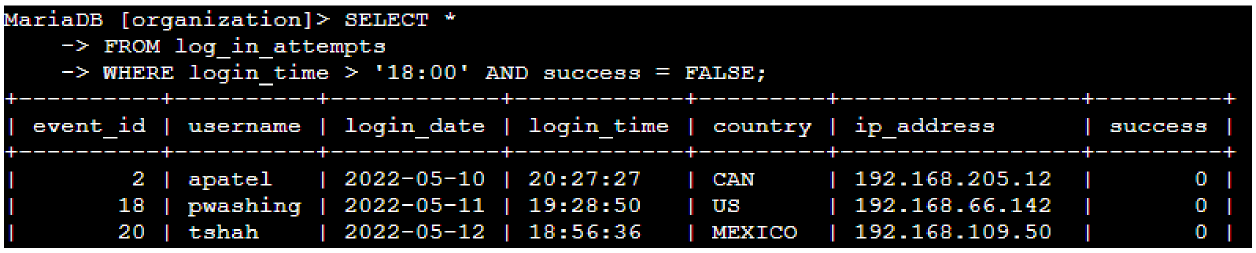
Apply filters to SQL queries

Project description

My organization is focused on enhancing system security, which involves investigating potential security incidents and updating employee machines as required. As a security professional, it is my responsibility to ensure the system remains safe by querying relevant databases to identify suspicious activity and target specific machines for updates. The following steps demonstrate how I applied SQL filters to retrieve data from the `employees` and `log_in_attempts` tables to perform these security-related tasks.

Retrieve after hours failed login attempts

To investigate a potential security incident that occurred after business hours (after 18:00), I needed to identify all failed login attempts during that time frame.



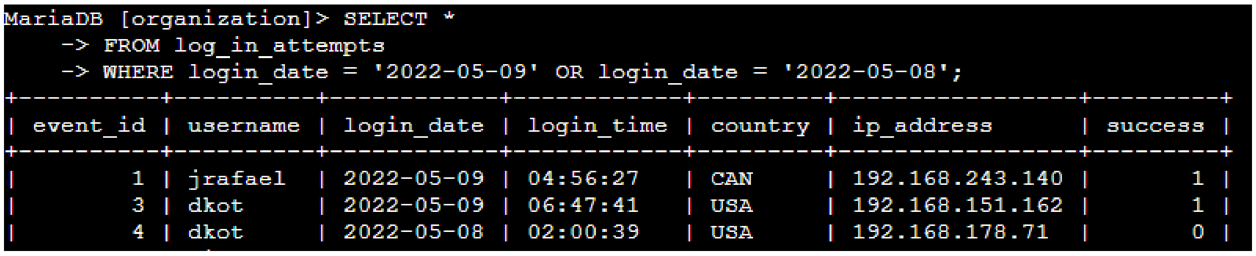
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+---------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

How it works:

- I started by selecting all data from the `log_in_attempts` table.
- I used a `WHERE` clause with the `AND` operator to combine two conditions.
- The first condition, `login_time > '18:00'`, filters for attempts made after 18:00.
- The second condition, `success = FALSE` , filters for failed attempts.

Retrieve login attempts on specific dates

I investigated a suspicious event that occurred on 2022-05-09 and the 2022-05-08 by querying all login activity for those two dates.



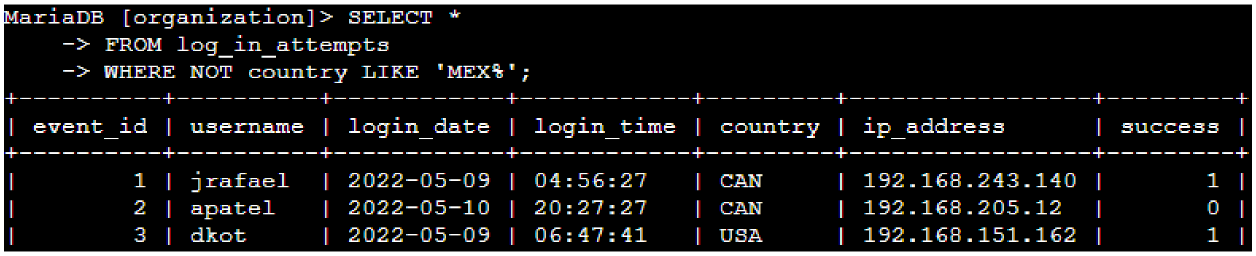
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success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

How it works:

- I selected all data from the `log_in_attempts` table.
- The `WHERE` clause uses the `OR` operator to include results that match either specified date.
- The conditions `login_date = '2022-05-09'` and `login_date = '2022-05-08'` ensure the query returns activity from both target days.

Retrieve login attempts outside of Mexico

The security team determined that a specific suspicious event did not originate from Mexico, so I queried for login attempts occurring in all other countries.



