Apply filters to SQL queries

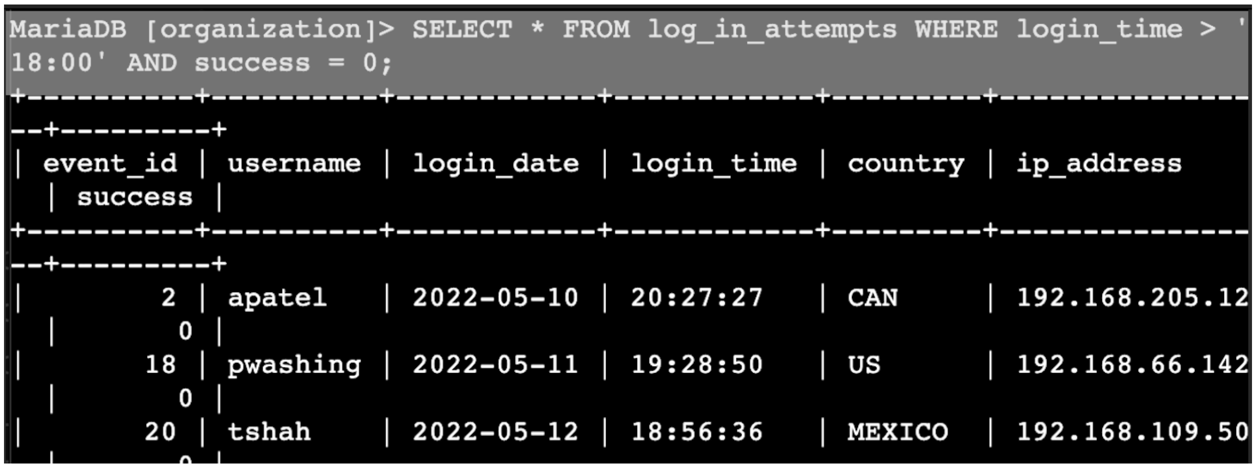
Project description

This project focuses on applying various SQL filters and operators to query data from a relational database in support of IT and security operations. The goal of this project is to show my efficiency in using SQL operation to retrieve specific information of employees and login data based on department, location, login success or country. I used various SQL conditions such as WHERE, LIKE, NOT, AND, and OR to practice filtering and retrieving information from the database.

Retrieve after hours failed login attempts

I began with investigating the suspicious activity that occurred outside of regular business hours. I queried the log_in_attempts table to identify all failed login attempts that took place after 18:00 (6pm). This was done to detect unauthorised access attempts when systems are less likely to be monitored.

The SQL query used:
SELECT * FROM log_in_attempts
WHERE login_time > '18:00'
AND success = 0;



```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_time > '
18:00' AND success = 0;
+----------+----------+------------+------------+---------+---------------
--+---------+
| 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
```

This query filters the data to return only those records where:

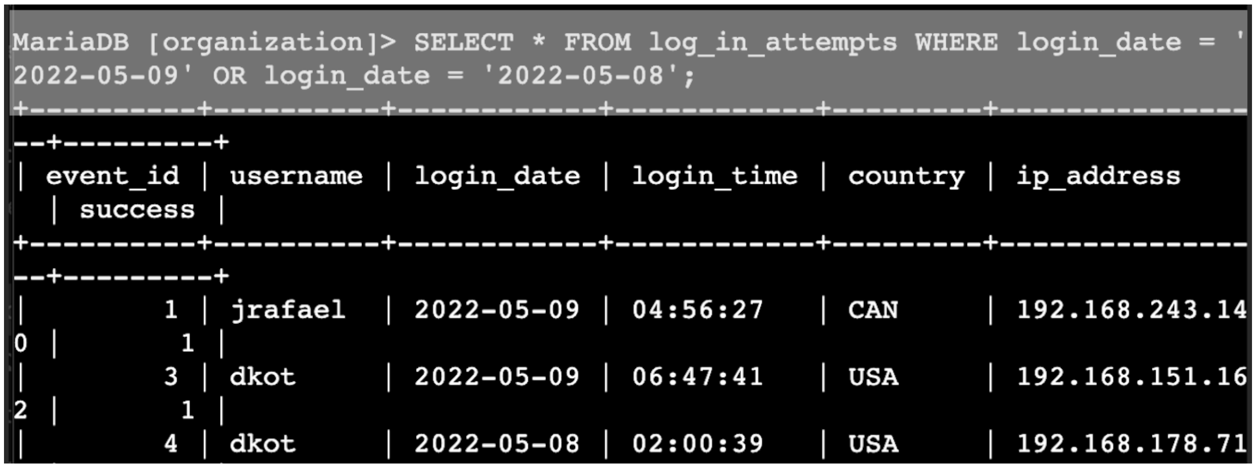
- login_time is later than 18:00 (i.e., after business hours), and
- success equals 0, indicating a **failed login attempt**.

