability and its Constituent Factors. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 55, 15 pages. https://doi.org/10.1145/3411764.3445696

[14] G.M. Donahue. 2001. Usability and the bottom line. *IEEE Software* 18, 1 (2001), 31–37. https://doi.org/10.1109/52.903161

[15] Andrew Ettles, Andrew Luxton-Reilly, and Paul Denny. 2018. Common Logic Errors Made by Novice Programmers. In *Proceedings of the 20th Australasian Computing Education Conference* (Brisbane, Queensland, Australia) *(ACE '18)*. Association for Computing Machinery, New York, NY, USA, 83–89. https://doi.org/10.1145/3160489.3160493

[16] Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot Text-to-SQL Translation using Structure and Content Prompt Learning. *Proc. ACM Manag. Data* 1, 2, Article 147 (jun 2023), 28 pages. https://doi.org/10.1145/3589292

[17] Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton. 2007. Architecture of a Database System. *Foundations and Trends in Databases* 1, 2 (2007), 141–259. https://doi.org/10.1561/1900000002

[18] James J. Horning. 1974. *What the Compiler Should Tell the User.* Springer Berlin Heidelberg, Berlin, Heidelberg, 525–548. https://doi.org/10.1007/978-3-662-21549-4_20

[19] Hsiu-Fang Hsieh and Sarah E Shannon. 2005. Three Approaches to Qualitative Content Analysis. *Qualitative Health Research* 15, 9 (2005), 1277–1288. https://doi.org/10.1177/1049732305276687

[20] ISO/IEC. 2016. ISO/IEC 9075-1:2016, "SQL - Part 1: Framework". https://www.iso.org/standard/63555.html

[21] ISO/IEC. 2016. ISO/IEC 9075-2:2016, "SQL - Part 2: Foundation". https://www.iso.org/standard/63556.html

[22] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* Technical Report. New York, NY, USA. https://doi.org/10.1145/2534860 999133.

[23] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-SQL. *The VLDB Journal* 32, 4 (Jan. 2023), 905–936. https://doi.org/10.1007/s00778-022-00776-8

[24] Bailey Kimmel, Austin Geisert, Lily Yaro, Brendan Gipson, Taylor Hotchkiss, Sidney Osae-Asante, Hunter Vaught, Grant Wininger, and Chase Yamaguchi. 2024. Enhancing Programming Error Messages in Real Time with Generative AI. In *ACM CHI EA '24*. https://doi.org/10.1145/3613905.3647967

[25] Bailey Kimmel, Austin Lee Geisert, Lily Yaro, Brendan Gipson, Ronald Taylor Hotchkiss, Sidney Kwame Osae-Asante, Hunter Vaught, Grant Wininger, and Chase Yamaguchi. 2024. Enhancing Programming Error Messages in Real Time with Generative AI. In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 608, 7 pages. https://doi.org/10.1145/3613905.3647967

[26] Sarah K. Kummerfeld and Judy Kay. 2003. The Neglected Battle Fields of Syntax Errors. In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20* (Adelaide, Australia) *(ACE '03)*. Australian Computer Society, Inc., AUS, 105–111.

[27] Junho Lee, Dowon Song, Sunbeom So, and Hakjoo Oh. 2018. Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 158 (oct 2018), 30 pages. https://doi.org/10.1145/3276528

[28] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 563–569. https://doi.org/10.1145/3545945.3569770

[29] Christian Murphy, Eunhee Kim, Gail Kaiser, and Adam Cannon. 2008. Backstop: A Tool for Debugging Runtime Errors. *SIGCSE Bull.* 40, 1 (mar 2008), 173–177. https://doi.org/10.1145/1352322.1352193

[30] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. 2008. Compiler error messages: what can help novices? *SIGCSE Bull.* 40, 1 (mar 2008), 168–172. https://doi.org/10.1145/1352322.1352192

[31] J.F. Pane, B.A. Myers, and L.B. Miller. 2002. Using HCI techniques to design a more usable programming system. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments.* IEEE Computing Society. https://doi.org/10.1109/hcc.2002.1046372

[32] Raymond S. Pettit, John Homer, and Roger Gee. 2017. Do Enhanced Compiler Error Messages Help Students? Results Inconclusive.. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 465–470. https://doi.org/10.1145/3017680.3017768

[33] Kai Presler-Marshall, Sarah Heckman, and Kathryn Stolee. 2021. SQLRepair: Identifying and repairing mistakes in student-authored SQL queries. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 199–210.

