Onni Poutanen

SQL ERROR MESSAGES THAT HINDER THE SYNTAX ERROR CORRECTION

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2022

TIIVISTELMÄ

Poutanen, Onni
Syntaksivirheiden korjaamista haittaavat SQL virheilmoitukset
Jyväskylä: Jyväskylän yliopisto, 2022, 60 s.
Tietojärjestelmätiede, Pro gradu -tutkielma
Ohjaaja(t): Taipalus, Toni

Niin tavalliset ihmiset kuin ohjelmoijatkin törmäävät päivittäisessä elämässään virheilmoituksiin, jotka ovat vaikeaselkoisia tai niiden ehdottama korjaus ei toimi. Tämän Pro gradu -tutkielman on tarkoitus tutkia, onko olemassa SQL virheilmoituksia, jotka mahdollisesti aiheuttavat enemmän ongelmia virheenkorjaamisessa kuin ne ratkaisevat. Tätä tutkielmaa varten on tutkittu SQL:n syntaksia ja tehty katsaus sitä koskevaan kirjallisuuteen. Tämän lisäksi SQL:n ja ohjelmoinnin virheilmoituksia koskevaan kirjallisuuteen on perehdytty. Tutkimuksessa käytetty data kerättiin opiskelijoilta, jotka ovat käyneet SQL:n ja tietokantojen perusteet sisältävän kurssin. Opiskelijoille annettiin virheellinen SQL-lause ja tunnetun tietokantahallintajärjestelmän antama virheilmoitus kyseiseen lauseeseen. Väärin korjattuja lauseita tutkittiin mahdollisten yhteyksien löytämiseksi virheilmoituksen ja lausekkeeseen tehdyn muutoksen väliltä. Tulokset osoittavat, että jotkin virheilmoitukset saattavat johtaa harhaan näyttäessään virheellistä tietoa tai kohdistaa käyttäjän huomion väärän asiaan lauseessa. Myös epäselvät ja epämääräiset virheilmoitukset aiheuttivat ongelmia. Tulevaisuudessa tutkimusten on mahdollista kiinnittää huomiota näihin ongelmiin ja mahdollisesti parantaa virheilmoituksien laatua.

Asiasanat: SQL, virheilmoitus, syntaksi

ABSTRACT

Poutanen, Onni
SQL Error messages that hinder the syntax error correction
Jyväskylä: University of Jyväskylä, 2022, 60 pp.
Information Systems, Master's Thesis
Supervisor(s): Taipalus, Toni

Not only programmers but many people in their daily lives have experienced the frustration of error message that does not make sense or the suggested fix does not work. This thesis aims to study SQL error messages that might cause more problems in the error correction process than they solve. SQL as language and syntax is studied for this thesis, and literature is reviewed. In addition, error message literature is examined in the context of programmers and SQL. The data is collected from students who have completed introductory level SQL and databases course. The students were presented with the incorrect statements, and with the help of an error message provided by a popular database management system, the participant submitted the corrected statements. Incorrectly altered statements are studied to discover any connections between the error message and modifications made by the user. The results revealed that there is a possibility that some error messages might mislead, provide misinformation, or fixate the user's attention to the wrong parts of the statement. In the future is possible to focus on these problems and improve the quality of the error messages.

Keywords: SQL, error messages, syntax

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

1.1 The problem

Programmers rely heavily on the hints given by compilers in the form of error messages and use significant time reading them. (Barik et al., 2017) Compiler error messages are often seen as lackluster and unavailing, causing obstacles in learning. (Becker, 2016; Denny et al., 2014; Traver, 2010) Practical and helpful error messages are something to thrive for. (Brown, 1983) Eminently Structured Query Language (SQL) error messages can be considered a fruitful research focus. (Taipalus et al., 2021) Especially syntax errors are seen as a significant obstacle for novice users. (Ahadi et al., 2016)

This thesis aims to identify these error messages that might cause hindrances regarding SQL syntax error correction. Furthermore, unhelpful error messages are analyzed to understand why they are not performing as expected. This can improve and enhance the error messages in the SQL domain in the future and improve learning and efficiency when using SQL. The research question driving this thesis is the following: "Are there SQL syntax error messages that have a negative impact on user error correction performance?". The additional research question is a natural continuum for the primary research question: "Why do certain SQL syntax error messages have a negative impact?".

1.2 Approach in this thesis

This thesis' main themes are SQL, error messages, and understanding user action based on error messages. The approach is to examine especially novice users and how they are affected by the error messages provided to them. This is done by conducting an empirical study and analyzing the results. The focus is on two of the largest database management systems, MySQL and Oracle Database.

The theory part of this thesis is to establish an understanding of the outline of what SQL is and the key concepts around it. It will cover some reasoning why SQL has become the popular query language it is today and how it emerged. The theory chapter also explains technicalities like the SQL syntax and how database management systems (DBMS) handle it. Other concepts discussed include databases, the relational model, and how SQL is attached to them. The second part of the theory chapter explains what error messages are and what can be found about SQL-related error messages. Theory on error messages focuses on the purpose of error messages, the primary findings of studies, and what types of error messages there are. Error research also covers how different kinds of errors can be classified.

The literature selected for this thesis includes scientific articles, books, and electronic sources. Scientific publications were accessed through Google Scholar Keywords for searches were: "error messages," "SQL," and "SQL error messages," but not limited to them. From error message studies, only programming or SQL related were considered as references.

In the research chapter, data collection methods are explained and examined, and the analysis approach is reviewed. This chapter should cover any questions about how and where the data was collected. Elaborating why the research is done in a certain way and what is the angle taken.

The results chapter presents all the findings of the analysis. In this part, the findings are iterated with the support of a visual presentation. This part also includes the incorrect statements and the error messages used in the empirical part.

The second to last chapter of this thesis includes discussion, answers to research questions, and future remarks. It aims to cover the significant findings from the analysis of results and what connections can be made from them. Any possible limitations are also disclosed in this part.

The last chapter of this thesis is conclusions. It summarizes the contents of this thesis. This chapter aims to give an overall picture of what was done and the main contributions.

2 THEORETICAL BACKGROUND

First, the theoretical background aims to cover a basic understanding of how databases and database management systems work and how SQL is related to them. Secondly, understanding syntax and the types of errors are most associated with SQL and syntactic errors. The last thing discussed is SQL error messages and error messages in general.

2.1 SQL and databases

This part is to establish a base understanding of what SQL is. This includes background and what is the correct syntax of it. SQL is also closely connected to the relational database model, which this chapter explains.

2.1.1 SQL standardization and a short history

SQL, or "structured query language" by full name, was originally developed by the IBM research laboratory during the 1970s. Oracle Database became the first commercially available DMBS utilizing SQL in 1979 (National Research Council, 1999). The first version of SQL was developed in the 1970s, but not until 1986 was it standardized by American National Standards Institute (ANSI) (Kelechava, 2018) and by International Organization for Standardization (ISO) in 1987 (ISO, 2022). Both ANSI and ISO have agreed on using SQL as the default language in relational databases. It has then become the de-facto query language used with relational databases. (Elmasri & Navathe, 2016)

Standardization benefits a product and is an important step for a product to become the mainstream option. It provides better interchangeability for the product. A standardized product can be transferred to a different setup and still function properly in the new environment. (Farrell & Seloner, 1984)

For SQL, standardization benefits are comparable to a normal product. In databases, a standardized query language enables the use of multiple different databases that all understand the same language. Different DBMSs might have variations between them. Since 1999 (SQL:1999), the standard of SQL has been separated into core specifications and extensions. The core specifications are something that SQL compliant DBMSs are required to implement. The extensions are optional and extend the functionality of the DBMS. Extensions can provide utilities like data mining or a more versatile database by providing analytics or additional data types, like multimedia data. (Elmasri & Navathe, 2016) In an ideal situation, two or more different DBMSs use standardized SQL, and all queries and other statements are interchangeable between the solutions. The subject becomes complicated as there are multiple different dialects and versions of SQL in deployment. Theoretically, interchangeability is a characteristic of SQL, but the minor differences between dialects force changes in an application when maneuvering between DBMSs. (Groff & Weinberg, 1999)

2.1.2 Purpose of SQL

SQL is a programming language created for managing relational databases. The abbreviation SQL stands for Structured Query Language, which gives a way that it is a language used with computers and other information systems. SQL is not a separate entity from the database. SQL should be considered an essential part of the database management system, a tool to interact with the database management system and the database itself. It can be used to access, manage, modify, update, and organize data within a database. It is thought to be one of the greatest influencers of the huge popularity of relational databases. (Elmasri & Navathe, 2016)

SQL can be identified as a declarative language. A declarative language is considered one of the main programming paradigms. The main characteristic of a declarative language is that the user declares what is required from the program rather than telling the program how to achieve it. (Fehily, 2008)

2.1.3 Role of SQL

The process of creating a functional database system includes choosing the appropriate DBMS. A database management system provides the users with different ways to interact with the database itself; this could include different interfaces and languages. The languages have been divided into types based on the purpose of the language. There are four main categories of DBMS languages: DDL, SDL, VDL, and DML. (Elmasri & Navathe, 2016)

Data definition language (DDL) defines different designs or schemas. Schema is what could be called the blueprint of the database. Database administrators and database designers utilize DDL to define conceptual and internal schemas for the database. For example, CREATE statement is one of the main functions of a DDL language. Storage definition language (SDL) can be utilized to define internal schemas. Although, the role of SDL is nowadays fulfilled by different functions, parameters, and specifications that are associated with the storage of files. Modern DBMSs do not have a specific language to fulfill the role of SDL. View definition language (VDL) is a tool to create views for the users and mappings. The role of VDL is overlapped in modern DBMSs by DDL, which, as mentioned, is used to define schemas. In relational database management systems, SQL queries and their results are used as the views for the application or user. DML or data manipulation language is used to fetch, add, and remove data from the database. DBMS can provide a language like SQL to achieve this functionality. (Elmasri & Navathe, 2016)

Mostly the modern DBMSs do not provide specific languages to fulfill the different roles of mentioned languages. For example, SQL is called a comprehensive database language that includes DDL, VDL, and DML. SDL is no longer included in SQL because the language aimed to be more conceptual and external levels. (Elmasri & Navathe, 2016)

2.1.4 How DBMS handles SQL statements

As the user inputs a DDL or DML statement into the database, the DBMS processes the statement before executing. The statement will never be executed if the preliminary process is not completed. Although all errors cannot be pruned out, for example, in the statement's execution, data translation errors might occur even with a correctly composed statement. (ORACLE, 2022)

Oracle describes the process: the process begins with parsing of the statement; this includes syntax check, semantic and shared pool check. Syntax check, which is related to the main topic of this thesis, makes sure that keywords are correctly spelled and the syntax is correct. A semantic check confirms if the required tables, columns, and other objects are implemented in the database. For example, if the statement requires data from a column that does not exist. With every query, the database creates a hash value of the statement. The hash represents that statement. The database system compares the hash to an existing pool of hashes. If a match for the hash can be found, the database can use the already existing execution plan with the statement (soft parse), or the database must create a new execution plan based on the statement (hard parse). After this, the statement is ready for the execution phase, where the database either fetches requested rows or fulfills the DDL statement. (ORACLE, 2022)

MySQL has a similar process compared to the oracle. One of the main differences is related to prior query storage. As the Oracle database had the hash-based check if there is an existing execution plan, MySQL also checks something called query cache, even before parsing. If the same query is done, MySQL will return the stored result set immediately if the result is still valid. Otherwise, MySQL has similar internals with a parser, query optimizer, and query execution plan. (Pachev, 2007)

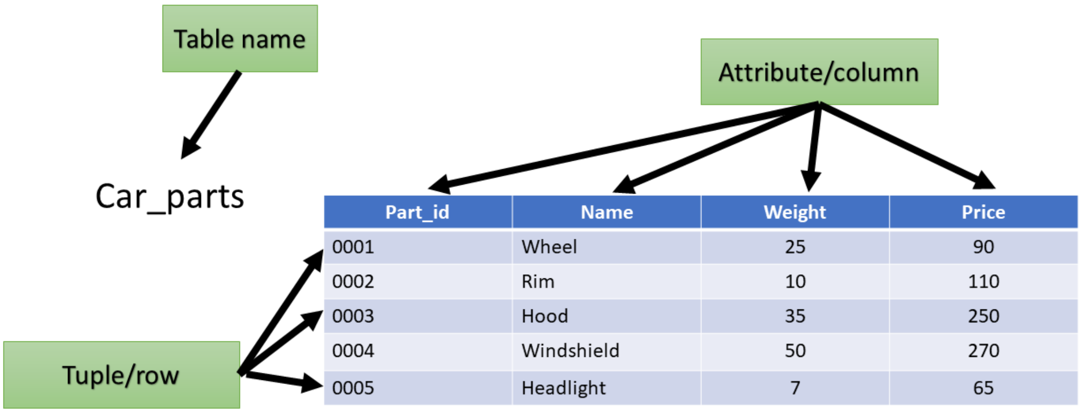
2.1.5 Databases and relational model

Databases are one of the cornerstones of the modern world. It is a collection of data or information, often electronically stored in a computer system. DBMS is used to control the database. Online stores, games, and many other systems rely on quick and efficient database access.

Data models are used to describe how the database is built and structured. Data models can be categorized by the concept. Conceptual or high-level data models represent structure as the user would view it. Conceptual models present entities, attributes, and relationships, for example, the name of a product(attribute), warehouse(entity), and in what warehouse the product is located (relationship). The conceptual model is considered a high-level data model, and the low-level or physical data model is used to represent how the data is saved physically, for example, on hard drives within a concrete computer. The physical data model also depicts information like access paths and recorded formats. (Elmasri & Navathe, 2016)

This thesis discusses SQL, which is associated with relational databases. The relational data model is categorized as a representational or implementation data model. Most modern DBMSs utilize representational data models, which can be thought to be a middle ground between conceptual and physical data models. (Elmasri & Navathe, 2016)

The relational model refers to an approach in database management first introduced by Edgar F. Codd (1970), a database that utilizes this model is called a relational database. As the name suggests, the data is stored in the form of relations, commonly referred to as tables. These relations have attributes, also referred to as columns. Attributes have names or properties that describe the contents of the table, for example, “Name” or “Part_id.” Figure 1 presents a simple table called “Car_parts” in a relational database, holding different car parts with the unique identifier “Part_id,” the primary key.



Part_id	Name	Weight	Price
0001	Wheel	25	90
0002	Rim	10	110
0003	Hood	35	250
0004	Windshield	50	270
0005	Headlight	7	65

Figure 1 Simple relational database table for car parts

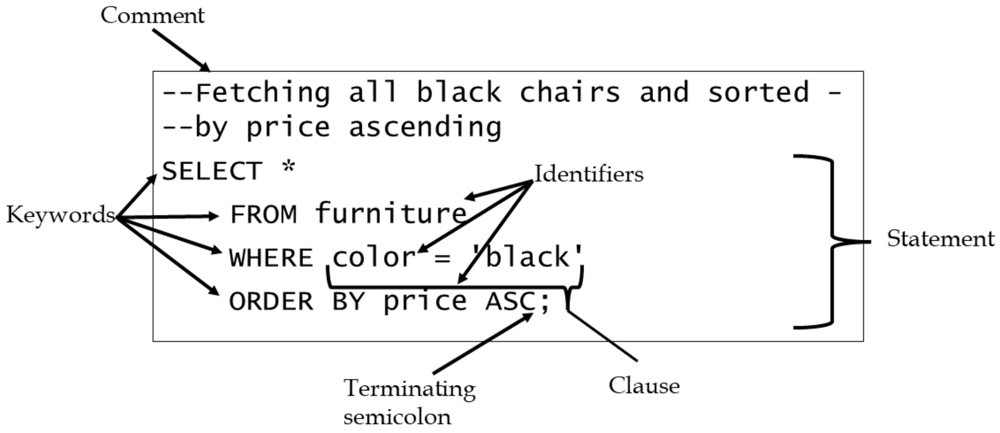
Most relational database management systems (RDBMS) or simply database management systems (DBSM) are built around utilizing SQL language. These database management systems sometimes include proprietary extensions, which extend the functionality of the SQL. As it might enhance the experience of using one DBSM, it also can cause incompatibility issues if working between multiple different DBSMs. Table 1 shows a few of the largest and most popular DBSMs currently available. There are various possibilities to choose from. These might have dialect differences, but at least in theory, interchangeability is possible. An indicator of the success and importance of SQL is that some of the DBSMs have been initially released over three decades ago and are still used to this day. For example, SQLite is the most widely used DBMS in the world. SQLite is found in Android, iOS, windows, car media systems, and other applications. (SQLite, 2022)

Table 1 Eight most popular database management systems

DBMS	Developer	Initial release	Reference
Microsoft SQL Server	Microsoft, Sybase, and Ashton-Tate.	1989	(Preston, 2007)
Microsoft Access	Microsoft	1992	(FMSinc, 2022)
Oracle Database	Oracle Corporation	1979 (Oracle V2)	(ORACLE, 2022)
IBM DB2	IBM	1983	(IBM, 1983)
SAP HANA	SAP SE	2011	(SAP, 2022)
MySQL	MySQL AB	1995	(Pachev, 2007)
PostgreSQL	PostgreSQL Global Development group	1996	(PostrgeSQL, 2022)
SQLite	D. Richard Hipp	2000	(Owens, 2006)

2.1.6 SQL syntax

The SQL statement can be divided into six components to help understand the syntax, also presented in Figure 2. A comment is not a necessary part of the SQL syntax and is considered optional, but writing comments is commonly depicted as a good programming habit. Explaining the query in a comment will help other users understand the statement's purpose. A comment works by inserting "--"before a line, and the compiler will ignore it. The second component of the statement is the statement itself. By "SQL statement" is referred to a correct sequence of keywords, identifiers, operators, literals, and punctuation symbols. The statement can be, for example, a query for data. The third component is clauses, which are parts of the SQL statement that follow a keyword. Some clauses are mandatory, but some are not, depending on the structure of the statement. Noteworthy is that the clauses must be written in the statement in a correct succession. As mentioned, keywords are required before the clauses. These are words used by SQL for specific tasks; using them as something else will cause an error. There are keywords and other phrases restricted by the DBMS, which cannot be used outside their intended usage. Identifiers are part of the syntax and component of the statement that is utilized to refer to a database object like tables and attributes. Figure 2 shows identifiers price, furniture, and color. The last sixth component is the terminating semicolon, which is a part of the correct SQL syntax and often forgotten. (Fehily, 2008)



```
--Fetching all black chairs and sorted -
--by price ascending
SELECT *
    FROM furniture
    WHERE color = 'black'
    ORDER BY price ASC;
```

Figure 2 Components of SQL statement

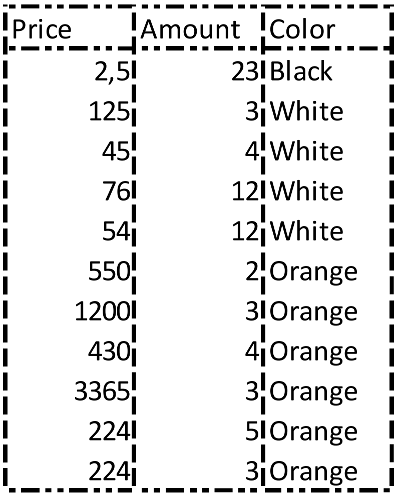
The SQL statement always begins with a keyword and ends with a semicolon. Otherwise, the internals can vary. There is a possibility that two different statements provide the same correct result-set. Therefore, it is challenging to formulate rules on how SQL statements should be built. The statement can include multiple SELECT clauses and subclauses to achieve the desired results. Building SQL statements requires expertise in SQL, database understanding, and a clear vision of desired results.

2.1.7 Building a SQL statement with example

Assumed that the database connection is made and other configurations are done, the user can start writing queries to interact with the database. Often DBMS provides some interface. Upcoming examples assume that there is a database for a small online store. The store database has tables for furniture and suppliers. The database could contain tens or hundreds of tables and thousands of items per table in a real-life scenario.

