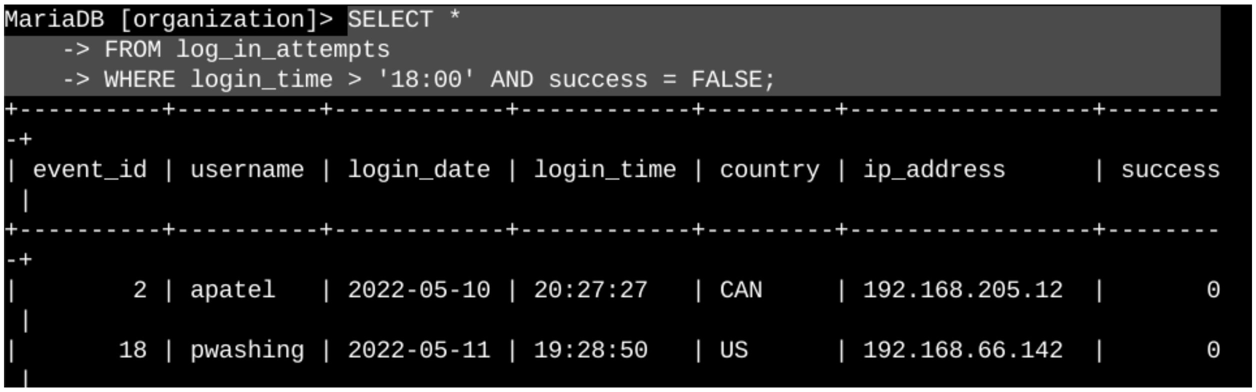
# Apply filters to SQL queries

## Project description

In this lab scenario, I assumed the role of a cybersecurity professional tasked with extracting crucial information from a database. The goal was to gather specific details about employees, their devices, and their respective departments. This data was essential for my team's investigation into potential security threats and for updating computers. My responsibility is to filter and retrieve the necessary information from the database to aid in the investigation process.

## Retrieve after hours failed login attempts

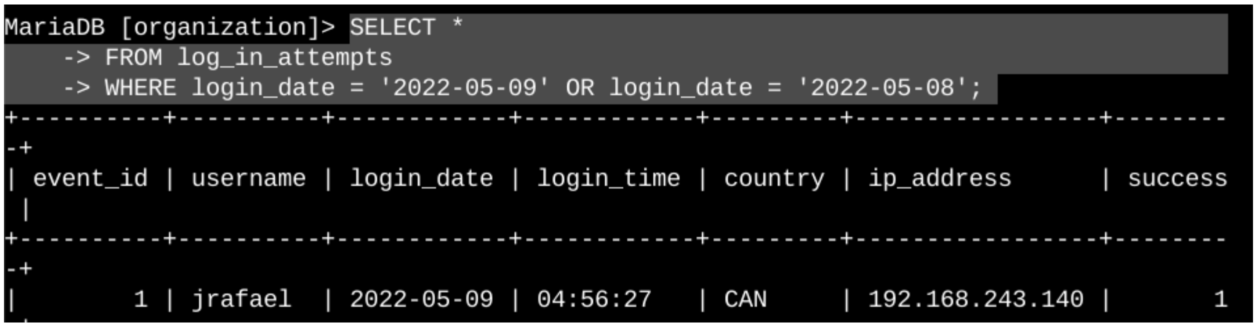
I wrote a query to retrieve data of all unsuccessful attempts after 18:00, the business's after-hours period. The lab starts with the organization database in the MariaDB shell, which is already open. I can write my query right away. As shown in the image below, I used the SELECT* command to retrieve all the data from the log_in_attempts table. Then I used the WHERE clause to filter the data required. This required me to use the AND command since I needed two conditions to be true simultaneously.



```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address      | success
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0
 |
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0
 |
```

