# Apply filters to SQL queries

## Project description

My company is dedicated to enhancing the security of its system. I am responsible for guaranteeing the system's safety, examining any potential security concerns, and maintaining employee computers as required. Below are instances illustrating how I employed SQL with filters to execute security-related activities.

## Retrieve after hours failed login attempts

After regular business hours (after 18:00), a possible security breach incident took place. It's essential to investigate all unsuccessful login attempts during these off-hours.

Here's the SQL query I crafted to filter for failed login attempts that happened after regular business hours:



The query depicted in the screenshot focuses on extracting failed login attempts that happened post 18:00. Initially, I selected all records from the **log\_in\_attempts** table. Then, I utilized a **WHERE** clause along with an **AND** operator to refine the results, ensuring that only unsuccessful login attempts occurring after 18:00 were displayed. The first condition, **login\_time > '18:00',** isolates attempts after 18:00, while the second condition, **success = FALSE**, specifically targets failed login attempts.

## Retrieve login attempts on specific dates

## An incident of concern arose on 2022-05-09. It's imperative to scrutinize all login activities from that day as well as the preceding day. Here's the SQL code I formulated to filter for login attempts on these specific dates:



The query displayed in the screenshot retrieves all login attempts from either 2022-05-09 or 2022-05-08. Initially, I selected all data from the **log\_in\_attempts** table. Subsequently, I employed a **WHERE** clause with an **OR** operator to filter the results, ensuring only login attempts from either 2022-05-09 or 2022-05-08 were included. The first condition, **login\_date = '2022-05-09'**, focuses on logins from 2022-05-09, while the second condition, **login\_date = '2022-05-08'**, targets logins from 2022-05-08.

## Retrieve login attempts outside of Mexico

## Based on my analysis of the organization's login attempt data, I've identified a concern regarding login attempts originating from outside Mexico. These attempts warrant further investigation. Here's the SQL code I developed to filter for login attempts occurring outside Mexico:



The query in the screenshot retrieves all login attempts originating from countries other than Mexico. Initially, I selected all data from the log\_in\_attempts table. Then, I utilized a **WHERE** clause with **NOT** to exclude Mexico from the results. I employed the **LIKE** operator with the pattern **MEX%** to match Mexico, as the dataset represents it as **MEX** or **MEXICO**. The percentage sign **(%)** serves as a wildcard, representing any number of unspecified characters when used with **LIKE**.

## Retrieve employees in Marketing

## To facilitate updating computers for specific Marketing department employees, I've crafted a SQL query to filter for machines belonging to employees in the Marketing department located in the East building. Here's the code:



The query depicted in the screenshot retrieves all employees in the Marketing department situated in the East building. Initially, all data from the employees table is selected. Subsequently, a **WHERE** clause with an **AND** operator is employed to narrow down the results to employees working in both the Marketing department and the East building. The **LIKE** operator with the pattern **East%** is utilized to match the East building, as the data in the office column represents it with specific office numbers. The first condition, **department = 'Marketing'**, focuses on employees in the Marketing department, while the second condition, office LIKE **'East%**', targets employees in the East building.

## 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.



The query presented in the screenshot retrieves all employees in the Finance and Sales departments. Initially, it selects all data from the employees table. Then, a **WHERE** clause with an **OR** operator is applied to filter for employees in either the Finance or Sales departments. The **OR** operator is used because the intention is to include all employees who are in either department. The first condition, **department = 'Finance'**, filters for employees from the Finance department, while the second condition, **department = 'Sales**', filters for employees from the Sales department.

## Retrieve all employees not in IT

## To gather information on employees who are not part of the Information Technology department and facilitate the security update, I've developed a SQL query to filter for their machines. Here's the query:

## Screenshot of the command line with the SQL query for this task and its output

The query displayed in the screenshot retrieves all employees who are not part of the Information Technology department. Initially, it selects all data from the employees table. Then, a **WHERE** clause with **NOT** is utilized to filter for employees not in this specific department.

## Summary

I effectively utilized SQL queries to apply filters for extracting specific information from the log\_in\_attempts and employees tables. Employing operators such as **AND**, **OR**, and **NOT** allowed me to refine my queries to precisely target the required data for each task. Additionally, the use of the **LIKE** operator along with the percentage sign **(%)** wildcard facilitated filtering for patterns within the data.