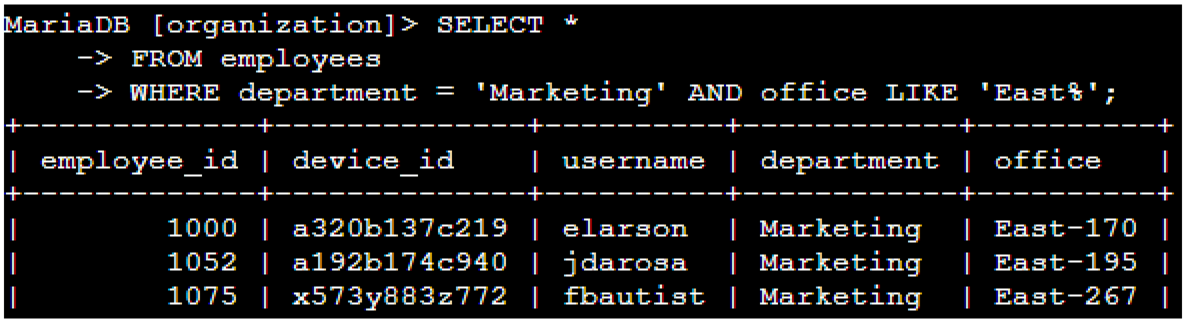
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+--------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

How it works:

- I filtered the `log_in_attempts` table using the `NOT` operator to exclude specific results.
- I used the `LIKE` keyword with the pattern `'MEX%'` because the database lists Mexico as both "MEX" and "MEXICO".
- The percentage sign (`%`) acts as a wildcard representing any number of unspecified characters, ensuring all variations of Mexico are excluded from the results.

Retrieve employees in Marketing

To prepare for security updates on machines in the Marketing department located in the East building, I queried the `employees` table for relevant records.



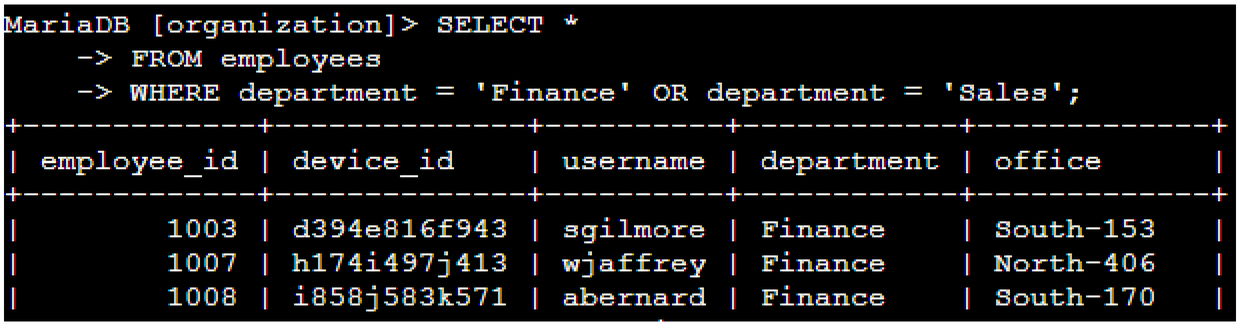
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+-----------+
| employee_id | device_id     | username | department | office    |
+-------------+---------------+----------+------------+-----------+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

How it works:

- I selected all columns from the `employees` table.
- The `AND` operator was used to ensure the results satisfied both department and location requirements.
- The condition `department = 'Marketing'` isolated the target team.
- I used `LIKE 'East%'` for the `office` column because specific office numbers follow the building name, and the `%` wildcard matches any office located in the East building.

Retrieve employees in Finance or Sales

A different security update was required for employees in either the Finance or Sales departments.



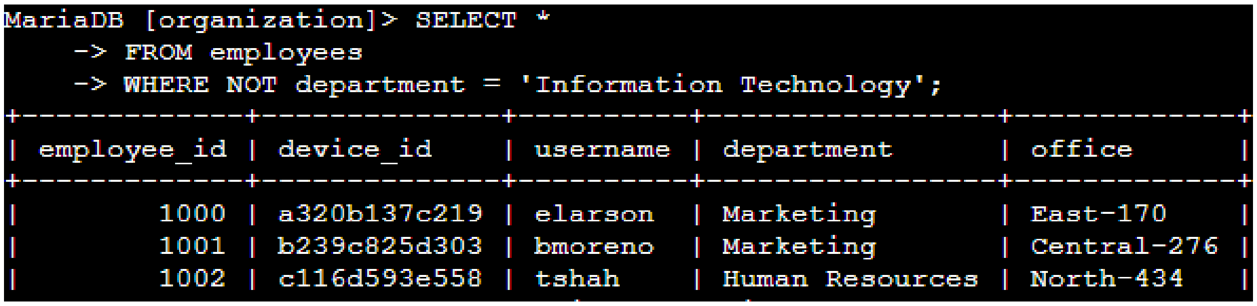
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+---------------+----------+------------+---------------+
| employee_id | device_id     | username | department | office        |
+-------------+---------------+----------+------------+---------------+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

How it works:

- I queried the `employees` table.
- I used the `OR` operator to include all employees who belong to *either* the Finance or the Sales department.
- This ensures that employees from both departments are captured in a single result set for the update.

Retrieve all employees not in IT

The final update was required for all employees across the organization, except for those in the Information Technology (IT) department who had already been updated.



```
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   |
```

How it works:

- After selecting from the `employees` table, I used the `NOT` operator in the `WHERE` clause.
- The condition `NOT department = 'Information Technology'` excludes the IT department from the list, returning everyone else who still needs the security update.

Summary

I successfully applied SQL filters using `AND`, `OR`, and `NOT` operators to extract specific security and employee data from the database. By utilizing pattern matching with the `LIKE` keyword and wildcards, I was able to handle variations in data entry, such as country codes and office locations. These queries provided the precise information needed to investigate security incidents and efficiently target machines for critical updates