# Apply filters to SQL queries

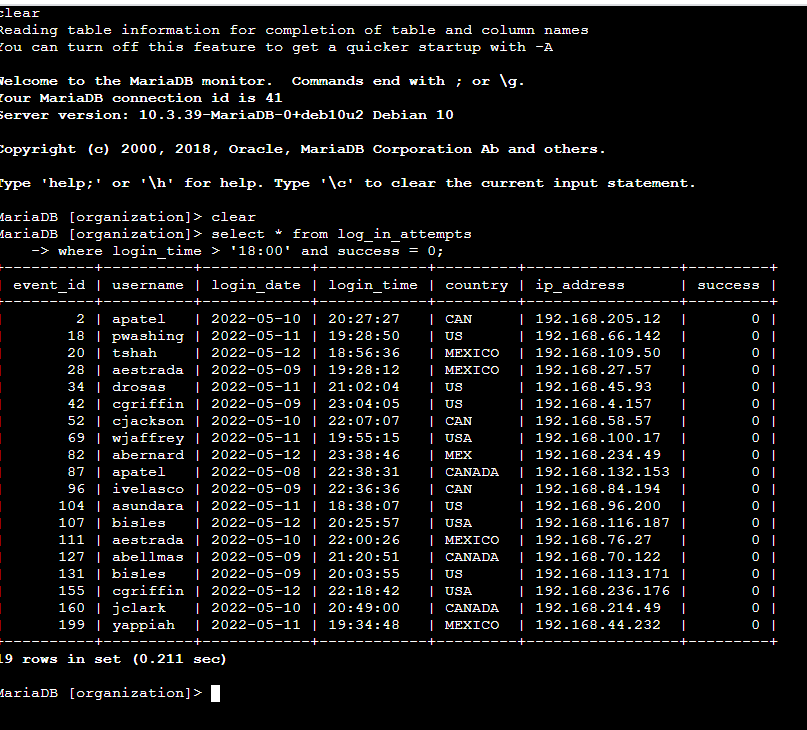
## Project description

As part of my role in improving cybersecurity within my organization, I use SQL to investigate potential threats and ensure our systems and user data remain secure. This project highlights how I applied SQL filtering techniques to review login activity, identify abnormal behavior, and assist with targeted updates to employee machines. These queries demonstrate my ability to extract meaningful insights from data using logical operators, date and time filters, and pattern matching.

## Retrieve after hours failed login attempts

Our team was alerted to suspicious login attempts that may have occurred after standard working hours. I needed to identify all failed attempts made after 18:00 to help assess whether these were related to unauthorized access.

The SQL query below was used to find this data:

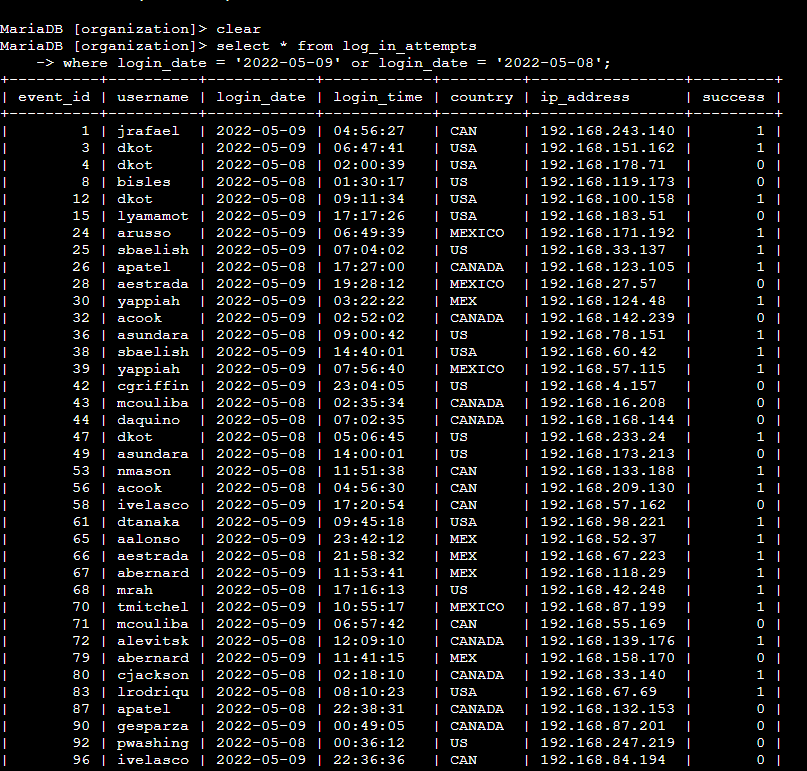


This query returns only the records where a login was attempted after 18:00 and failed. Using select \* from log\_in\_attempts, I pulled all login data, then filtered the results with where login\_time > '18:00' and success = 0. This combination ensures we only capture after-hours failures, which are more likely to indicate a threat.

## Retrieve login attempts on specific dates

Following an incident report tied to May 9, 2022, I was tasked with reviewing login activity from both that day and the day before. This would help identify patterns or suspicious actions leading up to the event.

