# Apply filters to SQL queries

## Project description

My organization is enhancing system security. My responsibility is to ensure its safety by investigating potential security issues and updating employee computers as necessary. The following steps illustrate how I used SQL queries with filters to carry out security-related tasks.

## Retrieve after hours failed login attempts

A potential security incident occurred after business hours (after 18:00). All failed login attempts during this period need to be investigated.

The following SQL query demonstrates how I filtered for these failed login attempts after 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 |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of its output. This query filters for failed login attempts occurring after 18:00. I began by selecting all records from the `log_in_attempts` table. Then, using a `WHERE` clause with an `AND` operator, I narrowed the results to include only login attempts made after 18:00 that were unsuccessful. Specifically, the condition `login_time > '18:00'` filters for attempts after 6 PM, and `success = FALSE` selects only the failed attempts.

## Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. All login activity from that day and the previous day must be investigated.

The following SQL query demonstrates how I filtered for login attempts occurring on these 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 |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of the output. This query retrieves all login attempts from either 2022-05-09 or 2022-05-08. I began by selecting all data from the `log_in_attempts` table. Then, using a `WHERE` clause with an `OR` operator, I filtered the results to include only login attempts on those two specific dates. The condition `login_date = '2022-05-09'` filters for logins on May 9, 2022, and `login_date = '2022-05-08'` filters for logins on May 8, 2022.

## Retrieve login attempts outside of Mexico

After reviewing the organization's login attempt data, I identified suspicious activity originating from outside Mexico. These login attempts require further investigation.

The following SQL query demonstrates how I filtered for login attempts made 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 |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of the output. This query retrieves all login attempts from countries other than Mexico. I began by selecting all records from the `log_in_attempts` table. Then, using a `WHERE` clause with `NOT` and the `LIKE` operator, I filtered out entries matching Mexico.

The pattern `MEX%` was used to cover both "MEX" and "MEXICO" since the `%` wildcard matches any sequence of characters following the prefix.

## Retrieve employees in Marketing

My team needs to update the computers assigned to employees in the Marketing department. To identify which machines require updates, I created a SQL query to filter for employee machines belonging to Marketing staff located in the East building.

The following code demonstrates this query:

```
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 |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of the output. This query returns all employees who work in the Marketing department and are located in the East building. I began by selecting all records from the `employees` table. Then, using a `WHERE` clause with an `AND` operator, I filtered the results to include only employees in the Marketing department whose office matches the pattern `East%`. The `LIKE 'East%'` condition accounts for office entries that start with "East" followed by specific office numbers. The `department = 'Marketing'` condition filters for the Marketing department employees.

## Retrieve employees in Finance or Sales

The machines for employees in the Finance and Sales departments also require updates. Since a different security update applies, I need to identify employees exclusively from these two departments.

The following SQL query demonstrates how I filtered for employee machines belonging to 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   |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of the output. This query retrieves all employees who work in either the Finance or Sales departments. I started by selecting all records from the `employees` table. Then, using a `WHERE` clause with the `OR` operator, I filtered the results to include employees from both departments. The condition `department = 'Finance'` selects employees in Finance, while `department = 'Sales'` selects those in Sales. The `OR` operator ensures employees from either department are included.

## Retrieve all employees not in IT

My team needs to apply one more security update to employees outside the Information Technology department. To proceed, I first need to identify these employees.

The following SQL query demonstrates how I filtered for employee machines belonging to 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   |
```

The first part of the screenshot shows my SQL query, and the second part displays a sample of the output. This query retrieves all employees who are not in the Information

Technology department. I began by selecting all records from the `employees` table. Then, I applied a `WHERE` clause with the `NOT` operator to exclude employees from that department.

## Summary

I applied filters to SQL queries to extract specific information about login attempts and employee machines using two tables: `log_in_attempts` and `employees`. To refine the results, I used the `AND`, `OR`, and `NOT` operators, as well as the `LIKE` operator with the `%` wildcard to match patterns.