# Assessing Peer Correction of SQL and NoSQL Queries

Wensheng Wu
University of Southern California
Los Angeles, CA, USA
wenshenw@usc.edu

## ABSTRACT

Students in database courses often make varied syntax and semantic mistakes in writing SQL and NoSQL queries. We report on a study where we designed two styles of multiple-choice questions based on students' mistakes in midterms and used the questions in the final exam to assess student's capabilities in identifying and correcting other students' mistakes. The study found that (1) students had similar performance on both styles of the questions; (2) the average accuracy rate of students in peer correction was about 83%; (3) over 80% of the students thought it was helpful to see and correct others' mistakes; and (4) students' performance in peer correction was moderately correlated with their overall performance. This study is the *first* to address students' mistakes in writing NoSQL queries and assess peer correction of both SQL and NoSQL queries.

## CCS CONCEPTS

• **Information systems** → **Data management systems**; • **Social and professional topics** → **Computing education**.

## KEYWORDS

SQL, NoSQL, query mistakes, peer correction, assessment

**ACM Reference Format:**
Wensheng Wu. 2023. Assessing Peer Correction of SQL and NoSQL Queries . In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569884

## 1 INTRODUCTION

Students in database courses often make varied syntax and semantic mistakes in writing SQL and NoSQL queries [1, 11, 18]. In this paper, we report on a study where we designed multiple-choice questions based on students' common mistakes in midterms, and then used the questions in the final exam to assess students' capabilities in identifying and correcting the mistakes.

Students in the graduate database course of our applied data science program offered in Spring 2022 participated in the study. The course had two sections, where section A had 91 students and section B had 73 students. We designed two styles of questions. Style 1 questions focus on correcting a single incorrect query given by a student in his/her midterm, while style 2 questions ask students to identify mistakes in the alternative queries from different students for the same information need. Students in section A used the style 1 questions, while students in section B used the style 2 questions. In addition to the peer correction questions, we also included a short survey at the end of the final exam to solicit students' feedback on error recall, identification, and correction.

The key findings of our study are:

- Students had similar performance in both styles of the questions: success rate on style 1 questions ranges from .64 to .91, and rate on style 2 questions ranges from .52 to .91.
- The average accuracy rate of students in peer correction, measured by $F$-score, was about 83% in both sections.
- 55% of the students in section A and 44% of the students in section B reported that they have made similar mistakes in the midterms.
- 81% of the students in section A and 84% in section B thought that peer correction was helpful.
- Students' performance on peer correction was moderately correlated with their total grade (the correlation coefficient $\rho$ = .45 for section A and .49 for section B).

Furthermore, students' detailed comments indicate that they felt the peer correction questions in the final exam helped "illustrate where the errors may occur", "prevent others from making the same mistakes", and "find different ways of answering the question".

While prior research has touched upon peer correction of SQL queries [9], we believe that this study is the *first* to address students' mistakes and peer corrections in writing NoSQL queries.

## 2 RELATED WORK

**SQL query mistakes:** There have been many efforts on understanding students' mistakes in writing SQL queries and reasons behind the mistakes [18, 19]. Ahadi et. al. [1, 2] conducted several studies to analyze syntax and semantic errors in students' SQL queries and found that (1) students made more syntax errors than semantic ones; (2) undefined columns and grouping errors were the two most common syntax errors among novices; (3) most of the semantic errors occurred in queries involving self-join, natural join, group by with having, and correlated subqueries; and (4) more than half of the semantic errors were due to omission, e.g., omitting column in select and omitting aggregate function. Their studies also suggested that the main reason for the errors was lack of practice.

Miedema et. al. [11] found out that student's mistakes in writing SQL queries may be caused by (1) the interference from their prior knowledge of other programming languages, e.g., confusing "=" in SQL with "==" in Python, (2) misunderstandings of SQL syntax, e.g., attributes in the select clause are separated by commas, while conditions in the where clause are connected by AND or OR, and (3) the lack of experiences in using SQL constructs such as group by, join, and subqueries.

