Renaldo Schmidt

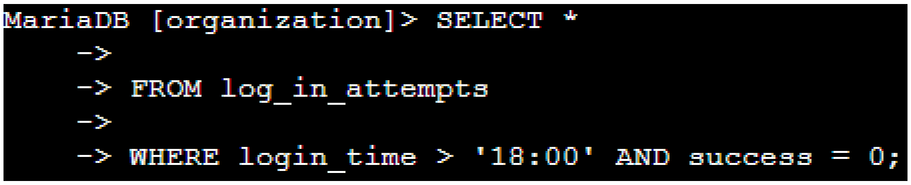
Apply Filters to SQL Queries

Project description

In this project, I aim to show employers that I am proficient in using SQL to filter databases for information that is necessary for the analysis of security-related data. Throughout this project, I acted as a security professional where part of my role was to investigate security issues.

Retrieve after-hours failed login attempts

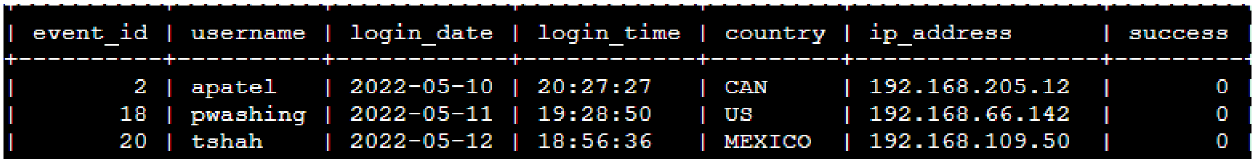
It was reported that after business hours there was a security incident, so I used SQL to see failed login attempts after business hours.



```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE login_time > '18:00' AND success = 0;
```

In order to use SQL, you have to use the clause "SELECT" to determine which columns of data you desire to return, and "FROM" to specify which table to query. You can also use "WHERE" to filter tables for specific data.
In this instance, I used SELECT followed by a "*". The asterisk returns all columns of data from the table.
I also used FROM "log_in_attempts" to specify that I want data from the login attempts table.

To filter for attempts after business hours I used the clause "WHERE" followed by the column "login_time" and the operator ">" (greater than) and lastly '18:00'. The greater than operator is used for numerical data such as login time, and here it was used to give me every login attempt that was made after 18:00 (6 pm or business closing time). This was combined with the keyword AND, which is used to filter for two specific conditions at the same time. In this case, AND was followed by "success = 0", which means that it will only return data from the login attempts table where the data in the success column was 0 (0 means failed and 1 means success). This gave me a filtered list of all of the failed login attempts after 6 pm.

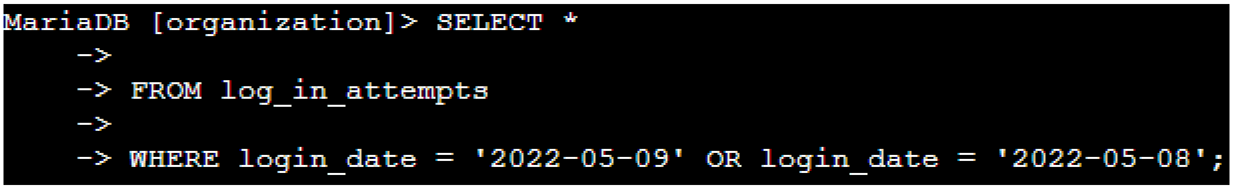


```
+----------+----------+------------+------------+---------+-----------------+---------
| event_id | username | login_date | login_time | country | ip_address      | success
+----------+----------+------------+------------+---------+-----------------+---------
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0
|       20 | tshah    | 2022-05-12 | 18:56:36   | MEXICO  | 192.168.109.50  |       0
```

This is a screenshot of the first three rows of data returned by this SQL query.

Retrieve login attempts on specific dates

A suspicious event occurred on May 9th, so I need to review all login attempts on that day and the day prior to help me analyze the situation.



