# Apply filters to SQL queries

## Project description

In my role as a security professional at a large organization, my responsibilities include investigating potential security issues and ensuring the safety of our system. To achieve this goal, I have been instructed to investigate potential security issues and take necessary actions to ensure the system's safety. Recently we have experienced indications of compromise (IoC) relating to login attempts and employee machines. In the following paragraphs, I will provide examples of how I examined the organization’s data stored in the employees and log\_in\_attempts tables using SQL filters. The investigations performed are listed below:

* Investigate failed login attempts that were made after business hours, after 6p.m. (18:00).
* Investigate a suspicious event that occurred on 2022-05-09.
* Investigate logins that did not originate in Mexico.
* Obtain information about employees in the Marketing department who are located in all offices in the East building to assist the team in updating employee machines.
* Obtain information about employees in the Finance and Sales departments.
* Obtain information about employees in the Information Technology department.

## Retrieve after hours failed login attempts

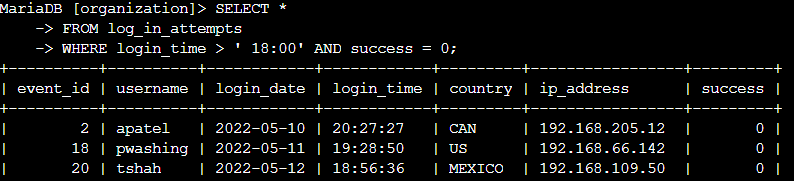
After normal business hours (office hours end at 18:00), there was a security incident that occurred. For this reason, all failed after hours login attempts must be reviewed to ensure no malicious activity is occurring. The log\_in\_attempts table contains information on when login attempts were made, the login\_time column. This table also contains the success column, which holds values of TRUE or FALSE to indicate whether the login was successful. It is worth noting that MySQL stores Boolean values as 1 for TRUE, and 0 for FALSE.

Given the information I have, I began to query the log\_in\_attempts database by selecting all rows of data (\*) from the log\_in\_attempts database and modifying that result with the WHERE clause to only return the login\_time column, supplemented by the AND operator to include all failed login attempts (FALSE or 0) after 18:00. The WHERE clause is the filter that will parse through the selected data. The AND operator includes data that meets the specified conditions. I specified this parameter with a condition, login\_time > ’18:00’. The last condition I used was used to filter for the failed login attempts, success = 0. Since I like numbers and they are less characters than TRUE or FALSE, I chose to use 0 in my filter. Please refer to the following code and screenshot for reference:

SELECT \*

FROM log\_in\_attempts

WHERE login\_time > '18:00' AND success = 0;



## Retrieve login attempts on specific dates

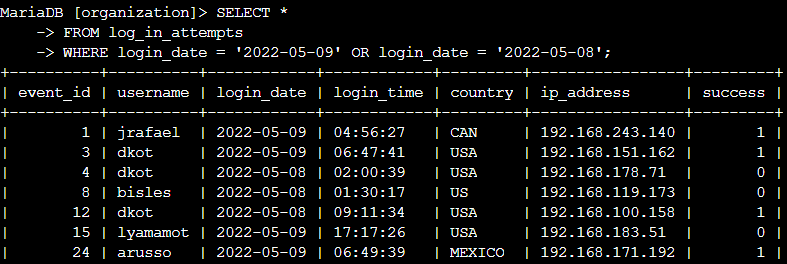
During the investigation a suspicious login attempt was noticed on 2022-05-09. I began to investigate the day as well as one day prior. As stated earlier, the login\_date column in the log\_in\_attempts table contains information on the dates when login attempts were made.

I began with selecting all data from the log\_in\_attempts table. Next, I used the WHERE clause and OR operator to filter for login attempts on 2022-05-09 or 2022-05-08. Filtering for this requires two conditions, login\_date = ‘2022-05-09’ and login\_date = ‘2022-05-08’ with the OR operator in between the two conditions. For refence, please refer to the code and screenshot below:

SELECT \*

FROM log\_in\_attempts

WHERE login\_date = '2022-05-09' OR login\_date = '2022-05-08';



## Retrieve login attempts outside of Mexico

After initial investigation, I determined that the login attempts occurred outside of Mexico. To assist my team in investigating the attempts, I ran another query.

