Apply filters to SQL queries

Project description

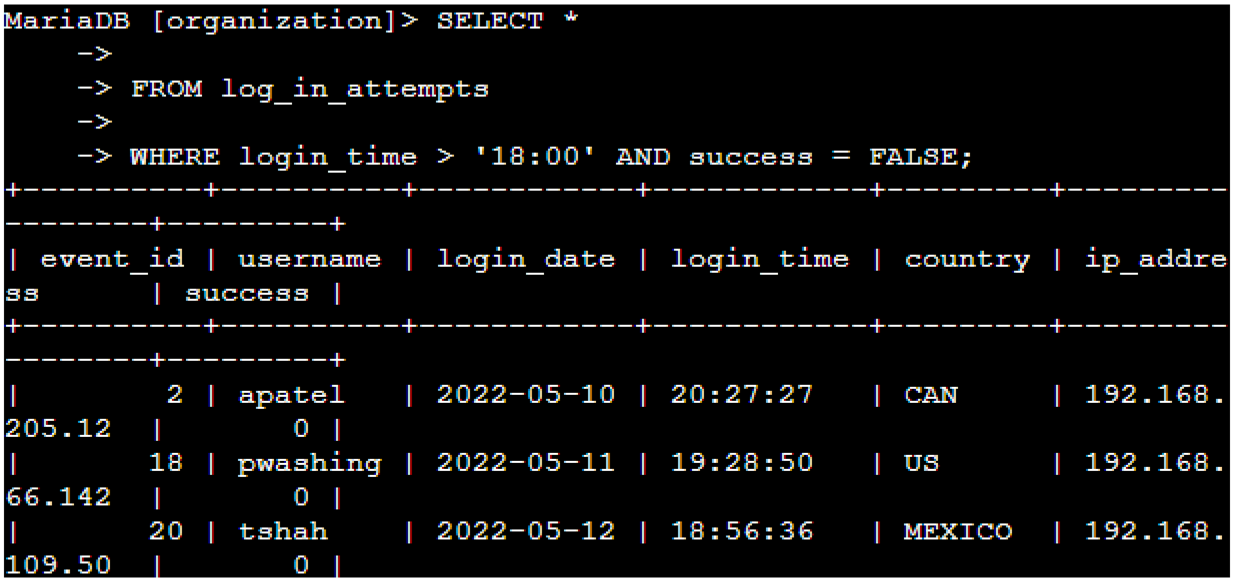
This project was completed as part of a cybersecurity training course. As a student, I practiced using SQL queries with filters to analyze system data, investigate potential security issues, and understand how databases can support security-related decisions.

The exercises simulate real-world scenarios involving login activity and employee information. The goal was to apply SQL filtering techniques correctly and explain the logic behind each query.

Retrieve after hours failed login attempts

A potential security incident occurred outside of normal business hours (after 18:00). All failed login attempts that happened after this time needed to be investigated.

To retrieve this information, I queried the log_in_attempts table and applied filters to identify login attempts that occurred after 18:00 and were unsuccessful.



```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+---------
--------+---------+
| event_id | username | login_date | login_time | country | ip_addre
ss       | 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 |
```