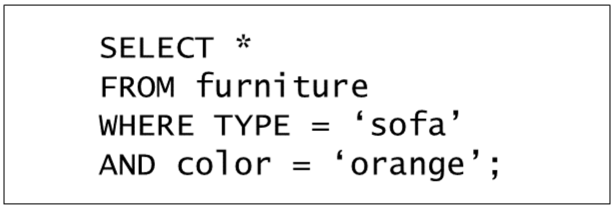
The SELECT statement is used to fetch data from the database. It is necessary at the beginning of every query. Although the statement can begin with some other keywords, the focus of this thesis is on querying data. Next, the user defines what columns or data is wanted, for example, price. For this purpose, SQL contains FROM keyword. FROM keyword specifies the table data is fetched from. A simple query on the database to retrieve all furniture prices can be done only with the help of these two keywords. The whole query would be: “SELECT price FROM Furniture.” If more data is required and more information displayed on the results, the query can be modified to show price, amount, and color: “SELECT price, amount, color FROM Furniture.” The DBMS returns them, as shown in Table 2.

Table 2 Result-set for simple query with price and amount



Price	Amount	Color
2,5	23	Black
125	3	White
45	4	White
76	12	White
54	12	White
550	2	Orange
1200	3	Orange
430	4	Orange
3365	3	Orange
224	5	Orange
224	3	Orange

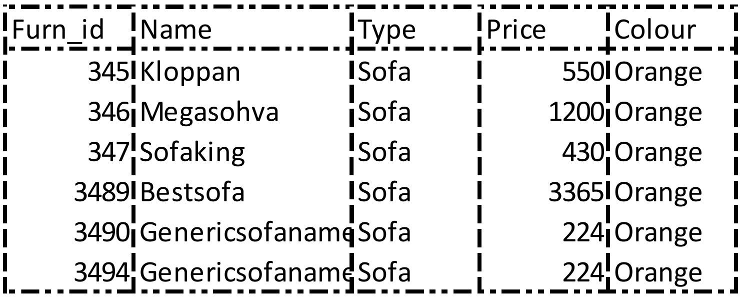
The SELECT statement is used to retrieve data. If entered as shown in the example, it provides the user with all the possible data found in the defined columns. To specify more precisely the desired results, the statement can be complemented with a WHERE clause. If the user wants to find out all orange sofas, the additional clause could be: "WHERE TYPE='sofa' AND color='orange'." The asterisk used in Figure 3 with the SELECT clause is to specify that all columns should be displayed in the result-set shown in Table 3. Single quotation marks are used to delimit strings but are not used with numerals. Double quotation marks are rarely used and generally only with table or column identifiers, but it varies between databases. Some DBMSs accept double quotation marks instead of single ones when defining strings.



```
SELECT *
FROM furniture
WHERE TYPE = ‘sofa’
AND color = ‘orange’;
```

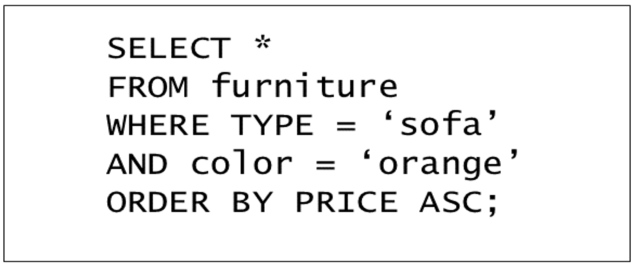
Figure 3 SQL query with WHERE keyword

Table 3 Result set for querying orange sofas



Furn_id	Name	Type	Price	Colour
345	Kloppan	Sofa	550	Orange
346	Megasohva	Sofa	1200	Orange
347	Sofaking	Sofa	430	Orange
3489	Bestsofa	Sofa	3365	Orange
3490	Genericsofanam	Sofa	224	Orange
3494	Genericsofanam	Sofa	224	Orange

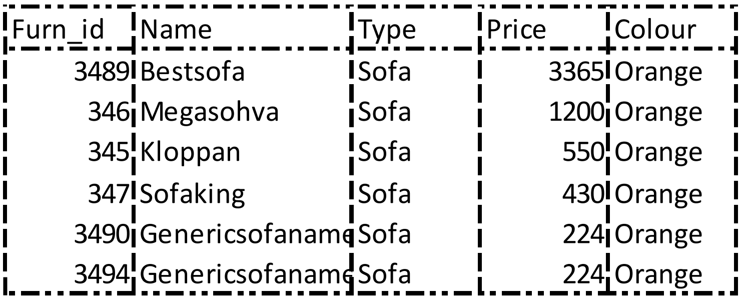
SQL can also be utilized to sort the results if required. Statement "ORDER BY" is added to the end of the statement. If the user wants to list all sofas by price, the statement complete statement is "SELECT * FROM FURNITURE WHERE TYPE='sofa' ORDER BY PRICE." In addition, the sort can be descending or ascending: "ORDER BY PRICE DESC" or "ORDER BY PRICE ASC." Query in Figure 4 would provide the same result set as the previous example, but in a different order, as shown in Table 4. The ORDER BY clause can include multiple columns to arrange the results further.



```
SELECT *
FROM furniture
WHERE TYPE = 'sofa'
AND color = 'orange'
ORDER BY PRICE ASC;
```

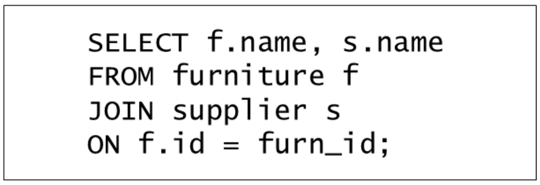
Figure 4 SQL query with ORDER BY keyword

Table 4 Result set sorted by price



Furn_id	Name	Type	Price	Colour
3489	Bestsofa	Sofa	3365	Orange
346	Megasohva	Sofa	1200	Orange
345	Kloppan	Sofa	550	Orange
347	Sofaking	Sofa	430	Orange
3490	Genericsofanam	Sofa	224	Orange
3494	Genericsofanam	Sofa	224	Orange

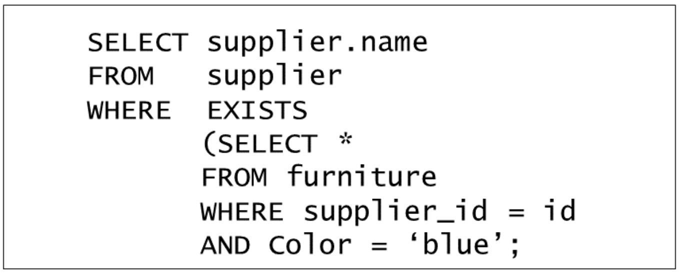
Query from one table might not provide sufficient results. To formulate a query that retrieves data from multiple tables, the JOIN statement can be deployed with the help of the subclause ON. For this thesis's scope, the INNER JOIN clause is required, which is the most common JOIN type. The INNER JOIN compares the tuples or rows between the selected tables based on the equivalent column. If the JOIN type is not specified in the statement, DBMS by default, utilizes INNER JOIN. The example in Figure 5 has the keyword JOIN, which defaults as INNER JOIN.



```
SELECT f.name, s.name
FROM furniture f
JOIN supplier s
ON f.id = furn_id;
```

Figure 5 SQL query with JOIN

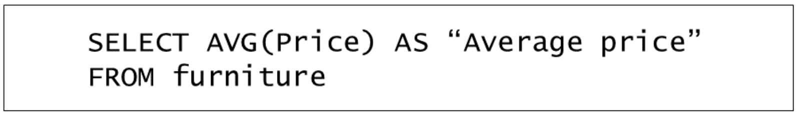
EXISTS keyword can be utilized in cases that do not require results from multiple tables but require information if a table holds a suitable record. It does not provide the results but only a Boolean answer if the subquery returns at least one record. For example, Figure 6 presents a query that would give results only of suppliers that provide blue sofas.



```
SELECT supplier.name
FROM   supplier
WHERE  EXISTS
       (SELECT *
       FROM furniture
       WHERE supplier_id = id
       AND Color = ‘blue’;
```

Figure 6 SQL query with EXISTS keyword

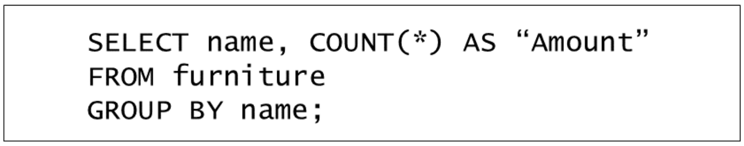
SQL also makes use of aggregate functions. These functions include count, sum, average (avg), minimum (min), and maximum (max). Aggregate functions work on sets of rows and perform mathematical operations on them. Aggregate functions are also called group functions and are deployed often with GROUP BY. Figure 7 presents a simple query that will provide the user with the average price of all items in table “furniture.”



```
SELECT AVG(Price) AS “Average price”
FROM furniture
```

Figure 7 SQL query with aggregate function AVG

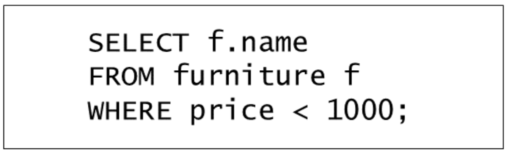
With GROUP BY, users can group the result set into logical groups. For example, if the user would like to calculate the amount of each sofa and group them by name, the query in Figure 8 could be used. This query would provide all furniture in distinct rows with the amount of each piece of furniture. With only a single column in the SELECT clause, there must be only a single column in the GROUP BY. The columns in SELECT must be included in the GROUP BY clause if they are not aggregate functions.



```
SELECT name, COUNT(*) AS “Amount”
FROM furniture
GROUP BY name;
```

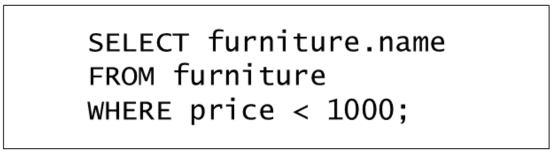
Figure 8 SQL query with aggregate function COUNT and GROUP BY keyword

Some attributes or columns may have identical names in more extensive databases with numerous tables. SQL uses temporary naming of tables and columns, also called aliases, to prevent errors. The naming persists only for the duration of one query. The syntax for alias uses AS keyword, but it is not necessary to use with all DBMSs. The syntax is the same in both column and table naming, but columns are named in the SELECT clause, and tables are named in FROM clause. In the Figure 9 table, “furniture” is given the alias “f,” and it is also referred to in the SELECT clause by defining that column “name” should be taken from the “f.” It is possible that other tables also include a column called “name,” which would cause an issue without the help of aliasing. This specifying can be done without assigning the table an alias like Figure 10. However, assigning an alias can reduce the length of the query if dealing with multiple columns and tables.



```
SELECT f.name
FROM furniture f
WHERE price < 1000;
```

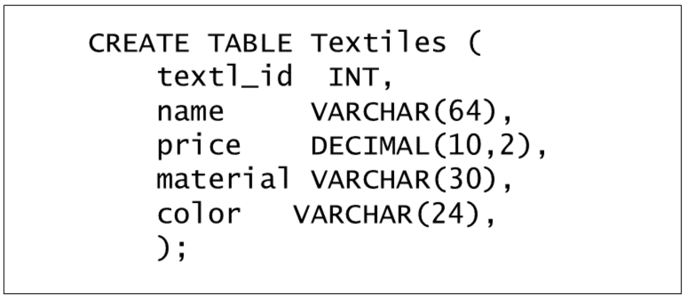
Figure 9 SQL query with an alias for table "furniture"



```
SELECT furniture.name
FROM furniture
WHERE price < 1000;
```

Figure 10 SQL query specifying that column "name" is in table "furniture"

The SQL toolset does not end up querying for data. SQL can be used to create new tables, update, and delete them. The CREATE TABLE statement can be used to establish a new table for the database. Figure 11 presents an example of a statement to create a new table for the database. As shown in Figure 11, the statement can be lengthy. Every line presents a column in the actual database. Every column has a name, the data type, and possibly the maximum length of data.



```
CREATE TABLE Textiles (
    textl_id  INT,
    name     VARCHAR(64),
    price    DECIMAL(10,2),
    material VARCHAR(30),
    color   VARCHAR(24),
    );
```

Figure 11 SQL statement to create a new table within the database

Additionally, the user can create and design the wanted SQL schema. It is a collection of the objects in the database. These objects include tables, triggers, and other database objects.

2.2 Error messages

2.2.1 Purpose of error messages

Error messages can affect and guide people's lives every day without paying attention to them. These messages can interact with people in most common ways, like when a car has a blinking light or entering an incorrect password for an email. These error messages are relatively simple and efficiently guide to action. Error messages with information systems equivalently inform the user about the faulty operation. Computer error messages can provide the user with information about the cause of the error. Additionally, error messages within computer systems can and often do navigate the user towards action to resolve the issue. (Maglio & Kandogan, 2004)

The definition of an error message can be said in multiple ways. To define error messages in the context of IT, some dictionaries have different ways. As shown in Table 5, the main constructors of this definition are that the message is displayed on the computer screen, and something has gone wrong and caused an error in the system. An error message is to provide the user with the information that there is a problem. Providing additional information and helpful guidance should be one of the key components of computer error messages. (Brown, 1983

Table 5 Definitions of error messages in information systems

"Information that appears on a computer screen or other device to state that you have made a mistake or that something has gone wrong in a program"	(Cambridge Dictionary, 2022)
"a message indicating that an error has occurred"	(Merriam-Webster, 2022)
"a message displayed on a visual display unit, printout, etc. indicating that an incorrect instruction has been given to the computer"	(Collins, 2022)

2.2.2 Error message research

Error message vagueness, quality, and usefulness have been questioned for a long time. Brown (1983) already states that error messages are part of information systems that should receive additional attention to enhance them. Error messages in general, have been studied for over half a century. The way different programming and other environments produce these error messages can vastly differ. It can be argued that error messages and how they are brought to the user are one of the most significant aspects of programming.

Becker et al. (2019), in a literature review on the topic of "text-based programming error message research," classified different guidelines suggested by multiple papers. Different guideline classifications:

- Increase Readability
- Reduce Cognitive Load
- Provide Context
- Use a positive tone
- Show examples
- Show solutions or Hints
- Allow Dynamic Interaction
- Provide Scaffolding
- Use Logical Argumentation
- Report errors at the right time

The consensus among papers is that readability is one of the most important aspects of error messages. (Becker et al., 2019) This translates into how well and efficiently the user understands the meaning of the error message.

A study by Karvelas et al. (2020) suggests that novice programmers take the initiative to get more feedback from the compiler. Implicating those novices are likely to feel that they need more information from the error messages. Their study also suggests that novice users rely on error messages and tend to run manual compilations to ensure that the code runs if there are no error messages present. Both findings show a glimpse of the importance of error messages for novice programmers. Even if error messages are an asset to novice programmers, some studies show that enhancing error messages might not have as desired effect as it could be. (Denny et al., 2014; Pettit et al., 2017) There is some contradiction to this statement, as Becker (2016) finds that enhanced error messages do diminish the total amount of compiler errors among students. Becker is also able to further identify eight specific error messages that benefitted more from enhancing, resulting in fewer errors. In a recent study by Denny et al. (2020), novice programmers debugging capabilities were benchmarked. The results showed that error messages altered with guidelines provided by most recent and comprehensive research on error messages resulted in fewer attempts to correct the error and time spent on the problem.

According to a study done with the help of eye-tracking, programmers tend to use a significant amount of time in reading and trying to understand error messages. There was also an evident connection between the programmer not correctly interpreting the message and difficulties solving the error. This leads to the conclusion that the non-readable error message could potentially hinder the user's performance. (Barik et al., 2017) An experienced programmer possibly comprehends the meaning of the error message swiftly, but this might not be the case for novice programmers. As professional programmers spend time reading the message, there are implications that novice programmers do not benefit from a longer error message. (Nienaltowski et al., 2008)

2.2.3 Types of errors in SQL and reasons

There are a few different types of errors possible in SQL. Brass & Goldberg (2005) Define errors in semantic and syntactic categories. A syntactic error is caused by incorrect syntax in the inserted query, which means the user or application tries to enter a sequence of characters that is not a proper SQL query. These types of invalid inputs receive immediate feedback from the BDSM in the form of an error message. The content and format of the error message can vary between different BSDM providers.

The other type of error Brass and Goldberg (2005) define is a semantic error. This error happens when a valid SQL sentence is used, but it is not suitable for the task. To elaborate, the semantic error does not cause an error message, and can even provide results, but the results are not like the user expects. These semantic errors can cause grievance as there is a possibility that the user is deluded into believing that the SQL statement is adequately formulated. This could happen because of inconsistency in the used SQL query, for example, a clause that ultimately contradicts itself and provides no results.

Brass and Goldberg (2005) also state that needlessly complex queries could be considered a semantic error. The reasoning behind this is manifold, but the main reason is that the user does not likely understand the SQL perfectly. In this case, the results returned should be correct even if it is considered a semantic error.

In this thesis, the focus is on SQL syntax error messages. Even if Goldbergs and Brass's definition of syntactic error is adequate in many cases, it could be helpful to elaborate on the subject. Taipalus et al. (2018), in their research have classified syntactic errors into six main categories:

1. *Ambiguous Database Object.* In some cases, the objects in the database might contain identical attributes, as objects on various levels form their namespaces. Without a proper definition of the object, this can lead to a situation where the DBMS might be unable to decide what object query is referring to. For example, multiple tables inside a database might have the attribute "name."
2. *Undefined Database Object.* This type of error is caused by a query pointing at an object that does not exist in the database. This can result from a simple typographic error or a lack of information about the database on hand. For example, the correct table could be "users" instead of "user."

3. *Data type mismatch.* Taipalus et al. (2018) identified four different situations which commonly caused this type of mistake according to their research. First, the utilization of an incorrect operator in the query that does not match the datatype with operators like IS and LIKE. According to Taipalus et al. (2018), the following common mistake is the incorrect use of quote markings inside the query. This type of error is caused by using quotes around string type or numeric values. The third is the lack of understanding operators that compare Booleans, for example, not correctly utilizing AND and OR operators. The last erroneous query observed was using the column as a part of the function, even if it does not match the required data type.
4. *Illegal Aggregate Function Placement.* Aggregate functions, also known as COUNT, SUM, AVG, etc., can be only placed with the SELECT statement or HAVING clause. Any other placement would cause a syntactic error. This might not be true in all cases, as Taipalus et al. (2018) specify that these rules apply specifically to the environment in which their data was collected.
5. *Illegal or Insufficient Grouping.* This error is caused by not having GROUP BY inserted in the statement when using the aggregate function. If the primary SELECT clause of the SQL statement has at least one grouping function and at least one column, there should be a GROUP BY clause which incorporates the grouping column.
6. *Common Syntax Error.* The number of different possibilities of causing a syntax error is enormous. As some syntax errors can be classified, some errors are not easily grouped with others. These errors include simple spelling errors in SQL statements, missing semicolon, inappropriate order of clauses, improper brackets, and missing FROM clause.

A study by Ahadi et al. (2016) has implications that inexperienced SQL users, in this case, students, would be more likely to make a syntactic error rather than a semantic error. The authors collected an estimated 160 000 SQL SELECT statements from students. In conclusion, the authors suggest that teaching should have greater attention to the correct SQL syntax. The results also revealed that students would cease to try to answer the question in case of a syntactic error. (Ahadi et al., 2016) The research does not consider the effectiveness of the displayed error message in terms of sequential attempts on the SQL query.

