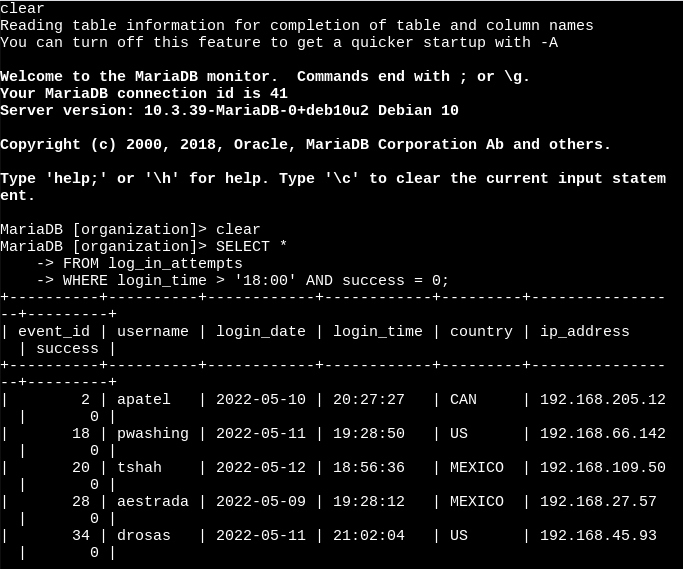
# **Apply filters to SQL queries**

## **Project description**

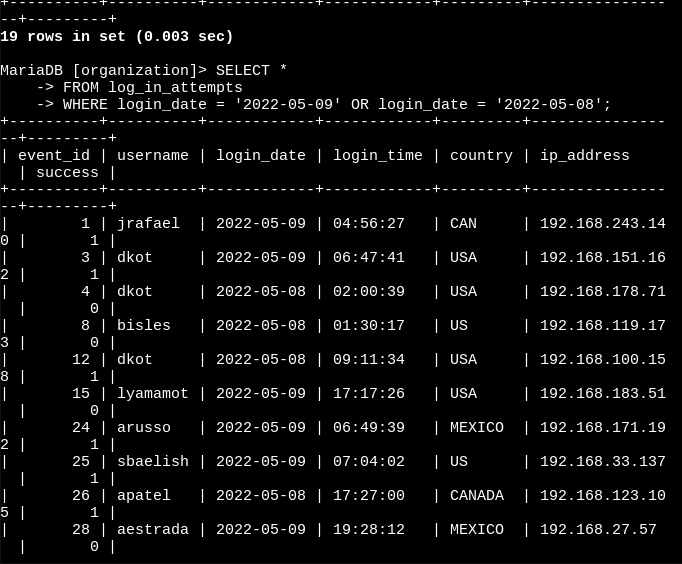
In this project, I demonstrate how to utilize SQL commands, keywords and clauses in order to query as well as filter various types of data of interest from the tabular structure of a relational database via the Linux-driven Bash CLI. Specifically, I apply a variety of comparison or logical operators, filters and conditions, to deliberate query statements in order to sift through a deluge of employee as well as login attempt data, that is stored within a SQL database.

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



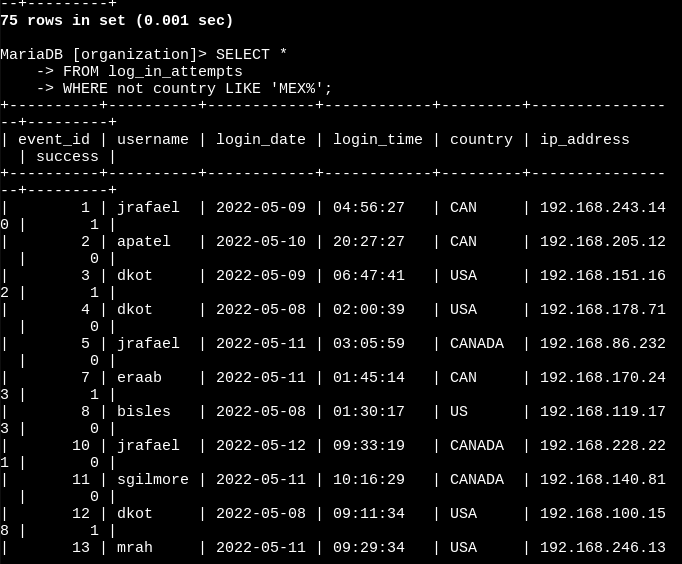
Firstly, I selected all organizational data from the log\_in\_attempts table, where the value within the login time column was greater than 6:00 p.m and was accompanied with a failed attempt within the login attempt column. Explicitly, I used the SELECT keyword followed by the asterisk(\*) as well as the FROM clause, to SELECT all data entries from the log\_in\_attempts table. After that I applied the WHERE login\_time > ‘18:00:00’; statement in order to filter as well as query the failed login attempt data relative to that specified timeframe.

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



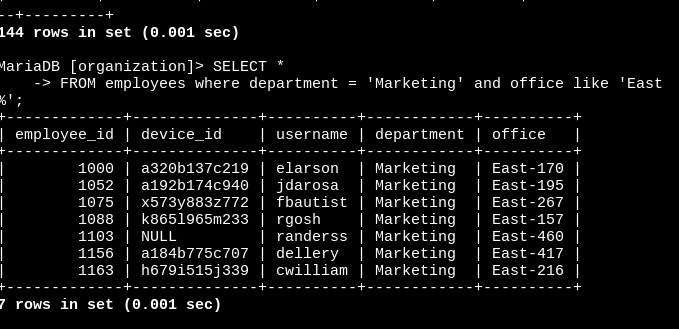
In the next query statement, I selected all organizational data from the log\_in\_attempts table, where the value within the login date column is equal to May 5, 2009 or May 5, 2008. Precisely, I used the SELECT keyword followed by the asterisk(\*) as well as the FROM clause, to SELECT all data entries from the log\_in\_attempts table. After that I applied the WHERE login\_date = ‘2022-05–09’ OR login\_date = ‘2022-05-08’; filter, to establish the necessary data conditions in order to retrieve login attempt data relative to these two desired dates.

