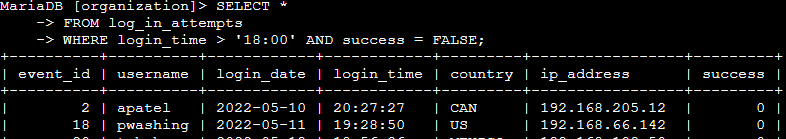
# Apply filters to SQL queries

## Project description

Fictional Scenario: My organization is working to make their systems more secure. I have been tasked to make sure the system is safe, taking time to update the employee’s computers and investigating potential security issues. The following steps provide examples of how I am able to use SQL with filters to perform security-related tasks.

## Retrieve after hours failed login attempts

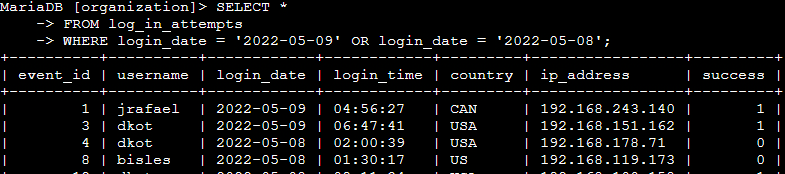
In this first scenario, our team is investigating a potential security incident that happened after business hours. We are looking into all login attempts that were not successful after business hours which is any time after 18:00. For this, we can use a SQL query to filter and search for all failed login attempts that occurred during these hours as seen below:



The first half of the screenshot shows the query that I used to accomplish this task and the bottom half shows a section of the output that it gave. Breaking this query down, we are selecting all columns (indicated by the \*) from the “log\_in\_attempts” table that meet certain criteria. We use the “WHERE” clause to make sure that it only selects rows that meet these criteria. In this situation, we wanted to search for any attempts after business hours so we put “login\_time > ’18:00’” but, in addition to that, we ONLY want unsuccessful attempts so we also included the “success = FALSE” condition. Since, we want the rows to meet BOTH of these conditions, we must the “AND” command to make sure that both conditions are met and not just one or the other.

## Retrieve login attempts on specific dates

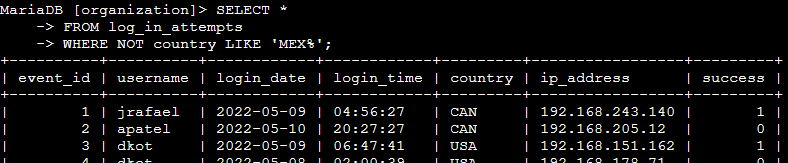
In this next scenario, our organization had suspicious activity occur on 2022-05-09. Our team needs to look into any login activity that occurred on this date as well as the day before so we have as much information on this incident as possible and able to take action. For this scenario, I can use the following SQL query to filter for login attempts that may have occurred on these dates:



The first half of the screenshot shows the query that I used to accomplish this task and the bottom half shows a piece of the output of the query. If we take a closer look at this query, we are selecting all the columns (indicated by the \*) from the “log\_in\_attempts” table just like in the previous task. In this task, we wanted to get rows that match one of the dates we are searching for but not both. Since we only need the results to match one of the conditions, we can use the “OR” command instead of “AND”. This means that it only needs to meet one of the specified queries after the “WHERE” clause which in this task is a login date of either 2022-05-09 or 2022-05-08. These conditions are set by using “login\_date = 2022-05-09” and “login\_date = 2022-05-08”.

## Retrieve login attempts outside of Mexico

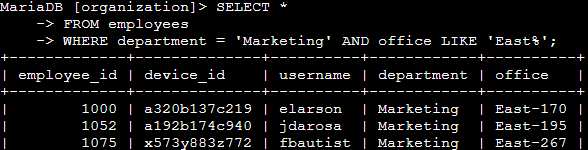
For the next task, we are investigating login attempts that occurred outside of Mexico because we believe there is a security risk occurring outside of these attempts. Just like in the previous tasks, we are able to use a SQL query to filter for any login attempts that occurred outside of Mexico as shown below:



As shown above, we can see the query that was used to accomplish this task in the first half and a piece of the output in the second half. As shown previously, we will be selecting all of the columns from the “log\_in\_attempts” table. We will still be using the “WHERE” command to set conditions to be used to filter for results but this time we will be using “NOT” to pull all rows that do not include certain characters since we want all attempts outside of Mexico. To provide the conditions that we need to filter out, I used “LIKE” and “MEX%” as the means of how we will filter out countries other than Mexico. We use the “%” in this example because the dataset being used represents Mexico as “MEX” and “Mexico” so using “MEX%” with “LIKE” is essentially stating to pull all results that don’t include anything that starts with “MEX” so we can get the most accurate results possible.