The results revealed multiple(19 failed attempts) failed attempts from different users across various locations including the **US, Canada, and Mexico**, confirming a need for further review into the source and intent of these login attempts.

Retrieve login attempts on specific dates

To investigate a suspicious event reported on the 2022-05-09, I used a SQL query to retrieve all login attempts that occurred on both May 9th, 2022 and the day before that (May 8th, 2022). I was able to review the login activity leading up to the incident.

The SQL query used:
SELECT * FROM log_in_attempts
WHERE login_date = '2022-05-09'
OR login_date = '2022-05-08';



```
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     | 192.168.243.14
0 |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.16
2 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71
```

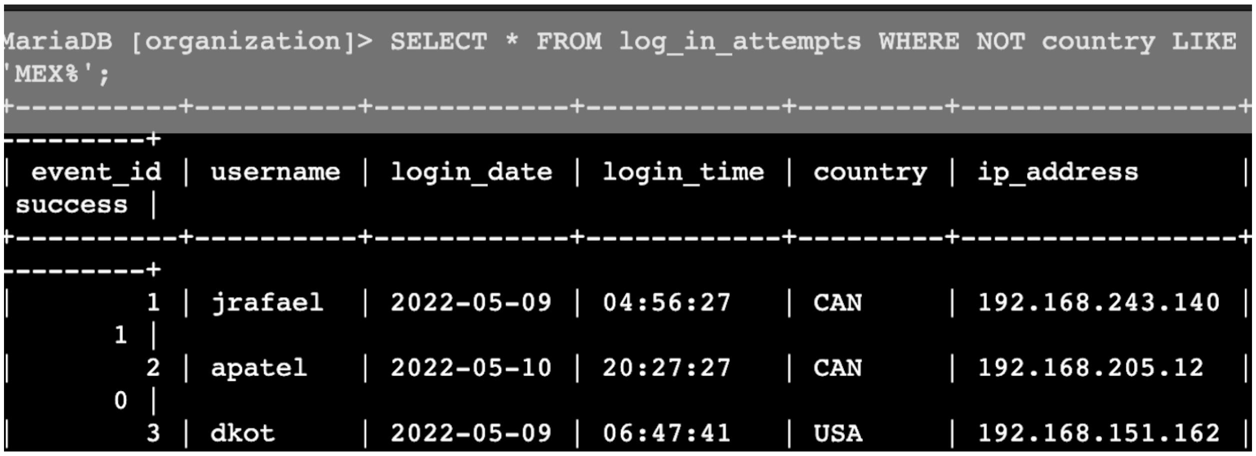
This query uses the OR operator to include rows where the login_date matches **either of the two specified dates**. The goal was to ensure no relevant activity before or on the day of the event was missed.

This returned login attempts from users across various countries and times, including both **successful and failed attempts**, giving a full view of access behaviour around the time of the incident.

Retrieve login attempts outside of Mexico

To investigate suspicious activity while excluding all login attempts from **Mexico,** I used the SQL query that filters out any entries where the country column contains the pattern '**MEX**' or '**MEXICO**'.