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



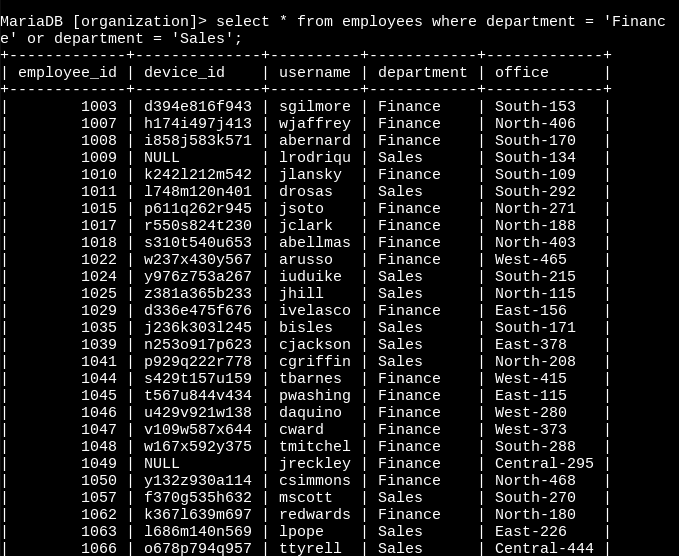
Subsequently, I then selected all organizational data from the log\_in\_attempts table, where the string entry within the country column was not like MEX followed by additional string, which tells the DB to return all failed login attempt entries, that have country entries other than those containing the string ‘MEX’ and ‘MEXICO’. Descriptively, I used the SELECT keyword as well as the asterisk(\*) followed by the FROM clause, to retrieve all data entries from the log\_in\_attempts table within the relational DB. Following the SELECT and FROM statement, I added a WHERE NOT condition such as the following: WHERE NOT country LIKE ‘MEX%’; operation and filter to return all failed login attempt entries, that have country entries other than those containing the string ‘MEX’ and ‘MEXICO’.

## **Retrieve employees in Marketing**



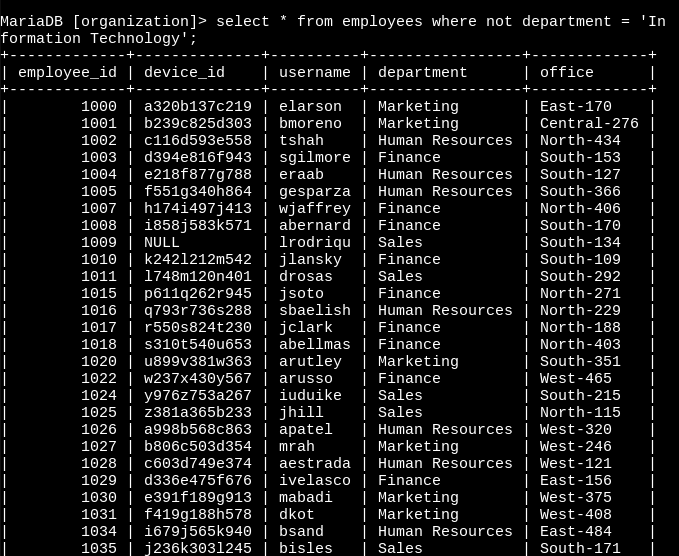
In this screenshot, I selected all organizational data from the employees table, where the string within the department column equaled ‘Marketing’ and office column began with the word East. Definitively, I carried out this query by utilizing the SELECT keyword followed by the asterisk(\*) symbol as well as the FROM clause, to SELECT all data entries from the employees table. Following that query statement, I applied the WHERE department = ‘Marketing’ and office LIKE ‘EAST%’ filter statement in order to establish the necessary conditions or operators needed to query the desired employee data relative to the marketing department and east office wings.

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



Following that step, I selected all organizational data from the employees table, where the string within the department column equaled ‘Marketing’ or department column = ‘Sales’;. Explicitly, I performed this query by inputting the SELECT keyword followed by the asterisk(\*) symbol as well as the FROM clause, to SELECT all data entries from the employees table. Following that query statement, I attached the WHERE department = ‘Finance’ and department = ‘Sales’; filter in order to query the desired employee data relative to both the Finance and Sales department.

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



Lastly, I selected all organizational data from the employees table, where the value within the department column was not equal to the string ‘Information Technology’; Basically, I reiterated the SELECT keyword followed by the asterisk(\*) symbol as well as the FROM clause in order to SELECT all data entries from the employees table. Subsequently, I applied the WHERE NOT department condition, so that I can filter entries that were equal to the string ‘Information Technology’;. Which allowed me to establish the necessary conditions and logical operators needed to query data with department entries containing the desired string.

## **Summary**

Throughout this SQL project, I demonstrate how to effectively iterate SQL commands, keywords and clauses in conjunction with various comparison or logical operators, filters and conditions, to deliberately extract key data from an ocean of employees as well as login attempt data being stored within a SQL database. Technically, I chose to use the SELECT clause to identify the data or relative columns that I wished to query and I used the FROM CLAUSE in order to specify where or from which table I sought for the SQL processor to extract this data from. Additionally, I applied significant keywords, conditions, or filters to these clauses such as OR, NOT, OR NOT, AND, BETWEEN as well as mathematical operators in order to specify the data parameters of each individual query.