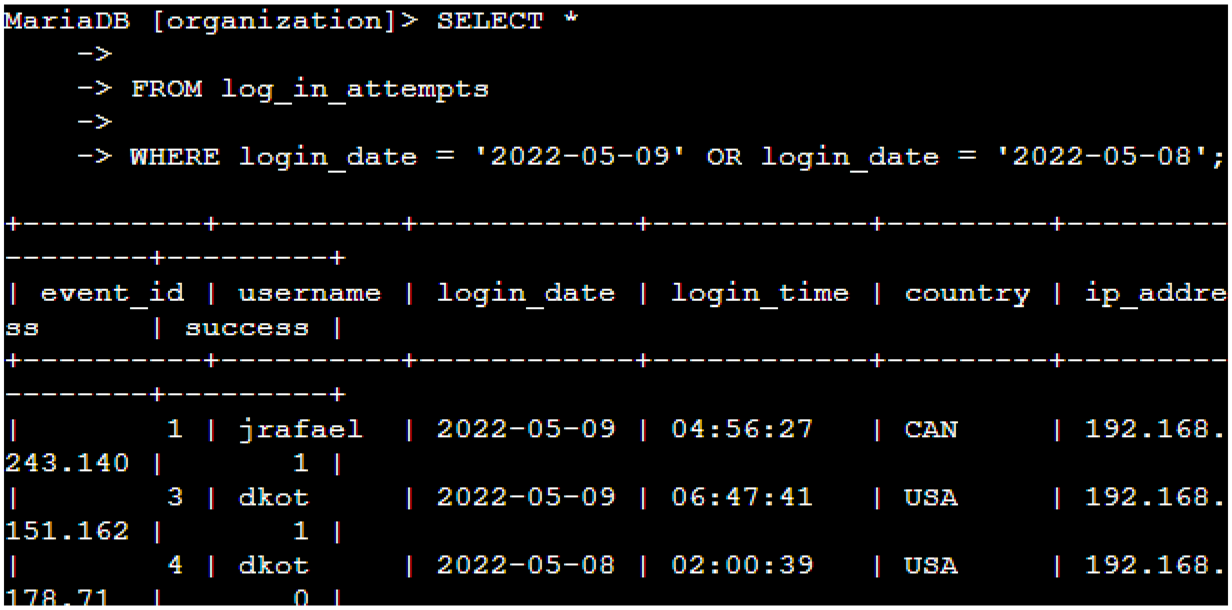
I used a WHERE clause with the AND operator to combine both conditions:

- login_time > '18:00' to filter for after-hours activity
- success = FALSE to filter for failed login attempts

This query returned only the records that matched both conditions.

Retrieve login attempts on specific dates

A suspicious event was reported on 2022-05-09. To investigate further, all login attempts that occurred on that date or the previous day needed to be reviewed.



```
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_addre
ss      | success |
+----------+----------+------------+------------+---------+---------
--------+----------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.
243.140 |        1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.
151.162 |        1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.
178.71  |        0 |
```

I queried the log_in_attempts table and used a WHERE clause with the OR operator to filter login activity on the two relevant dates:

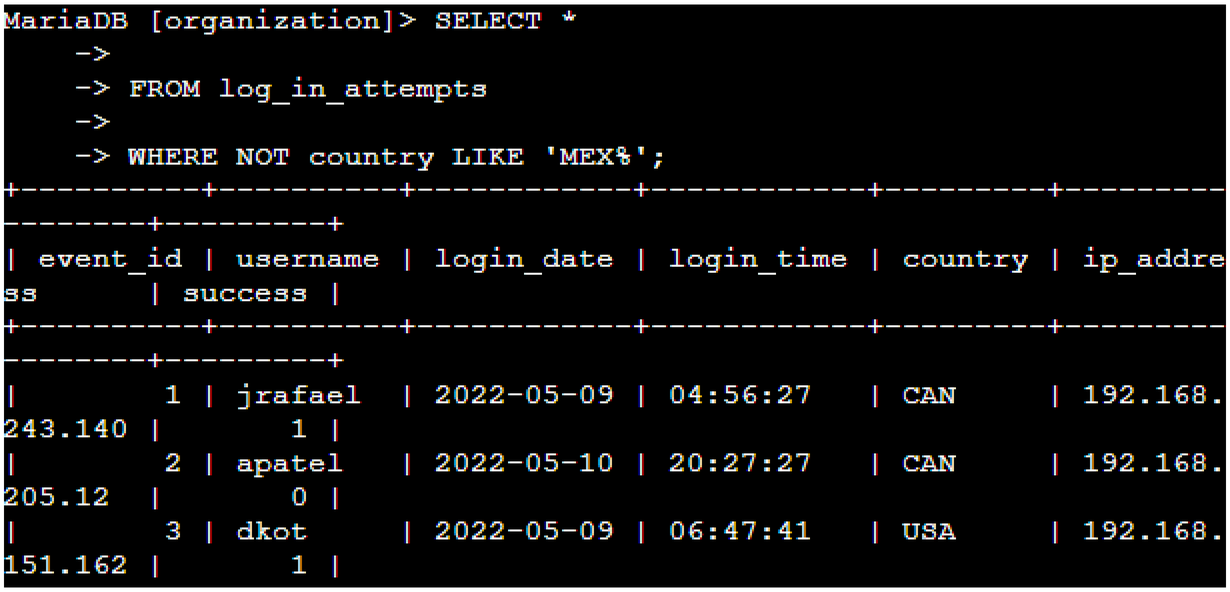
- login_date = '2022-05-09'
- login_date = '2022-05-08'