Migler and Dekhtyar [12] found that students had difficulties in mastering self-joins, correlated subqueries, and outer joins. Poulsen



*SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada*

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569884

et. al. [15] also found that the majority of students' mistakes in writing SQL queries were syntax errors and it was especially challenging for students to write queries that involve group by and correlated subqueries. Brass and Goldberg [5] compiled a list of semantic errors in SQL queries such as inconsistent condition (e.g., where job = "clerk" and job = "manager") and unnecessary joins.

**Error correction:** There are also several efforts on developing tools to assist in the identification, evaluation, and correction of students' mistakes in writing SQL queries. For example, Sadiq et. al. [17] developed SQLator, an online SQL learning tool to automatically evaluate the queries formulated by students. Presler-Marshall et. al. [16] developed SQLRepair to help correct certain errors in students' SQL queries. Yang et. al. [21] computed edit distances among students' multiple submissions to the same SQL assignments, and used visualization tools to help instructors locate the areas where the students struggled the most. Leinonen et. al. [9] developed a crowdsourcing system where students can design, share, and peer review SQL assignments and answers.

**Learning from errors:** Experimental investigations in [10] showed that corrective feedback, including the analysis of reasoning leading up to the mistakes, is beneficial to learning. Booth et. al. [3] suggested that incorrect examples, either alone or in combination with correct examples, are especially useful in improving student learning in algebra. Große and Renkl [8] found out that learning from worked examples that contain errors was effective for students in a probability course, especially when the students had good mathematics background. Pea et. al. [14] suggested that students' mistakes in programming courses may be caused by the misunderstandings on the general programming concepts, inadequate knowledge of syntax and semantics of data and control logic, and failure to retrieve such knowledge.

## 3 MISTAKES IN SQL & NOSQL QUERIES

As described, our study was conducted in the context of a graduate database course in the data science program offered in Spring 2022. The course provides students with a broad overview to the landscape of modern data management. It covers data models, query languages, and query execution of SQL and NoSQL databases, as well as parallel data processing systems, such as Hadoop [20] and Spark [6]. The course teaches students how to write SQL queries to retrieve data from MySQL databases, REST requests to retrieve and update JSON data managed in Firebase (a Google cloud database) [13], XPath expressions to retrieve data in XML files, and MongoDB scripts [4] to retrieve JSON data stored in a MongoDB database. In this study, we focus on REST and XPath for NoSQL queries.

For example, Figure 1 shows an SQL query with subquery and group by. Figure 2 shows a curl command that sends a REST request (through HTTP methods, e.g., GET, PUT, POST, and PATCH) to the Firebase server. The request specifies the location of the endpoint, e.g., 'https://.../students.json', and details on the parameters for result ordering and filtering, e.g., orderBy and endAt. Figure 3 shows a segment of an XML file containing person data. An XPath expression is a path-like expression that specifies how to navigate through an ordered tree representation of the file, in order

Consider the following SQL query:

```
Select Beers.manf, count(Beers.name)
from Beers
where Beers.name in
     (select beer from Sells)
group by Beers.manf;
```

This query is an attempt to find, for each manufacturer, **how many times** its beers are listed for sale in bars, from the beers database with the following tables:

Beers(name, manf) // name and manufacturer of a beer
Sells(bar, beer, price) // a bar lists a beer at a given price

Which of the following statements about the above query is (are) correct?

**a**) The query does NOT have any syntax errors — syntax assertion
**b**) The subquery in the query may return the same beer multiple times — observation — distraction
**c**) It is NOT ok to omit Beers in Beers.manf and Beers.name
**d**) The query will only count the beer once even when the beer is sold at multiple bars — explanation

**Figure 1: An example of SQL question (style 1)**

Consider the following curl command:

```
curl -X PUT
    'https://dsci123.firebaseio.com/students.json?
     orderBy="age"&endAt=25'
```

This command is an attempt to find students who are **at most** 25 years old from the Firebase database.

Which of the following statements about the above command is (are) correct?

