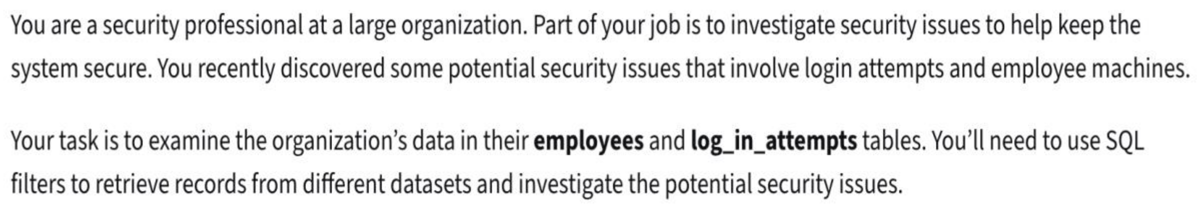
Apply filters to SQL queries



You are a security professional at a large organization. Part of your job is to investigate security issues to help keep the system secure. You recently discovered some potential security issues that involve login attempts and employee machines.

Your task is to examine the organization's data in their **employees** and **log_in_attempts** tables. You'll need to use SQL filters to retrieve records from different datasets and investigate the potential security issues.

Project description

In this project, I used SQL to investigate possible security issues involving employee login activity. My goal was to review data from two tables, employees and log_in_attempts, to find anything unusual that might point to a security concern.

I wrote SQL queries using filters like AND, OR, and NOT to narrow down the results and spot patterns such as repeated failed logins, access from unfamiliar machines, or activity that didn't match normal work hours. By connecting the information from both datasets, I was able to identify signs of suspicious behavior that could indicate a security risk.

This project helped me strengthen my skills in using SQL to investigate potential threats and understand how data analysis can play an important role in keeping systems secure.

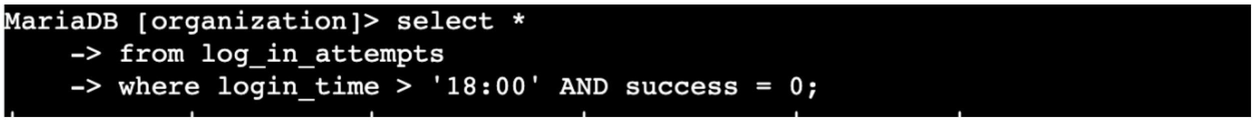
Retrieve after hours failed login attempts

After-hours login attempts can signal possible unauthorized access. I used SQL to retrieve all failed logins that occurred after 6:00 PM.

Query Explanation:
The query selects all data from the `log_in_attempts` table and filters results using two conditions combined with **AND**:

- `login_time > '18:00'` isolates login activity that occurred after normal work hours.
- `success = 0` retrieves only the failed attempts.

Together, these filters display all failed logins that occurred after business hours, which could indicate password-guessing attempts or compromised credentials.



```
MariaDB [organization]> select *
    -> from log_in_attempts
    -> where login_time > '18:00' AND success = 0;
```

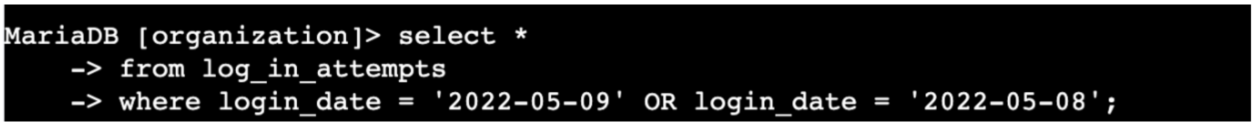
Retrieve login attempts on specific dates

I needed to investigate login attempts that occurred during and around a specific incident date.

Query Explanation:
Using **OR**, I filtered for activity that took place on **2022-05-09** or the day before:

- `login_date = '2022-05-09'` finds all logins on the target date.
- `login_date = '2022-05-08'` expands the range to include possible precursors.

This helped me identify any suspicious login behavior leading up to or following the event.