There are multiple reasons why users might make mistakes while formulating queries. As Taipalus et al. (2018) presented, reasons like misspelling, lack of knowledge and expertise, some errors are derived from more cognitive origin. Smelcer (1995) gives insight into the possible sources of errors. Smelcer identifies five significant reasons for cognitive errors, particularly in the context of SQL:

1. *Working memory overload.* Similar to a computer, humans also have limited amounts of working memory at their disposal. This can cause a simple mistake of forgetting a part necessary for the correct SQL statement.
2. *Absence of retrieval cues.* Related to working memory overload, a hint would remind the user of the critical component even before attempting the query. This has similarities with the purpose of an error message, as they aim to hint toward the correct statement.
3. *Procedural fixedness.* There is a risk of using the same formula for solving the problem on hand, even in the case of a very different problem, which would require altering the query. This cognitive error could happen to a user who has extensive amounts of similar queries to write, and the following statement suddenly requires a new approach.
4. *Incorrect procedural knowledge.* This cognitive error is related to multiple causes mentioned by Taipalus et al. (2018). All six mentioned types of errors can be caused for this reason. This cognitive reason is that sometimes the user does not have the proficiency to construct the correct query statement.
5. *Misperception.* An error in user observation and attention. For example, a misread sentence or a word can cause the user to write a statement reckoned to be correct but is inherently wrong as the user honestly has just the incorrect word in mind. The statement can be otherwise correct but still returns an error.

3 RESEARCH SETTING

3.1 Data collection

For data collection, we selected the sixteen most common syntax errors presented in Taipalus et al. (2018). We constructed sixteen erroneous queries, each pertaining to one of the most common syntax errors. Additionally, we recorded the corresponding error messages from Oracle Database 19c and MySQL 8 with the InnoDB storage engine.

The data were collected from an introductory database course. Prior to participation, the students in the course were given lectures and practical exercises concerning the relational model, database design and implementation, as well as data manipulation and querying using SQL. Participation was voluntary, and the students were shown a data privacy statement prior to participation. A total of 87 students chose to participate. These participants were randomly assigned to either Oracle Database or MySQL group, and shown a questionnaire of sixteen erroneous SQL queries, data demands, and DBMS error messages. The error messages were shown according to the group a student was assigned to, i.e., a student in the Oracle Database group was shown error messages generated by Oracle Database. Next, with the help of the error messages and general knowledge on SQL, the participant was instructed to fix the erroneous SQL query.

3.2 Analysis

Data collected has one entry for each question, and all participants were presented with 16 incorrect statements and error messages. One entry in refined data has the following information:

- The database management system presenting the error message
- If the statement is correctly modified
- Participant number
- Question number
- The corrected statement from the user

The data for this thesis includes a voluminous number of entries, which all must be examined with precision. Every entry has attributes that can be effortlessly analyzed automatically, like the percentage of statement correction success. Important information for the study is the statement written by the participant. Creating, for example, a script to find patterns and other possibly interesting matters is outside the scope of this thesis. Because nature of some parts of the data, the data must be gone through manually.

In accordance with the research question, the data analysis begins from erroneous attempts to fix the query. This is due to the goal of discovering possible hindrances in error correction. Perfectly corrected statements in data hint that the error message presented might have been helpful instead of a hindrance. In case of divisive results on error correction success, the incorrectly altered statements must be reviewed.

Analyzing the data involves iterating through all the questions one by one. Every entry for every question must be observed to identify connections between the error message and the incorrectly fixed statement. Furthermore, some statements might be altered correctly to some degree, but not perfectly, and still cause an error when inserted into the DBMS.

To achieve an efficient and most practical approach to analysis, some preliminary analysis was done. For this purpose, one question was selected and analyzed. During this process, it was documented if a specific technique of analyzing would increase efficiency. It was noticed that selecting answers for one statement and grouping the incorrect answers in groups by similarities, was an effective way of iterating through the data and recognizing patterns.

4 RESULTS

4.1 Introduction to findings

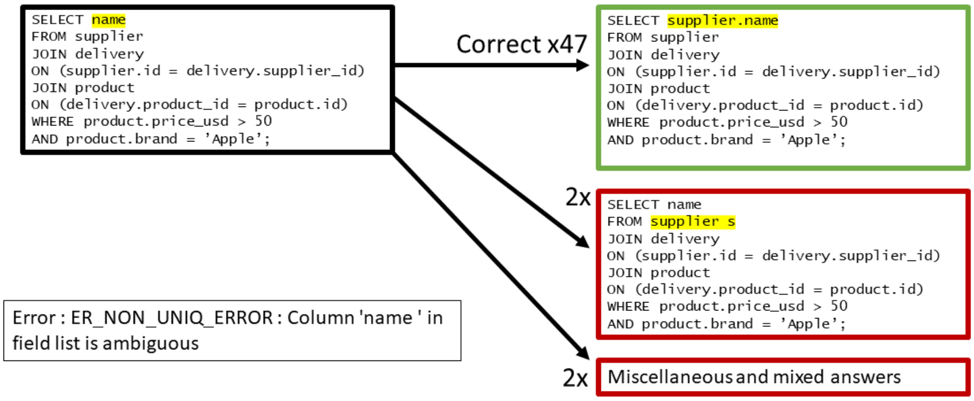
The findings chapter comprises what could be perceived from the answers to the incorrect statements with the error message as additional guidance. Headlines for 4.2.x are representative of common syntax errors in SQL. Figures have the error message provided by the DBMS and correct and incorrect answers grouped in their respective categories.

For clarification, the Figures presented in the findings have the incorrect answers grouped by possible similarities, like altering the same part of the statement. Some Figures have included "no changes" as a separate finding if it seems that a significant amount of the participants has made no changes. The "miscellaneous and mixed answers" can include answers that had no changes; this group contains answers that have minor changes and have no recognizable similarities.

4.2 Findings

4.2.1 Ambiguous column name

As shown in Figures 12 and 13, the user must correct the column name to supplier.name to avoid ambiguous column naming. This error is caused by multiple similarly named attributes/columns inside the database.



```
SELECT name
FROM supplier
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

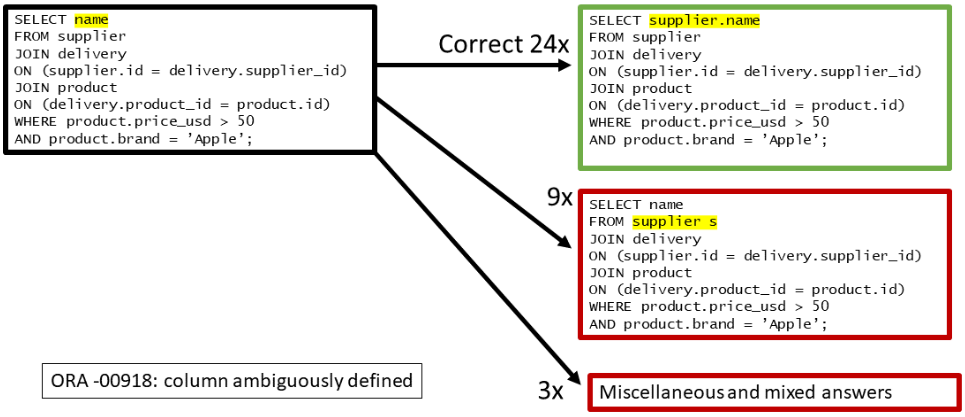
Error : ER_NON_UNIQ_ERROR : Column 'name ' in field list is ambiguous

```
SELECT supplier.name
FROM supplier
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

```
SELECT name
FROM supplier s
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

Figure 12 MySQL error message and results for statement #1

The error message by MySQL in Figure 12 details some information about the error. 'Non uniq' refers to the fact that the name is not unique. It shows directly that the entered column name is ambiguous. There are only four incorrect statements. Two wrong answers had altered nearly the correct part of the statement and indicated an understanding of the issue. The rest of the incorrect answers had no collective theme. Based on the high success rate, there is little indication that this error message should be considered unhelpful. Correctly corrected statements are almost identical, with the main differences in alias naming.



```
SELECT name
FROM supplier
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

ORA -00918: column ambiguously defined

```
SELECT supplier.name
FROM supplier
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

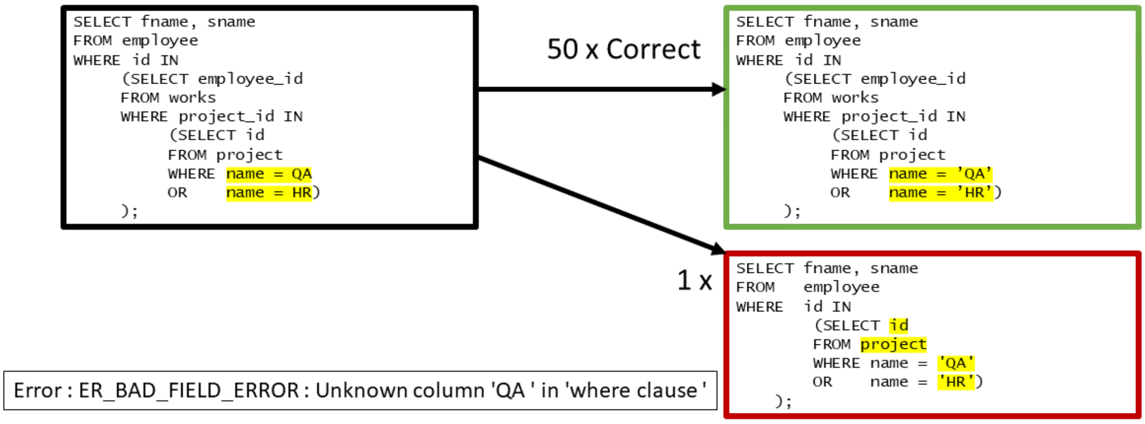
```
SELECT name
FROM supplier s
JOIN delivery
ON (supplier.id = delivery.supplier_id)
JOIN product
ON (delivery.product_id = product.id)
WHERE product.price_usd > 50
AND product.brand = 'Apple';
```

Figure 13 Oracle Database error message and results for statement #1

Compared to MySQL, as seen in Figure 13, the Oracle Database gives less information on the location of the error but does define the problem similarly. The error message does inform the user that a column is ambiguous. It does not provide the column that causes the error. Even if the success rate is noticeably lower, all incorrect answers address the correct problem besides a few. The most common mistake is assigning an alias to table "supplier" but not referencing that in the SELECT clause.

4.2.2 Omitting quotes around character data

In this case, the user must add a single quotation to the second WHERE clause. In SQL, character data is defined with single quotation marks. Incorrect and correct statements can be seen in Figures 14 and 15.



```
SELECT fname, sname
FROM employee
WHERE id IN
     (SELECT employee_id
     FROM works
     WHERE project_id IN
          (SELECT id
          FROM project
          WHERE name = QA
          OR    name = HR)
     );
```

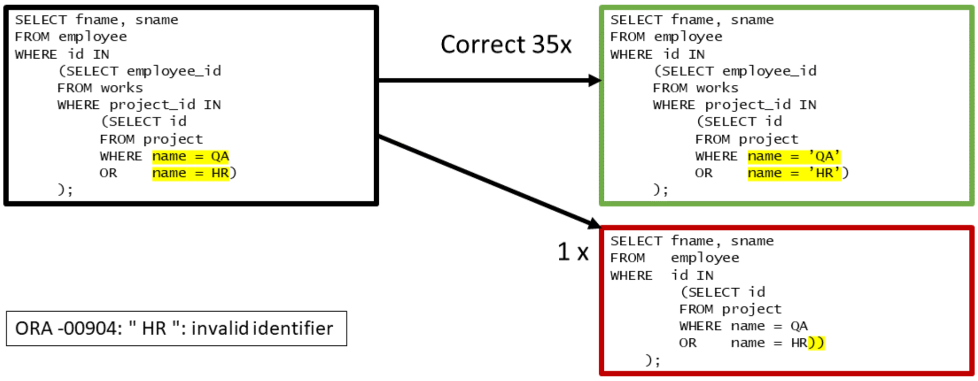
```
SELECT fname, sname
FROM employee
WHERE id IN
     (SELECT employee_id
     FROM works
     WHERE project_id IN
          (SELECT id
          FROM project
          WHERE name = ’QA’
          OR    name = ’HR’)
     );
```

```
SELECT fname, sname
FROM   employee
WHERE  id IN
         (SELECT id
         FROM project
         WHERE name = 'QA'
         OR    name = 'HR')
      );
```

```
Error : ER_BAD_FIELD_ERROR : Unknown column 'QA ' in 'where clause '
```

Figure 14 MySQL error message and results for statement #2

Error message by MySQL seen in Figure 14 provides the syntax error’s location, including a reference to ‘QA.’ It does not mention quotation marks. Interestingly as seen in Figure 14, the only incorrect answer does add the quotation marks to the correct place but incorrectly deploys the WHERE and IN clause to refer between tables.



```
SELECT fname, sname
FROM employee
WHERE id IN
     (SELECT employee_id
     FROM works
     WHERE project_id IN
          (SELECT id
          FROM project
          WHERE name = QA
          OR    name = HR)
     );
```

```
SELECT fname, sname
FROM employee
WHERE id IN
     (SELECT employee_id
     FROM works
     WHERE project_id IN
          (SELECT id
          FROM project
          WHERE name = ’QA’
          OR    name = ’HR’)
     );
```

```
SELECT fname, sname
FROM   employee
WHERE  id IN
        (SELECT id
        FROM project
        WHERE name = QA
        OR    name = HR))
     );
```

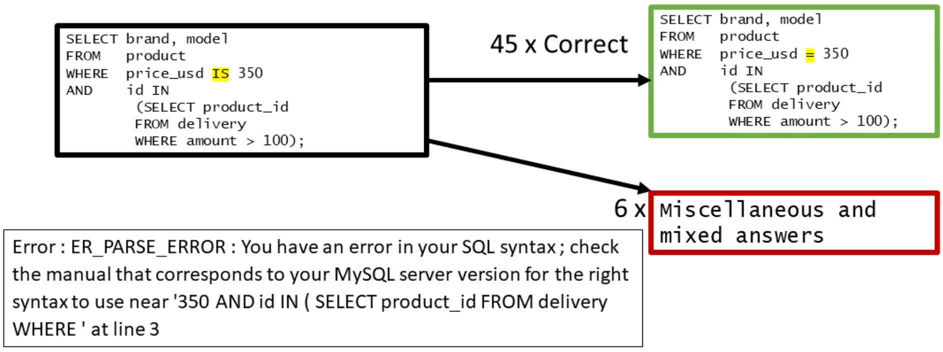
```
ORA -00904: " HR ": invalid identifier
```

Figure 15 Oracle Database error message and results for statement #2

As seen in Figure 15, the error message by Oracle Database is shorter than the MySQL counterpart. The message informs the user that HR is an invalid identifier. Also, in this case, it is up to the user to have the expertise to notice the missing quotation marks. The single incorrect statement seen in Figure 15 adds additional parenthesis.

4.2.3 Faulty use of IS as operand

This statement has invalid use of the operator, as shown in Figures 16 and 17. When using the WHERE clause to find something equal, a "=" is used. IS as an operator does not exist in SQL.



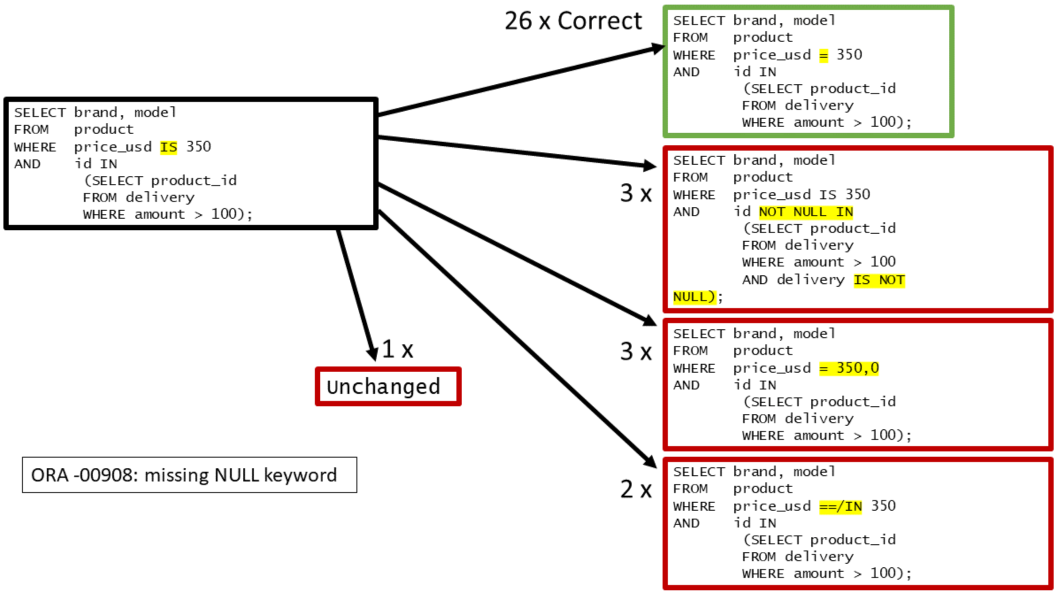
```
SELECT brand, model
FROM   product
WHERE  price_usd IS 350
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
SELECT brand, model
FROM   product
WHERE  price_usd = 350
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check
the manual that corresponds to your MySQL server version for the right
syntax to use near '350 AND id IN ( SELECT product_id FROM delivery
WHERE ' at line 3
```

Figure 16 MySQL error message and results for statement #3

The error message from MySQL itself has the correct line, as shown in Figure 16, helping to locate the error on line 3. The message also delivers information that the error is near '350 AND id IN'. Otherwise, the syntax error is not explained. The user must notice that the operator is invalid. Incorrectly altered statements are not locating the error correctly. All six have different faulty changes, including one with no contribution.



```
SELECT brand, model
FROM   product
WHERE  price_usd IS 350
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
SELECT brand, model
FROM   product
WHERE  price_usd = 350
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
SELECT brand, model
FROM   product
WHERE  price_usd IS 350
AND    id NOT NULL IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100
        AND delivery IS NOT
NULL);
```

```
SELECT brand, model
FROM   product
WHERE  price_usd = 350,0
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
SELECT brand, model
FROM   product
WHERE  price_usd ==/IN 350
AND    id IN
        (SELECT product_id
        FROM delivery
        WHERE amount > 100);
```

```
ORA -00908: missing NULL keyword
```

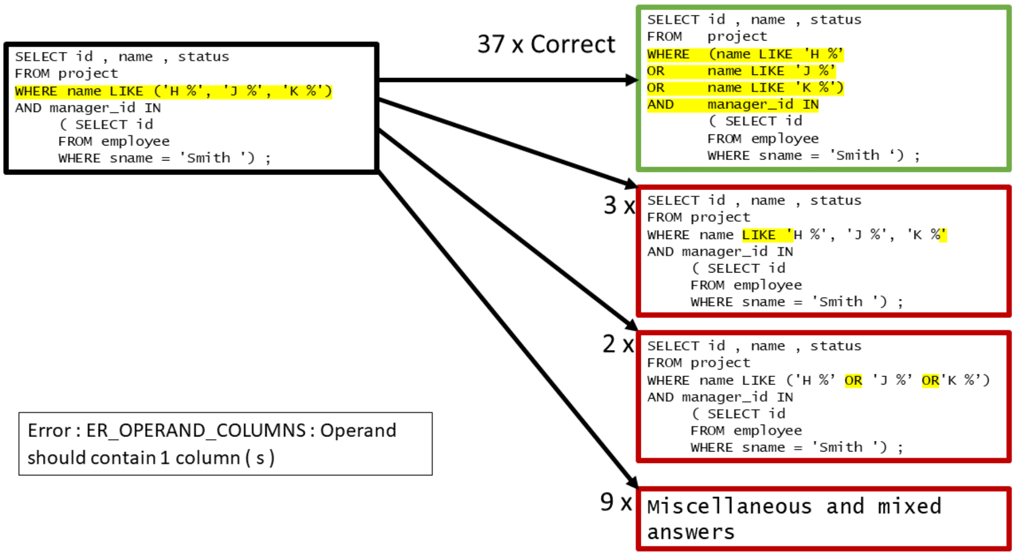
Figure 17 Oracle Database error message and results for statement #3

This error message by Oracle Database in Figure 17 does not seem to provide information on the location or type of error. Keywords are words reserved for utility in SQL, and NULL itself is not a keyword.

Three out of nine incorrect answers have added NOT NULL as part of the statement. Another three participants decided to alter the WHERE clause, adding punctuation points for the numeral. Apart from one unaltered statement, the rest of the incorrect answers changed the IS keyword to something else.

4.2.4 Incorrect syntax with LIKE keyword

Correct and incorrect statements can be seen in Figures 18 and 19. The user must first move the parenthesis to cover the statement correctly to correct this statement. Secondly, adding OR operator and stating different desired results separately with the help of the LIKE keyword.