[34] Gary B Randolph. 2003. The Forest and the Trees: Using Oracle and SQL Server Together to Teach ANSI-Standard SQL. *Design* (2003), 234–236.

[35] Phyllis Reisner. 1977. Use of Psychological Experimentation as an Aid to Development of a Query Language. *IEEE Transactions on Software Engineering* SE-3, 3 (1977), 218–229. https://doi.org/10.1109/tse.1977.231131

[36] Phyllis Reisner. 1981. Human Factors Studies of Database Query Languages: A Survey and Assessment. *Comput. Surveys* 13, 1 (March 1981), 13–31. https://doi.org/10.1145/356835.356837

[37] Phyllis Reisner, Raymond F. Boyce, and Donald D. Chamberlin. 1975. Human factors evaluation of two data base query languages. In *Proceedings of the national computer conference and exposition AFIPS '75*. ACM Press. https://doi.org/10.1145/1499949.1500036

[38] Maria Mercedes T. Rodrigo and Christine Lourrine S. Tablatin. 2023. How Do Programming Students Read and Act upon Compiler Error Messages?. In *Augmented Cognition*, Dylan D. Schmorrow and Cali M. Fidopiastis (Eds.). Springer Nature Switzerland, Cham, 153–168.

[39] Ben Shneiderman. 1982. Designing computer system messages. *Commun. ACM* 25, 9 (1982), 610–611. https://doi.org/10.1145/358628.358639

[40] Toni Taipalus. 2023. SQL: A Trojan Horse Hiding a Decathlon of Complexities. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research* (Seattle, WA, USA) *(DataEd '23)*. Association for Computing Machinery, New York, NY, USA, 9–13. https://doi.org/10.1145/3596673.3603142

[41] Toni Taipalus and Hilkka Grahn. 2023. Framework for SQL Error Message Design: A Data-Driven Approach. *ACM Transactions on Software Engineering and Methodology* 33 (2023). Issue 1. https://doi.org/10.1145/3607180

[42] Toni Taipalus and Hilkka Grahn. 2023. NewSQL Database Management System Compiler Errors: Effectiveness and Usefulness. *International Journal of Human–Computer Interaction* 39 (2023), 3936–3947. Issue 20. https://doi.org/10.1080/10447318.2022.2108648

[43] Toni Taipalus and Hilkka Grahn. 2024. Building Blocks Towards More Effective SQL Error Messages. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 241–247. https://doi.org/10.1145/3649217.3653552

[44] Toni Taipalus, Hilkka Grahn, and Hadi Ghanbari. 2021. Error messages in relational database management systems: A comparison of effectiveness, usefulness, and user confidence. *Journal of Systems and Software* 181 (2021), 111034. https://doi.org/10.1016/j.jss.2021.111034

[45] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and Complications in SQL Query Formulation. *ACM Transactions on Computing Education* 18, 3, Article 15 (2018), 29 pages. https://doi.org/10.1145/3231712

[46] The Joint Task Force on Computing Curricula. 2015. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Technical Report. New York, NY, USA. https://dl.acm.org/citation.cfm?id=2965631

[47] Heikki Topi, Kate M. Kaiser, Janice C. Sipior, Joseph S. Valacich, J. F. Nunamaker, Jr., G. J. de Vreede, and Ryan Wright. 2010. *Curriculum Guidelines for Undergraduate Degree Programs in Information Systems*. Technical Report. New York, NY, USA. https://dl.acm.org/citation.cfm?id=2593310