**a**) -X PUT should be -X GET — correction
**b**) endAt should be startAt instead — distraction
**c**) orderBy="age" should be orderBy="$age"
**d**) The above command has NO syntax error

**Figure 2: An example of Firebase REST API question (style 1)**

to find the desired information. For example, the expression "/persons/person[address[city='LA']]/name/text()" will find the names of people who live in LA from XML data shown in Figure 3.

Table 1 shows the common mistakes students made in writing SQL queries, REST requests, and XPath expressions in their homework assignments and two midterm exams. Mistakes are divided into two categories: syntax and semantic. A *syntax* mistake is a mistake where the query does not conform to the specification of query language and will be rejected by the database server. If a query has no syntax errors but does not produce the correct results, we say the query has a *semantic* error.

**SQL:** Students tend to make mistakes in writing queries that involve group by, having, aggregation, subquery, and join. For example, the query "select beer, count(*) from Sells group by beer having

| Language | Type | Common mistake | Example | Correction |
|---|---|---|---|---|
| SQL | Syntax | confuse "having" with "where" | ... group by beer having price > 3 | ... where price > 3 group by beer |
| | | ambiguous attribute names | select name from beers, bars | select beers.name from beers, bars |
| | | missing table alias for subquery | select * from (select ...) | select * from (select ...) as t |
| | | using table name as subquery | select * from ... where beer not in Likes | ... beer not in (select beer from Likes) |
| | | missing "select ..." in theta join | select Likes join Beers on ... | select * from Likes join Beers on ... |
| | | misuse of aggregation function | select name from ... having count > 1 | ... having count(*) > 1 |
| | Semantic | taking A != x to mean none of A's values is equal to x | (find bars that do not sell 'Budweiser')<br>select bar from Sells<br>where beer != 'Budweiser' | select bar from Sells where bar not in<br>(select bar from Sells<br>where beer ='Budweiser') |
| | | missing group by | select maker, count(*) from Beers | ... from Beers group by maker |
| | | incorrect group by attribute | select maker, ... group by name | select maker, ... group by maker |
| | | omitting distinct for unique values | select maker from Beers | select distinct maker from Beers |
| | | using "in" when 'not in' is intended | select * from Beers where maker in (...) | ... where maker not in (...) |
| | | missing group by attribute in select | select count(*) from ... group by bar | select bar, count(*) ... group by bar |
| REST Request | Syntax | wrong format for JSON data | curl -X PUT ... -d '{100: name="john"}' | ... -d '{"id": 100, "name": "john"}' |
| | | confused about orderBy syntax | curl -X GET '...&orderBy="$value"' | ...&orderBy="value"' |
| | | quoting numerical values | curl -X GET '...&startAt="25"' | ...&startAt=25' |
| | Semantic | confuse PUT with PATCH | *(add age attribute for student 100 )*<br>curl -X PUT '.../students/100.json' -d '{"age": 25}' | curl -X PATCH '.../students/100.json' -d '{"age": 25}' |
| | | incorrect structuring of data | curl -X PUT '...students/id/100.json' ... | .../students/100.json ... |
| | | confuse POST with PATCH | curl -X POST '.../100.json' ... | curl -X PATCH '.../100.json' ... |
| XPath | Syntax | missing @ for attributes | person[id = "123"] | person[@id = "123"] |
| | | assume position index starts from 0 | /students/student[0] | /students/student[1] |
| | | using () instead [] for predicate | //student(age = 25) | //student[age=25] |
| | | missing / at the beginning of path | students/student/name | /students/student/name |
| | | missing argument for contains() | contains("123") | contains(., "123") |
| | Semantic | placing predicate on wrong element | person/name[age = 25] | person[age=25]/name |
| | | returning wrong data | person[age = 25]/text() | person[age = 25]/name/text() |
| | | quoting numeric values | person[age > "25"] | person[age > 25] |
| | | confuse predicate with axes | person/city="LA" | person[city="LA"] |
| | | confused about usage of / and // | /student | //student or /students/student |
| | | misuse of contains for exact match | person[contains(city, "LA")] | person[city = "LA"] |

**Table 1: Common mistakes of students in writing SQL and NoSQL queries (REST requests and XPath expressions)**

price > 3" will be rejected by MySQL with an error "unknown column price in having clause". This is an example of syntax error, since the SQL language requires attributes in the having clause to either appear in the group by clause or be aggregated. On the other hand, omitting "group by beer" in "select beer, count(*) from Sells group by beer"' is allowed in MySQL. But the query returns only the first beer and the total count of *all* beers, instead of each individual beer and its corresponding count as expected.