```
SELECT id , name , status
FROM project
WHERE name LIKE ('H %', 'J %', 'K %')
AND manager_id IN
      ( SELECT id
      FROM employee
      WHERE sname = 'Smith ') ;
```

```
Error : ER_OPERAND_COLUMNS : Operand
should contain 1 column ( s )
```

```
SELECT id , name , status
FROM   project
WHERE  (name LIKE 'H %'
OR      name LIKE 'J %'
OR      name LIKE 'K %')
AND     manager_id IN
         ( SELECT id
        FROM employee
        WHERE sname = 'Smith ') ;
```

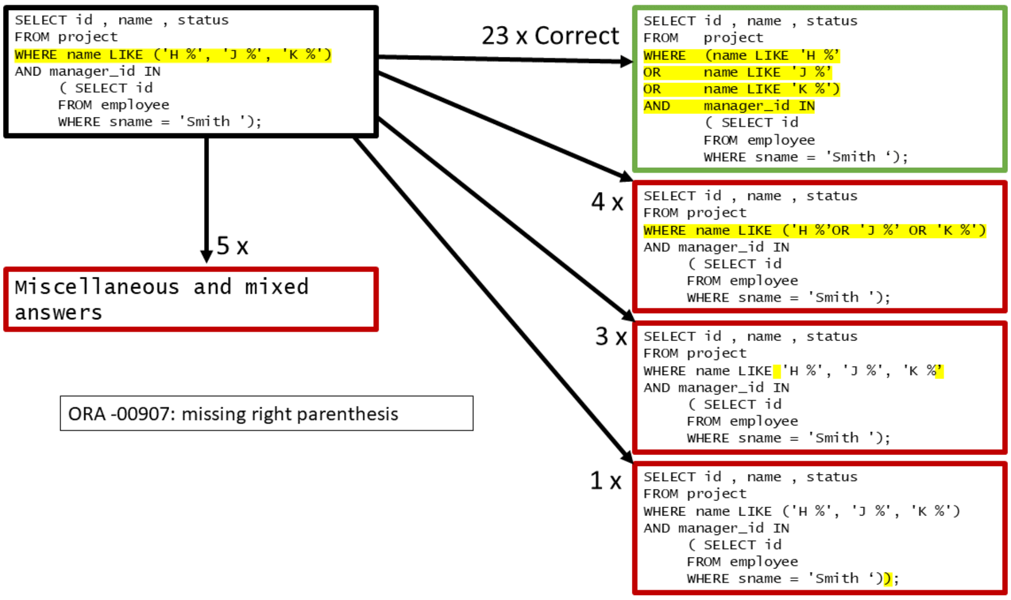
```
SELECT id , name , status
FROM project
WHERE name LIKE 'H %', 'J %', 'K %'
AND manager_id IN
      ( SELECT id
      FROM employee
      WHERE sname = 'Smith ') ;
```

```
SELECT id , name , status
FROM project
WHERE name LIKE ('H %' OR 'J %' OR'K %')
AND manager_id IN
      ( SELECT id
      FROM employee
      WHERE sname = 'Smith ') ;
```

```
Miscellaneous and mixed
answers
```

Figure 18 MySQL error message and results for statement #4

Error message by MySQL in Figure 18 points at operators and states that there should be one column. The message does not provide information on the location or clarify which operand. Three of the participants removed the parenthesis entirely. Only two users tried adding OR operator but failed to otherwise correct it. There is no visible theme in failed answers; the inputs are mostly different. Correct answers had multiple different variations. Some users deleted parenthesis, but the statement still provides wanted results.



```
SELECT id , name , status
FROM project
WHERE name LIKE ('H %', 'J %', 'K %')
AND manager_id IN
     ( SELECT id
     FROM employee
     WHERE sname = 'Smith ');
```

```
SELECT id , name , status
FROM   project
WHERE  (name LIKE 'H %'
OR     name LIKE 'J %'
OR     name LIKE 'K %')
AND    manager_id IN
       ( SELECT id
       FROM employee
       WHERE sname = 'Smith ‘);
```

```
SELECT id , name , status
FROM project
WHERE name LIKE ('H %’OR 'J %’ OR 'K %')
AND manager_id IN
     ( SELECT id
     FROM employee
     WHERE sname = 'Smith ');
```

```
SELECT id , name , status
FROM project
WHERE name LIKE 'H %', 'J %', 'K %’
AND manager_id IN
     ( SELECT id
     FROM employee
     WHERE sname = 'Smith ');
```

```
SELECT id , name , status
FROM project
WHERE name LIKE ('H %', 'J %', 'K %')
AND manager_id IN
     ( SELECT id
     FROM employee
     WHERE sname = 'Smith ‘));
```

Miscellaneous and mixed answers

ORA -00907: missing right parenthesis

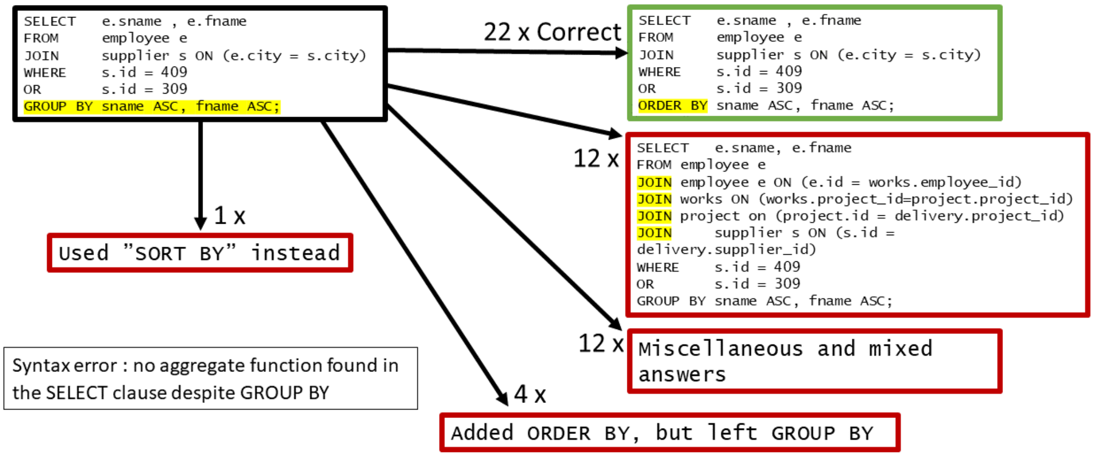
Figure 19 Oracle Database error message and results for statement #4

The error message seen in Figure 19 by Oracle Database informs the user that the right parenthesis is missing. It is possible to interpret the message so that the rightmost parenthesis is missing. The error message also fails to mention the operator or any other related information on locating the problem.

One user decided to add extra parenthesis to the end of the statement, which could be thought of as direct action suggested by the error message. Three users have removed the first pair of parentheses completely. Some users understand the error is related to the operators but cannot correctly change the statement. A large portion of the incorrect answers has no visible theme.

4.2.5 Confusing logic of the keywords

The incorrect statement shown in Figures 20 and 21 has the wrong keyword. Appendix A in Table 6 states that the results should be sorted, not grouped.



```
SELECT   e.sname , e.fname
FROM     employee e
JOIN     supplier s ON (e.city = s.city)
WHERE    s.id = 409
OR       s.id = 309
GROUP BY sname ASC, fname ASC;
```

```
SELECT   e.sname , e.fname
FROM     employee e
JOIN     supplier s ON (e.city = s.city)
WHERE    s.id = 409
OR       s.id = 309
ORDER BY sname ASC, fname ASC;
```

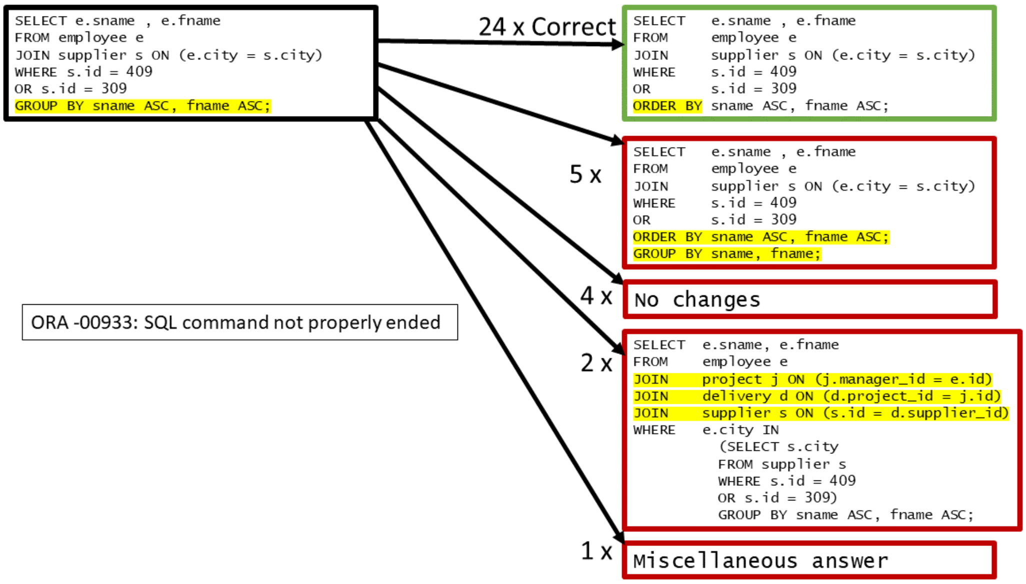
```
SELECT   e.sname, e.fname
FROM employee e
JOIN employee e ON (e.id = works.employee_id)
JOIN works ON (works.project_id=project.project_id)
JOIN project on (project.id = delivery.project_id)
JOIN     supplier s ON (s.id =
delivery.supplier_id)
WHERE    s.id = 409
OR       s.id = 309
GROUP BY sname ASC, fname ASC;
```

Syntax error : no aggregate function found in the SELECT clause despite GROUP BY

22 x Correct
12 x
12 x
Miscellaneous and mixed answers
1 x
Used "SORT BY" instead
4 x
Added ORDER BY, but left GROUP BY

Figure 20 MySQL error message and results for statement #5

Error message by MySQL in Figure 20 has GROUP BY, but the user must understand it is the wrong keyword for this statement; it does not provide the correct results. Twelve participants added more JOIN keywords. Twelve answers had no visible theme. Four of the answers had added correctly ORDER by but had not removed the incorrect GROUP BY. One participant had used SORT BY, which is not a SQL keyword. In Figure 20 presented miscellaneous and mixed answers; all had left the invalid GROUP BY.



```
SELECT e.sname , e.fname
FROM employee e
JOIN supplier s ON (e.city = s.city)
WHERE s.id = 409
OR s.id = 309
GROUP BY sname ASC, fname ASC;
```

```
SELECT   e.sname , e.fname
FROM     employee e
JOIN     supplier s ON (e.city = s.city)
WHERE    s.id = 409
OR       s.id = 309
ORDER BY sname ASC, fname ASC;
```

```
SELECT   e.sname , e.fname
FROM     employee e
JOIN     supplier s ON (e.city = s.city)
WHERE    s.id = 409
OR       s.id = 309
ORDER BY sname ASC, fname ASC;
GROUP BY sname, fname;
```

```
SELECT  e.sname, e.fname
FROM    employee e
JOIN    project j ON (j.manager_id = e.id)
JOIN    delivery d ON (d.project_id = j.id)
JOIN    supplier s ON (s.id = d.supplier_id)
WHERE   e.city IN
          (SELECT s.city
          FROM supplier s
          WHERE s.id = 409
          OR s.id = 309)
          GROUP BY sname ASC, fname ASC;
```

ORA -00933: SQL command not properly ended

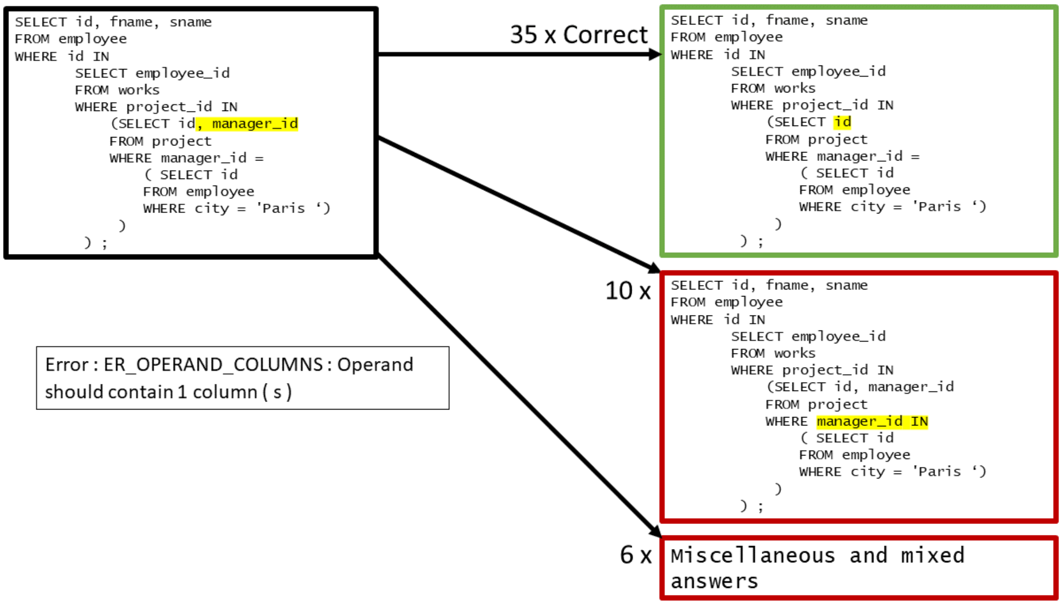
Figure 21 Oracle Database error message and results for statement #5

Few immediate notions can be made from the error message by Oracle Database seen in Figure 21. Firstly, the message implies that the statement is not correctly ended. Secondly, it does lead the user in the right direction, as the error is located near the end of the statement. The message could be considered vague but possibly not as misleading. Five of the users managed to add the ORDER

BY but forgot to remove the redundant GROUP BY. Four users made no changes. Two answers had added more JOINs similarly to MySQL participants.

4.2.6 Too many columns in the subquery

This erroneous query in Figures 22 and 23 is different from the prior. Instead of changing or adding parts of the statement, the user must remove a part of the middlemost SELECT clause. There can be only one column or attribute referred to in a subquery like this.



```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
             (SELECT id, manager_id
             FROM project
             WHERE manager_id =
                  ( SELECT id
                  FROM employee
                  WHERE city = 'Paris ')
             )
        ) ;
```

Error : ER_OPERAND_COLUMNS : Operand should contain 1 column (s)

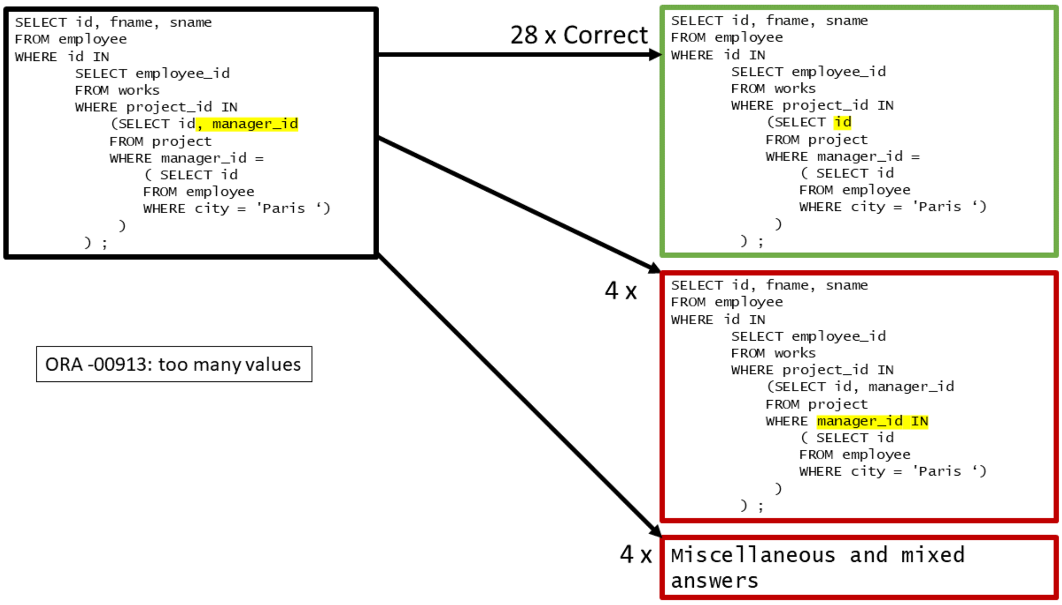
```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
             (SELECT id
             FROM project
             WHERE manager_id =
                  ( SELECT id
                  FROM employee
                  WHERE city = 'Paris ')
             )
        ) ;
```

```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
             (SELECT id, manager_id
             FROM project
             WHERE manager_id IN
                  ( SELECT id
                  FROM employee
                  WHERE city = 'Paris ')
             )
        ) ;
```

Miscellaneous and mixed answers

Figure 22 MySQL error message and results for statement #6

The error message by MySQL in Figure 22 provides information that the operator should contain only one column. The message is correct as IN operator has a redundant column on line seven. 10 out of 16 incorrect responses seen in Figure 22 changed the line 'WHERE manager_id = ' to 'WHERE manager_id IN' and failed to correct the actual error. There are six miscellaneous answers.



```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
            (SELECT id, manager_id
            FROM project
            WHERE manager_id =
                ( SELECT id
                FROM employee
                WHERE city = 'Paris ‘)
              )
          ) ;
```

ORA -00913: too many values

```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
            (SELECT id
            FROM project
            WHERE manager_id =
                ( SELECT id
                FROM employee
                WHERE city = 'Paris ‘)
              )
          ) ;
```

```
SELECT id, fname, sname
FROM employee
WHERE id IN
        SELECT employee_id
        FROM works
        WHERE project_id IN
            (SELECT id, manager_id
            FROM project
            WHERE manager_id IN
                ( SELECT id
                FROM employee
                WHERE city = 'Paris ‘)
              )
          ) ;
```

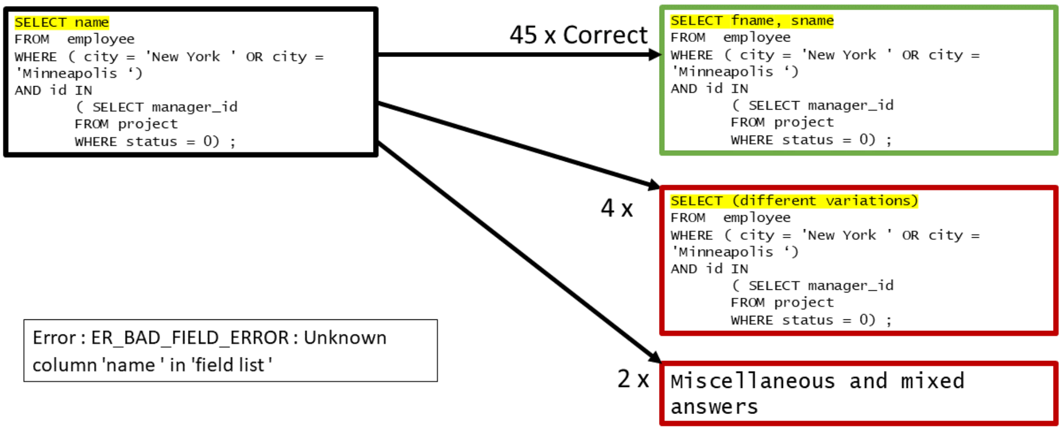
Miscellaneous and mixed answers

Figure 23 Oracle Database error message and results for statement #6

As seen in Figure 23 majority correctly altered the statement based on Oracle Databases error message. Like MySQL, most incorrect answers focused on the different operator ‘=’ and proceeded to ignore the rest of the statement. Two of the answers could be considered unfinished, and two of the answers have changed the statement otherwise incorrectly.

4.2.7 Undefined column

This statement has an incorrect attribute/column name in the SELECT clause. The user must check the database schema to identify the correct attribute naming (appendix A). In addition, the database has first and last names separately, requiring both columns in the SELECT clause to have full names in the results, as can be seen in Figures 24 and 25.



