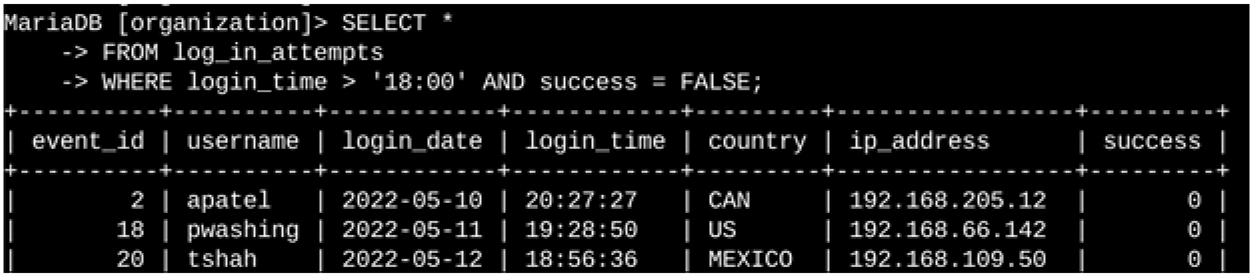
# Apply filters to SQL queries

## Project description

This portfolio demonstrates my ability to use SQL to filter and extract meaningful data from a database, which is essential for security investigations. By leveraging SQL filters, I analyzed login attempts and employee records to identify suspicious activities and ensure appropriate security measures were in place.

## Retrieve after hours failed login attempts

**Scenario:** A security incident potentially occurred after business hours, and I needed to identify all failed login attempts that happened after 18:00.

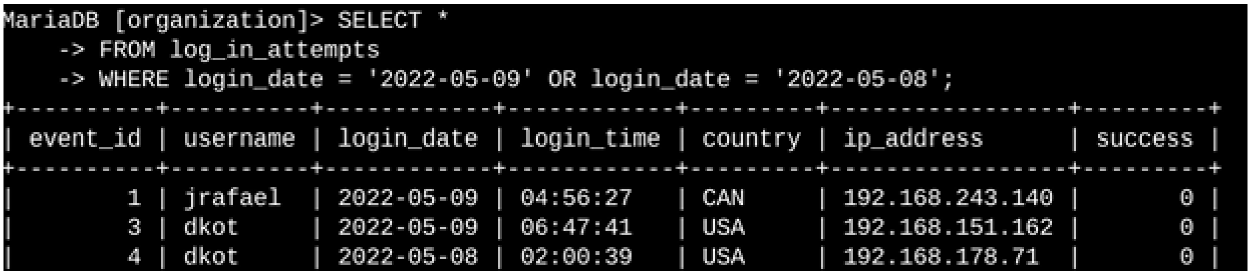


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
|       20 | tshah    | 2022-05-12 | 18:56:36   | MEXICO  | 192.168.109.50  |       0 |
```

This query retrieves all records from the log_in_attempts table where the login_time is later than 18:00 and the success column indicates a failed attempt (FALSE). This helps in identifying unauthorized access attempts after hours.

## Retrieve login attempts on specific dates

Scenario: A suspicious event occurred on May 9, 2022. To analyze activity surrounding this event, I retrieved all login attempts from both May 8 and May 9.

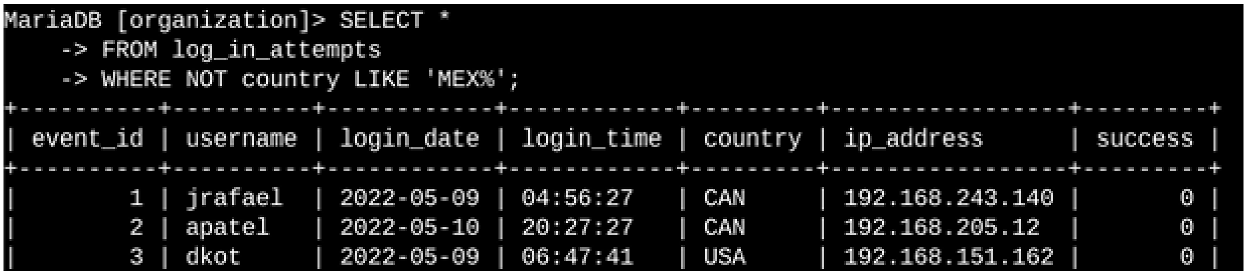


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

This query filters login attempts based on specific dates, ensuring that any unusual activity before or during the suspicious event is captured for further investigation.

## Retrieve login attempts outside of Mexico

Scenario: The security team discovered suspicious login activity but determined it did not originate in Mexico. I needed to filter out login attempts from Mexico and identify all other locations.

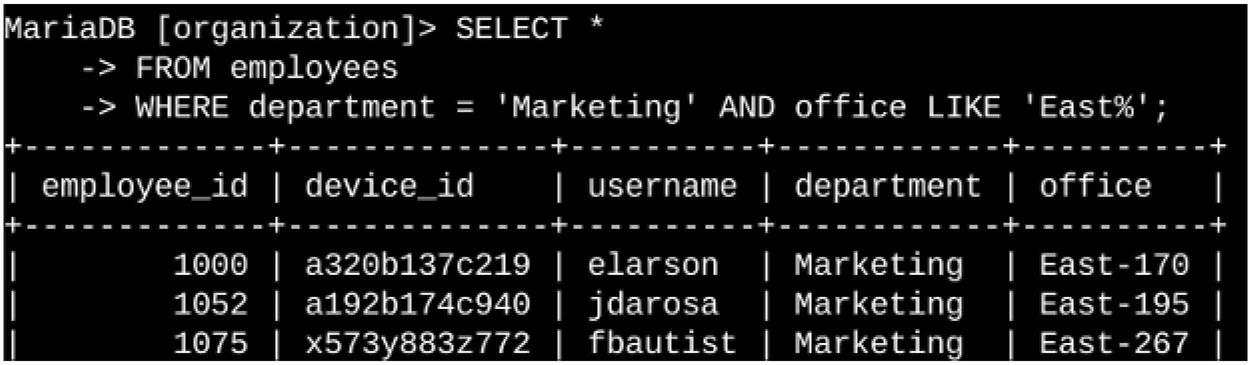


