# Apply filters to SQL queries

## Project description

In this scenario we will be investigating potential security issues and retrieving records from data sets using SQL. This will be accomplished by examining the data in the *employees* and *log\_in\_attempts tables*.

## Retrieve after hours failed login attempts

In this scenario we want to investigate failed login attempts after business hours (18:00). To query a database using SQL we must always use *SELECT* and *FROM*. *SELECT* identifies the column you are searching for separated by a comma or with an ‘ \* ’ to signify that all columns will be searched for. *FROM* designates which table is being searched. All queries are ended with a semicolon. To identify unsuccessful attempts, we’ll have to use Where, indicates a condition that filters the table, to find login attempts after 18:00 and *AND* to identify unsuccessful attempts.

*Note: SQL stores the value ‘0’ for ‘false’ and ‘1’ for ‘true’.*

The query for this task is written as follows:

***SELECT \****

***FROM log\_in\_attempts***

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

This is the result.

*A screen shot of a computer

Description automatically generated with medium confidence* *There were 19 unsuccessful login attempts after business hours.*

## Retrieve login attempts on specific dates

In this one we’ll be looking at a suspicious event that occurred on 05/09/2022. To accomplish this the logs from the day and the day before with the *WHERE* and *OR*, an additional condition.

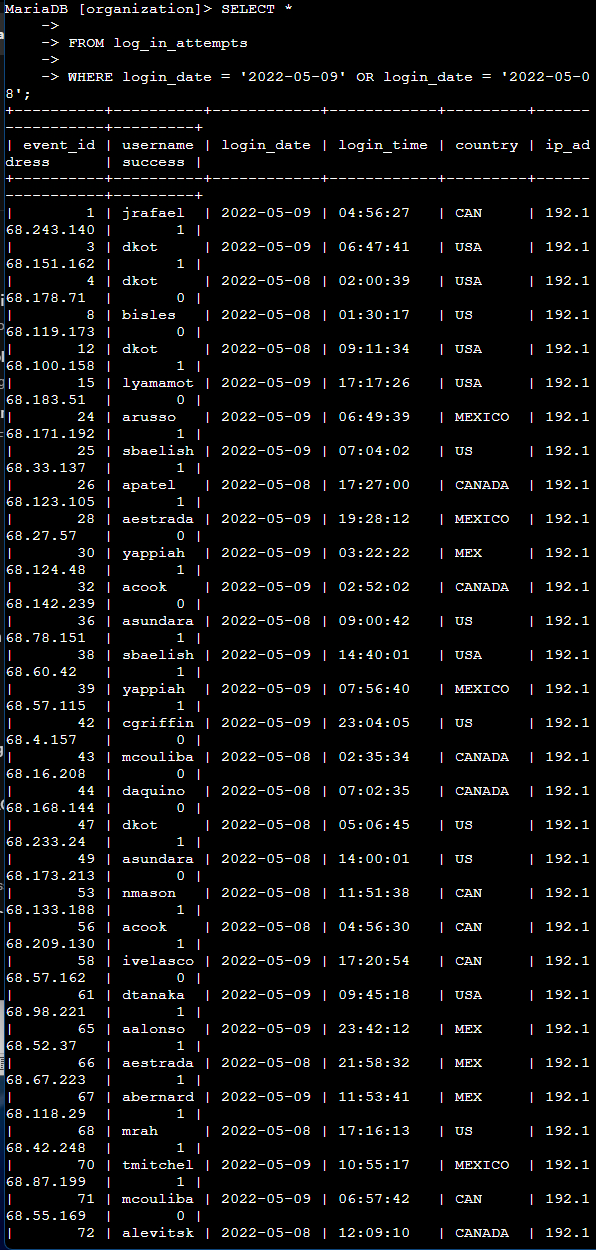
This is the query:

***SELECT \****

***FROM log\_in\_attempts***

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

And result:

A screenshot of a computer screen

Description automatically generated with medium confidence *There were 75 login attempts on these days.*

## Retrieve login attempts outside of Mexico

Now we’ll use the *LIKE*, replaces ‘=’ when used with ‘%’, operator as well as a wildcard or ‘%’ to return all results that start with ‘MEX”. This well help us retrieve all logins outside of Mexico.

