1/4- Selecting Data

Course Description

Learn SQL, the go-to language for transforming raw database data into useful insights. This course focuses on filtering, comparing, summarizing with aggregate functions, sorting, grouping, and presenting data neatly. With hands-on practice queries, master the skills to analyze data using SQL and apply your knowledge today!

SQL

Selecting Data

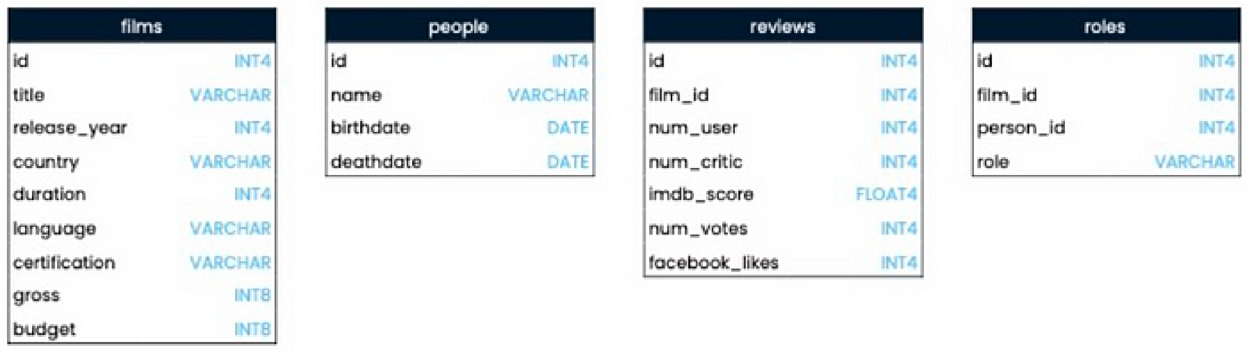
In this article, we will build on our foundational SQL, learn how to reveal insights, and how present our results clearly.While SQL can be used to create and modify databases, the focus of this article will be querying databases.

Course Roadmap

What is a Query?

In databases, a query is a way to ask for specific information. This guide teaches you to use keywords to execute queries, count records, and view data. Learn about common SQL errors, coding style, execution order, filtering techniques, aggregate functions, and sorting/grouping results.

Database
The data base that is used is as follows



films	
id	INT4
title	VARCHAR
release_year	INT4
country	VARCHAR
duration	INT4
language	VARCHAR
certification	VARCHAR
gross	INT8
budget	INT8

people	
id	INT4
name	VARCHAR
birthdate	DATE
deathdate	DATE

reviews	
id	INT4
film_id	INT4
num_user	INT4
num_critic	INT4
imdb_score	FLOAT4
num_votes	INT4
facebook_likes	INT4

roles	
id	INT4
film_id	INT4
person_id	INT4
role	VARCHAR

Schema of Film’s database

Keywords

COUNT()

Learn how to use COUNT() to count records with values in a specific field, using the example of counting something from the people's table in this article.

Example:

We number of birthdates present at the people table,

```sql
SELECT COUNT(birthdate) AS count_birthdates
FROM people;
```

Count birthdate

Result

```
|count_birthdates|
|----------------|
|6152            |
```

Result set

We have used AS to make the result more readable.

If we want to count more than one field, we need to use COUNT() multiple times.

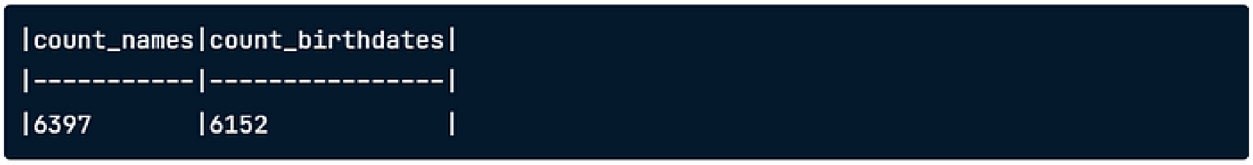
Example:

Count both the number of names and birthdates present in the people table.

```
SELECT COUNT(name), COUNT(birthdate)
```

```
FROM people;
```

Result



```
|count_names|count_birthdates|
|-----------|----------------|
|6397       |6152            |
```

Result set

Using * with COUNT()

if we want to know the number of records in a table, we can call

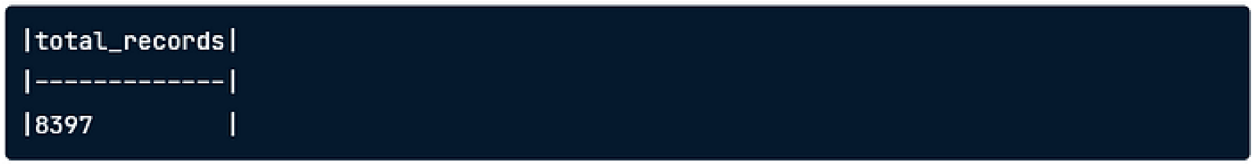
the COUNT with an asterisk(*).

Example:

```
SELECT COUNT(*) AS total_records
```

```
FROM people;
```

Result



```
|total_records|
|-------------|
|8397         |
```

Result set

DISTINCT

Most of the time our result includes duplicates. We can use the DISTINCT keyword to select all the unique values from a field.

