# Apply filters to SQL queries

## **Project description**

In this project, we will delve into the practical applications of SQL logical operators (AND, OR, NOT) within real-world contexts, particularly emphasizing their relevance to security analysis. Our focus will encompass retrieving crucial data, such as failed login attempts during after-hours, pinpointing login activities on specific dates, identifying login attempts originating from locations outside of Mexico, and isolating employees belonging to the Marketing, Finance, or Sales departments. Additionally, we will explore how to extract information on employees not affiliated with the IT department, showcasing the versatility of these SQL operators in addressing diverse analytical scenarios.

## **Retrieve after hours failed login attempts**

SELECT \*

FROM log\_in\_attempts

WHERE login\_time > '18:00' AND success = FALSE;

The SQL query provided is designed to retrieve after-hours failed login attempts from the "log\_in\_attempts" table. Let's break down how it works in the "Retrieve after hours failed login attempts" section:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, "log\_in\_attempts."
* **FROM log\_in\_attempts:** Here, we specify the table we want to retrieve data from, which is "log\_in\_attempts."
* **WHERE login\_time > '18:00':** This is the filtering condition. It tells the database to only consider rows where the "login\_time" is greater than '18:00' (6:00 PM). In other words, it selects records for login attempts made after 6:00 PM.
* **AND success = FALSE:** This condition further refines the selection. It specifies that the "success" column must have a value of "FALSE." This ensures that only failed login attempts are included in the results.

In summary, this query effectively retrieves records of failed login attempts that occurred after 6:00 PM, making it a valuable tool for security analysis, as it helps identify potentially suspicious or unauthorized login activities during non-standard hours.

## **Retrieve login attempts on specific dates**

SELECT \*

FROM log\_in\_attempts

WHERE login\_date = '2022-05-09' OR login\_date = '2022-05-08';

The SQL query you've provided is designed to retrieve records from the "log\_in\_attempts" table where the "login\_date" matches either '2022-05-09' or '2022-05-08'. Here's a breakdown of how this query works:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, "log\_in\_attempts."
* **FROM log\_in\_attempts:** Here, we specify the table from which we want to retrieve data, which is "log\_in\_attempts."
* **WHERE login\_date = '2022-05-09' OR login\_date = '2022-05-08':** This is the filtering condition. It tells the database to include rows where the "login\_date" is equal to either '2022-05-09' or '2022-05-08.'

The query essentially collects login attempts that occurred on either of these two specific dates. This can be useful for various purposes, including security analysis, auditing, or tracking user activities on those specific days.

## **Retrieve login attempts outside of Mexico**

SELECT \*

FROM log\_in\_attempts

WHERE NOT country LIKE 'MEX%';

The SQL query you've provided is designed to retrieve records from the "log\_in\_attempts" table where the "country" column does not start with 'MEX'. Here's an explanation of how this query works:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, "log\_in\_attempts."
* **FROM log\_in\_attempts:** Here, we specify the table from which we want to retrieve data, which is "log\_in\_attempts."
* **WHERE NOT country LIKE 'MEX%':** This is the filtering condition. It tells the database to include rows where the "country" column does not start with 'MEX'.

In practical terms, this query can be used to identify and retrieve login attempts that originate from countries other than Mexico. It can be valuable for security analysis and monitoring login activities from different geographical regions.

## **Retrieve employees in Marketing**

SELECT \*

FROM employees;

The SQL query you've provided is a basic one that retrieves all records from the "employees" table. Here's a breakdown of what it does:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, "employees."
* **FROM employees:** Here, we specify the table from which we want to retrieve data, which is "employees."

Essentially, this query fetches all the data stored in the "employees" table, returning a result set containing information about all the employees in the database. It's a fundamental query used to view the entire contents of a table, and it can be useful for various purposes, such as reviewing or exporting employee data for analysis or reporting.

SELECT \*

FROM employees

WHERE department = 'Marketing' AND office LIKE 'East%';

The SQL query you've provided is retrieving data from the "employees" table based on certain conditions. Here's a breakdown of what it does:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, "employees."
* **FROM employees:** Here, we specify the table from which we want to retrieve data, which is "employees."
* **WHERE department = 'Marketing' AND office LIKE 'East%':** This is the filtering condition. It specifies that you want to retrieve rows where both of the following conditions are met:
  + The "department" column must have a value of 'Marketing'.
  + The "office" column must start with the letters 'East'.

So, this query will return all rows from the "employees" table where the employees belong to the Marketing department and work in an office whose name starts with 'East'.

## **Retrieve employees in Finance or Sales**

SELECT \*

FROM employees

WHERE department = 'Finance' OR department = 'Sales';

The SQL query you've provided is used to retrieve data from the "employees" table based on certain conditions. Here's what this query does:

* ***SELECT*** *: This part of the query instructs the database to select all columns (denoted by '*') from the specified table, which is "employees."
* **FROM employees:** Here, we specify the table from which we want to retrieve data, which is "employees."
* **WHERE department = 'Finance' OR department = 'Sales':** This is the filtering condition. It specifies that you want to retrieve rows where either of the following conditions is met:
  + The "department" column has a value of 'Finance'.
  + The "department" column has a value of 'Sales'.

So, this query will return all rows from the "employees" table where the employees belong to either the Finance or Sales department.

## **Retrieve all employees not in IT**

SELECT \*

FROM employees

WHERE NOT department = 'Information Technology';

The SQL query you've provided is used to retrieve data from the "employees" table based on a certain condition. Here's what this query does:

* *SELECT : This part of the query instructs the database to select all columns (denoted by '*') from the specified table, which is "employees."
* **FROM employees:** Here, we specify the table from which we want to retrieve data, which is "employees."
* **WHERE NOT department = 'Information Technology':** This is the filtering condition. It specifies that you want to retrieve rows where the "department" column does not have the value 'Information Technology.'

So, this query will return all rows from the "employees" table where the employees do not belong to the Information Technology department.

## **Summary**

In this project, we embark on a journey into the practical applications of SQL logical operators - the versatile tools of AND, OR, and NOT. Our primary focus revolves around the realm of security analysis, but these operators transcend this field and find relevance in various real-world scenarios. We explore a spectrum of queries that unveil actionable insights, starting with the identification of after-hours failed login attempts, a vital security metric. Moving on, we pinpoint login activities on specific dates, enabling meticulous auditing and analysis. Beyond this, we uncover login attempts originating from locations beyond Mexico, contributing to the global perspective of security monitoring. Lastly, we delve into the nuances of employee data, revealing those within the Marketing, Finance, or Sales departments and spotlighting those who stand apart from the Information Technology domain. This journey is an illumination of SQL's prowess in solving diverse analytical challenges.

In the first section, we peer into the intricacies of SQL queries aimed at retrieving after-hours failed login attempts. The query meticulously filters records based on login times after 6:00 PM (18:00 hours) and unsuccessful login attempts. In the second section, we shift our focus to specific date-based queries, enabling the extraction of login attempts on particular dates, facilitating precise event tracking and analysis. The third section delves into geographic insights as the query uncovers login attempts outside Mexico. The final two sections spotlight employee data, differentiating between departments and affiliations, showcasing SQL's adaptability in addressing a wide spectrum of analytical scenarios. This project not only enriches our understanding of SQL but also equips us with the tools to dissect and analyze data through a security lens and beyond.