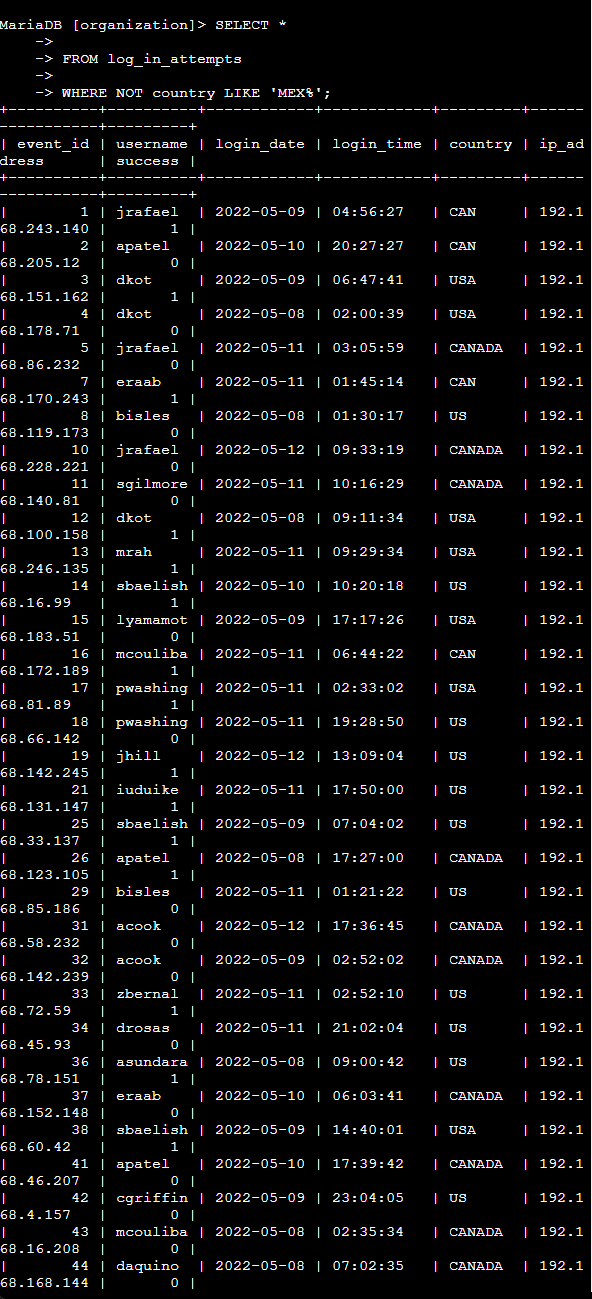
The query:

***SELECT \****

***FROM log\_in\_attempts***

***WHERE NOT country LIKE 'MEX%';***

And results:

A screenshot of a computer

Description automatically generated with medium confidence *There were 144 login attempts outside of Mexico.*

## Retrieve employees in Marketing

Now we’ll use SQL to pull information about the people in the marketing department. Let’s focus specifically on employees in the east buildings.

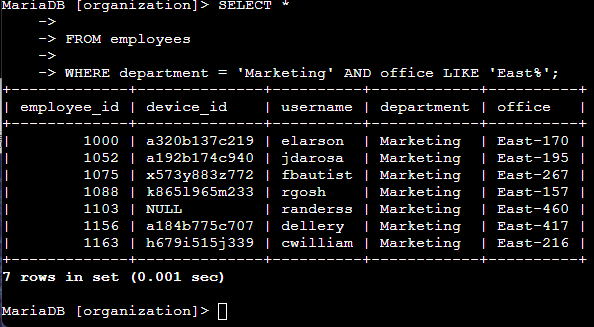
The query is as follows:

***SELECT \****

***FROM employees***

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

And results:



## Retrieve employees in Finance or Sales

Now, were going to look up employees in both the Marketing and Sales departments. To do this we’ll use *OR* to satisfy both conditions.

