# Apply filters to SQL queries

## Project description

As part of my role in strengthening organizational security, I am responsible for ensuring systems are protected, investigating potential security threats, and maintaining up-to-date employee devices. I used SQL with filters to perform several key security-related tasks, such as identifying suspicious activity, isolating vulnerable endpoints, and retrieving specific user data for audits and compliance checks.

## 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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query filters for failed login attempts that occurred after 18:00. First, I selected all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with an `AND` operator to filter my results to show only login attempts that:

- occurred after 18:00 (`login_time > '18:00'`)
- were unsuccessful (`success = FALSE`)

This helped me investigate possible unauthorized login activity outside normal working hours.

## 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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all login attempts that occurred on **2022-05-09** or **2022-05-08**.
First, I selected all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with an `OR` operator to filter my results to show only login attempts that:

- occurred on 2022-05-09 (`login_date = '2022-05-09'`)
- or occurred on 2022-05-08 (`login_date = '2022-05-08'`)

This allowed me to investigate login activity surrounding the suspicious event date.

## 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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all login attempts that occurred in countries **other than Mexico**.

First, I selected all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with `NOT LIKE` to filter for countries that **don't match** Mexico.

I used `LIKE 'MEX%'` as the pattern, because in the dataset, Mexico is represented as both **MEX** and **MEXICO**.

The percent sign `%` acts as a wildcard in SQL and represents any number of characters — so `MEX%` covers both cases.

This query helped me isolate potentially suspicious logins from outside the country.

## 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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all employees in the **Marketing department** in the **East building**.

First, I selected all data from the `employees` table. Then, I used a `WHERE` clause with `AND` to

filter for employees who work in both the **Marketing department** and in the **East building**.
I used `LIKE 'East%'` as the pattern to match, because the data in the `office` column represents the East building with specific office numbers (e.g., East-101, East-202).
The first condition is `department = 'Marketing'`, which filters for employees in the Marketing department.
The second condition is `office LIKE 'East%'`, which filters for employees working in offices located in the East building.

This helped me identify exactly which employee computers needed updates in the correct department and location.

## 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   |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all employees in the **Finance** and **Sales** departments.
First, I selected all data from the `employees` table. Then, I used a `WHERE` clause with `OR` to filter for employees who are in either the Finance or Sales department.
I used the `OR` operator instead of `AND` because I want **all employees who are in either department**, not both.
The first condition is `department = 'Finance'`, which filters for employees in the Finance department.
The second condition is `department = 'Sales'`, which filters for employees in the Sales department.

This query helped me retrieve only the relevant employee machine information needed for the Finance and Sales security update rollout.

## 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   |
```

The first part of the screenshot is my query, and the second part is a portion of the output.
This query returns all employees who are **not in the Information Technology department**.
First, I selected all data from the `employees` table. Then, I used a `WHERE` clause with `NOT` to exclude employees from this department.
The condition `department != 'Information Technology'` ensures that only employees from other departments are shown in the result.

This helped me retrieve a filtered list of employee machines where the final security update needs to be applied.

## Summary

I applied filters to SQL queries to get specific information on login attempts and employee machines. I used two different tables, `log_in_attempts` and `employees`. I used the `AND`, `OR`, and `NOT` operators to filter for the specific information needed for each task. I also used `LIKE` and the percentage sign (`%`) wildcard to filter for patterns.