Example

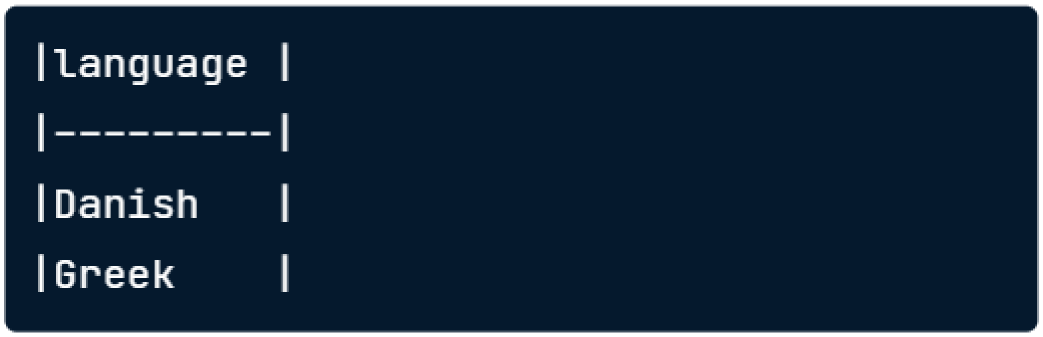
If we wanna know which languages are represented in the film's table?

```
SELECT DISTINCT language

FROM films;
```

Result

Adding DISTINCT removes duplicates, as we can see here.



```
|language |
|---------|
|Danish   |
|Greek    |
```

Result set

COUNT() With DISTINCT

Combining COUNT() with DISTINCT is also common to count

the number of unique values in a field.

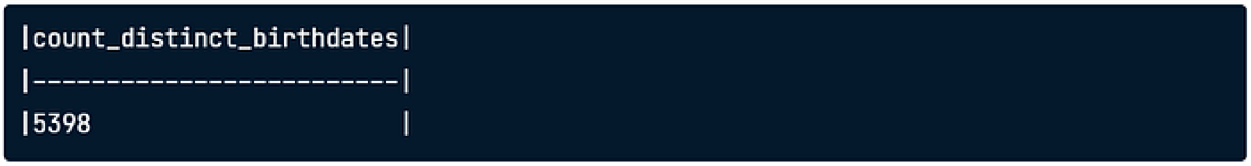
Example

count the number of DISTINCT birth dates in people table.

```
SELECT COUNT(DISTINCT birthdate) AS count_distinct_birthdates

FROM people;
```

Result



```
|count_distinct_birthdates|
|-------------------------|
|5398                     |
```

Result set

Learning to COUNT()

Here is a query counting `film_id`. Select the answer below that correctly describes what the query will return.

```
SELECT COUNT(film_id) AS count_film_id

FROM reviews;
```

Run the query in the console to test your theory!

Possible Answers

a) The number of unique films in the `reviews` table.

b) The number of records containing a `film_id`.

c) The total number of records in the `reviews` table.

d) The sum of the `film_id` field.

b is the right answer.

Practice with COUNT()

As you've seen, `COUNT(*)` tells you how many records are in a table. However, if you want to count the number of *non-missing* values in a particular field, you can call `COUNT()` on just that field.

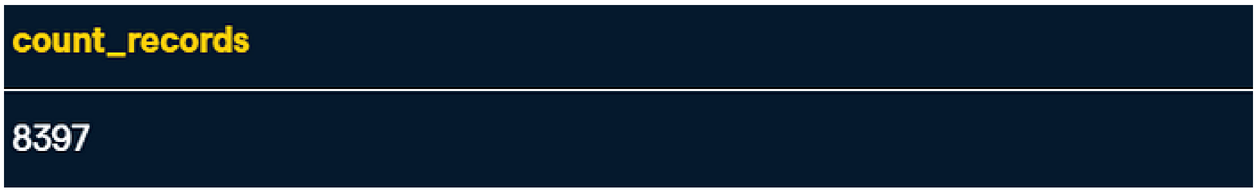
Let's get some practice with `COUNT()`! You can look at the data in the tables throughout these exercises by clicking on the table name in the console.

1. Count the number of records in the `people` table, aliasing the result as `count_records`.

```
SELECT COUNT(*) AS count_records

FROM people;
```

Result



count_records
8397

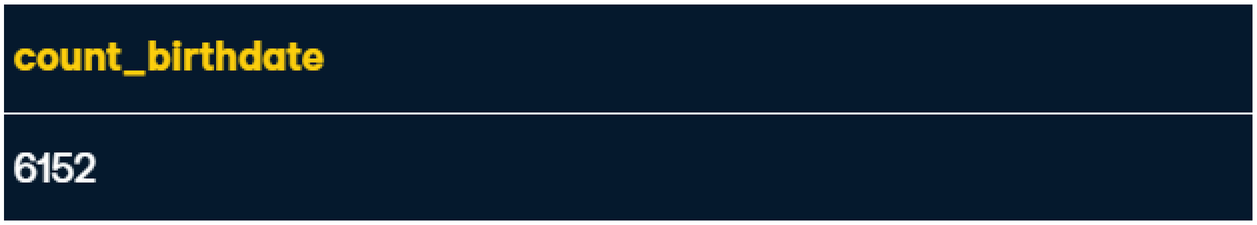
result set

2. Count the number of records with a `birthdate` in the `people` table, aliasing the result as `count_birthdate`.

```
SELECT COUNT(birthdate) AS count_birthdate

FROM people;
```

Result



count_birthdate
6152

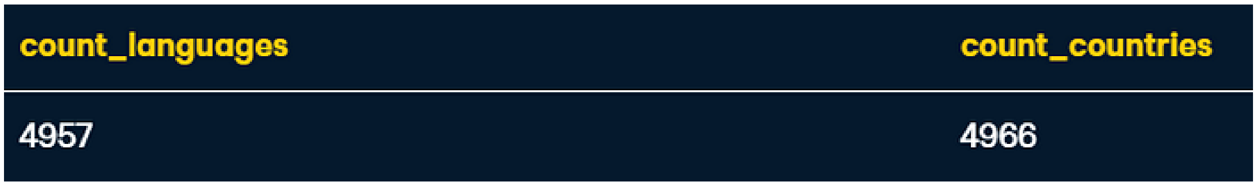
result set

3. Count the languages and countries in the `films` table; alias as `count_languages` and `count_countries`.

```
SELECT COUNT(language) AS count_languages, COUNT(country) AS
count_countries


FROM films;
```

Result



count_languages	count_countries
4957	4966

SELECT DISTINCT

Often query results will include many duplicate values. You can use the `DISTINCT` keyword to select the unique values from a field.