Here is the SQL query I used:

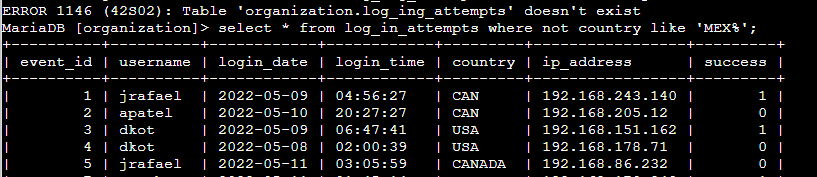


## This query pulls all login attempts from the log\_in\_attempts table and filters for specific dates using the or operator: login\_date = '2022-05-09' or login\_date = '2022-05-08'. This method allows quick comparison of user behavior across a critical time window.

## Retrieve login attempts outside of Mexico

After analyzing login locations, I focused on identifying attempts that originated from countries other than Mexico. This was necessary because some suspicious activity appeared to come from abroad.

The query I used is shown below:

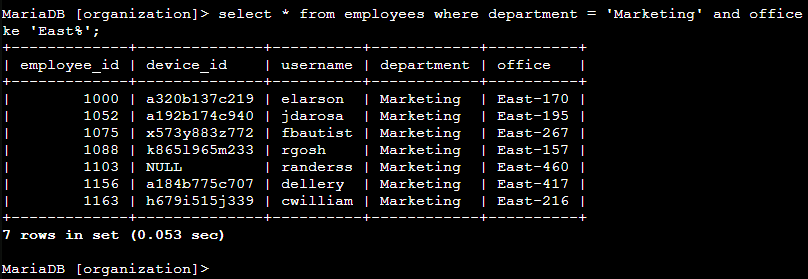


## In this case, I used not in combination with LIKE to exclude any countries whose name or code started with “MEX”. The condition NOT country LIKE 'MEX%' helped filter out both ‘MEX’ and ‘MEXICO’ values. The % wildcard matches any trailing characters after "MEX".

## Retrieve employees in Marketing

To prepare specific security updates, I had to identify employees in the Marketing department who are based in the East building offices.

Here is the SQL query I created to find this information:

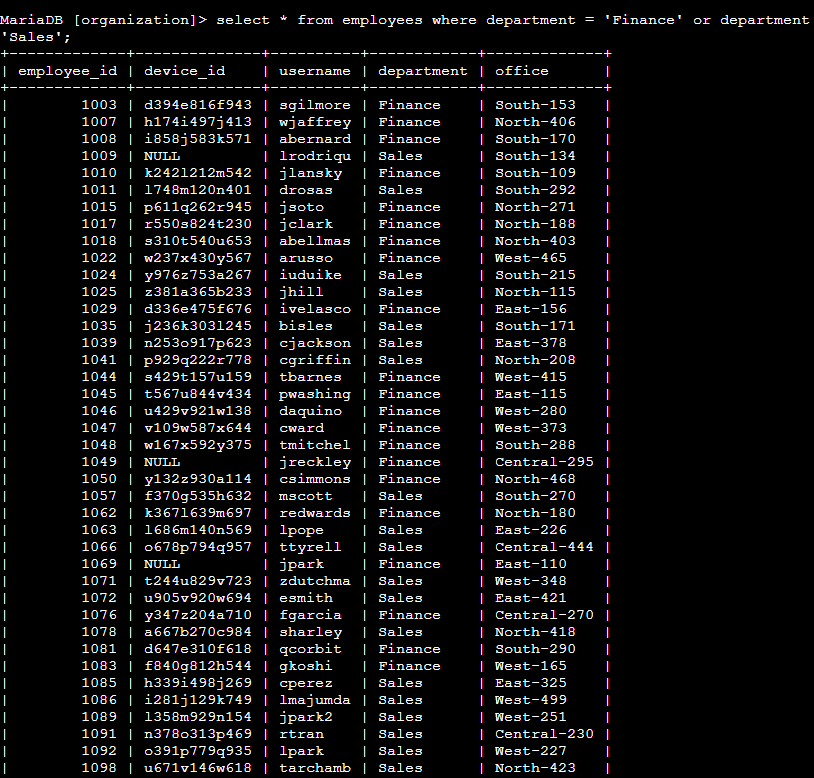


## I queried the employees table using the condition department = 'Marketing' and office like 'East%'. The and operator ensures that both criteria are met—employees must be in Marketing and located in an office that begins with "East", such as East-170 or East-320. The like keyword enables pattern matching to target just the East building.

## Retrieve employees in Finance or Sales

A separate security patch was needed for employees in either the Finance or Sales departments. To prepare for this, I queried the employee list to isolate users in those two departments.

The query I used is as follows:



The or operator is ideal for this scenario, since I wanted employees from either department. The query where department = 'Finance' OR department = 'Sales' ensured all relevant records were retrieved regardless of which group they belonged to.

## Retrieve all employees not in IT

Lastly, a general system update needed to be rolled out to all departments exceptInformation Technology, where the update had already been applied.

To locate those remaining employees, I ran the following query:

This query uses a where clause with != to exclude employees in the IT department. Specifically, department != 'Information Technology' filters out that group so that all others can be included in the update process.

## Summary

Throughout this project, I used SQL to apply precise filters for a range of security related tasks. These included narrowing down login attempts by time, date, and origin, and retrieving employees based on department and location. By applying operators like AND, OR, and NOT, along with date/time comparisons and pattern matching (LIKE), I was able to efficiently isolate the data needed to support critical security actions and system updates.