```
SELECT name
FROM  employee
WHERE ( city = 'New York ' OR city =
'Minneapolis ‘)
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0) ;
```

Error : ER_BAD_FIELD_ERROR : Unknown column 'name ' in 'field list '

```
SELECT fname, sname
FROM  employee
WHERE ( city = 'New York ' OR city =
'Minneapolis ‘)
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0) ;
```

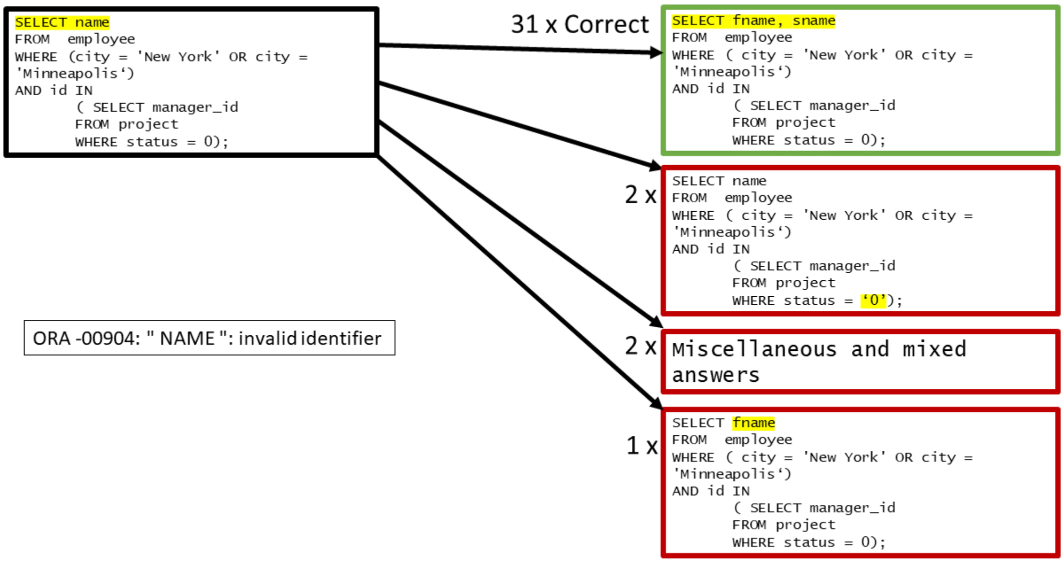
```
SELECT (different variations)
FROM  employee
WHERE ( city = 'New York ' OR city =
'Minneapolis ‘)
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0) ;
```

Miscellaneous and mixed answers

Figure 24 MySQL error message and results for statement #7

Error message from MySQL is accurate. As can be seen in Figure 24 error message describes the problem perfectly, stating that the DBMS does not recognize the column, guiding the user to check column names. It also points directly to ‘name,’ providing additional information on the location of the error.

From wrong answers, there are four that aim to correct the proper clause on the statement but are unable to. These users might understand the problem but do not have a clear idea of the correct phrasing. For example, one user has written only ‘fname,’ which is half correct but not entirely. Assignment in appendix A does not state that a full name is required.



```
SELECT name
FROM  employee
WHERE (city = 'New York' OR city =
'Minneapolis')
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0);
```

ORA -00904: " NAME ": invalid identifier

```
SELECT fname, sname
FROM  employee
WHERE ( city = 'New York' OR city =
'Minneapolis')
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0);
```

```
SELECT name
FROM  employee
WHERE ( city = 'New York' OR city =
'Minneapolis')
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = ‘0’);
```

Miscellaneous and mixed answers

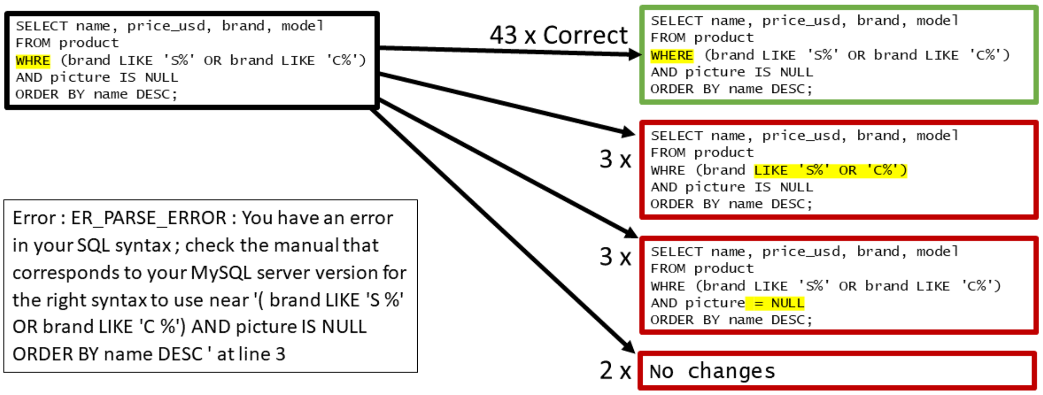
```
SELECT fname
FROM  employee
WHERE ( city = 'New York' OR city =
'Minneapolis')
AND id IN
        ( SELECT manager_id
        FROM project
        WHERE status = 0);
```

Figure 25 Oracle Database error message and results for statement #7

Like MySQL's error message, Oracle Database offers a descriptive error message, as shown in Figure 25. The location and the problem can be identified from the message. The message lacks the word ‘column,’ but the statement only has the word “name” in one part, narrowing the error’s location. Only one user has changed the SELECT clause but fails to specify both columns. The most common mistake among wrong answers is adding single quotation marks around the numeral at the end.

4.2.8 A common syntax error: typographic error

This statement has a typographic error shown in Figures 26 and 27, a common mistake in SQL syntax. Correcting this should not require much expertise as with basic English knowledge, users can identify misspelled word. The user must change “WHRE” to the correct form “WHERE.”



```
SELECT name, price_usd, brand, model
FROM product
WHRE (brand LIKE 'S%' OR brand LIKE 'C%')
AND picture IS NULL
ORDER BY name DESC;
```

Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check the manual that corresponds to your MySQL server version for the right syntax to use near '(brand LIKE 'S %' OR brand LIKE 'C %') AND picture IS NULL ORDER BY name DESC ' at line 3

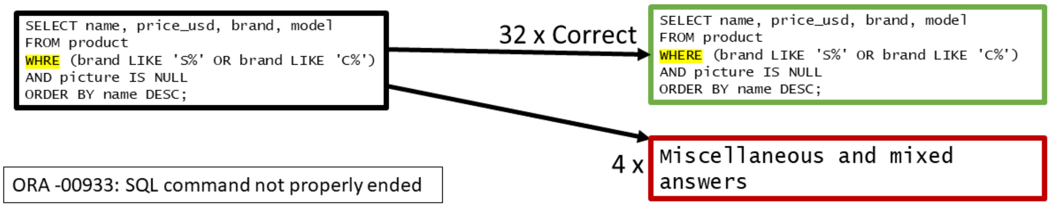
```
SELECT name, price_usd, brand, model
FROM product
WHERE (brand LIKE 'S%' OR brand LIKE 'C%')
AND picture IS NULL
ORDER BY name DESC;
```

```
SELECT name, price_usd, brand, model
FROM product
WHRE (brand LIKE 'S%' OR 'C%')
AND picture IS NULL
ORDER BY name DESC;
```

```
SELECT name, price_usd, brand, model
FROM product
WHRE (brand LIKE 'S%' OR brand LIKE 'C%')
AND picture = NULL
ORDER BY name DESC;
```

Figure 26 MySQL error message and results for statement #8

As shown in Figure 26, the error message by MySQL is long and contains a lot of information. It states that there is a syntax error, which is correct and refers to a manual. The message points to line 3 and has a quote of the clause. The error message provides a quote from the part after the error, after the WHERE keyword. Three of the eight incorrect answers modified the part quoted in the error message. The other Three participants had changed the IS null in various ways. The last two of the answers had no modifications.



```
SELECT name, price_usd, brand, model
FROM product
WHRE (brand LIKE 'S%' OR brand LIKE 'C%')
AND picture IS NULL
ORDER BY name DESC;
```

ORA -00933: SQL command not properly ended

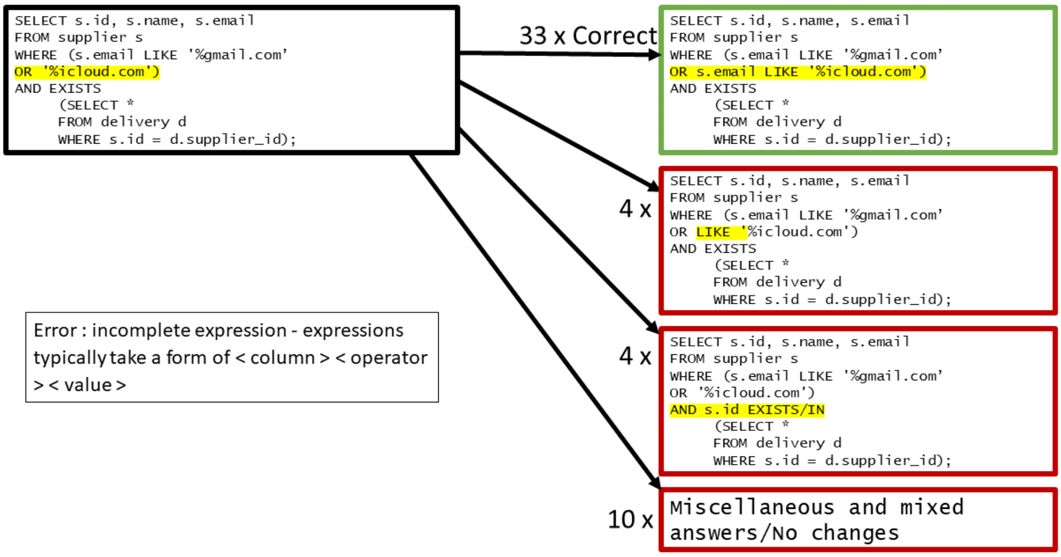
```
SELECT name, price_usd, brand, model
FROM product
WHERE (brand LIKE 'S%' OR brand LIKE 'C%')
AND picture IS NULL
ORDER BY name DESC;
```

Figure 27 Oracle Database error message and results for statement #8

In comparison to MySQL, Oracle Database gives a much shorter error message, as seen in Figure 27. Error message guides the user to check whether the SQL command has been ended correctly. It does not give the correct location or error as there is no missing semicolon or error even near the ending of the statement. Figure 27 presents the answers given. The incorrect answers have little in common. One answer is empty, and the other three have different minor modifications.

4.2.9 Confusing syntax of keyword LIKE

This statement needs another condition to go with the OR operator. In SQL, the OR operator is used to combine two Boolean expressions. In this example shown in Figures 28 and 29, the statement lacks the Boolean expression on the other side of the OR operator.



```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR '%icloud.com')
AND EXISTS
     (SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id);
```

Error : incomplete expression - expressions typically take a form of < column > < operator > < value >

```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR s.email LIKE '%icloud.com')
AND EXISTS
     (SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id);
```

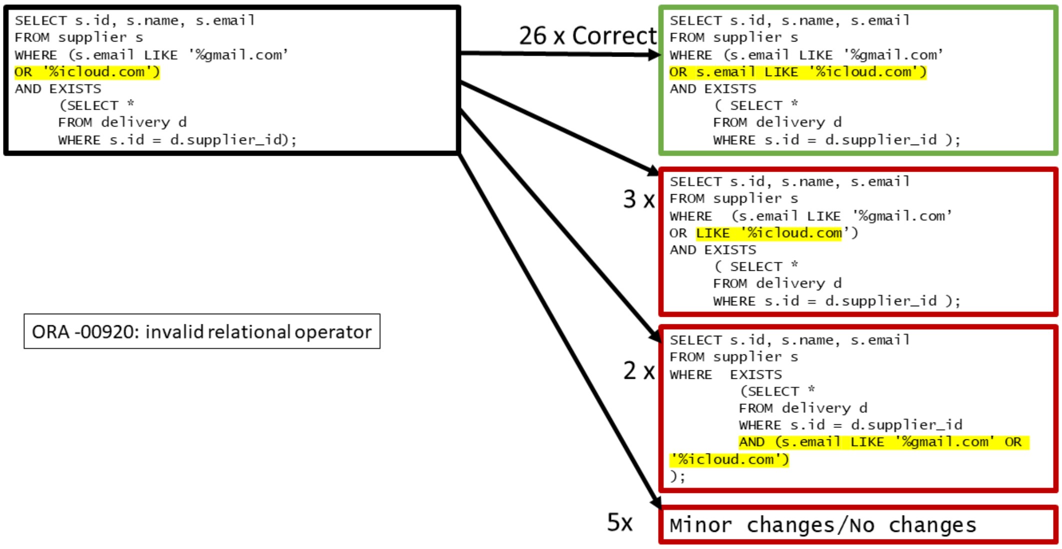
```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR LIKE '%icloud.com')
AND EXISTS
     (SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id);
```

```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR '%icloud.com')
AND s.id EXISTS/IN
     (SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id);
```

Miscellaneous and mixed answers/No changes

Figure 28 MySQL error message and results for statement #9

As seen in Figure 28, the error message by MySQL provides a lot of information and even guides on the correct syntax. The user is provided with the syntax to use with the OR operator. The message fails to include what expression is incomplete. Results are divisive, with only some noticeable trends. Four users identified the error in the correct part but could not correct it. For example, they add only another LIKE but no column name (s.email). Another four users have altered lines with AND EXISTS, and multiple users had added s.id before EXISTS keyword.



```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR '%icloud.com')
AND EXISTS
     (SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id);
```

ORA -00920: invalid relational operator

```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE (s.email LIKE '%gmail.com'
OR s.email LIKE '%icloud.com')
AND EXISTS
     ( SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id );
```

```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE   (s.email LIKE '%gmail.com'
OR LIKE '%icloud.com')
AND EXISTS
     ( SELECT *
     FROM delivery d
     WHERE s.id = d.supplier_id );
```

```
SELECT s.id, s.name, s.email
FROM supplier s
WHERE  EXISTS
          (SELECT *
          FROM delivery d
          WHERE s.id = d.supplier_id
          AND (s.email LIKE '%gmail.com' OR
'%icloud.com')
);
```

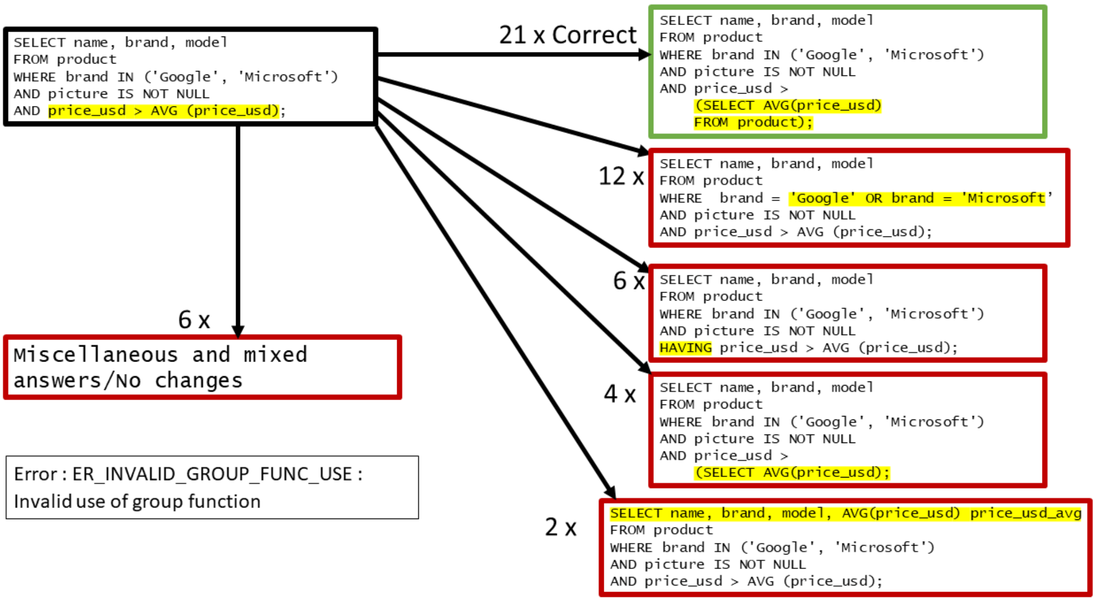
Minor changes/No changes

Figure 29 Oracle Database error message and results for statement #9

OR keyword is a relational operator. The error message by Oracle Database seen in Figure 29 provides misinformation because the operator is not incorrect, but the statement is otherwise incomplete. The message is relatively short and does not provide additional helpful information. Nothing conclusive is challenging to draw from the results as only three users made changes to the correct part but had similar mistakes as users exposed to the MySQL error message. Five of the incorrect answers were unchanged or had only removed parentheses. Interestingly, two participants moved the part with the OR operator to the end of the statement, but there is no connection between the action and the error message.

4.2.10 Illegal aggregate function placement

Aggregate functions like AVG (average) are used with the SELECT clause. Compared to other statements, this requires significant changes seen in Figures 30 and 31.



```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd > AVG (price_usd);
```

```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd >
     (SELECT AVG(price_usd)
     FROM product);
```

```
SELECT name, brand, model
FROM product
WHERE  brand = 'Google' OR brand = 'Microsoft'
AND picture IS NOT NULL
AND price_usd > AVG (price_usd);
```

```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
HAVING price_usd > AVG (price_usd);
```

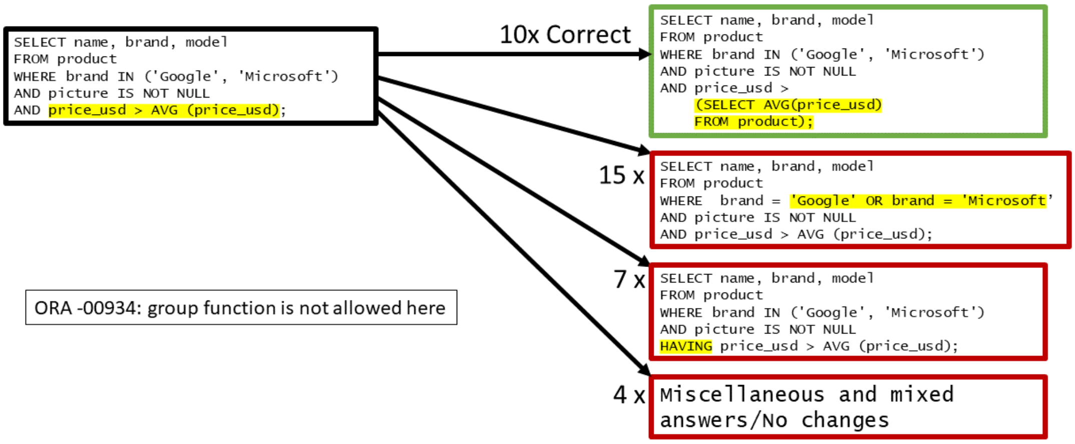
```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd >
     (SELECT AVG(price_usd);
```

```
SELECT name, brand, model, AVG(price_usd) price_usd_avg
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd > AVG (price_usd);
```

Error : ER_INVALID_GROUP_FUNC_USE :
Invalid use of group function

Figure 30 MySQL error message and results for statement #10

Aggregate functions are also called group functions. In Figure 30, the MySQL error message points directly at the statement's only group function (AVG). Otherwise, the message does not give information on the type of error or a more specific location. A considerable portion of 12 users altered the WHERE clause, adding OR keywords or other modifications. Six answers had changed HAVING on the last line. Only four incorrect answers either altered the ending and added SELECT or had the aggregate function elsewhere but did so incorrectly. Two answers had modified the first SELECT clause by adding an aggregate function there. The rest of the answers did not make any changes or made a unique change.