The SQL query used:
SELECT * FROM log_in_attempts
WHERE NOT country LIKE 'MEX%';



```
lariaDB [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 |
|        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 |
```

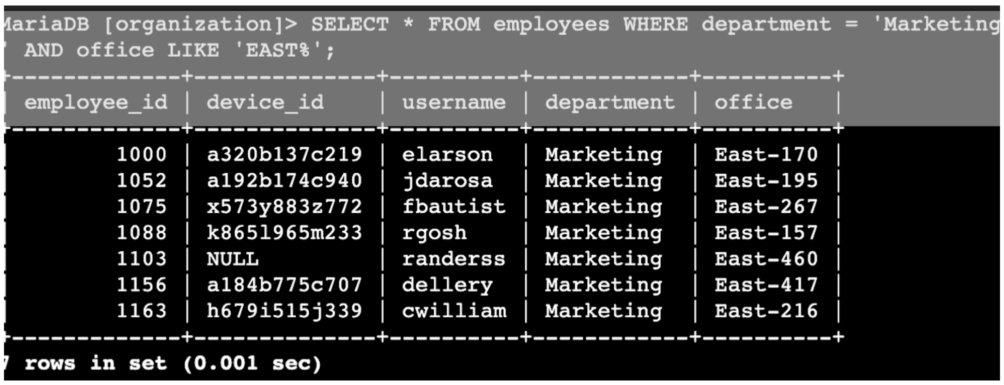
This query uses the NOT and LIKE keywords together. The LIKE 'MEX%' condition matches any country value that **starts with 'MEX'**, covering both 'MEX' and 'MEXICO'. By applying NOT to this condition, the query excludes these rows and retrieves only those login attempts **from other countries**.

This is helpful in narrowing down the investigation to activity that did **not** originate from Mexico, focusing the analysis on possibly foreign threats.

Retrieve employees in Marketing

To help with upcoming security updates I needed to identify all employees in the Market department in the East building. I used the employees table and filtered by both department and office columns.

The SQL query used:
SELECT * FROM employees
WHERE department = 'Marketing'
AND office LIKE 'East%';



```
lariaDB [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 (0.001 sec)
```

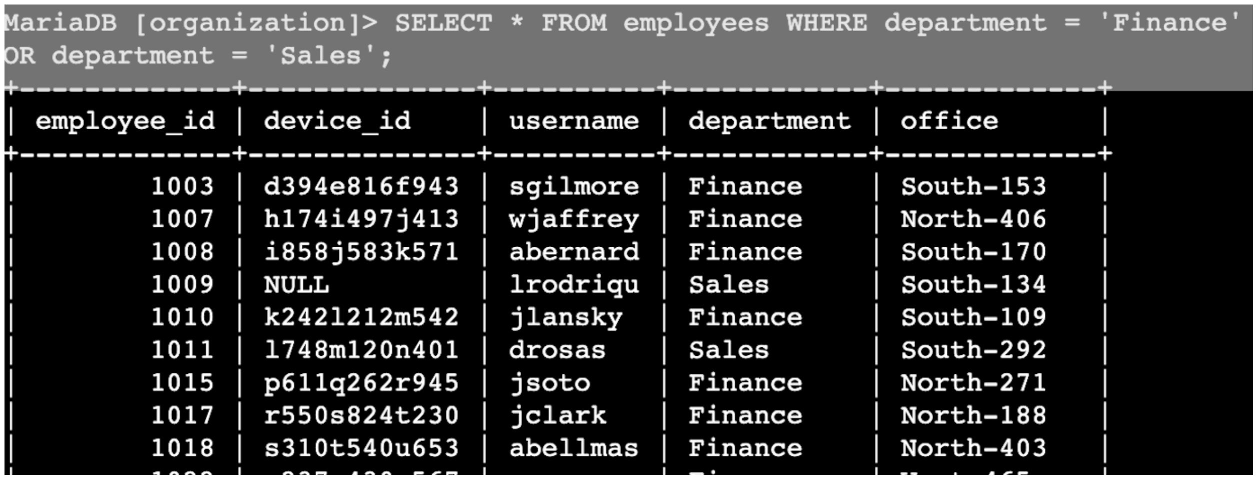
- department = 'Marketing' ensures we're only selecting rows for employees in the Marketing department.
- office LIKE 'East%' matches any office name that begins with "East", such as "East-170", "East-195", etc. This covers all East building locations, regardless of the specific room number.

This query returns a list of employees that are specifically in the **Marketing department** and **located in any office within the East building**. It is useful for targeting devices or users for security updates in that specific group.