I started the query by selecting all login attempts from the log\_in\_attempts table. The log\_in\_attempts table includes a country field and includes entries with 'MEX' and 'MEXICO'. Using the WHERE clause and NOT operator, I filtered for login attempts that occurred outside of Mexico. To avoid missing data, I used the NOT and LIKE operators and matched the query to the pattern 'MEX%'. The NOT operator selects all data except the specified value. The LIKE operator selects data that matches a specified pattern. Taking advantage of the percentage sign (%) we can look for all countries that start with the first three letters of Mexico and return anything that matches. Please refer to the code and screenshot below for reference:

SELECT \*

FROM log\_in\_attempts

WHERE NOT country LIKE 'MEX%';

A screen shot of a computer

Description automatically generated

## Retrieve employees in Marketing

I then assisted my team in updating computers for specific employees in the Marketing department to address the incident. The request was to select only employees who are in the Marketing department and in the East buildings.

I began by selecting all data from the employees table. This gave me a list of all the data in the table to help me to determine how I will enter the next query. Please refer to the code and screenshot for reference:

SELECT \*

FROM employees;

A screenshot of a computer screen

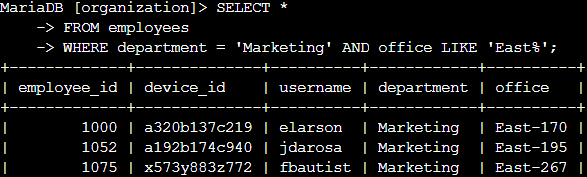
Description automatically generated

I noted that the office syntax was as such: 'East-170','East-320', etc.). Now that I understand the columns better, I began to query once more. I started by selecting all data from the employees table. Next, I used the WHERE clause with AND operator to filter for the employees who work in Marketing and in the East buildings. The first condition that I used was department = 'Marketing' and the second was office LIKE 'East%'. Combining these conditions with the AND operator returned Marketing employees from all East buildings. Please refer to the code and screenshot below for reference:

SELECT \*

FROM employees

WHERE department = 'Marketing' AND office LIKE 'East%';



## Retrieve employees in Finance or Sales

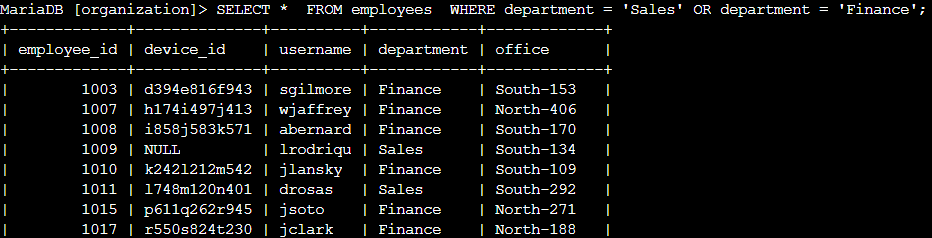
After helping to identify the Marketing department machines requiring an update, I then assisted my team in applying a different update to the computers of all employees in the Finance or the Sales department. Once again, I used data to obtain the data for this request.

I began this by first selecting all data from the employees table. Then, I used the WHERE clause and the OR operator to filter for the two departments. I did not use the AND operator because it would only include employees who work in both departments and not employees from either department. The first condition that I wrote was the department = ‘Sales’, the second condition I wrote was department = 'Finance'. With the OR operator separating the two conditions, I obtained the data I needed to help my team. Please refer to the code and screenshot below for reference.

SELECT \*

FROM employees

WHERE department = ‘Sales’ OR department = 'Finance';



## Retrieve all employees not in IT

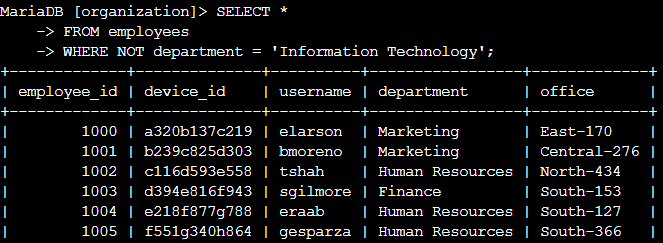
The final task essential to securing the systems at our organization was to apply a separate update to employee computers who are not in the Information Technology department. This update was already made to employee computers in the Information Technology department and must be deployed to the others.

I began the first step in this query by selecting all data from the employees table. I then used the WHERE clause and NOT operator. I filtered for all employees that are not listed in the Information Technology column/department. The NOT operator allowed this as it selects all data except the specified value. Please refer to the following code and screenshot for reference.

SELECT \*

FROM employees

WHERE NOT department = 'Information Technology';



## Summary

I efficiently obtained precise data regarding login attempts and employee machines from the log\_in\_attempts and employees tables by applying filters to SQL queries. Utilizing the power of AND, OR, and NOT operators, I tailored the filters to extract the exact information required for each task to help my team. Additionally, I harnessed the LIKE keyword with the percentage sign (%) wildcard to filter for specific patterns in the data. These efforts enable me to conduct in-depth investigations into potential security incidents and address security-related challenges within our system.