Apply filters to SQL queries

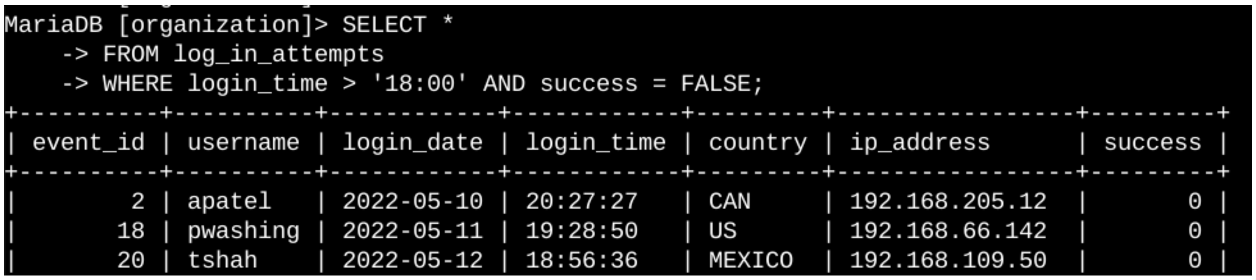
Project description

As a security professional at a large organization, I conducted a comprehensive security investigation using SQL queries to identify potential security threats and manage system updates. This project demonstrates my ability to leverage SQL filtering techniques to extract critical security insights from database logs and employee records, supporting data-driven security decisions and incident response protocols.

1. Retrieve after hours failed login attempts

There was a potential security incident that occurred after business hours (after 18:00). All after hours login attempts that failed need to be investigated.

The following code demonstrates how I created a SQL query to filter for failed login attempts that occurred after business hours.



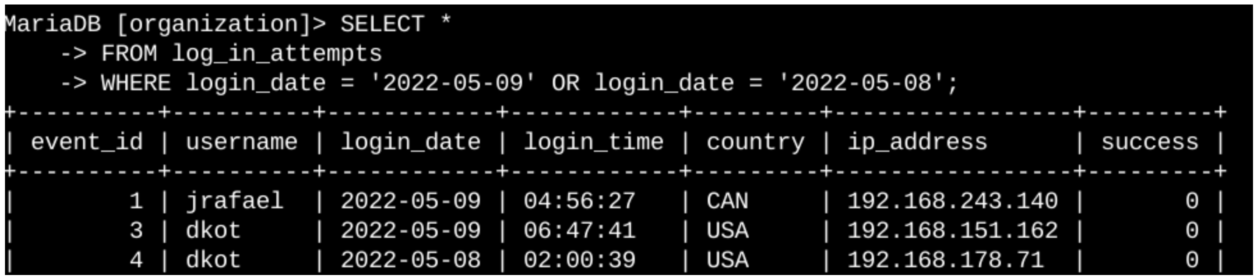
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 implements a multi-condition filter to identify potential unauthorized access attempts occurring outside standard business hours. By combining the temporal condition (`login_time > '18:00'`) with the authentication failure indicator (`success = FALSE`), I isolated suspicious activities that warrant further investigation. The AND operator ensures both conditions are met simultaneously, providing a focused dataset of high-risk login attempts that could indicate attempted security breaches or unauthorized access attempts during periods of reduced monitoring.

2. Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. Any login activity that happened on 2022-05-09 or on the day before needs to be investigated.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred on specific dates.



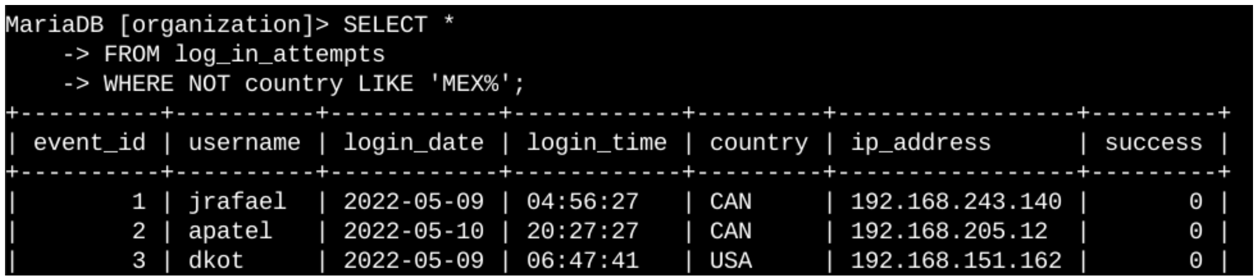
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 facilitates incident response by retrieving all authentication attempts within a specific temporal window surrounding a known security event. The OR operator enables the examination of login patterns across consecutive days, allowing for the identification of potential precursor activities or related incidents. This approach supports comprehensive incident timeline reconstruction and helps identify patterns that may indicate coordinated attack attempts or persistent threat actors.