Using the OR operator ensured that login attempts from either date were included in the results.

Retrieve login attempts outside of Mexico

During the investigation, I identified potentially suspicious login attempts originating outside of Mexico. These login attempts required additional review.

To retrieve this data, I selected all records from the log_in_attempts table and applied a WHERE clause with the NOT operator. I used the LIKE keyword with the pattern MEX% to account for different representations of Mexico in the dataset (such as MEX and MEXICO).



```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+---------
--------+---------+
| event_id | username | login_date | login_time | country | ip_addre
ss        | success |
+----------+----------+------------+------------+---------+---------
--------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.
243.140 |       1 |
|        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 |       1 |
```

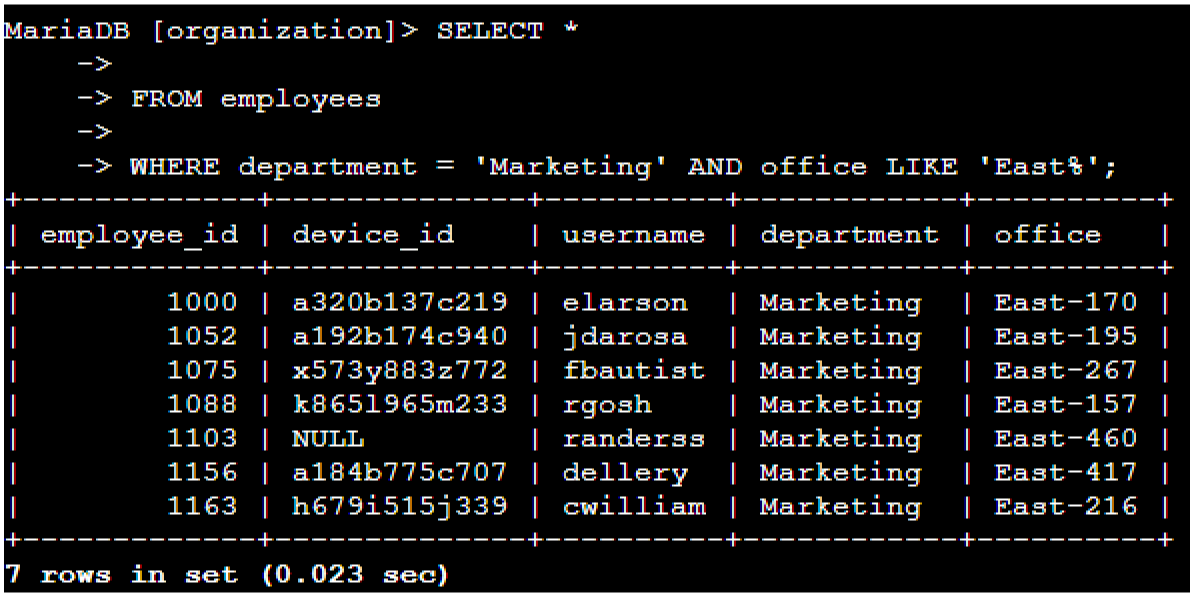
The ‘NOT LIKE 'MEX%' condition filtered out all login attempts originating from Mexico, returning only those from other countries.