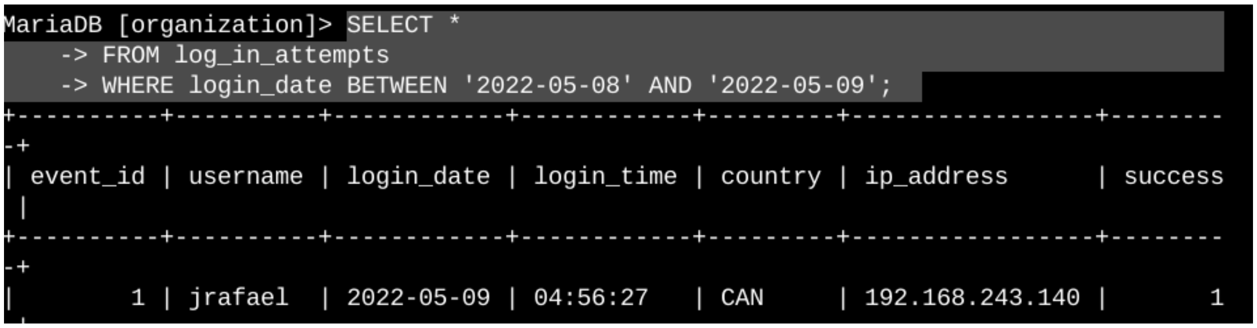
## Retrieve login attempts on specific dates

I want to retrieve all login attempts that occurred on ‘2022-05-08’ and ‘2022-05-09’ the days of the suspicious event. The lab required me to do this with an OR clause. Although the two conditions are from the same column, I still need to write two complete statements from each date as demonstrated below:



```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address     | success
 |
+----------+----------+------------+------------+---------+----------------+--------
-+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1
```

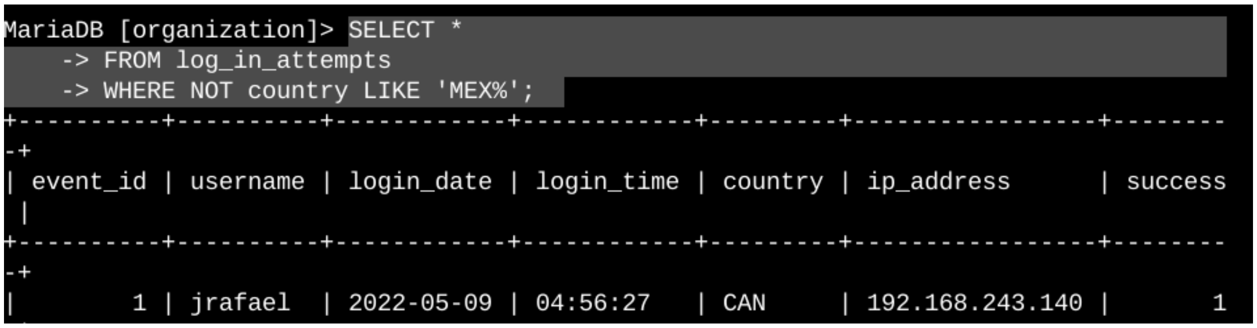
This can also be accomplished with BETWEEN command as shown below. I have to make sure to put the lower range data on the left and the higher range data on the right.



```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date BETWEEN '2022-05-08' AND '2022-05-09';
+----------+----------+------------+------------+---------+----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address     | success
 |
+----------+----------+------------+------------+---------+----------------+--------
-+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1
```

