# Apply filters to SQL queries

## Project description

In this lab I was tasked to retrieve various information from organizational databases using filtered SQL queries. Through these tasks I make use of various operators and conditional statements to retrieve only the desired information as efficiently as possible.

## Retrieve after hours failed login attempts

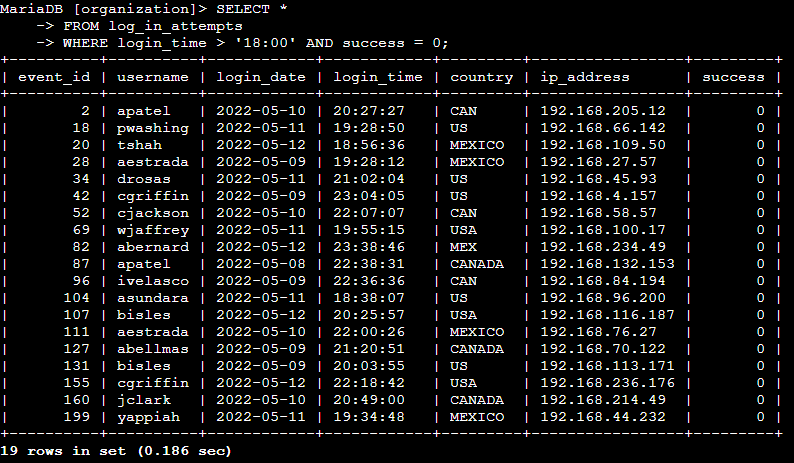
The query I used to retrieve all failed after hours login attempts is as follows:

SELECT \*

-> FROM log\_in\_attempts

-> WHERE login\_time > ‘18:00’ AND success = 0;

Since the organization in this case ends normal work hours at 6:00 PM (18:00) I applied a filter to only select entries where the time of the login attempt was later than 18:00. Since I’m only looking for the failed login attempts I use the AND operator to apply a second filter to sort out the successful login attempts from the results. Since in this case success or failure is a binary result I am able to represent my request for failed login attempts by either typing failure or 0.

The results of the query I ran are displayed in the following screenshot:

## Retrieve login attempts on specific dates

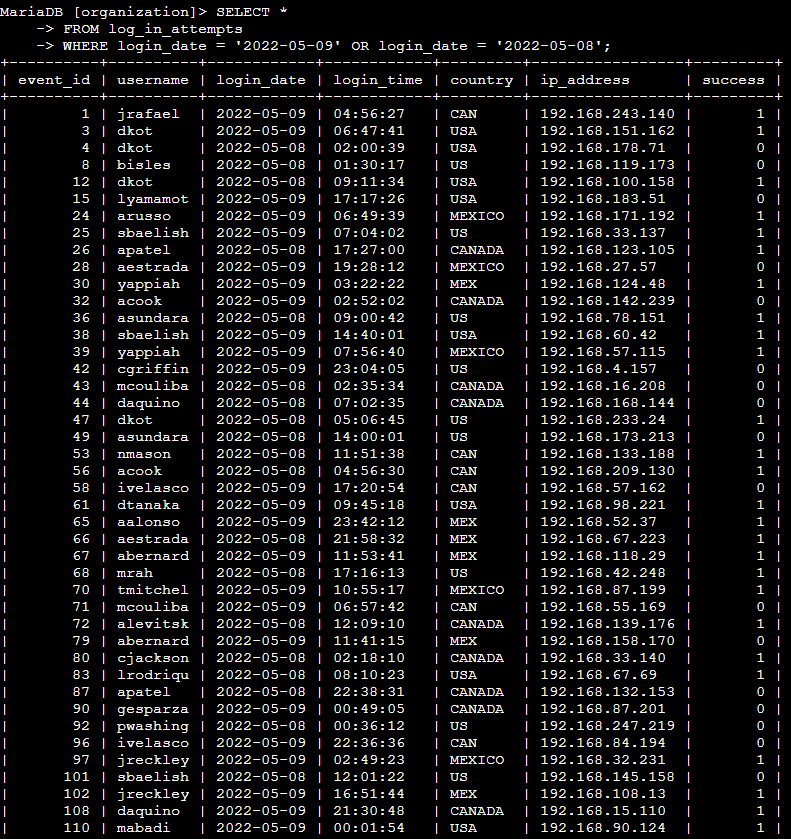
Next, I was tasked with retrieving login attempts on two specific days in the same query. I accomplished this with the following query:

SELECT \*

-> FROM log\_in\_attempts

-> WHERE login\_date = ‘2022-05-09’ OR login\_date = ‘2022-05-08’;

This query selects all results from the log\_in\_attempts table and then displays only results that occurred on the dates ‘2022-05-09’ or ‘2022-05-08’. It was important for me to remember to enclose the dates in single quotation marks to comply with proper SQL syntax. To specify two dates to look for I used the OR operator. It was also important to remember to specify that I was filtering the login\_date column after the OR operator so that SQL would know where to look for the information.

