# Apply filters to SQL queries

## Project description

My role involves overseeing the enhancement of security measures within my organization's system. I am responsible for guaranteeing the system's safety, examining any possible security concerns, and making necessary updates to employee computers. The subsequent actions offer instances of my utilization of SQL alongside filters to carry out tasks related to security.

## Retrieve after hours failed login attempts

An incident with potential security implications took place outside of regular business hours, specifically after 18:00. It is necessary to conduct an investigation into all unsuccessful login attempts that occurred during this after-hours period.

The provided code snippet illustrates how I crafted an SQL query to isolate failed login attempts that happened after business hours. The screenshot displays two sections: the initial part showcases the query I used, while the subsequent segment demonstrates a portion of the resulting output.



The query's objective is to filter out unsuccessful login attempts that transpired after 18:00. To achieve this, I initiated the process by extracting all data from the 'log\_in\_attempts' table. Subsequently, I incorporated a WHERE clause featuring an AND operator to refine the outcome. This refinement ensured that only login attempts meeting two conditions were displayed: those taking place after 18:00 and those that were unsuccessful. The first condition, 'login\_time > '18:00'', serves to filter for attempts occurring post 18:00, while the second condition, 'success = FALSE', targets failed login attempts.

## 

## Retrieve login attempts on specific dates

An incident arousing suspicion took place on the date of May 9, 2022. To thoroughly investigate the matter, it is imperative to scrutinize any login activities occurring on that specific date as well as the preceding day, May 8, 2022.

The provided code excerpt exemplifies how I constructed an SQL query to discern login attempts that unfolded on particular dates. The screenshot encompasses two segments: the initial part showcases the query formulation, while the subsequent portion displays a snippet of the resultant output.



The query's primary aim is to retrieve all login attempts transpiring on either May 9, 2022, or May 8, 2022. The procedure was inaugurated by extracting comprehensive data from the 'log\_in\_attempts' table. Subsequently, I employed a WHERE clause that integrates an OR operator to refine the findings. This refinement process ensured the presentation of solely those login attempts which align with two conditions: those transpiring on May 9, 2022 ('login\_date = '2022-05-09'') and those taking place on May 8, 2022 ('login\_date = '2022-05-08'').

## Retrieve login attempts outside of Mexico

Following an examination of the organization's login attempt data, I have identified an issue concerning login activities originating outside of Mexico. It is imperative to initiate an investigation into these particular login attempts.

The provided code snippet illustrates the methodology I employed to create an SQL query that isolates login attempts occurring beyond the borders of Mexico. The screenshot encompasses two segments: the initial part demonstrates the query formulation, while the subsequent segment presents a portion of the output achieved.



The query's primary objective is to retrieve all login attempts that transpired in countries other than Mexico. The procedure commenced by retrieving comprehensive data from the 'log\_in\_attempts' table. Subsequently, I employed a WHERE clause, utilizing the NOT operator, to refine the results in accordance with the stipulated criterion. To identify countries other than Mexico, I utilized the LIKE operator in conjunction with the pattern 'MEX%', given that the dataset represents Mexico using both 'MEX' and 'MEXICO'. The '%' sign in this context functions as a wildcard, representing any unspecified sequence of characters when utilized with the LIKE operator.

## Retrieve employees in Marketing

My team's objective is to update the computers for specific employees belonging to the Marketing department. This entails gathering information about which employee machines necessitate updates.

The provided code excerpt illustrates my approach to creating an SQL query that identifies employee machines from individuals within the Marketing department situated in the East building. The screenshot is divided into two parts: the initial segment demonstrates the query formulation, while the subsequent portion showcases a sample of the output generated.



The core purpose of the query is to retrieve details about all employees in the Marketing department who are stationed in the East building. The process was initiated by extracting comprehensive data from the 'employees' table. Subsequently, I integrated a WHERE clause utilizing the AND operator to refine the results based on specific criteria. In order to isolate employees from both the Marketing department and the East building, I utilized the LIKE operator with the pattern 'East%' to correspond with the data in the 'office' column, where the East building is represented alongside a unique office number. The first condition, 'department = 'Marketing'', filters for employees in the Marketing department, while the second condition, 'office LIKE 'East%'' narrows down the results to employees within the East building.

## Retrieve employees in Finance or Sales

Moreover, there is a necessity to update the machines utilized by employees within both the Finance and Sales departments. Given the distinct security update required, it is essential to gather information exclusively pertaining to employees belonging to these two specific departments.

The provided code excerpt demonstrates the methodology I employed to formulate an SQL query aimed at identifying employee machines associated with either the Finance or Sales departments. The screenshot encompasses two segments: the initial part showcases the query construction, while the subsequent section offers a glimpse of the resulting output.



The primary objective of the query is to retrieve data concerning all employees affiliated with the Finance or Sales departments. The procedure commenced with a comprehensive data extraction from the 'employees' table. Subsequently, I introduced a WHERE clause featuring the OR operator to tailor the results according to the specified criteria. The usage of the OR operator here is pivotal, as it ensures the inclusion of all employees who belong to either of the two departments. The first condition, 'department = 'Finance'', pinpoints employees in the Finance department, while the second condition, 'department = 'Sales'', refines the results to encompass employees from the Sales department.

## 

## Retrieve all employees not in IT

Furthermore, my team is required to implement an additional security update on employees who are not associated with the Information Technology department. In preparation for this update, the initial step involves gathering information specifically concerning these non-IT employees.

The provided content illustrates the methodology I employed to create an SQL query aimed at filtering employee machines belonging to individuals outside the Information Technology department. The screenshot comprises two sections: the initial segment demonstrates the query formulation, while the subsequent portion offers a snapshot of the obtained output.



The central goal of the query is to extract details about all employees who do not belong to the Information Technology department. The process commenced with a comprehensive data selection from the 'employees' table. Subsequently, I introduced a WHERE clause incorporating the NOT operator to tailor the results to encompass employees who are not affiliated with the specified department.

## Summary

I utilized filtering mechanisms within SQL queries to extract targeted data regarding login attempts and employee machines. Employing distinct tables—namely, 'log\_in\_attempts' and 'employees'—I harnessed operators such as AND, OR, and NOT to refine the results for each particular objective. Additionally, I employed the LIKE operator coupled with the '%' wildcard to effectively filter based on specific patterns within the data.