## Retrieve login attempts outside of Mexico

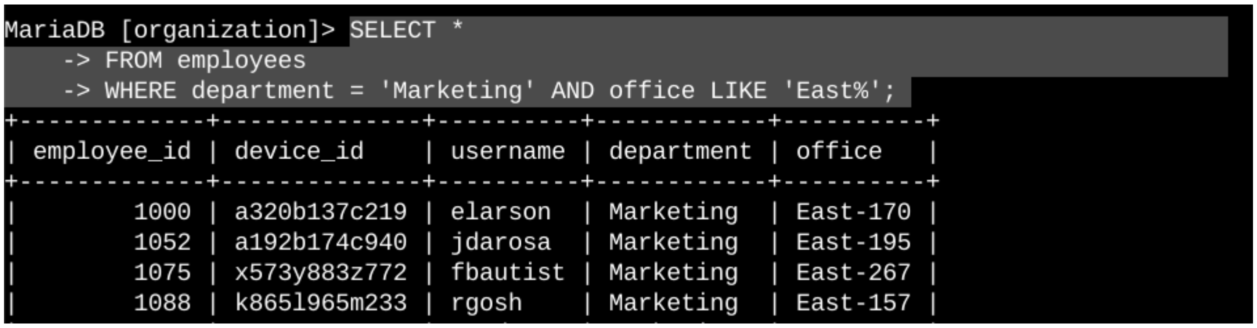
The scenario continues. The team is investigating logins that did not originate from Mexico. I need to find this information. It is noted that the country field includes entries with 'MEX' and 'MEXICO'. The lab requires me to use the NOT and LIKE operators and the matching pattern 'MEX%'. The next screenshot showed my query. The NOT operator lets me filter the country data that is not Mexico. The LIKE operator is used instead of operators like >, =, and < because I am looking for patterns of how Mexico can be written. The operator % acts as a wildcard, meaning that it will represent any length of string and different characters. It is useful when I am looking for different spellings of Mexico.



```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address     | success
 |
+----------+----------+------------+------------+---------+----------------+--------
-+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1
```

## Retrieve employees in Marketing

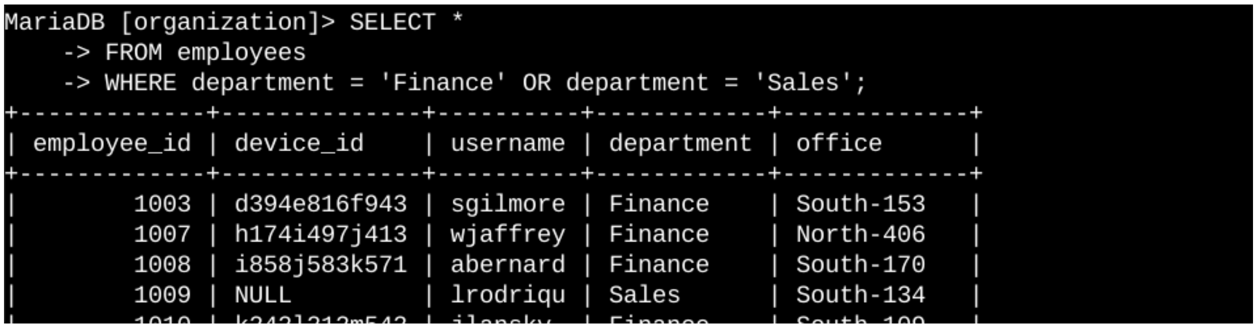
I need to retrieve information from the department and office columns in the employee table. My team is updating employee machines, and I need to obtain the information about employees in the ‘Marketing’ department who are located in all offices in the East building (such as ‘East-170’ or ‘East-320’). I used the % operator once more as shown below.



```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa  | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing  | East-267 |
|        1088 | k8651965m233 | rgosh    | Marketing  | East-157 |
```