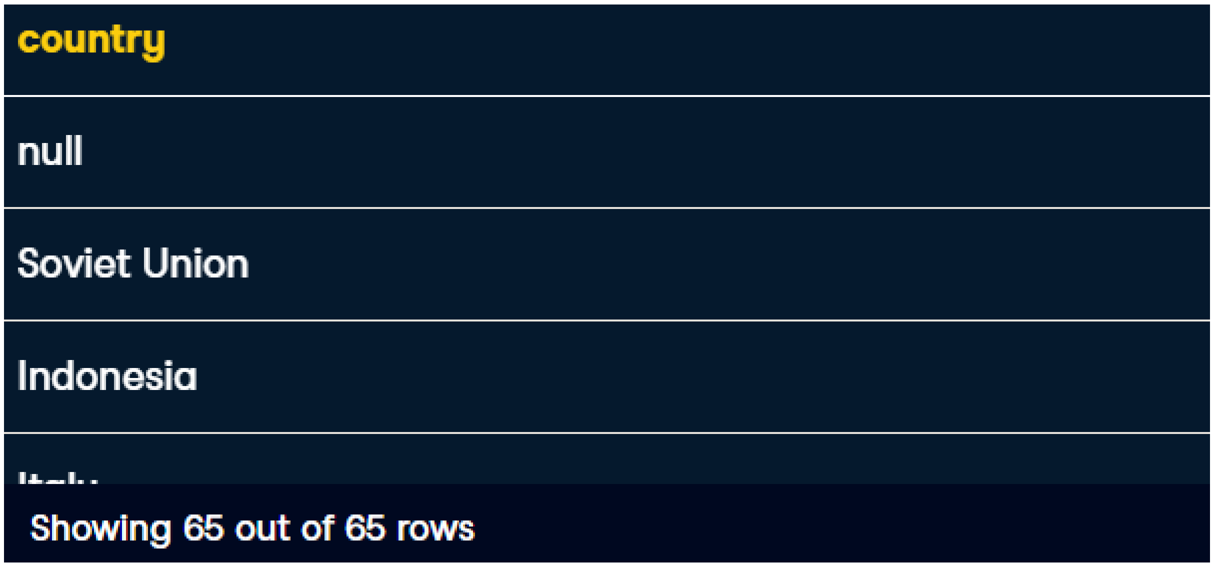
This might be useful if, for example, you're interested in knowing which languages are represented in the `films` table. See if you can find out what countries are represented in this table with the following exercises.

1. Return the unique countries represented in the `films` table using `DISTINCT`.

```
SELECT DISTINCT country

FROM films;
```

Result



country
null
Soviet Union
Indonesia
Italy

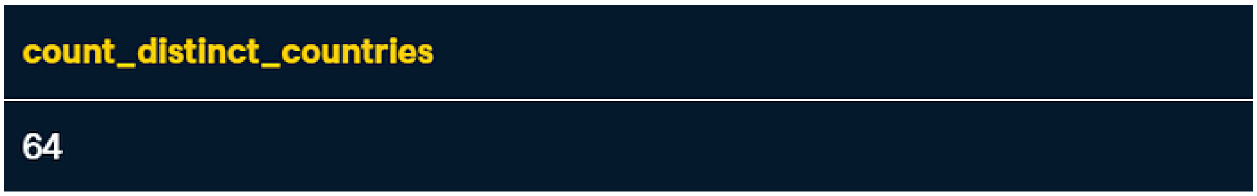
Showing 65 out of 65 rows

2. Return the number of unique countries represented in the `films` table, aliased as `count_distinct_countries`.

```
SELECT COUNT(DISTINCT country) AS count_distinct_countries

FROM films;
```

Result



count_distinct_countries
64

Query Execution

Now that we have flexed our SQL muscle a bit, we will take a small step back and better understand how SQL code works.

Unlike many programming languages, SQL code is not processed in the order it is written. Consider we want to grab a coat from a closet.

SQL CODE ORDER OF EXECUTION

First, we need to know which closet contains the coats. This is similar to the FROM statement.

```
FROM closet1
```

It is the first line to be processed.

Before any data can be selected, the table from which the data will be selected needs to be indicated.

Next, our selection is made.

```
SELECT coat

FROM closet1
```

Finally, the results are refined. Here we will use the **LIMIT** keyword that limits the results to a specified number of records. In this case, we only want to return the first 2 **coats** from the **closet1** table.

```
SELECT coat

FROM closet1

LIMIT 2;
```

Knowing the processing order is necessary when debugging and aliasing fields and tables.

Suppose we need to refer to an alias later on in our code. In that case, that alias will only make sense to a processor when its declaration in the SELECT statement is processed before the alias reference is made somewhere else in the query.

DEBUGGING SQL

Before we begin working with more advanced queries, it's useful to know more about debugging SQL code and how to read the error messages.