[48] V. Javier Traver. 2010. On Compiler Error Messages: What They Say and What They Mean. *Advances in Human-Computer Interaction* (2010), 1–26. https://doi.org/10.1155/2010/602570

[49] Kazuhiro Tsunoda, Hidehiko Masuhara, and Youyou Cong. 2023. Mind the Error Message: An Inverted Quiz Format to Direct Learner's Attention to Error Messages. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 382–388. https://doi.org/10.1145/3587102.3588823

[50] T.W. van der Schaaf. 1995. Human Recovery of Errors in Man-Machine Systems. *IFAC Proceedings Volumes* 28, 15 (1995), 71–76. https://doi.org/10.1016/s1474-6670(17)45211-6

[51] John Wrenn and Shriram Krishnamurthi. 2017. Error messages are classifiers: a process to design and evaluate error messages. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (Vancouver, BC, Canada) *(Onward! 2017)*. Association for Computing Machinery, New York, NY, USA, 134–147. https://doi.org/10.1145/3133850.3133862

[52] Dieter Zapf and James T. Reason. 1994. Introduction: Human Errors and Error Handling. *Applied Psychology* 43, 4 (Oct. 1994), 427–432. https://doi.org/10.1111/j.1464-0597.1994.tb00838.x

[53] Zihe Zhou, Shijuan Wang, and Yizhou Qian. 2021. Learning From Errors: Exploring the Effectiveness of Enhanced Error Messages in Learning to Program. *Frontiers in Psychology* 12 (2021). https://doi.org/10.3389/fpsyg.2021.768962

A EFFECTIVE ELEMENTS QUERY-BY-QUERY

This appendix contains all the queries with a syntax error, the resulting error message, and participant answers on which elements are effective in the error message. In Figures 2 through 17, the numbers inside parentheses indicate the number of participants who deemed the element effective. We list elements that were mentioned at least twice in the data. The theoretical maximum number of occurrences for a single element is the number of participants (165). The number of coded responses for each query is indicated in the figure caption. One response may mention several elements.

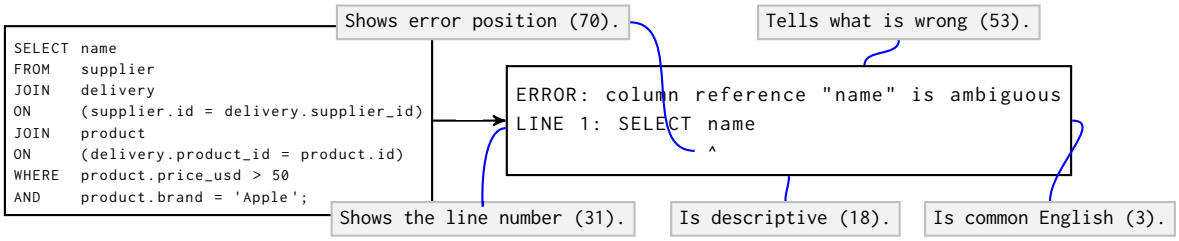

Fig. 2. Query T01 (156 responses)

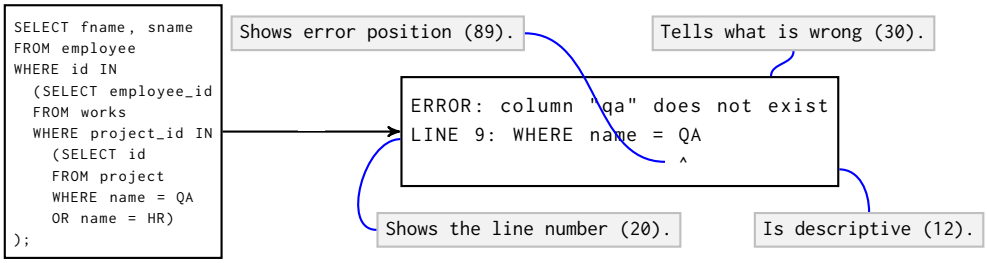

Fig. 3. Query T02 (117 responses)