Another common semantic mistake is to take "A != x" where $A$ is an attribute and $x$ is a value to mean that none of $A$'s values is equal to $x$. For example, "select bar from Sells where beer != 'Budweiser'" does not find bars that do not sell Budweiser. Instead, it finds bars that sell a beer which is not Budweiser.

**REST requests:** Common syntax errors include incorrect format used for JSON data and parameters. For example, name = "john" should be "name":"john"; orderBy ="value" should be orderBy = "$value"; and 25 should not be quoted in startAt = "25", when 25 refers to a numeric value instead of a string.

Common semantic mistakes include improper structuring of data and misuse of HTTP methods. For example, to add a new student with id = 100 and use id value as the key, there is no need to add "id" in the request URL "https://.../students/id/100.json"; using the PUT method in: curl -X PUT '. . . /students/100.json' -d '{"age": 25}' will overwrite all existing data for student 100, instead of adding a new attribute age for the student as intended.

**XPath:** Common syntax errors include missing @ for attributes, e.g., id in an XPath expression, person[id = "123"], should be @id if id is an attribute; using () instead of [] for predicate, e.g., //student(age = 25) should be //student[age = 25]; and assuming position index starts from 0, e.g., student[0] should be student[1] instead.

Common semantic mistakes include quoting numeric values, e.g., person[age > "25"] will treat age as a string and compare it with string "25" instead of number 25; placing predicate on a wrong element, e.g., person/name[age = 25] should be person[age = 25]/name, since age = 25 is a predicate on person instead of name.

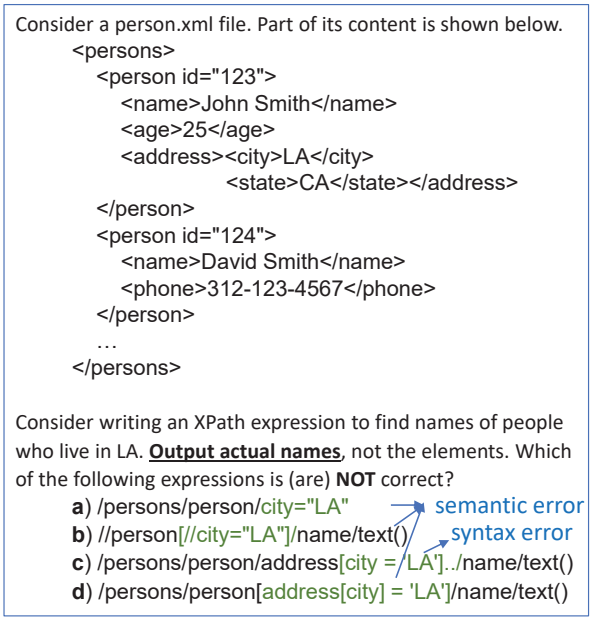


**Figure 3: An example of XPath question (style 2)**

In sum, students' mistakes are often caused by: (1) misunderstandings on the specification of query language and the structure of data, (2) incorrect structuring of queries, and (3) misuses of language constructs for what they are not intended to do.

## 4 DESIGN OF ASSESSMENT QUESTIONS

We design multiple choice questions based on students' common mistakes as described in Section 3. Each question has four choices and there may be multiple correct choices for a question. There are two styles of questions.

**Style 1:** In this style, each question consists of: (1) an incorrect SQL or NoSQL query from a student; (2) a description on what the query is intended to do, as well as the structure and content of the database the query will be executed on; and (3) a number of choices each of which is a statement about the query. Figures 1 and 2 show examples of SQL and NoSQL questions following this style. There are five types of statements.