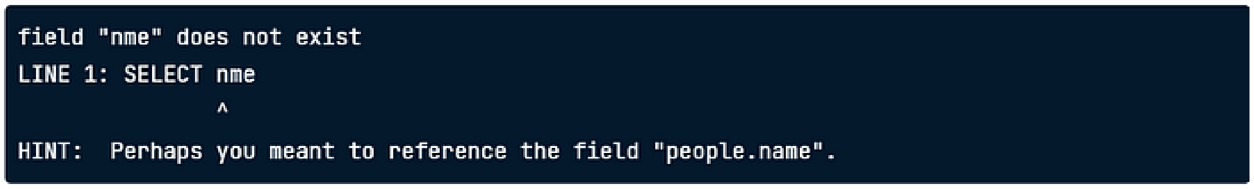
Some messages are extremely helpful, pinpointing and even suggesting a solution for the error. As this message does when we misspell the "name" field.

Look at the code and error below:

```
SELECT nme
```

```
FROM people;
```

ERROR:



```
field "nme" does not exist
LINE 1: SELECT nme
               ^
HINT:  Perhaps you meant to reference the field "people.name".
```

Common Errors

- Misspelling
- Incorrect capitalization
- Incorrect or missing punctuation

Other error messages are less helpful and require us to look at our code more closely.

Comma Errors

Forgetting a comma is a very common error.

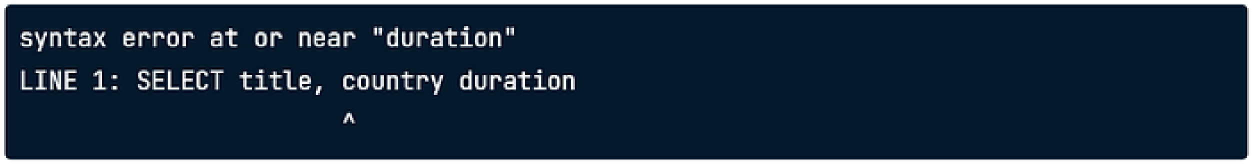
Let’s say we have drafted this code to find all titles, country of origin, and duration of films.

```
SELECT title, country duration

FROM films;
```

ERROR

The error message will alert us to the general location of the error using a caret (^) below the line of the code.



```
syntax error at or near "duration"
LINE 1: SELECT title, country duration
                              ^
```

We must examine the code a little further to discover the missing is between ‘country and duration’.

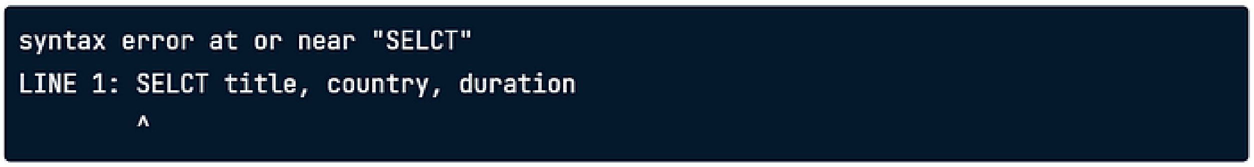
KEYWORD ERRORS

SQL displays a similar error message when a keyword is misspelt.

```
SELCT title, country, duration

FROM films;
```

ERROR



```
syntax error at or near "SELCT"
LINE 1: SELCT title, country, duration
        ^
```

But this time, the caret indicator below the offending line is spot on.

Final Notes on Error

There are a few more SQL errors out there, but the three mentioned in the article will be the most common ones that we will encounter.

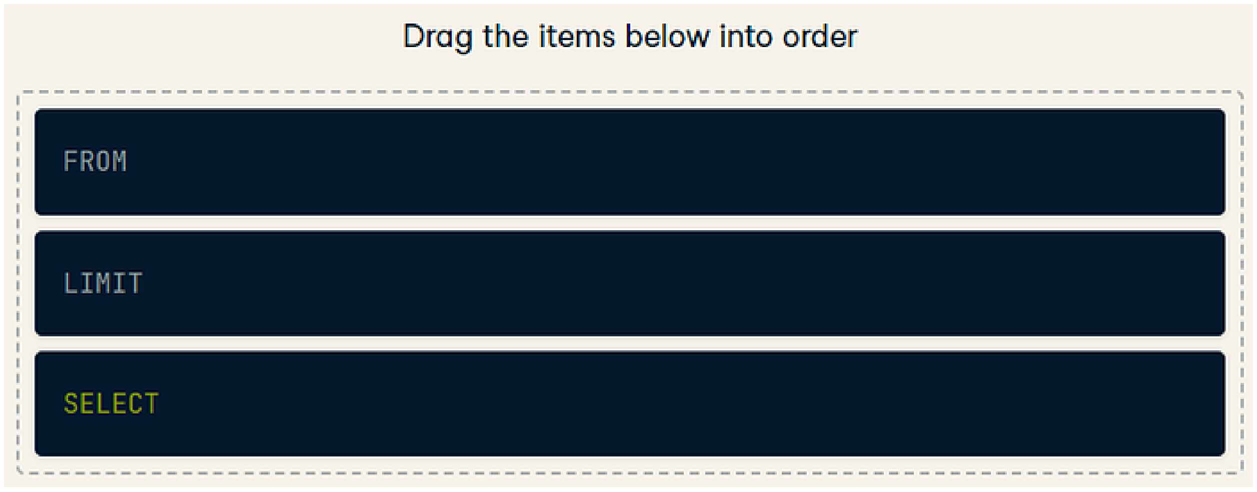
- Misspelling
- Incorrect capitalization
- Incorrect or missing punctuation

Debugging is a major skill, and the best way to master this skill is to make mistakes and learn from them