3. Retrieve login attempts outside of Mexico

After investigating the organization's data on login attempts, I believe there is an issue with the login attempts that occurred outside of Mexico. These login attempts should be investigated.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred outside of Mexico.



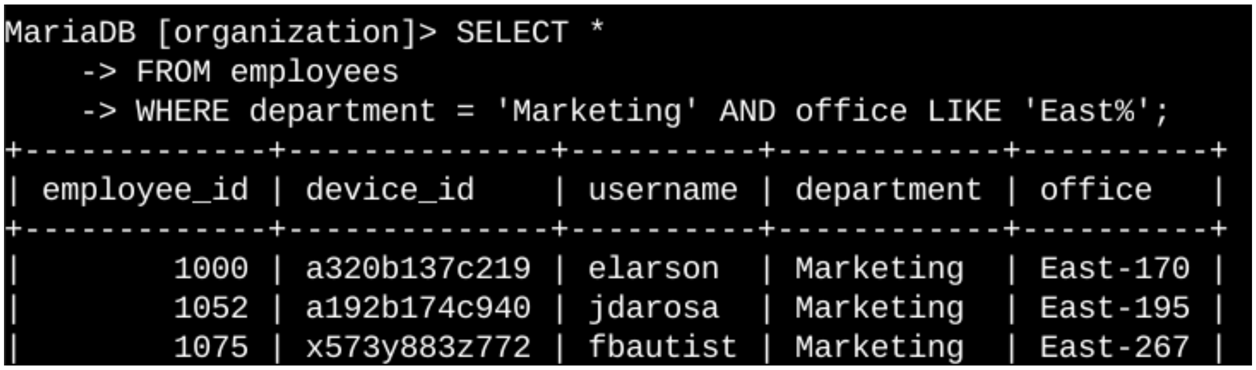
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+--------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

This query demonstrates advanced pattern matching using the LIKE operator with wildcard characters to handle data inconsistencies in country codes. The NOT operator effectively excludes all Mexico-originating traffic, where the pattern 'MEX%' captures both abbreviated ('MEX') and full ('MEXICO') country designations. This filtering technique is crucial for geographic threat analysis and helps isolate anomalous login attempts from unexpected locations, supporting the identification of potential compromised credentials or unauthorized access attempts.

4. Retrieve employees in Marketing

My team wants to update the computers for certain employees in the Marketing department. To do this, I have to get information on which employee machines to update.

The following code demonstrates how I created a SQL query to filter for employee machines from employees in the Marketing department in the East building.



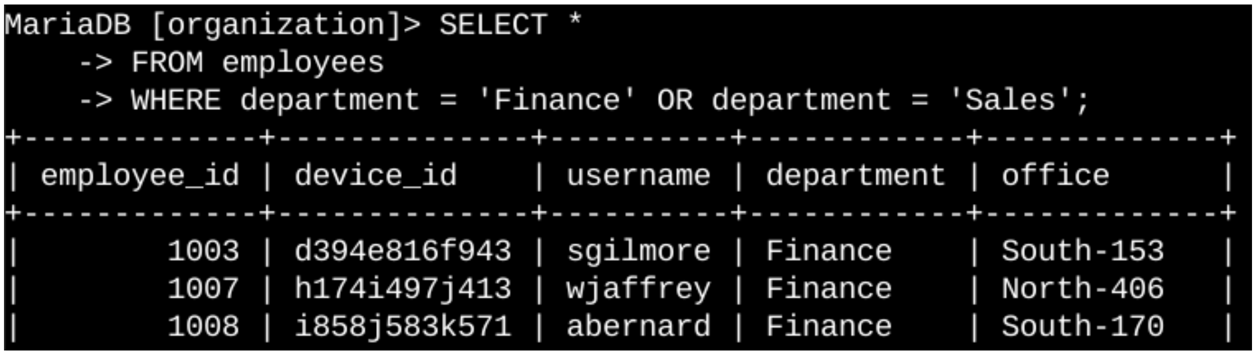
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+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