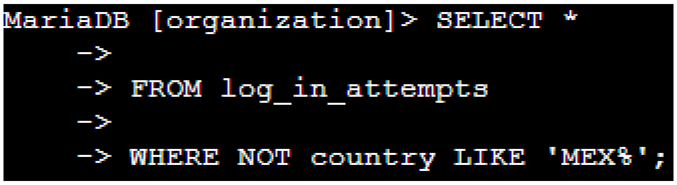
```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
```

I used the “SELECT *” again here to return all of the columns of data.
I used “FROM log_in_attempts” again to return data from the login attempts table.

To filter for login attempts on the two specific days I used the keyword WHERE followed by “login_date = ‘2022-05-09’ OR login_date = ‘2022-05-08’. The first part “login_date” specifies the column I wish to filter, the second part “=” is the operator that determines that the data in the column must be exactly what I specified after it “2022-05-09”. This was combined with the keyword “OR” which will filter for data that satisfies either condition. Lastly, the same format was used after the OR but the only thing that is changed is the date. This effectively filters the table and only returns login attempts on those two days.

Retrieve login attempts outside of Mexico

It was determined by members of the team that the suspicious activity did not originate from Mexico, so I will query login attempts excluding those from Mexico.



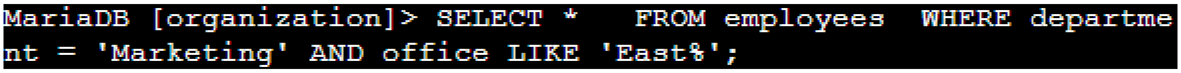
```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE NOT country LIKE 'MEX%';
```

The SELECT and FROM lines are the same as the previous two examples, as I am gathering all of the columns of data from the same login attempts table.

To exclude Mexico logins from the table I used the NOT operator after the WHERE keyword and before the column “country”. Following this was the operator LIKE followed by “MEX%”.
The NOT operator negates the condition that I specified following it.
The LIKE operator filters for a pattern defined by ‘MEX%’
The percent symbol after MEX is a wildcard that will include all data that starts with the characters MEX. This is important because sometimes Mexico is represented by “MEX” or “MEXICO” in the data.

Retrieve employees in Marketing

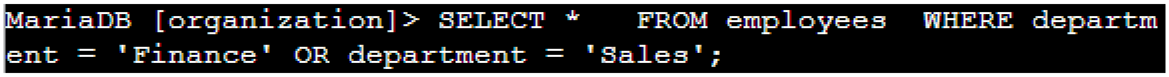
The team wants to perform security updates on marketing employees' computers in the east building.



```
MariaDB [organization]> SELECT *   FROM employees  WHERE departme
nt = 'Marketing' AND office LIKE 'East%';
```

“SELECT *” returns all columns
“FROM employees” returns data from the employee's table
“WHERE department = ‘Marketing” filters the data only to show data where the department column is “Marketing”
“Office LIKE ‘East%” filters the data to only return data where the office column that starts with the four characters “East”.
“AND” in between these two conditions filters for data that satisfy both conditions.

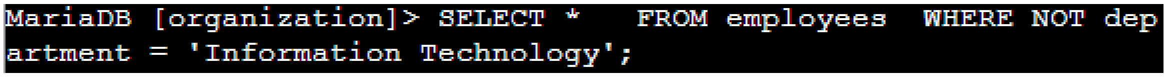
Retrieve employees in Finance or Sales



```
MariaDB [organization]> SELECT *   FROM employees  WHERE departm
ent = 'Finance' OR department = 'Sales';
```

SELECT and FROM lines are the same as in the previous example
In this case, we are looking for employees in the finance or sales department, so we use the OR operator to include both conditions in the return.

Retrieve all employees not in IT



```
MariaDB [organization]> SELECT *   FROM employees  WHERE NOT dep
artment = 'Information Technology';
```

SELECT and FROM lines are the same as in the previous example
For the WHERE filter, we use NOT to exclude data where the department is IT.

Summary

In this project, I used SQL to filter databases for information like login attempts and employees in certain departments. This was done through the use of the WHERE clause, and operators like =, NOT, %, LIKE, AND, and OR. If you are an employer who read through this project, thank you for giving me the opportunity to show my SQL skills!