```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
```

Since the country column may contain 'MEX' or 'MEXICO', the LIKE 'MEX%' condition ensures that all variations are excluded from the results, focusing on login attempts from other regions.

## Retrieve employees in Marketing

Scenario: The IT team needed to perform security updates on employee machines in the Marketing department located in the East building. I retrieved the relevant employee records to facilitate this task.

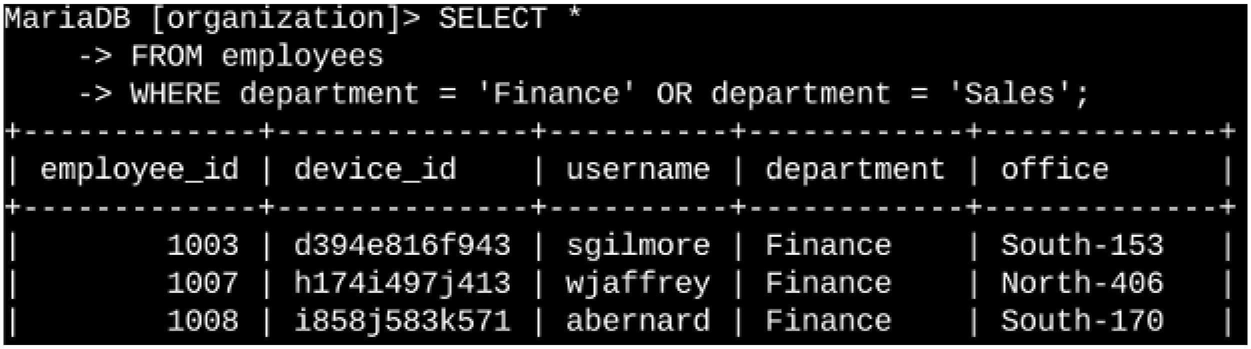


```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+----------+
| employee_id | device_id     | username | department | office   |
+-------------+---------------+----------+------------+----------+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

This query ensures that only employees from the Marketing department in offices that begin with 'East' (e.g., East-170, East-320) are retrieved.

## Retrieve employees in Finance or Sales

Scenario: A separate security update was required for employees in the Finance and Sales departments. I filtered the employee list accordingly

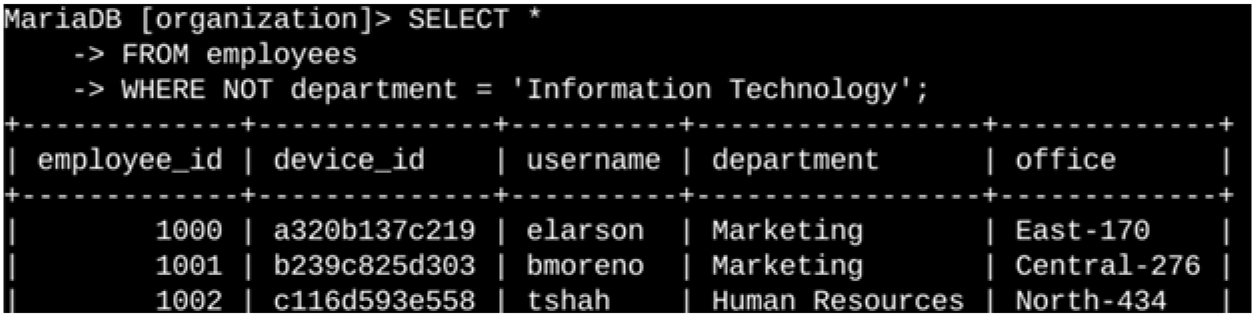


```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+--------------+----------+------------+-------------+
| employee_id | device_id    | username | department | office      |
+-------------+--------------+----------+------------+-------------+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

This query ensures that all employees in either the Finance or Sales department are included, helping IT target them for necessary security updates.

## Retrieve all employees not in IT

Scenario: Employees in the Information Technology (IT) department had already received security updates, so I needed to retrieve all employees who were not in IT.



```
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   |
```

This query excludes employees from the IT department, allowing IT administrators to focus on updating machines for all other employees.

## Summary

Through this activity, I effectively applied SQL queries with various filtering techniques, including logical operators (AND, OR, NOT), pattern matching, and date/time filtering. These skills are critical for investigating potential security incidents and ensuring that security policies are properly enforced across an organization.