The result of the query are displayed in the following screenshot:

## Retrieve login attempts outside of Mexico

Next, I was tasked with retrieving all login attempts from outside the country of Mexico. This was accomplished with the following query:

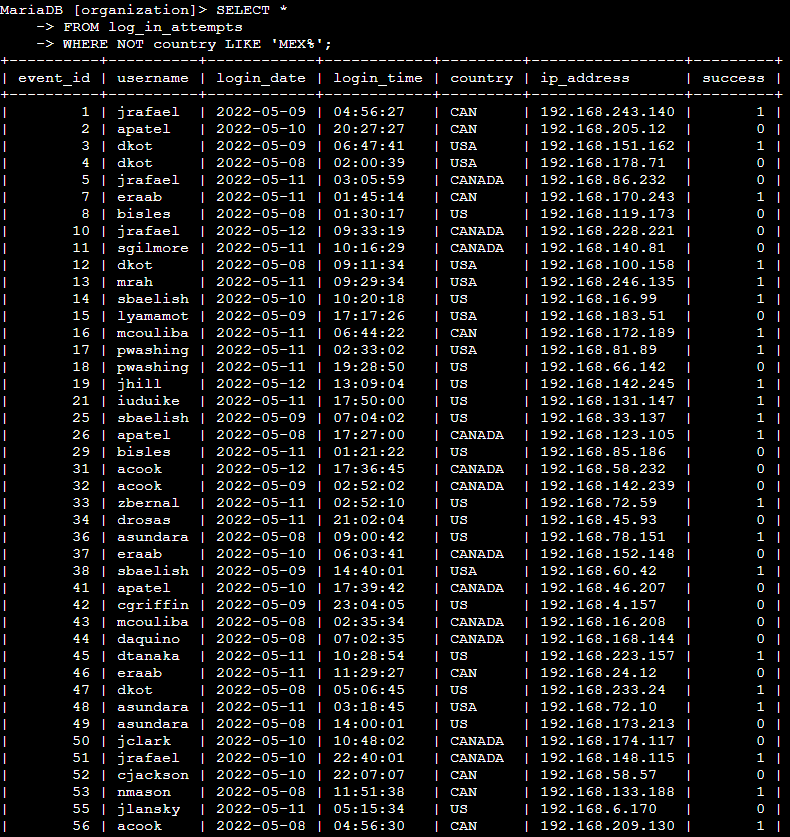
SELECT \*

-> FROM log\_in\_attempts

-> WHERE NOT country LIKE ‘MEX%’;

This query filters the results by the country column and removes any results that begin with the letters MEX. It was important to make use of the LIKE operator as some entries in the table have different values in the country column to represent Mexico, such as MEX or MEXICO. Combining the LIKE operator along with the ‘MEX%’ wildcard tell SQL to ignore any entries that begin with those three letters.

The results of the query are displayed in the following screenshot:



## Retrieve employees in Marketing

Next, I was tasked with retrieving all employees in the marketing department that are located in either the East-170 or East-320 offices. I accomplished this with the following query:

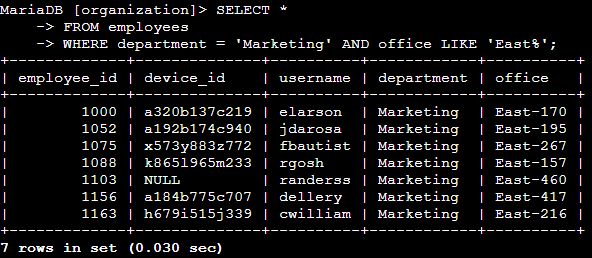
SELECT \*

-> FROM employees

-> WHERE department = ‘Marketing’ AND office LIKE ‘East%’;

This query was a bit more complicated to optimize as it would also be possible to filter the office column by specifying the two offices with the use of an OR operator. However, I found that using LIKE and the wildcard value for any East office to be a more efficient way to phrase this query.