Retrieve employees in Marketing

My team needed to update the computers of employees in the Marketing department who work in the East building.

To do this, I queried the employees table and applied two filters using the AND operator:

- department = 'Marketing' to identify Marketing employees
- office LIKE 'East%' to identify employees located 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 |
|        1088 | k8651965m233 | rgosh    | Marketing  | East-157 |
|        1103 | NULL         | randerss | Marketing  | East-460 |
|        1156 | a184b775c707 | dellery  | Marketing  | East-417 |
|        1163 | h679i515j339 | cwilliam | Marketing  | East-216 |
+-------------+-------------+----------+------------+----------+
7 rows in set (0.023 sec)
```

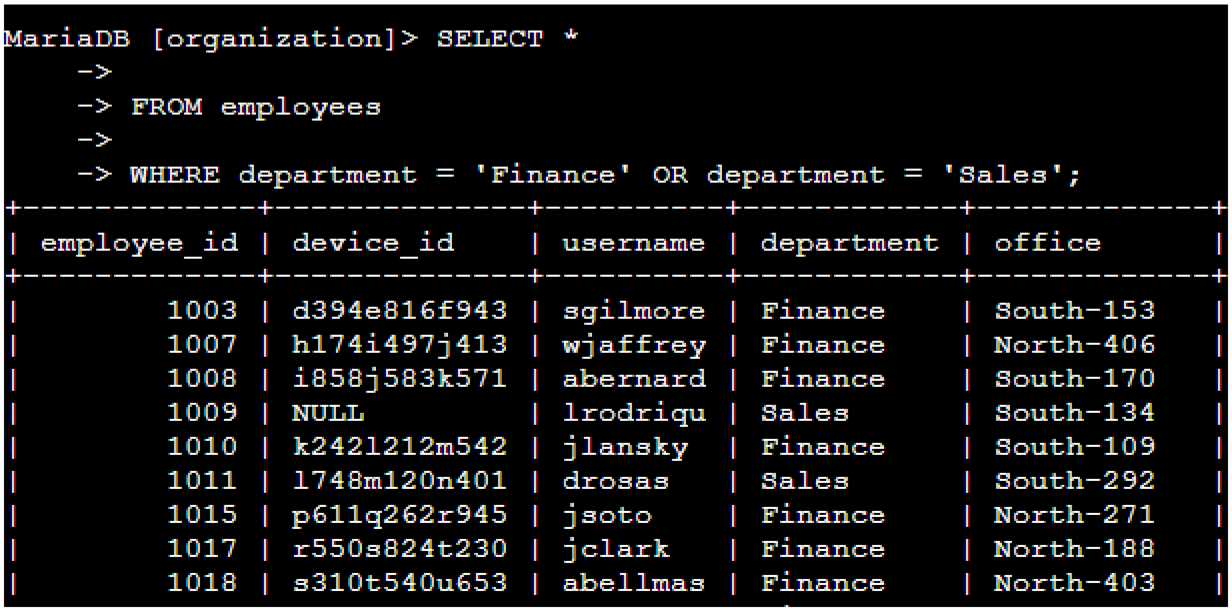
The LIKE operator was used because the office column includes both the building name and specific office numbers.

Retrieve employees in Finance or Sales

Different security updates were required for employees in the Finance and Sales departments. Therefore, I needed to retrieve information only for employees in these two departments.

I queried the employees table and used a WHERE clause with the OR operator:

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

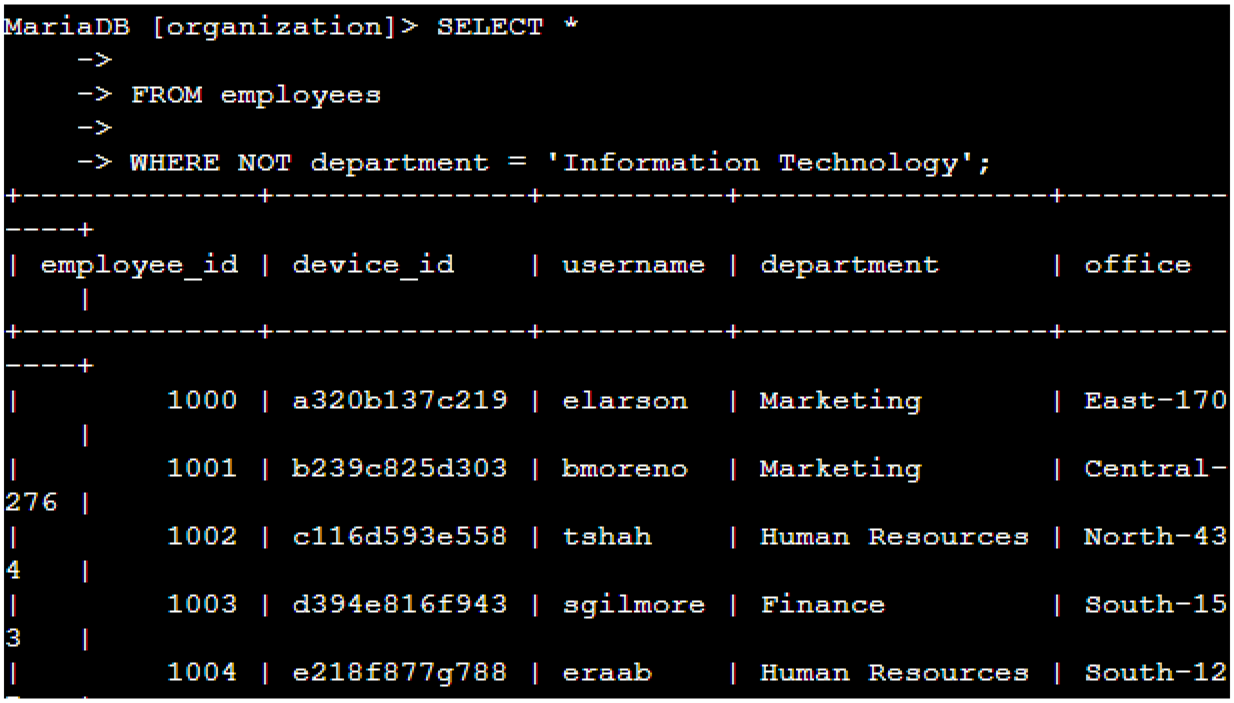