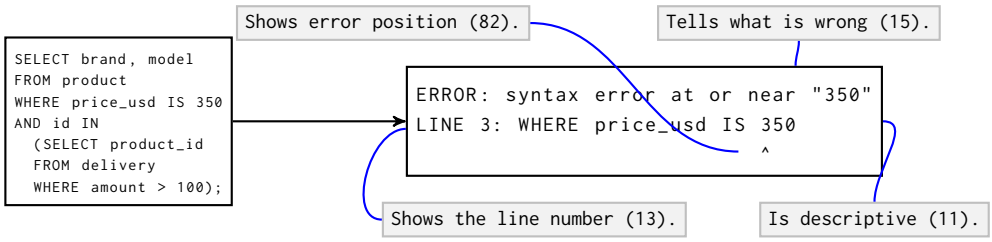

Fig. 4. Query T03 (122 responses)

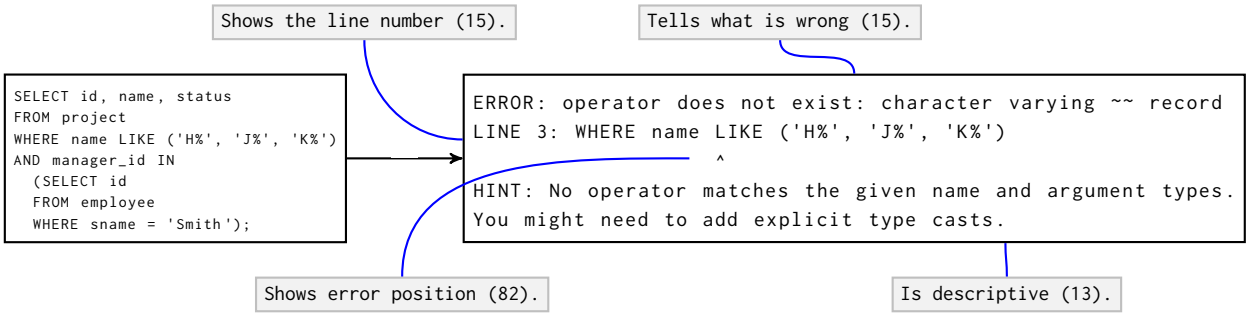

Fig. 5. Query T04 (123 responses)

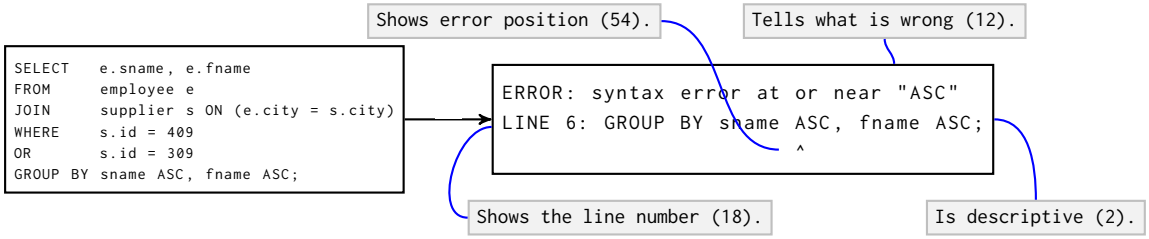

Fig. 6. Query T05 (113 responses)

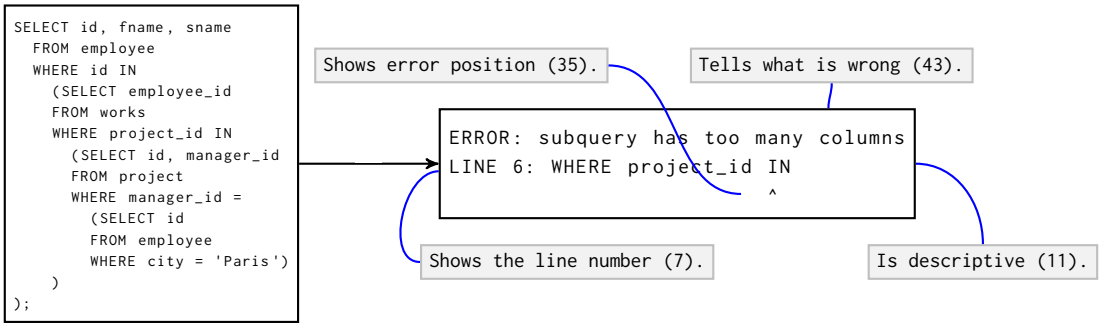

Fig. 7. Query T06 (117 responses)