## Retrieve employees in Finance or Sales

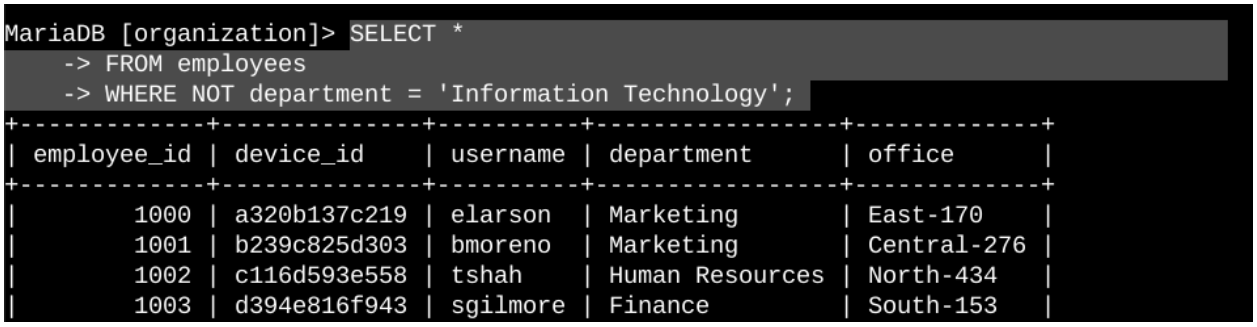
Now, my team needs to perform a different update to the computers of all employees in the Finance or the Sales department, and I need to locate information on these employees. OR clause will be best fit to use to find this information. Even though both conditions are based on the same column, I need to write out both full conditions. This means that I must specify department as the column in both conditions as demonstrated below.



```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+-------------+----------+------------+------------+
| employee_id | device_id   | username | department | office     |
+-------------+-------------+----------+------------+------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153  |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406  |
|        1008 | i858j583k571 | abernard | Finance    | South-170  |
|        1009 | NULL         | lrodriqu | Sales      | South-134  |
```

## Retrieve all employees not in IT

The scenario continues, and my team needs to make one more update. This update has already been made to employee computers in the Information Technology department. The team needs information about employees who are not in that department. In my query, I used the NOT operator to accomplish this as shown in the image below.



```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE NOT department = 'Information Technology';
+-------------+-------------+----------+-----------------+-------------+
| employee_id | device_id   | username | department      | office      |
+-------------+-------------+----------+-----------------+-------------+
|        1000 | a320b137c219 | elarson  | Marketing       | East-170    |
|        1001 | b239c825d303 | bmoreno  | Marketing       | Central-276 |
|        1002 | c116d593e558 | tshah    | Human Resources | North-434   |
|        1003 | d394e816f943 | sgilmore | Finance         | South-153   |
```

## Summary

With this lab, I gained and practiced practical experience in using SQL to:

- Run SQL queries to retrieve information from a database
- Apply AND, OR, and NOT operators to filter SQL queries
- Used LIKE and the % wildcard to filter for patterns

# Table formats

This document describes how the tables used for this portfolio activity are organized. The `organization` database contains the following two tables:

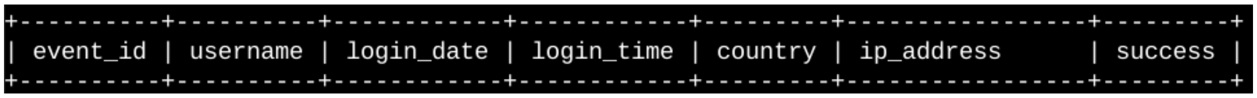
- `log_in_attempts`
- `employees`

## log_in_attempts

The `log_in_attempts` table has the following columns:

- `event_id`: The identification number assigned to each login event
- `username`: The username of the employee
- `login_date`: The date the login attempt was recorded
- `login_time`: The time the login attempt was recorded
- `country`: The country where the login attempt occurred
- `ip_address`: The IP address of that employee's machine
- `success`: The success of the login attempt; `FALSE` indicates a failed attempt

In the MariaDB shell, these columns are returned as:



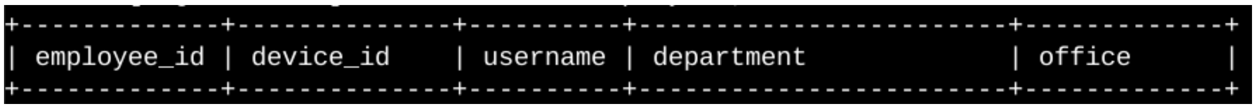
```
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
```

## employees

The `employees` table has the following columns:

- `employee_id`: The identification number assigned to each employee
- `device_id`: The identification number assigned to each device used by the employee
- `username`: The username of the employee
- `department`: The department the employee is in
- `office`: The office the employee is located in

In the MariaDB shell, these columns are returned as:



```
+-------------+---------------+----------+-----------------------+-------------+
| employee_id | device_id     | username | department            | office      |
+-------------+---------------+----------+-----------------------+-------------+
```