```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd > AVG (price_usd);
```

ORA -00934: group function is not allowed here

```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
AND price_usd >
    (SELECT AVG(price_usd)
    FROM product);
```

```
SELECT name, brand, model
FROM product
WHERE  brand = 'Google' OR brand = 'Microsoft'
AND picture IS NOT NULL
AND price_usd > AVG (price_usd);
```

```
SELECT name, brand, model
FROM product
WHERE brand IN ('Google', 'Microsoft')
AND picture IS NOT NULL
HAVING price_usd > AVG (price_usd);
```

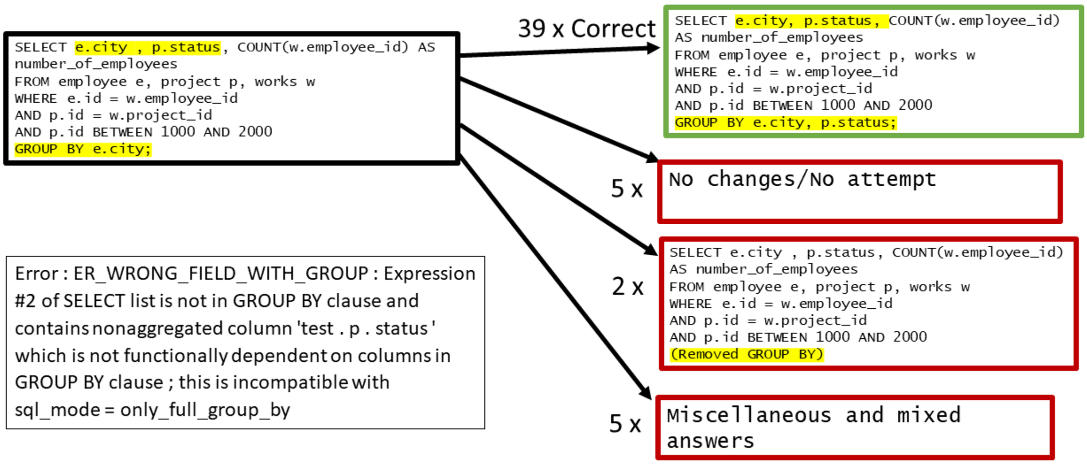
```
Miscellaneous and mixed
answers/No changes
```

Figure 31 Oracle Database error message and results for statement #10

Same as for MySQL, the aggregate function is named group function in error message by Oracle Database, as shown in Figure 31. The error message is controversial, implying that the function is not allowed. Compared to the MySQL "invalid use," the message by Oracle Database has a different approach. Most users had changed something inside the WHERE clause, not near the group function. Only one participant removed the aggregate function. Seven users added the HAVING keyword, which does not correlate with the error message or the statement. Only three answers had something related to the correct answer, all of them added a SELECT clause near the AVG function, but they included other incorrect changes.

4.2.11 Grouping error, omitting column from GROUP BY

The statement requires that all columns not handled by an aggregate function are included in the GROUP BY clause. The user must add 'p.status' to the GROUP BY clause to correct this. The correct and incorrect statements can be seen in Figures 32 and 33.



```
SELECT e.city , p.status, COUNT(w.employee_id) AS
number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
GROUP BY e.city;
```

Error : ER_WRONG_FIELD_WITH_GROUP : Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'test . p . status' which is not functionally dependent on columns in GROUP BY clause ; this is incompatible with sql_mode = only_full_group_by

```
SELECT e.city, p.status, COUNT(w.employee_id)
AS number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
GROUP BY e.city, p.status;
```

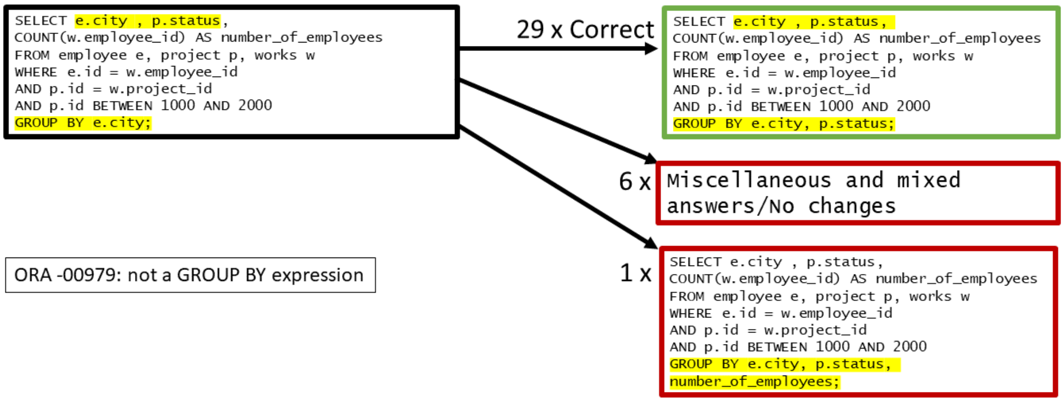
```
No changes/No attempt
```

```
SELECT e.city , p.status, COUNT(w.employee_id)
AS number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
(Removed GROUP BY)
```

```
Miscellaneous and mixed
answers
```

Figure 32 MySQL error message and results for statement #11

The error message for this statement by MySQL, seen in Figure 32, contains information on the location of the error, the type of the error, and hints of even possible solution. The message states that the second expression inside the SELECT clause is not in the GROUP BY statement. The error message contains more information and refers to 'test.p.status' as a nonaggregate column and informs that the column is not functionally dependent on columns in the GROUP BY clause. The success rate is high, implying that most participants have located and corrected the error based on the message or through their observations. The incorrect statements do not have an evident theme, apart from the 5 of the statements being unchanged or left blank. Only two users removed the GROUP BY clause completely.



```
SELECT e.city , p.status,
COUNT(w.employee_id) AS number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
GROUP BY e.city;
```

ORA -00979: not a GROUP BY expression

29 x Correct

```
SELECT e.city , p.status,
COUNT(w.employee_id) AS number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
GROUP BY e.city, p.status;
```

6 x Miscellaneous and mixed answers/No changes

1 x

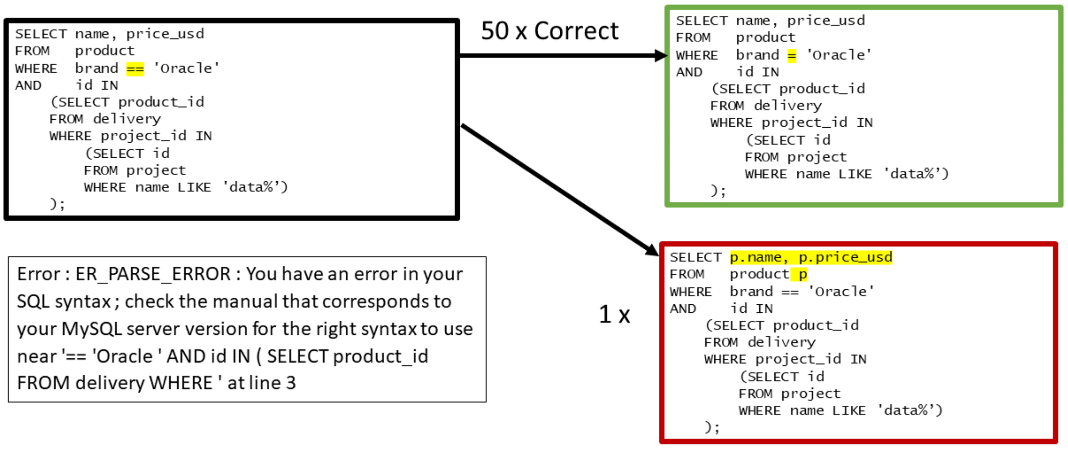
```
SELECT e.city , p.status,
COUNT(w.employee_id) AS number_of_employees
FROM employee e, project p, works w
WHERE e.id = w.employee_id
AND p.id = w.project_id
AND p.id BETWEEN 1000 AND 2000
GROUP BY e.city, p.status,
number_of_employees;
```

Figure 33 Oracle Database error message and results for statement #11

The message provided by Oracle Database in Figure 33 is shorter than the MySQL counterpart for the same incorrect statement. The information is controversial as it informs that something is not a GROUP by expression. It gives a vague hint, but in the right direction, if the user decides to check the GROUP BY clause. Of the seven incorrect answers, only two participants left the statement unchanged, and the rest had a diverse set of changes. One participant correctly added the column 'p.status' to the GROUP BY clause and the 'number_of_employees.'

4.2.12 Nonstandard operator "=="

The statement incorrectly uses the operator shown in Figures 34 and 35. Double '=' is sometimes used in programming to compare values. In SQL operator for checking whether the values are equal is single '='.



```
SELECT name, price_usd
FROM   product
WHERE  brand == 'Oracle'
AND     id IN
    (SELECT product_id
    FROM delivery
    WHERE project_id IN
        (SELECT id
        FROM project
        WHERE name LIKE 'data%')
    );
```

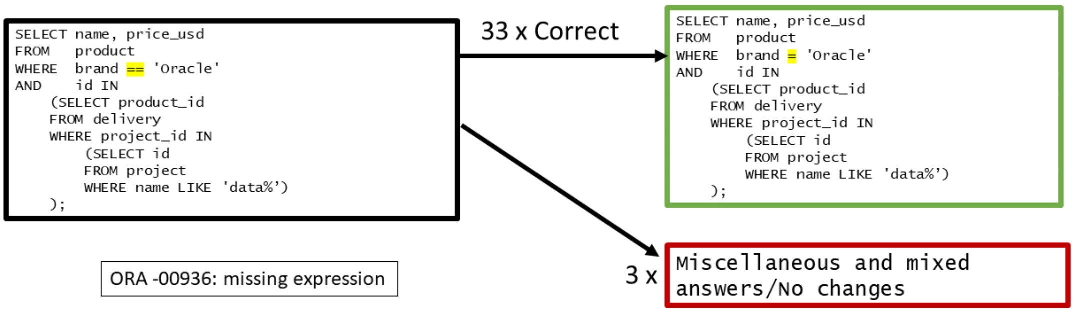
Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check the manual that corresponds to your MySQL server version for the right syntax to use near '== 'Oracle ' AND id IN (SELECT product_id FROM delivery WHERE ' at line 3

```
SELECT name, price_usd
FROM   product
WHERE  brand = 'Oracle'
AND     id IN
    (SELECT product_id
    FROM delivery
    WHERE project_id IN
        (SELECT id
        FROM project
        WHERE name LIKE 'data%')
    );
```

```
SELECT p.name, p.price_usd
FROM   product p
WHERE  brand == 'Oracle'
AND     id IN
    (SELECT product_id
    FROM delivery
    WHERE project_id IN
        (SELECT id
        FROM project
        WHERE name LIKE 'data%')
    );
```

Figure 34 MySQL error message and results for statement #12

The error message provided by MySQL in Figure 34 correctly locates the error in line 3. The message also has this vague hint that the error is near the quoted part. In this case, the erroneous part is included in the quoted part. The success rate is high. Only one user failed to correct the statement. The only incorrect answer was unable to change the double '=', only changing the alias of the columns.



```
SELECT name, price_usd
FROM   product
WHERE  brand == 'Oracle'
AND     id IN
    (SELECT product_id
    FROM delivery
    WHERE project_id IN
        (SELECT id
        FROM project
        WHERE name LIKE 'data%')
    );
```

ORA -00936: missing expression

```
SELECT name, price_usd
FROM   product
WHERE  brand = 'Oracle'
AND     id IN
    (SELECT product_id
    FROM delivery
    WHERE project_id IN
        (SELECT id
        FROM project
        WHERE name LIKE 'data%')
    );
```

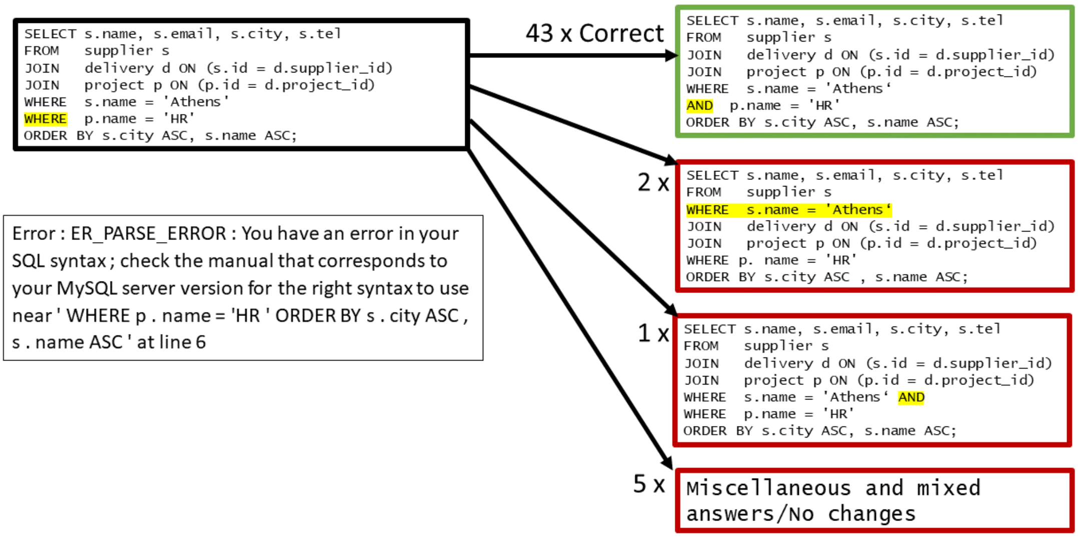
Miscellaneous and mixed answers/No changes

Figure 35 Oracle Database error message and results for statement #12

As seen in Figure 35, an error message by Oracle Database refers to a missing expression. The expression could be a Boolean, like in this case. It should include values, operators, and functions. The error message is correct but vague as the statement already contains multiple expressions. A small number of incorrect answers, similarly to MySQL counterpart. All three wrong answers have unique differences. Including one where '==' is changed to IS LIKE.

4.2.13 Using WHERE twice

This statement has an error as the keyword WHERE has been used twice with a single SELECT clause. The user must change the second WHERE to an AND keyword to correct this, as shown in Figures 36 and 37.



```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE  s.name = 'Athens'
WHERE  p.name = 'HR'
ORDER BY s.city ASC, s.name ASC;
```

Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check the manual that corresponds to your MySQL server version for the right syntax to use near ' WHERE p . name = 'HR ' ORDER BY s . city ASC , s . name ASC ' at line 6

```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE  s.name = 'Athens'
AND  p.name = 'HR'
ORDER BY s.city ASC, s.name ASC;
```

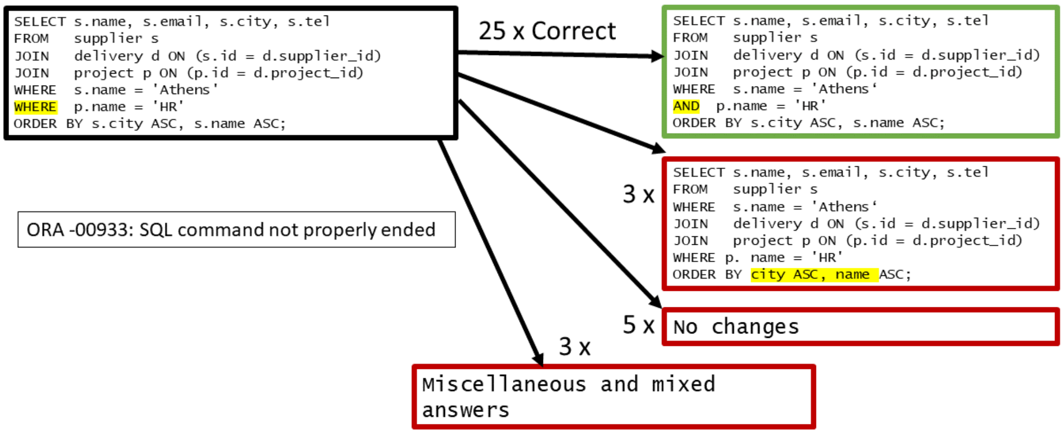
```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
WHERE  s.name = 'Athens'
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE p. name = 'HR'
ORDER BY s.city ASC , s.name ASC;
```

```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE  s.name = 'Athens' AND
WHERE  p.name = 'HR'
ORDER BY s.city ASC, s.name ASC;
```

Miscellaneous and mixed answers/No changes

Figure 36 MySQL error message and results for statement #13

The error message seen in Figure 36 by MySQL contains valid information and guides the user to check the manual. The location of the error is correct. In this case, the quoted part contains the incorrect keyword. Incorrect answers have no remarkable consistency. Only two of the responses have similarly changed the order of JOIN and WHERE statements. Only one participant managed to add the missing AND keyword but left the redundant WHERE. Five answers had unique differences or no changes.



```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE  s.name = 'Athens'
WHERE  p.name = 'HR'
ORDER BY s.city ASC, s.name ASC;
```

ORA -00933: SQL command not properly ended

```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE  s.name = 'Athens'
AND  p.name = 'HR'
ORDER BY s.city ASC, s.name ASC;
```

```
SELECT s.name, s.email, s.city, s.tel
FROM   supplier s
WHERE  s.name = 'Athens'
JOIN   delivery d ON (s.id = d.supplier_id)
JOIN   project p ON (p.id = d.project_id)
WHERE p. name = 'HR'
ORDER BY city ASC, name ASC;
```

No changes

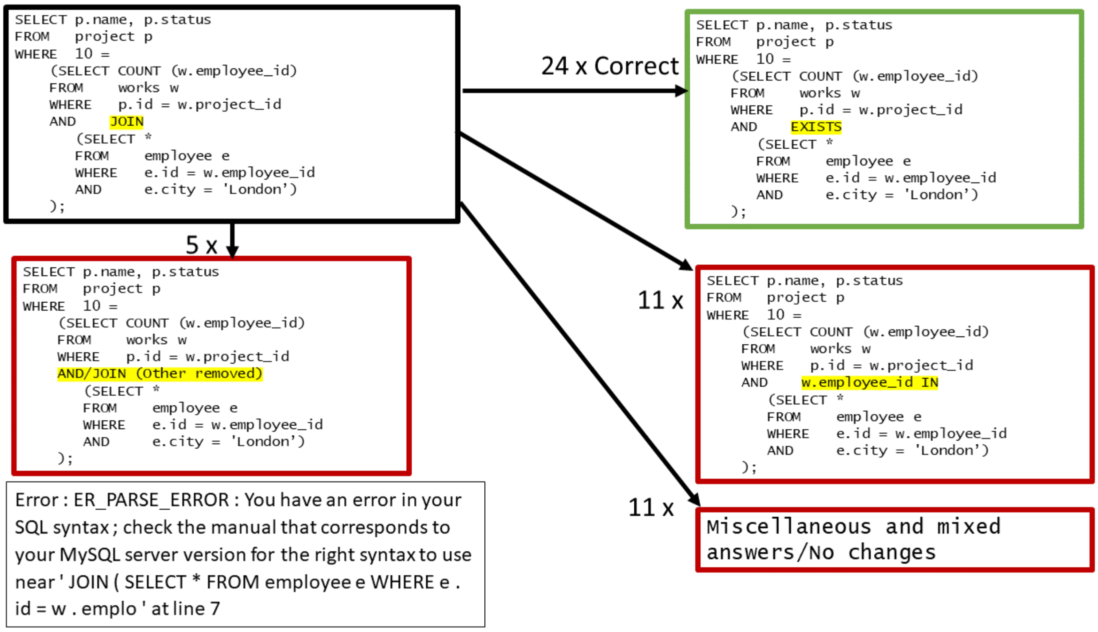
Miscellaneous and mixed answers

Figure 37 Oracle Database error message and results for statement #13

The error message in Figure 37 by Oracle Database does not give accurate information on the error location. It does provide a hint to the ending of the statement. Success rate could be considered significantly lower compared to MySQL. Five of the incorrect answers were unaltered or left empty. Three users had changed the ORDER BY clause of the statement.

4.2.14 Confusing the keywords JOIN and EXISTS

The statement requires a minor, one-word change, shown in Figures 38 and 39. The JOIN keyword is used if there is a need for the results from the other table, but in this case, EXISTS keyword is correct as there is only a need to check whether the employee is from London. JOIN and EXISTS keywords have different syntax.