- Correction: The statement is a correction to a specific mistake in the query. For example, choice a in Figure 2.
- Distraction: The statement is an incorrect "correction" to the query. For example, choice c in Figure 1, and choices b and c in Figure 2.
- Explanation: The statement explains on the semantic level why the query does not produce the desired results. For example, choice d in Figure 1.
- Observation: The statement is an observation on the semantics of *specific* part of the query, e.g., what a subquery in an SQL query returns. For example, choice b in Figure 1.
- Syntax assertion: The statement asserts that the given query does or does not have syntax error, e.g., choice a in Figure 1.

**Style 2:** In this style, the question just describes the data and information need, while each choice is a query (or an expression) given by a student for satisfying the request. The queries might have syntax or semantic errors. For example, choices a, b, and d in Figure 3 are syntactically correct but do not produce the desired results, while choice c has a syntax error.

So a key distinction between styles 1 and 2 is that a question in style 1 contains a single *incorrect* query from a student, while a question in style 2 may have multiple queries from different students where each query may be either correct or incorrect.

## 5 EVALUATION

### 5.1 Participants and Setup

All students in our database course in Spring 2022 participated in the study. The course had two sections. Section A had 91 students, and section B 73 students. We created 10 multiple-choice questions for each section, based on student's mistakes in the two midterms conducted in the week 6 and 11 of the semester. All questions used the same data and problem descriptions as that in the midterms. Four out of the ten questions were on SQL queries, three on REST requests, and three on XPath expressions. Each question had four choices and was worth 2 points. Selecting a wrong choice or missing a correct choice will result in a deduction of .5 point for the question.

These questions were then included in the accumulative final exam for the two sections, and accounted for 20% of the exam grade. The instructions in the exam made it clear to the students that the questions were based on their answers in the midterms which may have syntax and semantic errors. The remaining 80% of the final exam covered query execution algorithms, data analysis using MongoDB, map reduce in Hadoop, and Spark data frames and RDD. These topics were not covered in the midterms.

### 5.2 Style 1 Questions & Assessment

All questions for section A were of the style 1, where each choice of a question is a statement about an incorrect query given by a student in his/her midterm exam. The first three columns of Table 2 show the number of statements for different type of statement and level of difficulty. The level of difficulty was given by the instructor based on the expected time needed for students to make the right selection. For example, there are 12 statements about how to correct the mistakes in the query: 8 of them were considered to be easy and 4 had a medium level of difficulty.

For each statement, we computed a success rate which is the percentage of students who made the right choice in selecting the corresponding choice. The mean success rate and STDEV columns of the table show respectively the average and standard deviation of success rates over all students in Section A on the statement.

We can make the following observations:

- The mean success rate ranges from .62 (explanation, medium) to .91 (distraction, easy).
- Rates on medium and hard statements were typically lower than that on easy statements.
- Students did not perform well on the medium and hard-level explanations, with a mean rate of .62 and .64 respectively.

In particular, 62% of the students correctly selected a medium-level explanation on how to correct the position index error in an XPath expression. About 50% of the students correctly selected a

| Statement type | Difficulty | Count | Mean success rate | STDEV |
|---|---|---|---|---|
| Correction | Easy | 8 | 0.88 | 0.06 |
| | Medium | 4 | 0.73 | 0.16 |
| Distraction | Easy | 3 | 0.91 | 0.04 |
| | Medium | 4 | 0.85 | 0.09 |
| | Hard | 1 | 0.77 | 0.00 |
| Explanation | Easy | 2 | 0.79 | 0.04 |
| | Medium | 1 | **0.62** | 0.00 |
| | Hard | 4 | **0.64** | 0.12 |
| Observation | Easy | 2 | 0.85 | 0.05 |
| | Medium | 5 | 0.83 | 0.07 |
| Syntax assertion | Easy | 3 | 0.75 | 0.13 |
| | Medium | 3 | 0.71 | 0.01 |

**Table 2: Style 1 questions and student success rates**

| Expression type | Difficulty | Count | Mean success rate | STDEV |
|---|---|---|---|---|
| Syntax | Easy | 6 | 0.88 | 0.08 |
| | Medium | 4 | **0.56** | 0.15 |
| Semantic | Easy | 9 | 0.91 | 0.05 |
| | Medium | 19 | 0.72 | 0.19 |
| | Hard | 2 | **0.52** | 0.25 |

**Table 3: Style 2 questions and student success rates**

hard-level explanation on why the given SQL query failed to find "manufacturers that make some beers that nobody likes". A correct query would require proper use of subquery, which is one of the major challenges students have faced in writing SQL queries.