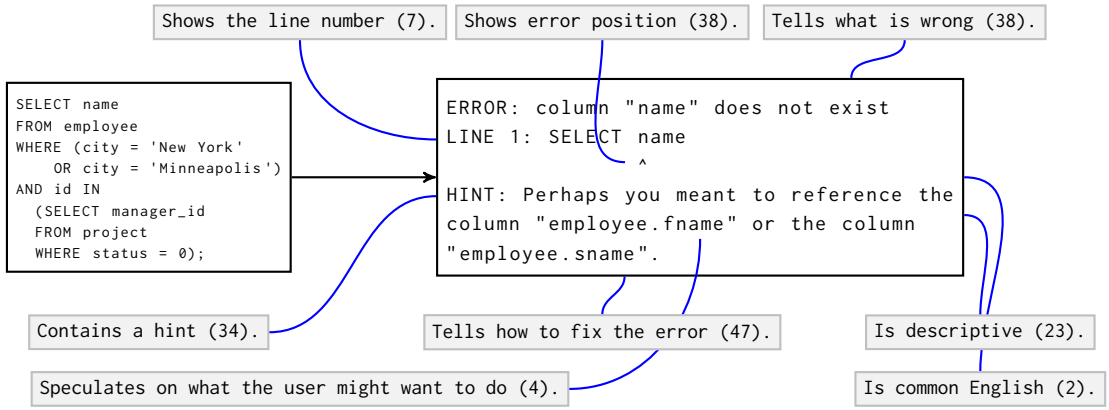

Fig. 8. Query T07 (165 responses)

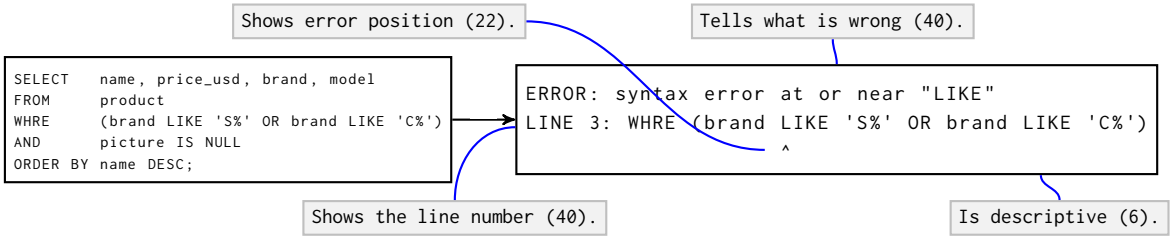

Fig. 9. Query T08 (119 responses)