```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM     works w
    WHERE    p.id = w.project_id
    AND      JOIN
       (SELECT *
       FROM     employee e
       WHERE    e.id = w.employee_id
       AND      e.city = 'London')
    );
```

```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM     works w
    WHERE    p.id = w.project_id
    AND      EXISTS
       (SELECT *
       FROM     employee e
       WHERE    e.id = w.employee_id
       AND      e.city = 'London')
    );
```

```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM     works w
    WHERE    p.id = w.project_id
    AND/JOIN (Other removed)
       (SELECT *
       FROM     employee e
       WHERE    e.id = w.employee_id
       AND      e.city = 'London')
    );
```

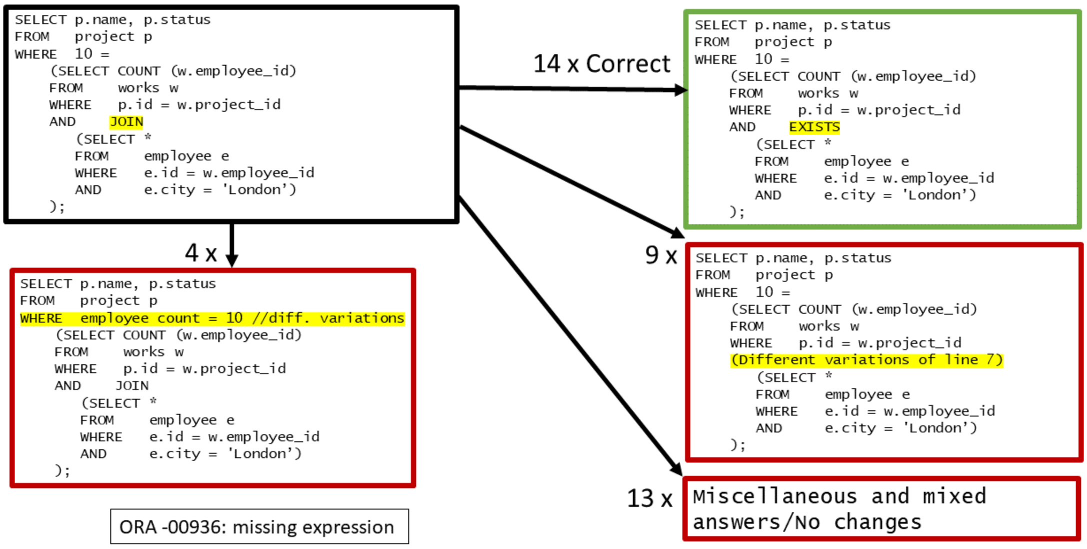
Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check the manual that corresponds to your MySQL server version for the right syntax to use near ' JOIN (SELECT * FROM employee e WHERE e . id = w . emplo ' at line 7

```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM     works w
    WHERE    p.id = w.project_id
    AND      w.employee_id IN
       (SELECT *
       FROM     employee e
       WHERE    e.id = w.employee_id
       AND      e.city = 'London')
    );
```

Miscellaneous and mixed answers/No changes

Figure 38 MySQL error message and results for statement #14

The error message by MySQL seen in Figure 38 contains the correct line of the error. The message also hints that the error is near the quoted part. Sixteen of the incorrect answer did alter line seven which included the wrong keyword. 11 of them added IN keyword with multiple different variations. Five of the incorrect answers had either removed AND or JOIN. The rest of the incorrect answers had no significant theme.



```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM    works w
    WHERE   p.id = w.project_id
    AND     JOIN
       (SELECT *
       FROM    employee e
       WHERE   e.id = w.employee_id
       AND     e.city = 'London')
    );
```

```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM    works w
    WHERE   p.id = w.project_id
    AND     EXISTS
       (SELECT *
       FROM    employee e
       WHERE   e.id = w.employee_id
       AND     e.city = 'London')
    );
```

```
SELECT p.name, p.status
FROM   project p
WHERE  10 =
    (SELECT COUNT (w.employee_id)
    FROM    works w
    WHERE   p.id = w.project_id
    (Different variations of line 7)
       (SELECT *
       FROM    employee e
       WHERE   e.id = w.employee_id
       AND     e.city = 'London')
    );
```

```
SELECT p.name, p.status
FROM   project p
WHERE  employee count = 10 //diff. variations
    (SELECT COUNT (w.employee_id)
    FROM    works w
    WHERE   p.id = w.project_id
    AND     JOIN
       (SELECT *
       FROM    employee e
       WHERE   e.id = w.employee_id
       AND     e.city = 'London')
    );
```

Miscellaneous and mixed answers/No changes

ORA -00936: missing expression

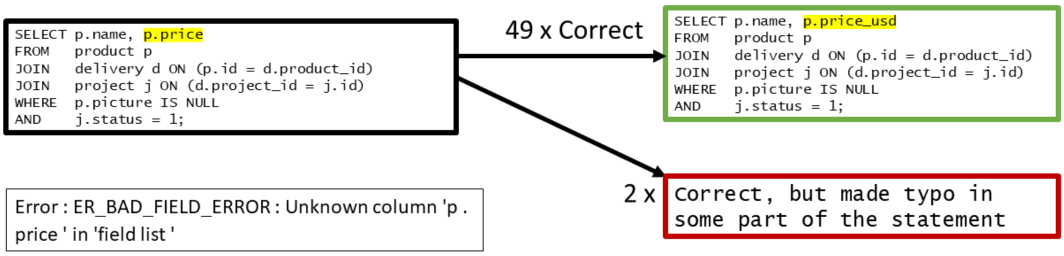
Figure 39 Oracle Database error message and results for statement #14

The message provided by Oracle Database in Figure 39 does not contain exact information about the error. Expressions in SQL change based on the user's needs and can be modified in multiple ways. There is no information on what expression is missing and from which part of the statement.

Observations seen in Figure 39 made from Oracle Database incorrect answers are like the results of MySQL. Nine users changed line seven by removing it, removing part of it, or adding keywords and column names. There is no cohesive theme. Four users had changed the first WHERE clause, but all with unique modifications. Over half of the answers (13) altered the wrong part or made no changes.

4.2.15 Misspelled column/attribute name

This statement has an incorrectly named column in the SELECT clause shown in Figures 40 and 41. This requires the user to check the correct naming of the column or attribute. In this case, the right is “p.price_usd” instead of “p.price”.



```
SELECT p.name, p.price
FROM   product p
JOIN   delivery d ON (p.id = d.product_id)
JOIN   project j ON (d.project_id = j.id)
WHERE  p.picture IS NULL
AND    j.status = 1;
```

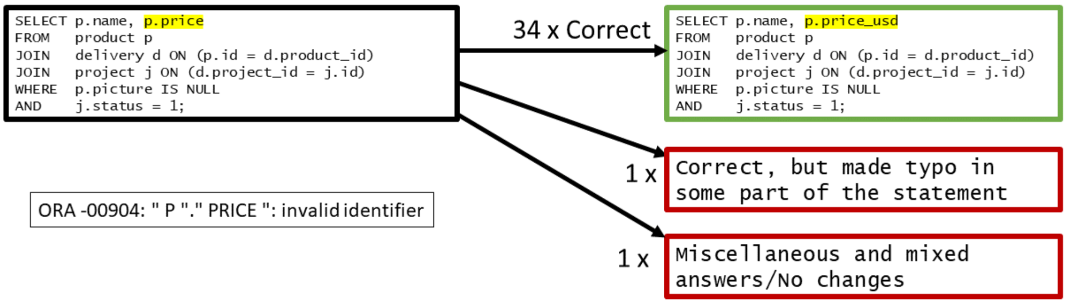
```
SELECT p.name, p.price_usd
FROM   product p
JOIN   delivery d ON (p.id = d.product_id)
JOIN   project j ON (d.project_id = j.id)
WHERE  p.picture IS NULL
AND    j.status = 1;
```

Correct, but made typo in some part of the statement

Error : ER_BAD_FIELD_ERROR : Unknown column 'p . price ' in 'field list '

Figure 40 MySQL error message and results for statement #15

The error message provided by the MySQL seen in Figure 40 for this erroneous statement is informative and exact. It informs the user that there is a column that cannot be identified. It also specifies that “p.price” is incorrect. Based on observations of the incorrect answers, the success rate could be 100%. There are only two incorrect answers, and both have changed the column name correctly but have typographic errors in the statement.



```
SELECT p.name, p.price
FROM   product p
JOIN   delivery d ON (p.id = d.product_id)
JOIN   project j ON (d.project_id = j.id)
WHERE  p.picture IS NULL
AND    j.status = 1;
```

ORA -00904: " P "." PRICE ": invalid identifier

```
SELECT p.name, p.price_usd
FROM   product p
JOIN   delivery d ON (p.id = d.product_id)
JOIN   project j ON (d.project_id = j.id)
WHERE  p.picture IS NULL
AND    j.status = 1;
```

Correct, but made typo in some part of the statement

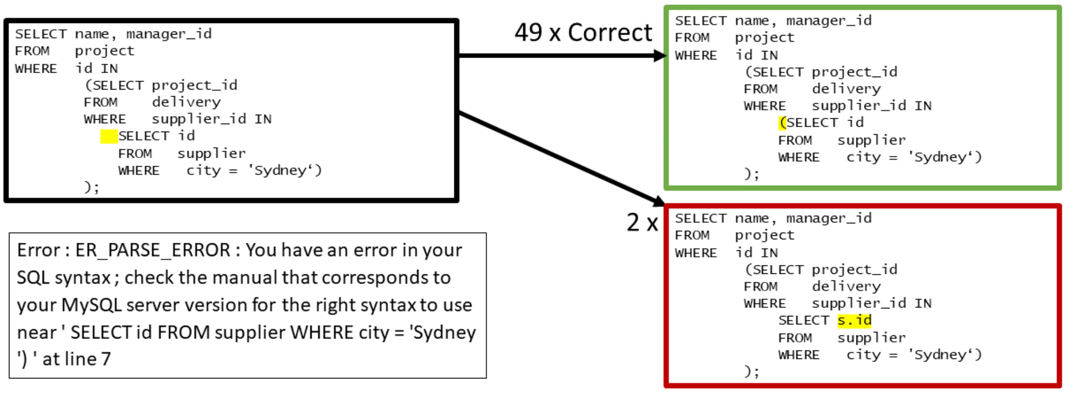
Miscellaneous and mixed answers/No changes

Figure 41 Oracle Database error message and results for statement #15

Similar to MySQL, Oracle Database provides accurate information about the error, as shown in Figure 41. The message specifies the incorrect column. Objectively, the error message by Oracle Database is more confusing and less polished with multiple quotation marks and dividing the wrong column name in two. Observing the incorrect answer provide little to discuss. Another response has a typographic error, and another has a change made that seemingly has no connection to the error message.

4.2.16 Missing parenthesis

This statement includes a common mistake and a missing parenthesis shown in Figures 42 and 43. Users must pay attention to the parentheses and notice that one parenthesis does not have a pair.



```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
         FROM    delivery
         WHERE   supplier_id IN
              SELECT id
              FROM   supplier
              WHERE   city = 'Sydney‘)
        );
```

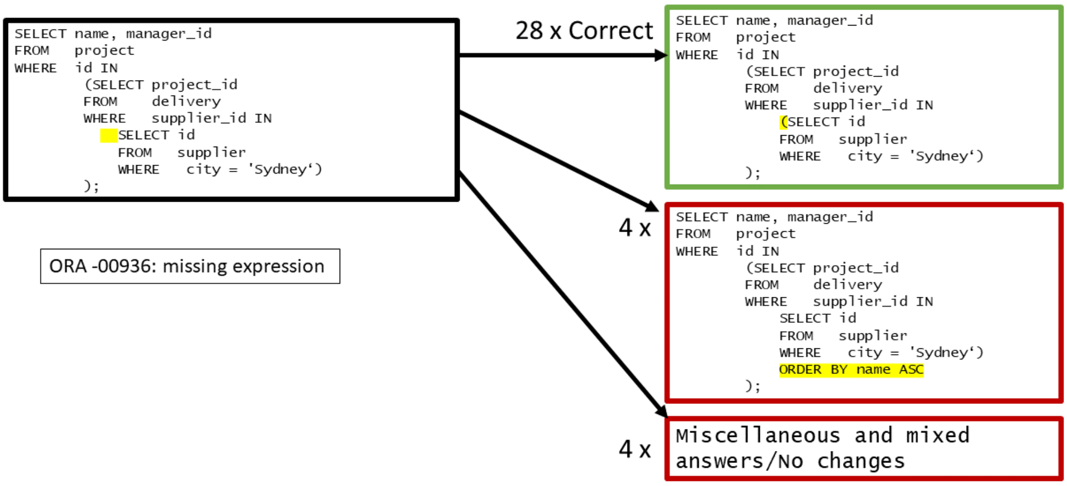
Error : ER_PARSE_ERROR : You have an error in your SQL syntax ; check the manual that corresponds to your MySQL server version for the right syntax to use near ' SELECT id FROM supplier WHERE city = 'Sydney ') ' at line 7

```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
         FROM    delivery
         WHERE   supplier_id IN
              (SELECT id
              FROM   supplier
              WHERE   city = 'Sydney‘)
        );
```

```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
         FROM    delivery
         WHERE   supplier_id IN
              SELECT s.id
              FROM   supplier
              WHERE   city = 'Sydney‘)
        );
```

Figure 42 MySQL error message and results for statement #16

The error message provided by MySQL in Figure 42 quotes near the erroneous part. It accurately informs the correct line. There is no indication of the type of error. Both incorrect answers have altered the correct line but failed to make the right change. Both users changed “id” to “s.id”. There is no apparent connection, besides the correct line, to the error message.



```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
        FROM    delivery
        WHERE   supplier_id IN
            SELECT id
            FROM   supplier
            WHERE   city = 'Sydney‘)
        );
```

ORA -00936: missing expression

```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
        FROM    delivery
        WHERE   supplier_id IN
            (SELECT id
            FROM   supplier
            WHERE   city = 'Sydney‘)
        );
```