The results of the query are shown in the following screenshot:



## Retrieve employees in Finance or Sales

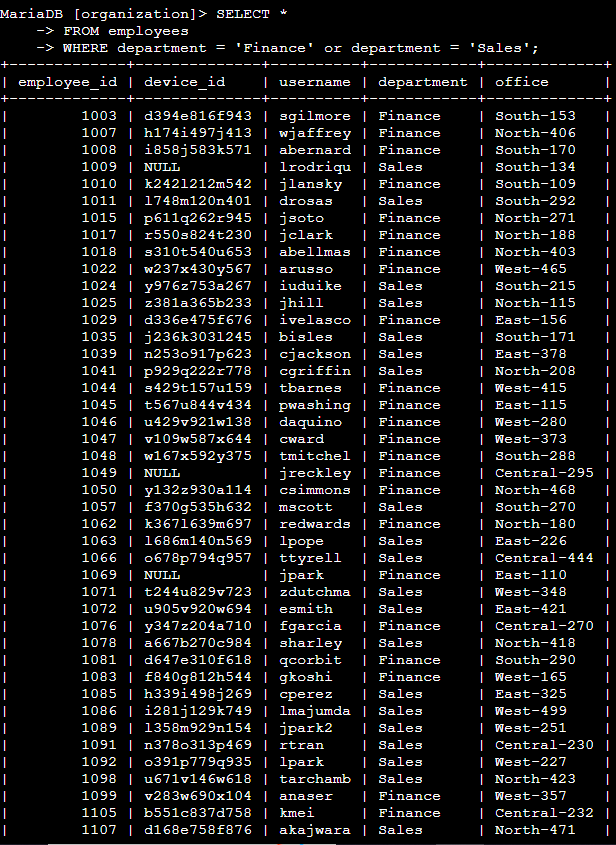
I was next tasked to retrieve all employees in the finance and sales departments. This was accomplished with the following query:

SELECT \*

-> FROM employees

-> WHERE department = ‘Finance’ OR department = ‘Sales’;

This query makes use of the OR operator to select all employees that have ‘Finance’ or ‘Sales’ as the values in the department column. It was important to remember to specify department as the column to filter by in both conditions.

The results of the query are displayed in the following screenshot:

## Retrieve all employees not in IT

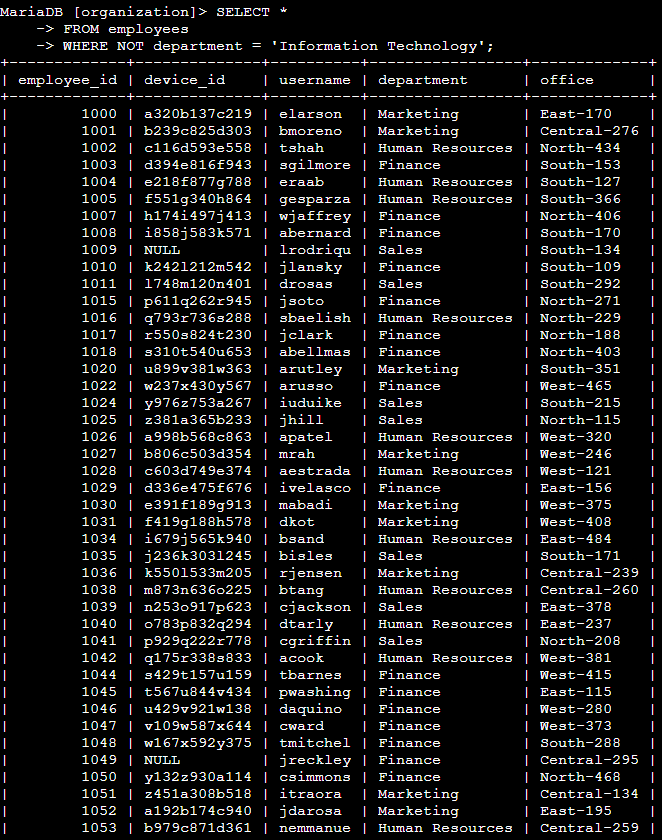
Finally, I was tasked with retrieving all employees outside of the IT department. This was accomplished with the following query:

SELECT \*

-> FROM employees

-> WHERE NOT department = ‘Information Technology’;

This query makes use of the NOT operator to only select employees that have anything other than ‘Information Technology’ as the value in the department column.

The results of the query are displayed in the following screenshot:

## Summary

Through this lab I gained valuable experience filtering databases through conditional statements attached to SQL queries. Proper use of operators such as AND, OR, and NOT allowed for me to filter results that meet multiple conditions or avoid results that meet the specified conditions, allowing for much more efficient retrieval of information from the database.