Retrieve employees in Finance or Sales

Our security team needed to identify machines belonging to employees who work in either the **Finance** or **Sales** departments so we can perform department-specific security updates. To do this, I queried the employees table using the following **SQL:**

SELECT * FROM employees
WHERE department = 'Finance'
OR 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   |
```

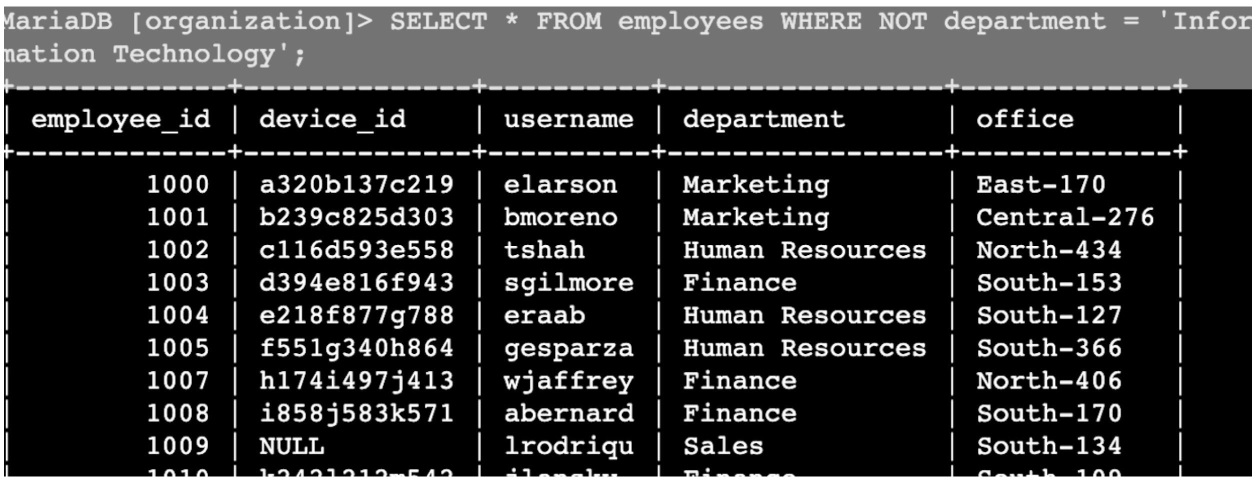
- department = 'Finance' filters for employees in the Finance department.
- OR department = 'Sales' adds employees in the Sales department to the results.
- The OR operator ensures that **either condition** being true will include the employee in the results.

This query returns **all employees who work in Finance or Sales**, regardless of their office location.

Retrieve all employees not in IT

Our team needs to apply an update to all employees **except those in the Information Technology department**, as they've already received the update. To get this list, I used the following **SQL query:**

SELECT * FROM employees
WHERE NOT department = 'Information Technology';



```
MariaDB [organization]> SELECT * FROM employees WHERE NOT department = 'Infor
mation 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    |
|        1004 | e218f877g788  | eraab     | Human Resources | South-127    |
|        1005 | f551g340h864  | gesparza  | Human Resources | South-366    |
|        1007 | h174i497j413  | wjaffrey  | Finance         | North-406    |
|        1008 | i858j583k571  | abernard  | Finance         | South-170    |
|        1009 | NULL          | lrodriqu  | Sales           | South-134    |
```

- The WHERE clause filters the data.
- NOT department = 'Information Technology' ensures that only employees whose department is **not**"Information Technology" are included.
- This filter is useful when we want to **exclude a specific group** while including everyone else.

This query returns a list of all employees **not working in the IT department**, helping us target the correct machines for the update.

Summary

In this project, I applied SQL filters to query specific information from the **`log_in_attempts`** and **`employees`** tables. I used filtering techniques like **`WHERE`, `AND`, `OR`,** and **`NOT`** demonstrating my knowledge on how to accomplish various security tasks and maintenance requests. I also used pattern matching with **`LIKE`** and `%` wildcards to identify entries based on partial matches, such as building locations or country codes. These SQL filtering techniques were crucial in helping my team monitor security events and manage employee device updates effectively.