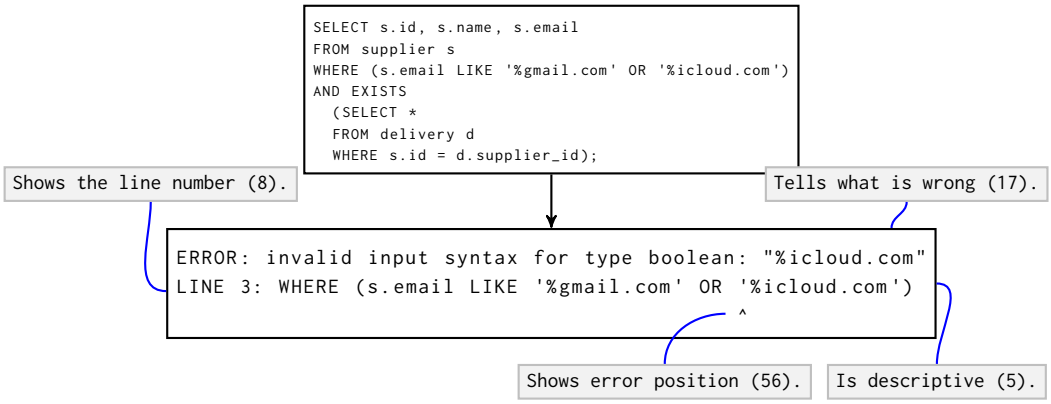

Fig. 10. Query T09 (119 responses)

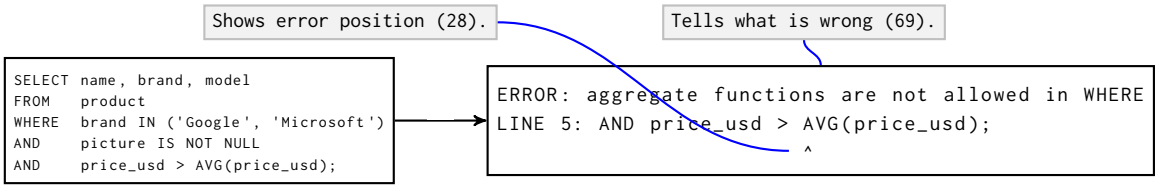

Fig. 11. Query T10 (132 responses)

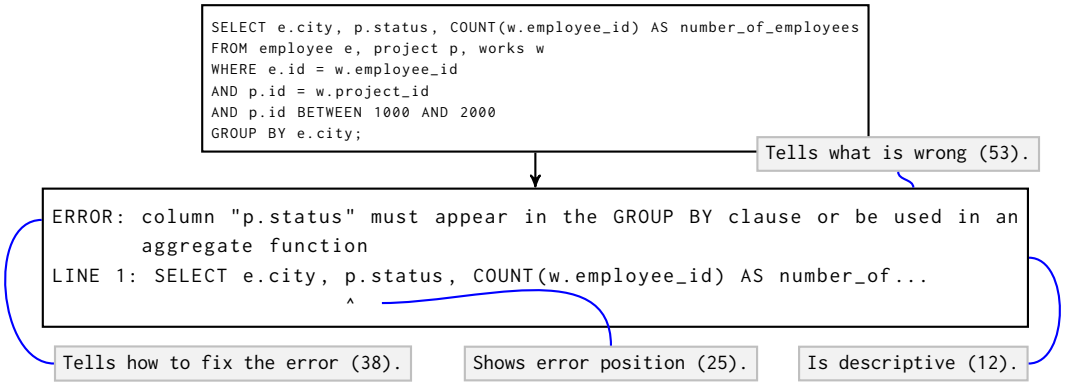

Fig. 12. Query T11 (119 responses)

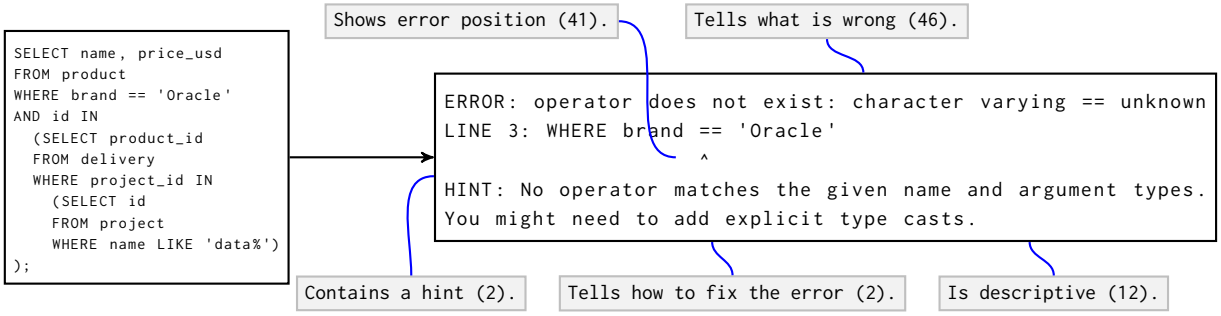

Fig. 13. Query T12 (140 responses)