Order of execution

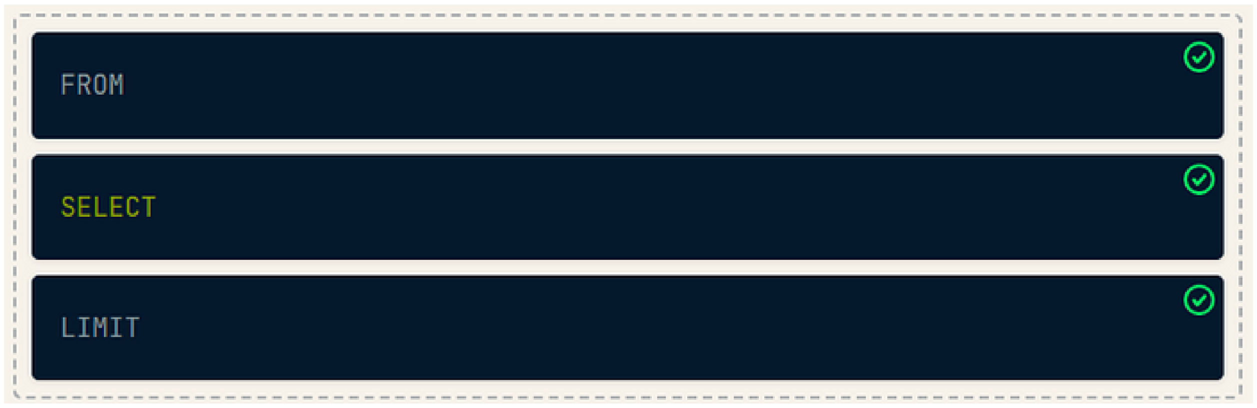
SQL code is processed differently than other programming languages in that you need to let the processor know where to pull the data from before making selections.

It’s essential to know your code’s order of execution to understand what results you’ll get from your query and how to fix any errors that may come up.



```
FROM
LIMIT
SELECT
```

Correct Order



```
FROM
SELECT
LIMIT
```

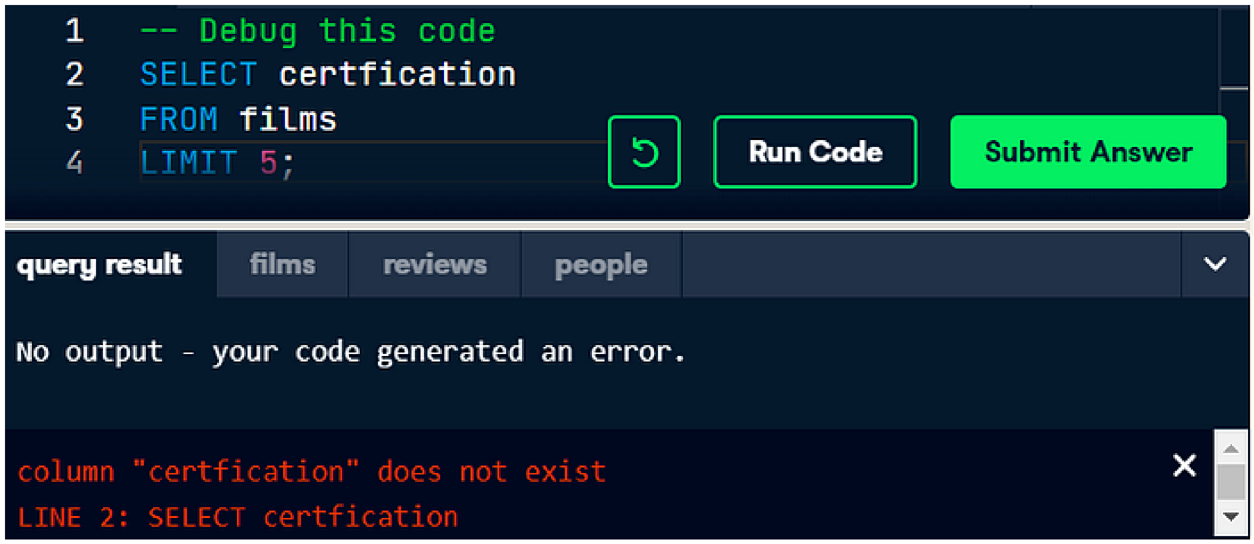
Debugging errors

Debugging is an essential skill for all coders, and it comes from making many mistakes and learning from them.

In this exercise, you'll be given some buggy code that you'll need to fix.

1. Debug and fix the SQL query provided.

Before Debugging



```sql
-- Debug this code
SELECT certfication
FROM films
LIMIT 5;
```

```
No output - your code generated an error.

column "certfication" does not exist
LINE 2: SELECT certfication
```

We can clearly see a 'Misspelling' error here. We need to recorrect the spelling of 'certification'