## Retrieve employees in Marketing

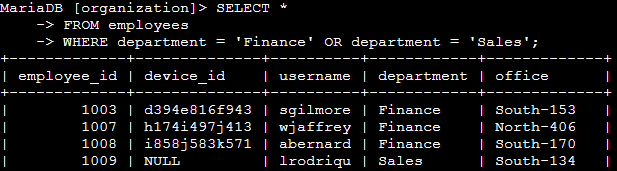
For the next task we were given, our team was assigned the task of updating computers for employees located in the East building that are part of our Marketing department. I need to get information on which computers exactly need to be updated and for this, we can use the following SQL query to filter exactly which employee’s machines we need to work with:



The above query shows how we accomplished the task of locating exactly which machines need to be updated and the bottom half of the image shows a part of that output. Just like in previous examples, we will be selecting all of the columns, but this time, we will be pulling the information from the “employees” table. We will be using the “WHERE” clause again to set the conditions of which rows we need to pull. We set 2 conditions, which both have to be met for the row to be pulled. We used “department = ‘Marketing’” to indicate the employee needs to be from the Marketing team but used “AND” to add a second condition which is “office LIKE ‘East%’”. This will search for any office that starts with “East” because as shown in the output, there are different sections to the East building so using the “%” will use any office that starts with “East” when paired with the “LIKE” command.

## Retrieve employees in Finance or Sales

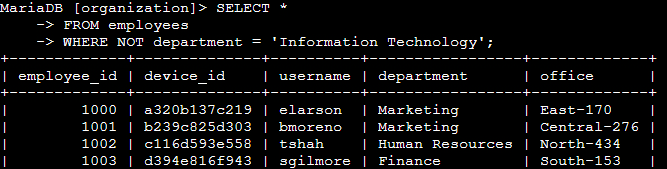
After completing the previous task, we were notified that all the machines in the Finance and Sales department also need to be updated. We needed to quickly gather the information for which machines this involves and can use a SQL query to filter for employee machines from the Finance and Sales departments. To accomplish this, I used the following SQL query to get that information:



The first part of the screenshot shows the query that I used to filter the database to quickly get the information I needed and the second half of the screenshot shows a piece of the output that displays the machines from the Finance and Sales departments. In the first line, we chose to select all columns. In the second line, we selected the “employees” table to pull the data from. We then used the “WHERE” clause to set the conditions for the query. Since we needed all machines from both the Finance and Sales team, I used the “OR” command between the conditions of “department = ‘Finance’” and “department = ‘Sales’”. We use “OR” rather than “AND” because we don’t need it to meet both conditions. As long as they are either from the Sales or Finance team, we need the record and that’s exactly what this query accomplishes and pulled the correct information.

## Retrieve all employees not in IT

For the final task, our team was tasked with updating all machines that are not within the Information Technology department. Just like with the previous tasks, we can use a SQL query to quickly get this information as shown below:



Similar to the previous screenshots, the first half shows the query used to accomplish this task and the second half shows a part of the output. I decided to select all columns from the “employees” table just like the previous examples. I then used the “WHERE” clause paired with “NOT” to pull all results that don’t have the department set as Information Technology shown in the “department = ‘Information Technology’” portion of the query. This pulls all rows that have any department other than Information Technology which is exactly what we needed to get this task completed.

## Summary

This project was to show how we can apply filters to SQL queries to get very specific information or sort through large datasets just like in this project where we used them to get information of login attempts or find machines that need to be updated based on a given task. I utilized 2 tables, one titled “log\_in\_attempts” and the other titled “employees”. We used a variety of operators to accomplish these tasks such as “AND”, “OR”, and “NOT” to filter through these tables. We also utilized other clauses such as “LIKE” and “%” when needed if we didn’t have an exact condition to set.