## 5.3 Style 2 Questions & Assessment

Questions in section B used the style 2 design, where choices are query expressions given by some students in the midterms for the specific task described in the question. The students were asked to select all the correct (or incorrect) expressions. For example, the task in the question shown in Figure 3 is to write an XPath expression to find names of people who live in LA. Students are expected to select all *incorrect* expressions from the given four choices.

The first three columns of Table 3 show the distribution of the number of expressions by expression type and level of difficulty. An expression is of the syntax type if it has syntax errors; otherwise, it is classified as a semantic type. Note that an expression of the semantic type might not have semantic errors. Among the 40 expressions, 29 (or 73%) are of the semantic type.

We can make the following observations:

- The mean success rate ranges from .52 (semantic, hard) to .91 (semantic, easy).
- The success rates on medium and hard expressions are much lower than that on easy expressions.
- The success rates were especially low for hard semantic-type expressions (.52) and medium syntax-type expressions (.56).

Figure 4 shows a question where students had low success rate. The question has four SQL queries taken from students' midterms. Queries a and d both have semantic errors: query a is missing "group by" and query d needs the "manf" attribute in its select clause. Note that, as described earlier, MySQL will accept these two queries, but

Recall the tables in the beers database as shown below.

```
Beers(name, manf)
Sells(bar, beer, price)
```

Consider writing an SQL query to find out, for each manufacturer, **how many times** its beers are sold in bars with a price of at least 3 dollars.

Which of the following queries is (are) **NOT** correct?

a)
```
SELECT manf, COUNT(beer) as count
FROM Beers as b join Sells as s on b.name = s.beer
WHERE price>= 3;
```
(missing group by)

b)
```
SELECT COUNT(name), manf
FROM Beers
GROUP BY manf
HAVING name IN (
    SELECT beer FROM sells WHERE price >= 3);
```
(syntax error: name is invalid in having)

c)
```
SELECT b.manf,COUNT(*)
FROM Beers AS b JOIN Sells AS s ON b.name=s.beer
GROUP BY b.manf
HAVING s.price>=3;
```
(syntax error: price is invalid in having)

d)
```
select count(*)
from Beers t1 join Sells t2 on t1.name = t2.beer
where t2.price >= 3
group by t1.manf;
```
(missing manf in select)

**Figure 4: A challenging style 2 question in the assessment**

does not produce the correct results. Queries b and c both used attributes (name in b and price in c) that are not valid in the having clause, a common mistake as described in Section 3. Recognizing these mistakes was considered to have a medium level of difficulty. 45% of the students correctly selected b; 42% correctly selected c.

Furthermore, only 37% of the students correctly identified an incorrect XPath expression: //person[//city="LA"]/name/text(). The expression will return names of all people as long as one person lives in LA, instead of only the names of people living in LA as requested. This error is considered to be hard for students to identify. Indeed, many students who gave a correct answer in the midterm thought the above expression was also acceptable.

## 5.4 Students' Overall Performance & Feedback

The *raw score* columns of Table 4 show the mean and standard deviation of students' raw scores on the peer correction questions. Note that the best possible score is 20 points. We can see that students from the two sections had similar average performance, with mean score of 15.9 for section A and 15.4 for section B, despite the different styles of questions for different sections.

In addition to the raw scores, we also computed the precision, recall, and $F$-score of student's answer for each question. Suppose the correct choices for a question are $a$ and $b$, but a student selected $a$, $c$, and $d$. The precision and recall of this student's answer on the question will be 1/3 and 1/2 respectively. The $F$-score is the harmonic mean of precision and recall, and computed as: $2PR/(P+R)$, where $P$ is precision and $R$ recall.