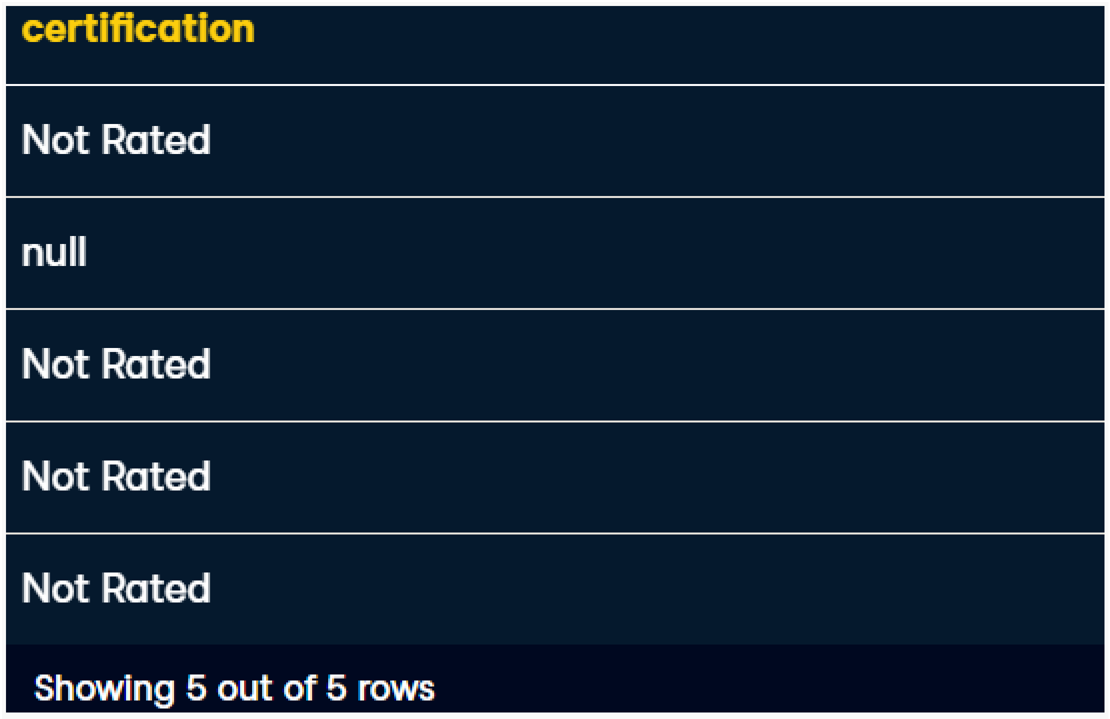
After Debugging

```sql
-- Debug this code

SELECT certification

FROM films

LIMIT 5;
```

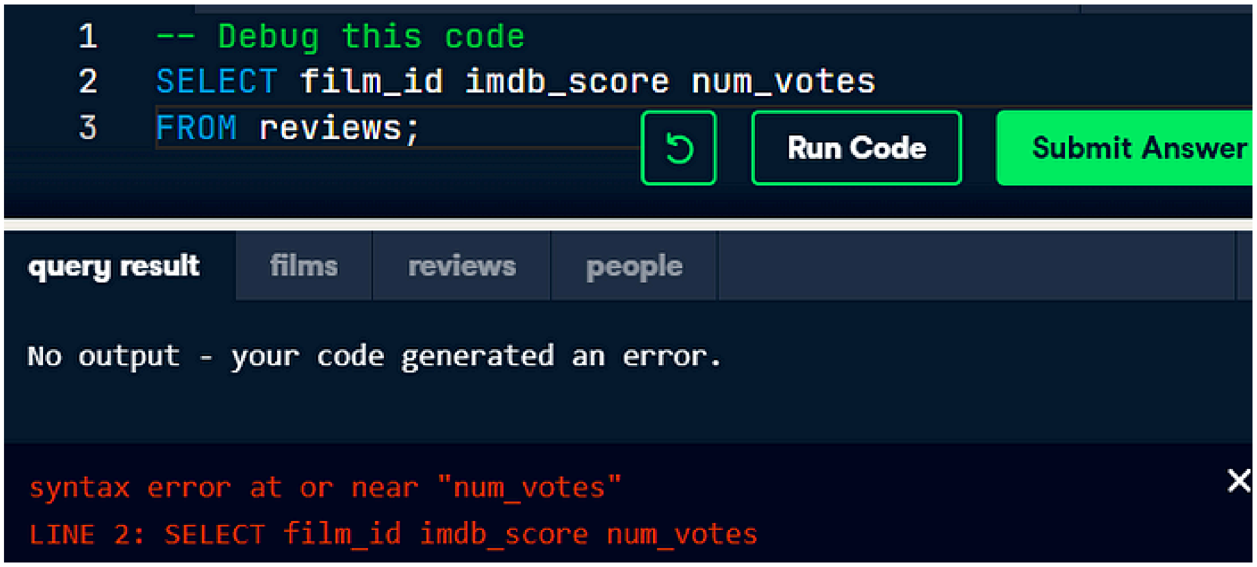


certification
Not Rated
null
Not Rated
Not Rated
Not Rated

Showing 5 out of 5 rows

Result set

2. Find the two errors in this code; the same error has been repeated twice.



```
-- Debug this code
SELECT film_id imdb_score num_votes
FROM reviews;
```

```
No output - your code generated an error.
```

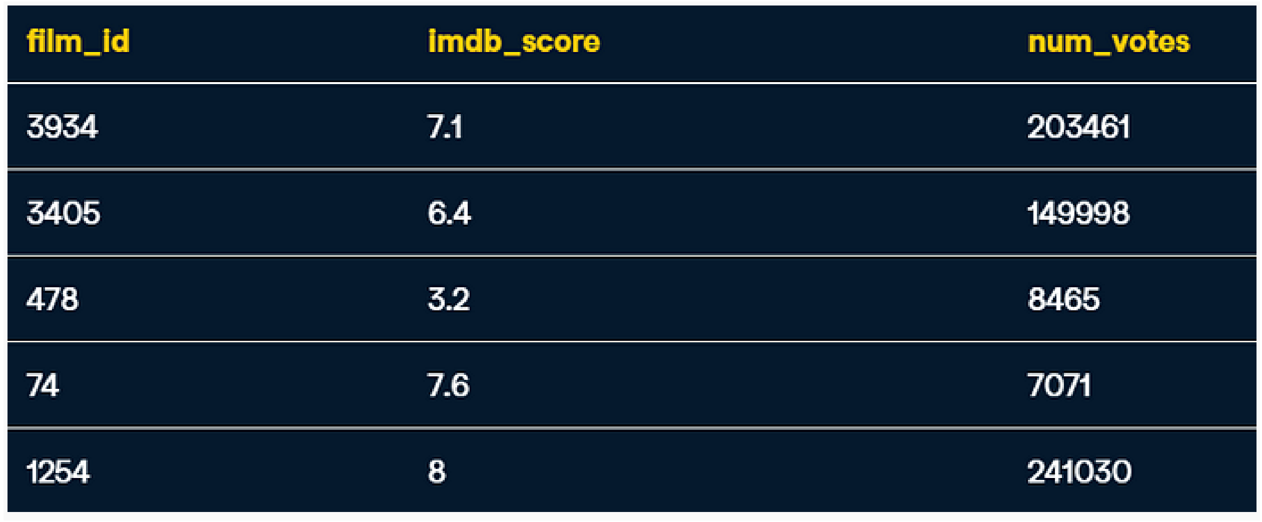
```
syntax error at or near "num_votes"
LINE 2: SELECT film_id imdb_score num_votes
```