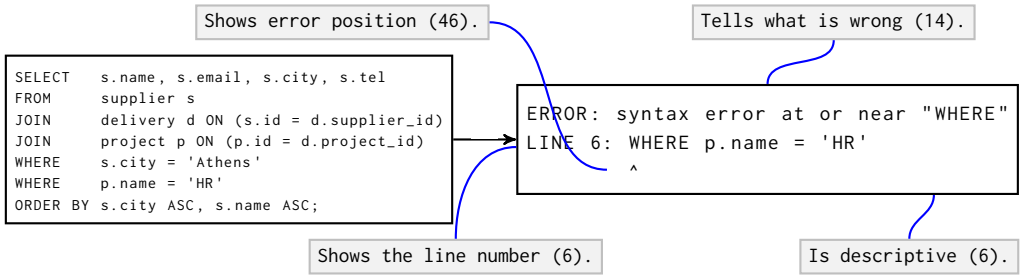

Fig. 14. Query T13 (120 responses)

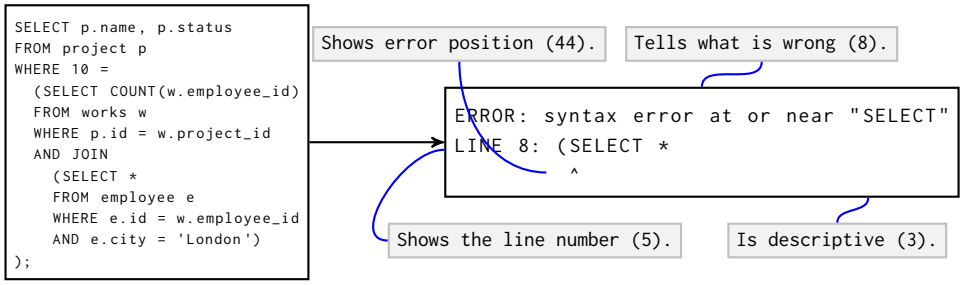

Fig. 15. Query T14 (128 responses)

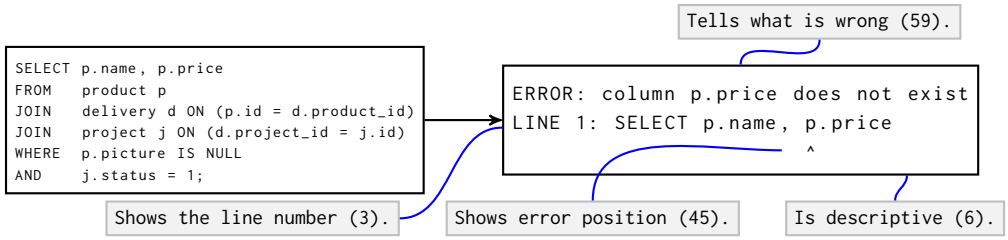

Fig. 16. Query T15 (134 responses)

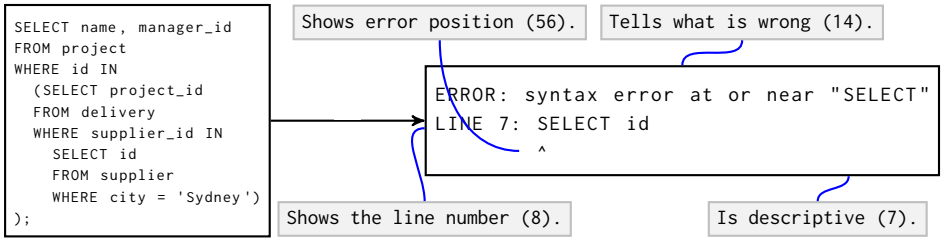

Fig. 17. Query T16 (128 responses)