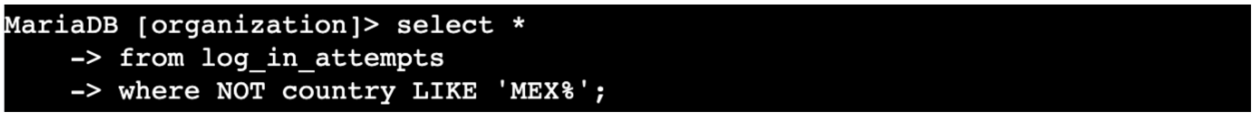


```
MariaDB [organization]> select *
    -> from log_in_attempts
    -> where login_date = '2022-05-09' OR login_date = '2022-05-08';
```

Retrieve login attempts outside of Mexico

Unusual login locations are potential red flags. I created a query to retrieve all login attempts from outside of Mexico.

Query Explanation:
The query filters the country field using **NOT LIKE 'MEX%'**, returning results where the country code does not start with "MEX."
The **%** wildcard allows matching both "MEX" and "MEXICO," while excluding them efficiently. This helps pinpoint logins from unfamiliar countries for further investigation.



```
MariaDB [organization]> select *
    -> from log_in_attempts
    -> where NOT country LIKE 'MEX%';
```

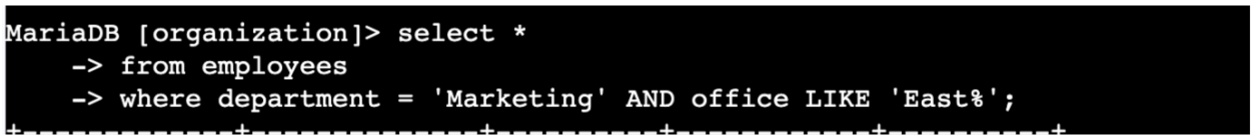
Retrieve employees in Marketing

To roll out a security update to Marketing department machines, I retrieved employees in that department who work in the East building.

Query Explanation:
The query uses **AND** with:

- `department = 'Marketing'`
- `office LIKE 'East%'`

The **LIKE** operator ensures the query captures all offices starting with “East,” such as “East-101.” This returns all Marketing employees located in the East building who require updates.



```
MariaDB [organization]> select *
    -> from employees
    -> where department = 'Marketing' AND office LIKE 'East%';
```

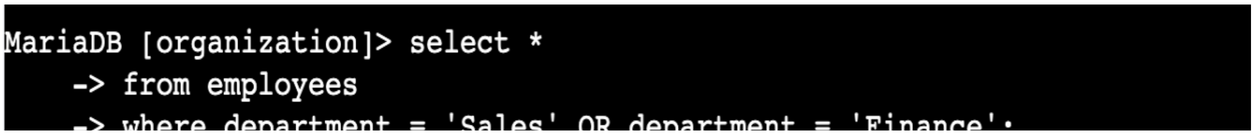
Retrieve employees in Finance or Sales

Different departments require different security configurations. I retrieved data for employees in either Finance or Sales.

Query Explanation:
The **OR** operator is used between:

- `department = 'Finance'`
- `department = 'Sales'`

This ensures the query returns employees from *either* department, rather than requiring both conditions.



```
MariaDB [organization]> select *
    -> from employees
    -> where department = 'Sales' OR department = 'Finance';
```

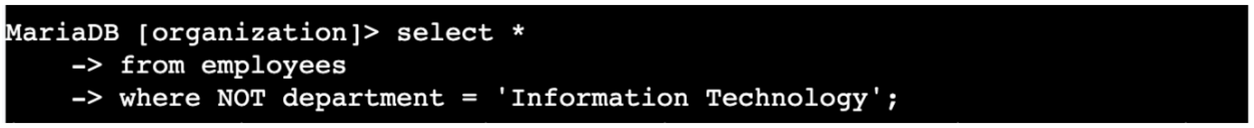
Retrieve all employees not in IT

To schedule general system updates, I needed to target employees outside of the IT department.

Query Explanation:
Using **NOT**, I filtered out IT employees:

- `WHERE NOT department = 'Information Technology'`

This returns all employees who do not belong to IT, allowing targeted updates without disrupting internal support systems.



```
MariaDB [organization]> select *
    -> from employees
    -> where NOT department = 'Information Technology';
```

Summary

Across these queries, I applied SQL filters to extract targeted data for security review and system maintenance. Using operators like **AND**, **OR**, **NOT**, and **LIKE**, I identified abnormal login activity, refined employee data, and demonstrated how SQL can be applied to real-world cybersecurity tasks to detect anomalies and improve organizational security.