Looking at the query, we can see commas are missing. So we just need to put commas to make the query work

After Debugging

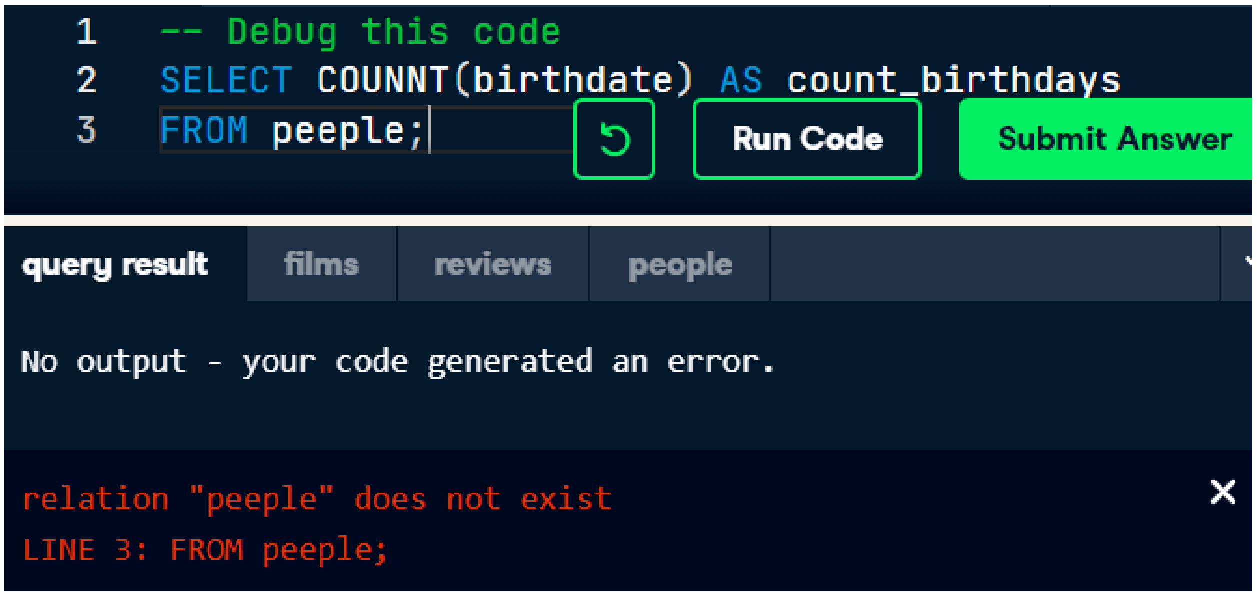
```
SELECT film_id, imdb_score, num_votes

FROM reviews;
```



film_id	imdb_score	num_votes
3934	7.1	203461
3405	6.4	149998
478	3.2	8465
74	7.6	7071
1254	8	241030

3. Find the two bugs in this final query.



```
-- Debug this code
SELECT COUNNT(birthdate) AS count_birthdays
FROM peeple;
```

```
No output - your code generated an error.
```

```
relation "peeple" does not exist
LINE 3: FROM peeple;
```

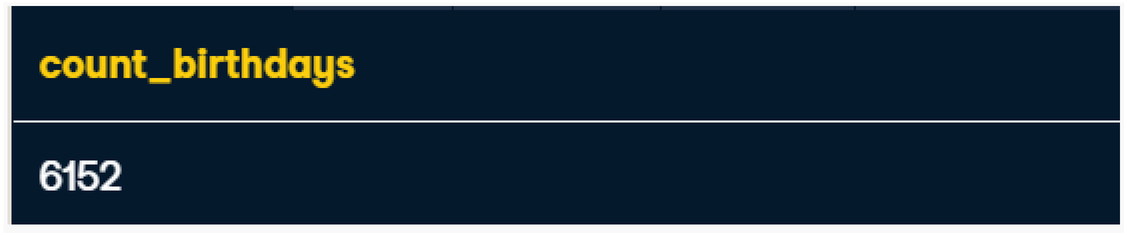
Another misspelling error. Check the spellings of COUNT() and people.