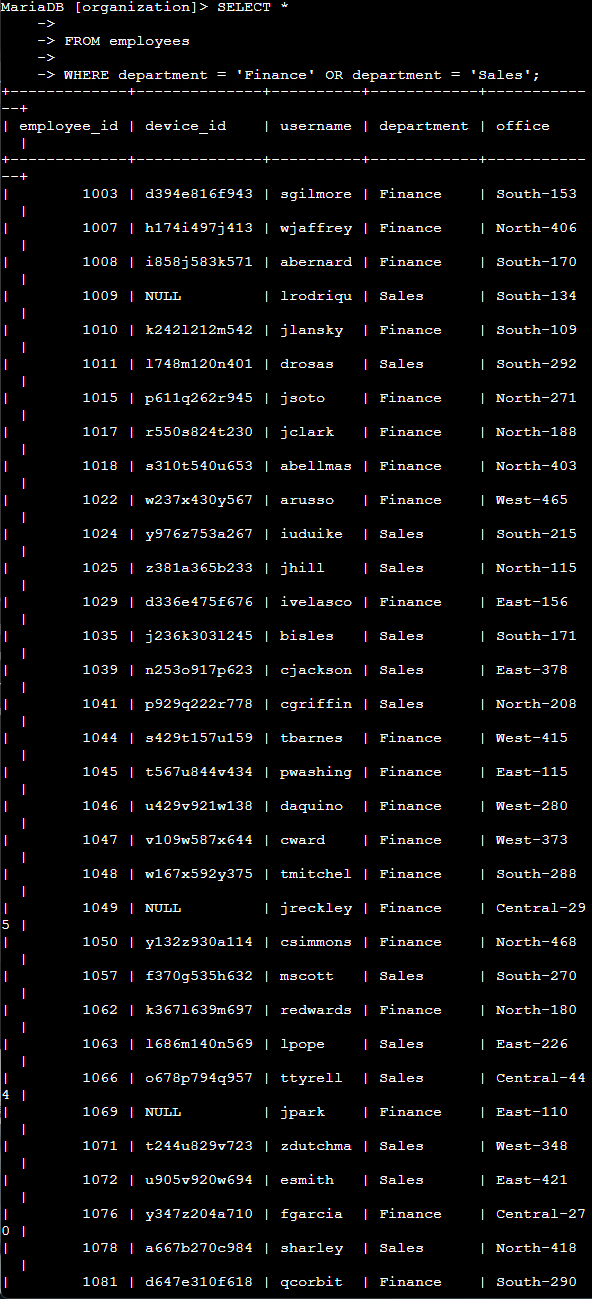
The query:

***SELECT \****

***FROM employees***

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

And results:

A screenshot of a computer program

Description automatically generated with medium confidence

## Retrieve all employees not in IT

In this last section we’ll retrieve info about employees not in the IT department using *NOT* or ‘!=’ which are considered the same in a query along with ‘<>’.

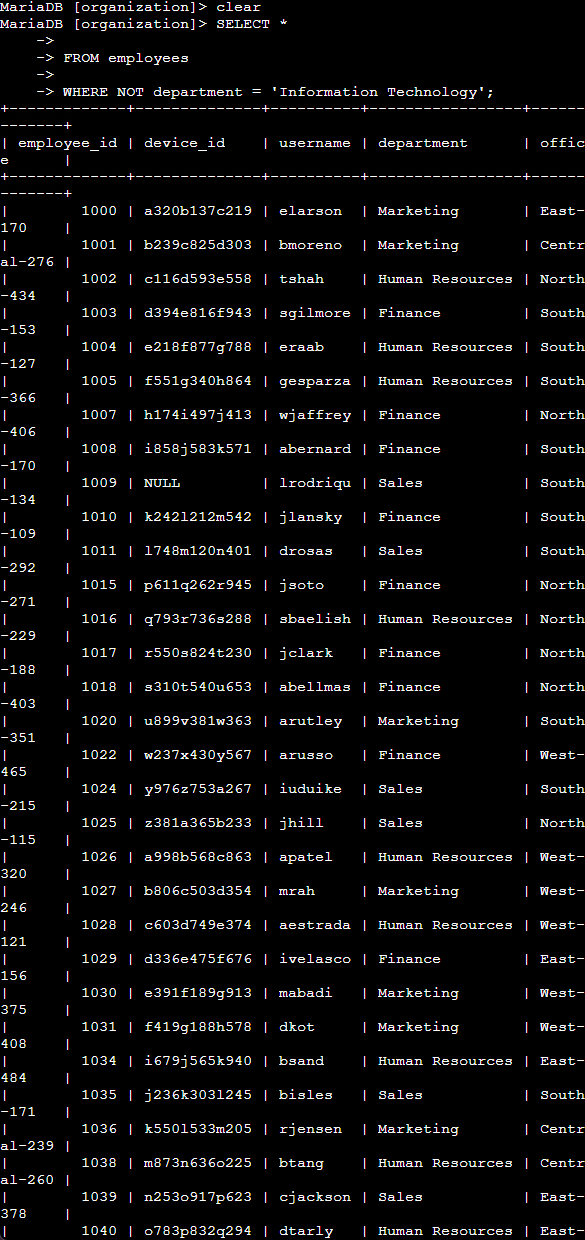
The final query:

***SELECT \****

***FROM employees***

***WHERE NOT department = 'Information Technology';***

And its result:



A screen shot of a computer screen

Description automatically generated with low confidence

*There are 161 employees not in the IT department.*

## Summary

In this project we used SQL queries to search databases. We even used operators *AND*, *OR*, and *NOT* to filter SQL queries. This helped us narrow down and solve problems and find information in an efficient and useful manner.