As Table 4 shows, students in the two sections had very similar performances: the average precision, recall, and $F$-score were about

| Section | No. of students | Raw score | | Precision | | Recall | | F-score | | Correlation with final exam grade | | Correlation with course total grade | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | STDEV | Mean | STDEV | Mean | STDEV | Mean | STDEV | ρ | p-value | ρ | p-value |
| A | 91 | 15.9 | 2.6 | 0.94 | 0.06 | 0.80 | 0.14 | 0.83 | 0.12 | 0.308 | 0.0030 | 0.453 | <.001 |
| B | 73 | 15.4 | 2.3 | 0.96 | 0.04 | 0.77 | 0.14 | 0.83 | 0.10 | 0.339 | 0.0034 | 0.498 | <.001 |

**Table 4: Students' overall performance on peer correction questions and its correlation with final exam and course total grade**

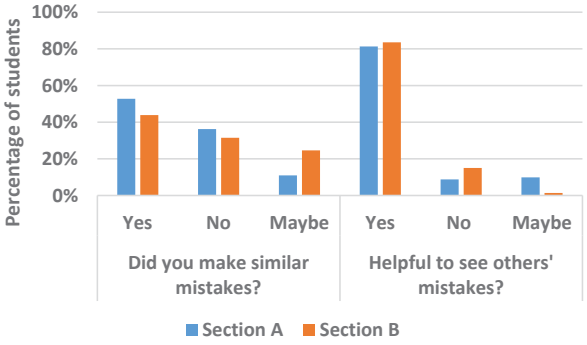


**Figure 5: Students' feedback on mistakes and helpfulness**

.95, .8, and .83 respectively. Note that recall is significantly lower than precision, indicating that students might have tried to avoid the penalties in selecting the wrong choices.

**Correlation analyses:** We further examined the correlation of students' performance on peer correction with their performance on the rest of the final exam and course grade. Students' course grade is formed by their grade on homework assignments (25%), lab tasks (5%), course project (20%), two midterms (30%), and final exam (25%). So the score on peer correction in the final exam accounted for 5% of students' total grade for the course.

The last four columns of Table 4 show the Pearson's correlation coefficients and corresponding $p$-values from the analyses. We can see that peer correction performance is moderately correlated (see [7], page 80) with students' performance on the rest of the final exam and the rest of total grade. The correlation with the rest of the course total grade was higher than the correlation with the rest of the final exam for both sections. This is likely due to that the peer correction questions were based on questions in midterm exams, while the rest of the final exam covered different topics than that in peer correction questions.

**Student feedback:** We also conducted a short survey at the end of the final exam. The survey consists of the following questions:

(1) Did you make mistakes in the midterms similar to those in the peer correction questions?
(2) What do you think are possible reasons for the mistakes or for **not** making the mistakes?
(3) Was it helpful to see and correct other students' mistakes?
(4) Explain why it was (not) helpful.

Results on survey questions 1 and 3 are shown in Figure 5. We can see that:

- 53% of the students in section A stated that they have made similar mistakes in the midterms, while 44% of the students in section B said so.
- 11% of the students in section A and 25% of the students in section B said that they might have made similar mistakes.
- 81% of the students in section A and 84% of the students in section B thought peer correction was helpful, and 10% of the students in section A said it might be helpful.

We also conducted thematic analysis on students' responses to questions 2 and 4. Students who did not make similar mistakes reported that the major factors contributing to their good performance in the midterms include: *review*, *practice*, and *prior experiences*. For example, some of the students' comments are:

- "important factors that contributed to the performance were the labs, and most importantly the homework"
- "good preparation by going through the lectures and slides thoroughly"
- "had a lot of familiarity with the subjects and used REST APIs and SQL on a daily basis"

On the other hand, the major reasons for the mistakes include: *careless*, *shallow understanding*, and *lack of practice*. For example,

- "not paying attention to the result which contains duplicates"
- "I did not understand the concept well"
- "not practicing enough for each topic"

Furthermore, students felt the peer correction process helped "illustrate where errors are made", "prevent people from making the same mistakes", and "find different ways of answering the question". There are also some students who felt that it was challenging to identify and correct other people's mistakes. For example, one student stated: "I think it is helpful, but also tricky. I have to read each query multiple times because on the surface level it makes sense, but I have to really consider whether it works or not."