After Debugging

```sql
-- Debug this code

SELECT COUNT(birthdate) AS count_birthdays

FROM people;
```



count_birthdays
6152

result set

SQL Style

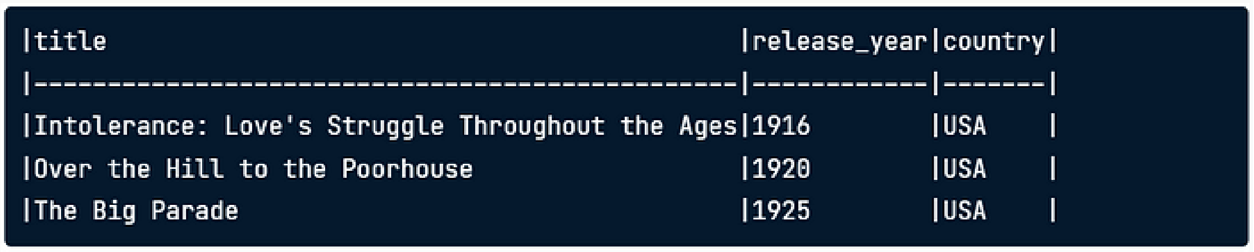
Now that we understand how SQL works, we will review how it looks.

SQL Formatting

SQL is a generous language when it comes to formatting. New lines, capitalization and indentations are not required in SQL as sometimes are in other programming languages.

For example, the code written below will run just fine.

```
select title, release_year, country from films limit 3
```



```
|title                                         |release_year|country|
|----------------------------------------------|------------|-------|
|Intolerance: Love's Struggle Throughout the Ages|1916        |USA    |
|Over the Hill to the Poorhouse                |1920        |USA    |
|The Big Parade                                |1925        |USA    |
```

Result set

The query runs and we got the right output but writing queries like this won't make us any friends in the SQL world. Because the lack of formatting makes the code difficult to read. Especially when queries become more complex.

SQL Formatting Best Practices

Over time, SQL users have developed style standards that are generally accepted across industries.

```
SELECT title, release_year, country
```

```
FROM films

LIMIT 3;
```

This code returns the same result as the previous code we discussed above. But it is much easier to read due to additional capitalized keywords and new lines between them. While capitalization and the new line are standard practices, many of the finer details of SQL styles are not.

Holywell's style guide

You can also check the link for a better understanding of standard practices.

1. https://www.sqlstyle.guide/

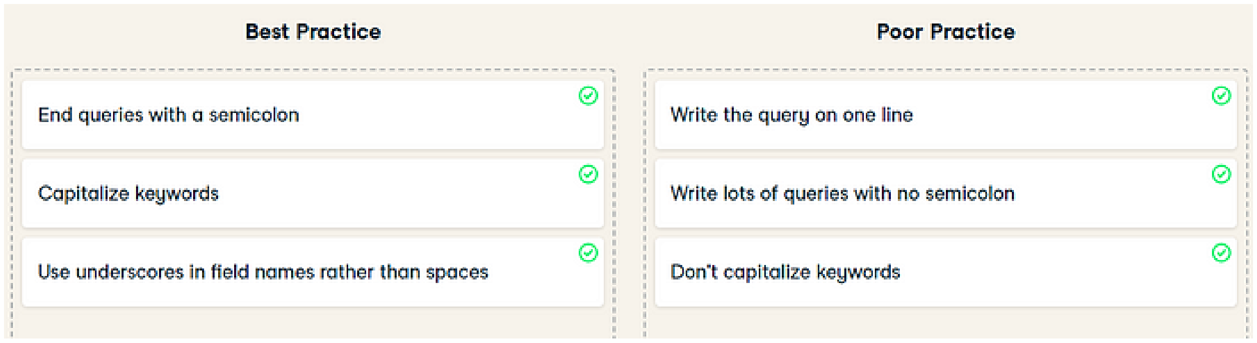
SQL best practices

SQL style guides outline standard best practices for writing code.

This exercise will present several SQL-style tips. Your job will be to decide whether they are considered best practices.

We'll be following **Holywell's style guide.**

1. Drag and drop the items into the correct zone.



Best Practice	Poor Practice
End queries with a semicolon ✓	Write the query on one line ✓
Capitalize keywords ✓	Write lots of queries with no semicolon ✓
Use underscores in field names rather than spaces ✓	Don't capitalize keywords ✓

Formatting

Readable code is highly valued in the coding community and professional settings. Without proper formatting, code and results can be difficult to interpret. You'll often be working with other people that need to understand your code or be able to explain your results, so having a solid formatting habit is essential.

In this exercise, you'll correct poorly written code to better adhere to SQL style standards.

2. Adjust the sample code so that it is in line with standard practices.

```
-- Rewrite this query

select person_id, role from roles limit 10
```

Standard practices

```
SELECT person_id, role

FROM roles
```

```
LIMIT 10;
```

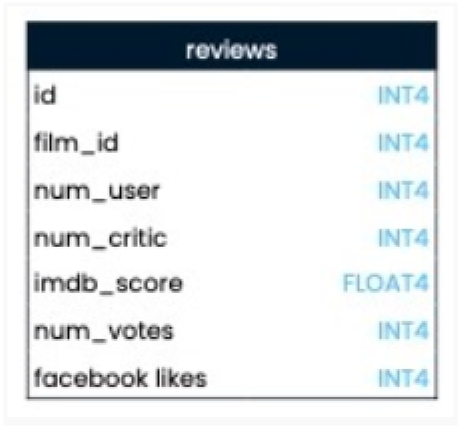
Non-standard fields

You may occasionally receive a dataset with poorly named fields. Ideally, you would fix these, but you can work around it with some added punctuation in this instance.

A sample query and schema have been provided; imagine you need to be able to run it with a non-standard field name. Select the multiple-choice answer that would correctly fill in the blank to return both a film's id and its number of Facebook likes for all reviews:

```
SELECT film_id, ___
```

```
FROM reviews;
```



reviews	
id	INT4
film_id	INT4
num_user	INT4
num_critic	INT4
imdb_score	FLOAT4
num_votes	INT4
facebook likes	INT4

Possible Answer

`1. facebook likes`

`2. "facebook likes"`

`3. facebook`, `likes`

2 is the right Answer.