```
SELECT name, manager_id
FROM   project
WHERE  id IN
        (SELECT project_id
        FROM    delivery
        WHERE   supplier_id IN
            SELECT id
            FROM   supplier
            WHERE   city = 'Sydney‘)
            ORDER BY name ASC
        );
```

Miscellaneous and mixed answers/No changes

Figure 43 Oracle Database error message and results for statement #16

Compared directly to the corresponding error message by MySQL, the one provided by Oracle Database is shorter and provides less information, as can be seen in Figure 43. The error message could be considered misleading, as the statement does not lack a whole expression. There is no information on the location or the type of error. Four users had added the ORDER BY clause to the end part of the statement. Interestingly all the four added the same clause, but it is not an expression. Otherwise, the other incorrect answers had little in common.

4.3 Summary of findings

There is a disparity between the success rates. The same statements could have different success rates depending on the DBMS that provides the error message. It is possible to recognize patterns and similarities in incorrect answers.

Statement 1 had a better success rate in cases of MySQL. Only two answers had a similar theme, and both could be considered half correct. Oracle Database answers had 9 with similar incorrect submits. The changes were like MySQL's incorrect answers.

There is little to interpret from **statement 2** answers. For both DBMS, the success rate is nearly perfect. MySQL had only one incorrect answer, which did have the correct change, but unnecessarily changed another part of the statement, which caused it to become faulty again.

For **statement 3,** MySQL had only six incorrect answers with no noticeable theme. The Oracle Database answers make it possible to identify three groups of similarly themed wrong answers. The groups are relatively small.

In the case of **statement 4,** there was a large portion of miscellaneous answers for both DBMSs. Both had one similar group of incorrect answers where the user had deleted parentheses. MySQL had two attempts that added OR keywords. Oracle Database had one with added parenthesis to the ending.

Statement 5 had a lower success rate than the average of all results. From MySQL, there is possible to identify a large group of 12 answers with similar changes. Four answers had the right idea, but the execution was not perfect. One user was very close. In the case of Oracle Database, the success rate is marginally better. It is possible to identify similar themes from incorrect answers, as in MySQL.

MySQL answers for **statement 6** had a larger group of similar answers, but all had slight differences. Oracle Database had four similar incorrect answers. It is noteworthy that the grouped incorrect answers were identical between the two DBMSs.

The success rate for **statement 7** was high for both DBMSs. MySQL answers had four answers that could be considered similar. In the case of Oracle Database, there are two answers considered similar and one answer with the right idea, but the execution is lacking.

Statement 8 had a slight disparity between DBMSs. MySQL answers had two identifiable groups of incorrect answers. Oracle Database answers had no visible theme and only four incorrect answers.

In the case of **statement 9,** there are two different groups of incorrect answers for MySQL. There is also a noticeably high number of incorrect answers that do not have a perceivable theme. Oracle Database answers have two small groups of incorrect answers. Compared to MySQL, the assorted answers grouping is also significant.

Statement 10 had a lower success rate compared to average. From MySQL answer, there was possible to distinguish four different groups of incorrect answers. One group of similar incorrect answers was noticeably higher. Oracle Database had an equivalent larger group of incorrect responses and one other grouping of incorrect answers. Both DBMSs had a relatively small number of miscellaneous answers compared to grouped incorrect answers.

For **statement 11,** MySQL had five unaltered changes, and two of the incorrect answers had similarities. For Oracle Database, answers had multiple different incorrect answers. One noteworthy submission for Oracle Database was an answer that had been nearly correct but added too many attributes to the GROUP BY.

In the case of **statement 12,** there is a high success rate on both DMBS. Only four incorrect answers spanned between the two. There is no grouping theme between the answers.

For **statement 13,** the MySQL part had only two answers with similarities and one response adding the correct keyword but could not fully correct the statement. Oracle Database answers had three similar incorrect answers. Five answers were left unchanged.

Both DBMSs and error messages had a low success rate with **statement 14**. Answers for MySQL had different groups of incorrect answers, with three different groups of incorrect answers. Oracle Database had similarly three groups of incorrect answers.

Statement 15 has a minimal number of incorrect answers regardless of the DBMS and error message. Statement 16 had a slight difference between MySQL and Oracle Database success rates. MySQL had only two incorrect responses and Oracle Database had 8, and four of those were similarly incorrectly altered.

5 DISCUSSION

As established prior in this thesis, the angle taken for the study is to identify potentially inappropriate error messages in the domain of RDBMs utilizing SQL as the query language. The focus of the analysis was primarily on the incorrect answers for the statements. The correctly altered statements are not considered preferred evidence because there is a suggestion that the user benefits from the error message.

5.1 Error messages with no solid evidence of adverse impact

Several statements and error messages across the two DBMSs can be interpreted as well performing with a high success rate, and incorrect submits that could be considered an attempt to correct the right part of the statement. The first error message and statement for MySQL and Oracle Database could be regarded as successful. Even though there are multiple incorrect answers, closer inspection reveals that most of the failed attempts seemingly address the problem of the ambiguous column but are unable to succeed for other reasons. This could be because of expertise. The second error message and statement could also be considered well-performing, with only two incorrect submissions across both DBMSs. There is little evidence that both DBMSs' first two error messages would cause more complications than they solve. Error message for statement 12 from both is performing well, and there is little to interpret. The theme continues with statement 15, where there are only four incorrect submissions across the two DBMS. Three of those have corrected the statement but made a typographic error, rendering the correction useless.

Oracle Database error message for statement 5 has no solid evidence of causing problems for the users. Even with a slightly lower success rate, it is hard to argue against the content of this error message. It has accurate information that the error is located at the end part of the statement but does not elaborate further. Too limited amount of information can become an obstacle.

Still, this case does not present evidence of that, as almost half of the incorrect answers find the wrong keyword but fail to remove the redundant GROUP BY clause.

Error message from Oracle Database for statement 6 could be described as vague. It does not provide the location of the error but hints that some part of the statement has too many values, which is accurate information. From incorrect answers could be interpreted that users have difficulties locating the error. Aside from the locating issues, there is little evidence to establish that this error message produces more issues for the user.

Error message for statement 7 from MySQL can be considered as well-performing. A minimal number of incorrect answers and incorrect answers seemingly address the correct issue, but there is another minor issue with the answers, a typographic error or insufficient correction.

Error message by Oracle Database for statement 7 has accurate information about the cause of the error. It is hard to argue that this error message would cause the user problems. There is no perceivable connection between the error message and incorrect answers.

Error message produced by Oracle Database for statement 9 cannot be deemed problematic. Even with a lower success rate, there is no evidence that the error message is causing more incorrect answers. Most incorrect answers have seemingly located the error, but user expertise to correct the statement is lacking.

Even with a low success rate, it is hard to find arguments that the error message from MySQL for statement 9 is misleading or causing problems. The error message provides the correct syntax to use. The user has the responsibility to locate and identify the invalid part. Four of the incorrect answers had altered the correct position, but insufficiently even with the help of proper syntax. The error message might not be perfect, but lackluster is not the same as misleading or problematic. This error message could benefit from the location of the error.

Error message for statement 10 from MySQL has a low success rate, but it provides correct information. In this case, a large portion of the incorrect answers attempt to change parts with group function but cannot perfect the syntax. There is little to argue that this error message causes problems because it does provide the user with the location and the type of error. In this case, it could be argued that user expertise and understanding of SQL is a factor.

Statement 10 with an error message from Oracle Database has produced interesting results. The information on the error message is correct, stating that the group function (an aggregate function) is not correctly placed. The success rate is low, and most incorrect answers focus on the WHERE clause instead of the group function. Seven of the incorrect answers have changed the correct line. The results could be interpreted so that the users do not recognize the problem with the group function and therefore misidentify errors at another part of the statement. Arguing that this error message is problematic has no proper foundation, even with a meager success rate.

Error message for statement 11 by Oracle Database is controversial. The error message shows that the problem is related to the GROUP BY; it strangely refers to it not being GROUP by expression. There is no solid evidence that this is causing problems because of the high success rate and one of the incorrect answers attempting to correct the right part.

5.2 Key findings for MySQL

There is evidence of error messages that provide correct guidance, but they are not specific. Imprecise error messages can cause problems. (Shneiderman, 1982) This would require the user to have the expertise to locate the error. Error message for statement 4 from MySQL could be an example. The error message points to the operator, and the statement includes five different operators. Five incorrect answers could be described as altering statements near the correct operator but cannot possibly understand or remember the proper syntax. MySQL error message for statement 6 is identical, and users are attempting to fix around an operator but are not addressing the correct operator.

For statements 3, 8, 12, 13, 14, and 16, MySQL error messages only refer to the syntax error located near a part quoted for the error message. In the case of statement 8, the incorrect part is outside the quoted part. Other error messages include the erroneous part inside the quoted part at the very beginning of the quote. When inspecting wrong answers, there are hints that users might not look directly at the beginning of the quote for the error. It is possible that the user would benefit from the information that the error is directly at the beginning of the quote, as many users have made changes inside the quoted part with little cohesion. This error message may make error correction a bit harder, as it gives a long quote for the user to check, increasing the chance that the user focuses on something else than the incorrect part. Unspecific error messages are considered undesirable. (Shneiderman, 1982)

Statement 5 for MySQL has a low success rate. From incorrect answers, only four understood that statement is missing ORDER BY. One user understood that it should replace GROUP BY but mistakenly used "SORT BY," which is not a valid keyword. This is interesting as the error message mentions the GROUP BY, and still, only five users notice that it is not the correct keyword for this occasion. Interestingly, none of the incorrect submits attempt to add aggregate function, which is the action error message suggests. The syntax in the statement is correct for the ORDER BY keyword, and the user only must detect the wrong keyword and change it. Even with a low success rate, it is hard to argue that the error message is the cause, as it delivers correct information. Appendix A states that results should be sorted, but some participants do not address this issue. Misperception is one of the common causes of error mentioned by Smelcer (1995). Although, if the error message had simply been "invalid GROUP BY", the user could pay more attention to it instead of trying to solve why the aggregate function is missing and what it means.

There is 76,5% success for the error message for statement 11 by MySQL. Objectively, the correction required for this statement is minor. The error message provides a lot of information, even explaining that the second column should be in the GROUP BY clause. The error message has the potential to confuse as it refers to the missing column as "test . p . status". There is no clear evidence to support this, but there is no apparent reason why the column is not referred to by its original name.

5.3 Key findings for Oracle Database

Error message for statement 3 by Oracle Database could be considered unhelpful because of inaccuracy and confusion. The error message states that the NULL keyword is missing, which is not required for the statement. The reason for the misleading error message is that the correct syntax around IS keyword requires the NULL keyword. It can confuse if the user fixates on the NULL keyword missing. Three users tried adding the NULL keyword to various parts of the statement. However, five of the incorrect answers address the right problem but cannot properly do so. There is little evidence that an error message could cause issues if the DBMSs misinterpret the statement's intended purpose and display an error message that it considers correct.

Error message for statement 4 from Oracle Database has inaccurate information about the syntax error. The message refers to missing right parenthesis, which is incorrect as the syntax error is related to the WHERE clause. Four incorrect answers have modified the parenthesis, with one user adding an extra parenthesis at the end. In conclusion, there is some evidence behind the inference that users might experience some confounding with this error message.

There is a high success rate with the error message by Oracle Database for statement 8. Even with users seemingly finding and correcting the right part of the statement, the information on the error message is incorrect. The error message states that the command is not properly ended, but the syntax error is located at the middlemost part of the statement. Albeit there is no evidence that this is causing problems, in this case, it could be argued that incorrect information does not help and could increase the time user spends on locating the error. (Denny et al., 2020)

Error message for statement 13 by Oracle Database has produced some incorrect answers. Three incorrect answers have altered the last line of the statement as the error message instructs. A vague location of the error could cause problems for the users considering locating and correcting the syntax error. The evidence is not strong enough to make definitive conclusions, but a study by Taipalus et al. (2021) supports this by stating that error messages by Oracle Database are not perceived that useful when locating the error.

Based on the success rate and incorrect answers, the error message for statement 14 by Oracle Database is not effective. Nine wrong answers altered the correct line. The error message states that the expression is missing. The statement has multiple expressions, and the line seven expression is not the most evident one. This error message could be considered so vague that it causes users problems locating and understanding the syntax error. Vague error messages are not advisable. (Molich & Nielsen, 1990)

Error message for statement 16 from Oracle Database could be considered misleading. Half of the incorrect answers have added the ORDER BY clause at the end, which does not seem to be related to the assignment. The error message prompts the user to add expressions, which half of the users have done. The information is not incorrect, as one parenthesis is missing rendering the expression started on line seven defective. The error could be identified by checking that parenthesis is an even number. There is some evidence that the vague error message has caused problems for some users.

5.4 Answers to research questions

Based on the results and analysis, it is reasonable to argue that the answer to the research question "Are there SQL syntax error messages that have a negative impact on user error correction performance?" is yes. There are indications that some error messages could mislead the user's attention to an incorrect part of the statement by providing vague information about the location. There is additional evidence that DBMS sometimes misinterprets the statement and provides a correct error message, but it is not the right information for the user to achieve what is required. It is also evident that some error messages had incorrect information about the error, affecting the user's ability to correct the statement. Lastly, suppose the error message includes the incorrect keyword. In that case, it is possible that the user might not understand that the keyword itself could be wrong for the occasion and instead searches for the problem elsewhere.

5.5 Limitations

Firstly, the analysis of the data was done manually. There is a possibility of human error, albeit multiple revisits and re-checks to the source data. The amount of processed information is vast, and it is impossible to consider everything inside the scope of this thesis.

Secondly, this thesis represents an early review of this subject. The lack of prior studies impacts the exploratory nature of this thesis. Therefore, the aim was to discover, and therefore even the minor evidence and proof might be considered a possibility rather than proving something for a fact.

Third, the sample population of users with an introductory level of SQL and databases completed. Some users might have excelled at that and have higher proficiency than others. This is not representative of users who have more experience or have worked with SQL for several years, are accustomed to the error messages, and have a trained eye for locating the errors. In addition, the sample size was 51 participants on MySQL and 36 on the Oracle database. To further research and prove some indications presented in this thesis, the sample size should be improved.

5.6 Future implications

There are some future remarks made from this thesis. There is evidence that some error messages could hurt the user error correction performance, but it needs more validation. It would be beneficial in the future to study SQL error messages with a focus on misleading, vague, and possibly user attention fixating error messages. Also, it could be interesting to learn why and when DBMS provides seemingly incorrect information about the error.

6 CONCLUSION

This thesis aimed to discover SQL error messages that could cause problems correcting SQL syntax errors. To conclude, there is a possibility that some error messages could cause hindrances in syntax error correction. The data collected made it possible to identify that some SQL error messages are misleading and include incorrect information. The results suggest that these error messages could negatively impact the user's ability to correct the statement. Some error messages were vague, and in some situations, there is slight evidence of users being affected by this negatively.

This thesis is limited by the lack of prior studies on the subject and novelty of the issue. In addition, the analysis method is prone to human error even with a careful and methodological approach. Lastly, the number of data points per error message could be higher in future studies on this subject.

These findings provide fertile ground to study further SQL error messages that cause hindrances and ultimately improve them for a better user experience. Studies in the future should focus on already discovered possible problematic SQL error messages. Findings made in this thesis require further affirmation. With additional evidence on the lackluster performance of specific error messages, it is possible to begin researching for possible improvements.

7 REFERENCES

Ahadi, A., Behbood, V., Vihavainen, A., Prior, J. & Lister, R. (2016). Students' syntactic mistakes in writing seven different types of SQL queries and its application to predicting students' success. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 401-406.

Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E. & Chris, P. (2017). Do Developers Read Compiler Error Messages? *IEEE/ACM 39th International Conference on Software Engineering (ICSE)* , 575-585.

Becker, B. A. (2016). An Effective Approach to Enhancing Compiler Error Messages. *SIGCSE '16 Proceedings of the 47th ACM Technical Symposium on Computing Science*, 126-131.

Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B. & Prather, J. (2019). Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. *Proceedings of the working group reports on innovation and technology in computer science education* , 177 - 210.

Brown, P. (1983). Error messages: the neglected area of the man/machine interface. . *Communications of the ACM, 26*(4), 246-249.

Cambridge Dictionary. (2022). *Cambridge Dictionary: error message*. Retrieved 3.31.2022 From https://dictionary.cambridge.org/dictionary/english/error-message

Codd, E. F. (1970). A relational model of data for large shared data banks. *MD Comput, 15*, 162-166.

Collins. (2022). *Collins Dictionary*. Retrieved 31.3.2022 From https://www.collinsdictionary.com/dictionary/english/error-message

Denny, P., Luxton-Reilly, A. & Carpenter, D. (2014). Enhancing Syntax Error Messages Appears Ineffectual. *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 273 - 278.

Denny, P., Prather, J. & Becker, B. A. (2020). Error Message Readability and Novice Debugging Performance. *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). Association for Computing Machinery,* 480- 486. doi:https://doi.org/10.1145/3341525.3387384

Elmasri, R. & Navathe, S. B. (2016). *Fundamentals of Database Systems* (Seventh edition p.). Pearson.

Farrell, J. & Seloner, Q. (1984). *STANDARIZATION, COMPABILITY AND INNOVATION.*

Fehily, C. (2008). *Visual Quickstart Guide SQL* (Third Edition p.). Peachpit Press.

FMSinc. (2022). *FMSinc: Microsoft Access History*. Retrieved 3.4.2022 From http://www.fmsinc.com/MicrosoftAccess/history/index.html

Groff, J. R. & Weinberg, P. N. (1999). *SQL: The Complete Reference.* Osborne/McGraw-Hill.

IBM. (1983). *Announcement Letter Number ZP83-0746.* IBM. Noudettu From https://www.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/6/877/ENUSZP83-0746/index.html&lang=en&request_locale=en

ISO. (2022). *Standar: ISO.org*. Retrieved 21.2.2022 From https://www.iso.org/standard/16661.html

Karvelas, L., Li, A. & Becker, B. A. (2020). The Effects of Compilation Mechanisms and Error Message Presentation on Novice Programmer Behavior. *In Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 759-765.

Kelechava, B. (2018). *ANSI.* Retrieved 21.2.2022 From https://blog.ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/

Maglio, P. P. & Kandogan, E. (2004). Error Messages: What's the Problem? *Queu, 2*(8), 50-55.

Merriam-Webster. (n.d.). *"Error message." Merriam-Webster.com Dictionary.* Retrieved 31.3.2022 From https://www.merriam-webster.com/dictionary/error%20message

Microsoft. (2021). *Microsoft Docs: error message guidelines*. Retrieved 3.4.2022 From https://docs.microsoft.com/en-us/windows/win32/debug/error-message-guidelines

Molich, R. & Nielsen, J. (1990). Improving a human-computer dialogue. *Communications of the ACM*, 338-348.

National Research Council. (1999). *Funding a Revolution: Government Support for Computing Research.* Washington D.C.: NATIONAL ACADEMY PRESS.

Nienaltowski, M.-H., Pedroni, M. & Meyer, B. (2008). Compiler Error Messages: What Can Help Novices? *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 168 - 172.

ORACLE. (2022). *Oracle Help Center: Database SQL Tuning Guide*. Retrieved 21.2.2022 From https://docs.oracle.com/database/121/TGSQL/tgsql_sqlproc.htm#TGSQL175

ORACLE. (2022). *Oracle: Help Center*. Retrieved 3.4.2022 From https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT001

Owens, M. (2006). *The Definitive Guide to SQLite* (First p.). Apress. doi:https://doi.org/10.1007/978-1-4302-0172-4

Pachev, S. (2007). *Understanding MySQL Internals.* O'Reilly Media, Inc.

Pachev, S. (2022). *O'REILLY: Understanding MySQL internals*. Retrieved 3.4.2022 From https://www.oreilly.com/library/view/understanding-mysql-internals/0596009577/ch01.html

Pettit, R., Homer, J. & Gee, R. (2017). Do Enhanced Compiler Error Messages Help Students? Results Inconclusive. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 465 - 570.

PostgreSQL. (2022). *PostgreSQL: History*. Retrieved 3.4.2022 From https://www.postgresql.org/docs/current/history.html

Preston, C. W. (2007). *Backup & Recovery: Inexpensive Backup Solutions for Open Systems.* O'Reilly Media, Inc.

SAP. (2022). *SAP: About*. Retrieved 3.4.2022 From https://www.sap.com/about/company/history/2011-present.html

Shneiderman, B. (1982). Designing Computer System Messages . *Communications of the ACM, 25*(9), 610-611.

Smelcer, J. B. (1995). User errors in database query composition. *International Journal of Human-Computer Studies, 42*(4), 353-381.

SQLite. (2022). *SQLite: most deployed*. Retrieved 3.4.2022 From https://www.sqlite.org/mostdeployed.html

Taipalus, T., Grahn, H. & Ghanbari, H. (2021). Error messages in relational database management systems: A comparison of effectiveness, usefulness, and user confidence. *Journal of Systems and Software* (181).

Taipalus, T., Siponen, M. & Vartianen, T. (2018). Errors and complications in SQL query formulation. *ACM Transactions on Computing Education, 18*(3), 1-29.

Traver, V. J. (2010). On compiler error messages: what they say and what they mean. Teoksessa *Advances in Human-Computer Interaction.*

APPENDIX 1 DATABASE SCHEMA AND DATA DEMANDS

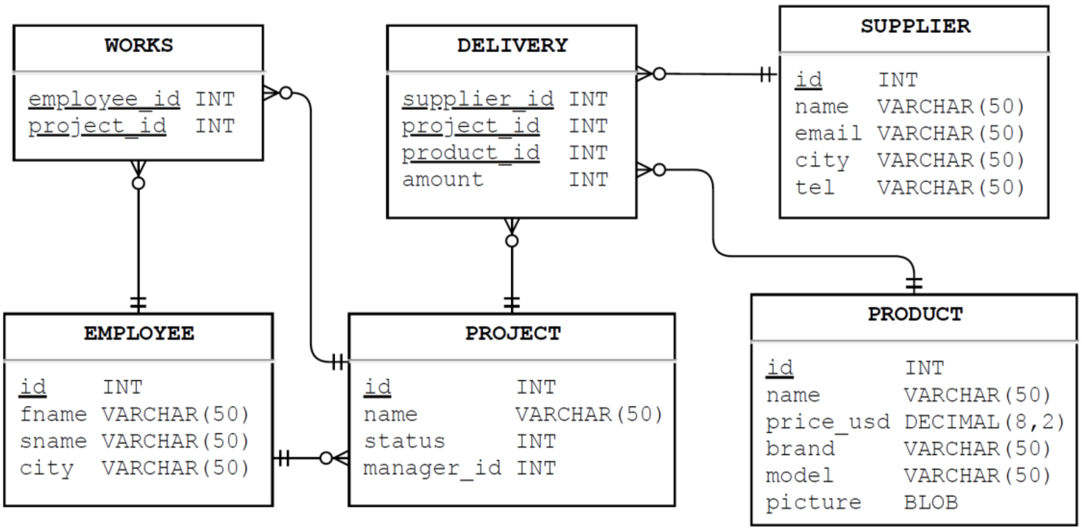


Figure 44 The database schema shown to the participants (Taipalus et al., 2021)

Table 6 Data demands for each statement (Taipalus et al., 2021)

#	Data demand
1	Find the names of suppliers who have delivered at least one Apple product priced over 50 USD.
2	Find the names of employees who work in a project called QA or in a project called HR.
3	Find the brands and models of products with a price of 350 USD which have been delivered at least once with an amount greater than 100 pcs.
4	Find the IDs, names and status of projects which start with an H, J, or K, and have a manager whose surname is Smith.
5	Find the names of employees who live in the same city as supplier 409 or 309 is located. Sort the results according to employee surname, then by first name, both in ascending order.
6	Find the IDs and names of employees who have worked for a project that is managed by an employee from Paris.
7	Find the names of employees from New York and Minneapolis, who manage at least one project with the status of 0.
8	Find the names, prices, brands, and models regarding products with a brand name which starts with an S or a C, and have no picture. Sort the results according to product name in descending order.
9	Find the IDs, names, and emails of suppliers who have an icloud or gmail address and who have made at least one delivery.
10	Find the names, brands and models of products from Google and Microsoft that have a picture, and a price more than average price of all products.
11	Find the number of employees in the projects 1000 through 2000 by employee city and project status.
12	Find the names and prices of products from Oracle which have been delivered to at least one project with a name starting with the word 'data'.
13	Find the names, emails, cities and telephone numbers of Athenian suppliers who have supplied at least one product to a project named HR. Order the results by supplier city, then by supplier name, both in ascending order.
14	Find the names and status of projects which have exactly 10 employees from London working in them.
15	Find the names and prices of products with no picture which have been delivered at least once to a project with the status of 1.
16	Find the names and manager ids of projects to which a supplier from Sydney has delivered at least one delivery.