## 6 CONCLUSION

We have presented a study on assessing students' capabilities in identifying and correcting mistakes in SQL and NoSQL queries written by other students. The study found that (1) students were able to achieve high accurate rates over different styles of assessment questions; and (2) the majority of students thought that peer correction can help them gain new perspectives on writing queries and avoid making similar mistakes in the future.

To the best of our knowledge, this is the first work on categorizing students' mistakes in writing NoSQL queries and assessing peer correction of both SQL and NoSQL queries. An interesting direction to extend the study is to have students *collaborate* on the error correction process. For example, we may compile students' common mistakes in homework and exams, and run *data science clinic* (e.g., on Piazza) to have students work collectively on correcting the mistakes and providing constructive feedback.

# REFERENCES


[1] Alireza Ahadi, Vahid Behbood, Arto Vihavainen, Julia Prior, and Raymond Lister. 2016. Students' syntactic mistakes in writing seven different types of SQL queries and its application to predicting students' success. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 401–406.

[2] Alireza Ahadi, Julia Prior, Vahid Behbood, and Raymond Lister. 2016. Students' semantic mistakes in writing seven different types of SQL queries. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 272–277.

[3] Julie L Booth, Karin E Lange, Kenneth R Koedinger, and Kristie J Newton. 2013. Using example problems to improve student learning in algebra: Differentiating between correct and incorrect examples. *Learning and Instruction* 25 (2013), 24–34.

[4] Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow. 2019. *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media.

[5] Stefan Brass and Christian Goldberg. 2004. Semantic errors in SQL queries: a quite complete list. In *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.* IEEE, 250–257.

[6] Bill Chambers and Matei Zaharia. 2018. *Spark: The definitive guide: Big data processing made simple*. " O'Reilly Media, Inc.".

[7] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Routledge.

[8] Cornelia S Große and ALEXANDER Renkl. 2004. Learning from worked examples: What happens if errors are included. *Instructional design for effective and enjoyable computer-supported learning* (2004), 356–364.

[9] Juho Leinonen, Nea Pirttinen, and Arto Hellas. 2020. Crowdsourcing Content Creation for SQL Practice. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 349–355.

[10] Janet Metcalfe. 2017. Learning from errors. *Grantee Submission* 68 (2017), 465–489.

[11] Daphne Miedema, Efthimia Aivaloglou, and George Fletcher. 2022. Identifying SQL misconceptions of novices: findings from a think-aloud study. *ACM Inroads* 13, 1 (2022), 52–65.

[12] Andrew Migler and Alex Dekhtyar. 2020. Mapping the SQL learning process in introductory database courses. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 619–625.

[13] Laurence Moroney, Anglin Moroney, and Anglin. 2017. *Definitive Guide to Firebase*. Springer.

[14] Roy D Pea, Elliot Soloway, and Jim C Spohrer. 1987. The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics* 9 (1987), 5–30.

[15] Seth Poulsen, Liia Butler, Abdussalam Alawini, and Geoffrey L Herman. 2020. Insights from student solutions to sql homework problems. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 404–410.

[16] 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.

[17] Shazia Sadiq, Maria Orlowska, Wasim Sadiq, and Joe Lin. 2004. SQLator: an online SQL learning workbench. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*. 223–227.

[18] Toni Taipalus and Ville Seppänen. 2020. SQL education: A systematic mapping study and future research agenda. *ACM Transactions on Computing Education (TOCE)* 20, 3 (2020), 1–33.

[19] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and complications in SQL query formulation. *ACM Transactions on Computing Education (TOCE)* 18, 3 (2018), 1–29.

[20] Tom White. 2012. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".

[21] Sophia Yang, Ziyuan Wei, Geoffrey L Herman, and Abdussalam Alawini. 2021. Analyzing Patterns in Student SQL Solutions via Levenshtein Edit Distance. In *Proceedings of the Eighth ACM Conference on Learning@ Scale*. 323–326.