This targeted query identifies specific employee machines requiring security updates by combining departmental and location-based criteria. The AND operator ensures precise filtering, while the LIKE operator with the 'East%' pattern efficiently matches all East building office designations (e.g., East-170, East-320). This granular approach enables systematic patch management and ensures comprehensive coverage of vulnerable systems within specific organizational segments, minimizing security exposure while optimizing resource allocation.

5. Retrieve employees in Finance or Sales

The machines for employees in the Finance and Sales departments also need to be updated. Since a different security update is needed, I have to get information on employees only from these two departments.

The following code demonstrates how I created a SQL query to filter for employee machines from employees in the Finance or Sales departments.



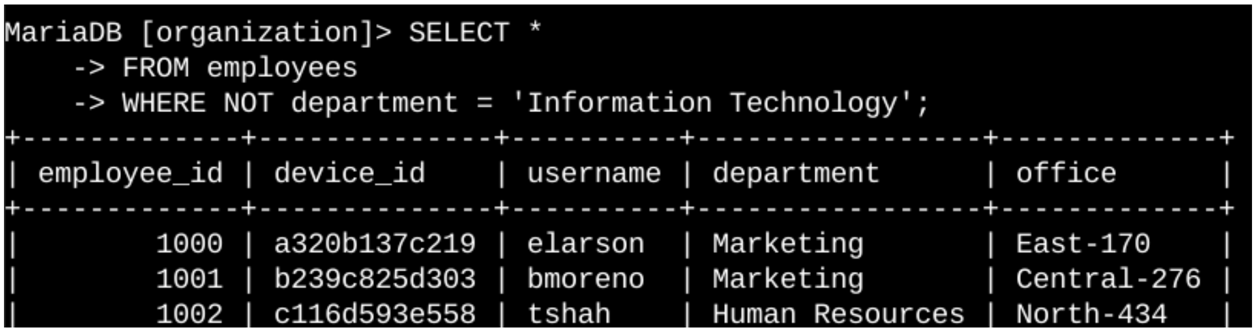
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+-------------+----------+------------+-------------+
| employee_id | device_id   | username | department | office      |
+-------------+-------------+----------+------------+-------------+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

This query implements inclusive filtering using the OR operator to identify employees across multiple departments requiring identical security updates. By targeting both Sales and Finance departments simultaneously, this approach streamlines the patch deployment process and ensures consistent security posture across critical business units that often handle sensitive financial and customer data. The query supports efficient bulk update operations while maintaining clear audit trails of affected systems.

6. Retrieve all employees not in IT

My team needs to make one more security update on employees who are not in the Information Technology department. To make the update, I first have to get information on these employees.

The following demonstrates how I created a SQL query to filter for employee machines from employees not in the Information Technology department.



```
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 exclusion-based query leverages the NOT operator to identify all employees requiring a specific security update, excluding the IT department which has already received the patch. This approach prevents redundant update attempts and potential system conflicts while ensuring comprehensive coverage across all non-IT departments. The query demonstrates efficient resource management and helps maintain version control integrity across the organization's infrastructure.

Summary

Through this comprehensive SQL analysis, I successfully identified and investigated suspicious login activities while managing targeted security updates across the organization. The queries demonstrated proficiency in using complex SQL operators (AND, OR, NOT) and pattern matching (LIKE) to extract actionable intelligence from security logs and employee databases. These techniques enabled rapid threat identification, geographic anomaly detection, and systematic patch management, contributing to enhanced organizational security posture and incident response capabilities. The ability to craft precise, efficient queries is essential for modern cybersecurity operations, enabling data-driven decision-making and proactive threat mitigation.