```
MariaDB [organization]> SELECT *
    ->
    -> FROM employees
    ->
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+---------------+----------+------------+-------------+
| employee_id | device_id     | username | department | office      |
+-------------+---------------+----------+------------+-------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153   |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406   |
|        1008 | i858j583k571 | abernard | Finance    | South-170   |
|        1009 | NULL          | lrodriqu | Sales      | South-134   |
|        1010 | k2421212m542 | jlansky  | Finance    | South-109   |
|        1011 | 1748m120n401 | drosas   | Sales      | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance    | North-271   |
|        1017 | r550s824t230 | jclark   | Finance    | North-188   |
|        1018 | s310t540u653 | abellmas | Finance    | North-403   |
```

This query returned all employees who belong to either department.

Retrieve all employees not in IT

Finally, my team needed to apply an additional security update to all employees who are not part of the Information Technology department.

To retrieve this information, I selected all data from the employees table and used a WHERE clause with the NOT operator to exclude employees from the IT 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-43
4   |
|        1003 | d394e816f943 | sgilmore | Finance         | South-15
3   |
|        1004 | e218f877g788 | eraab    | Human Resources | South-12
```

This allowed me to identify all employees who required the update.

Summary

In this project, I used SQL filters to retrieve targeted information related to system security and employee management. I worked with two tables, log_in_attempts and employees, and applied filtering techniques using:

- WHERE clauses
- Logical operators such as AND, OR, and NOT
- Pattern matching with LIKE and the % wildcard

These queries allowed me to efficiently investigate security incidents and identify employee machines that required updates.