Apply filters to SQL queries

Project description

In this project, the scenario was that I was a security professional at a large organization whose job was to investigate security issues to help keep the system secure. I had recently discovered some potential security issues that involved login attempts and employee machines. The task was to examine the organization's data in their `employees` and `log_in_attempts` tables using SQL filters to retrieve records from different datasets and investigate the issues.

Retrieve after-hours failed login attempts

In this step, I was investigating failed login attempts that were made after business hours, which is after `18:00`. I start by selecting which columns of data I want to see; in this case, I'll `SELECT *`, or all, `FROM` the `log_in_attempts` table. Then I set the filter criteria, starting with the `login_time` column being after, or greater than, `18:00`. Since I'm only looking for failed attempts, and SQL stores the boolean values `TRUE` and `FALSE` as `1` and `0`, respectively, I set the additional criteria to be `success = 0`, and combine the two conditions using `AND`, where `success` is another column in the `log_in_attempts` table. It is acceptable to use `FALSE` instead of `0` as well, and it will return the same results.

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

Retrieve login attempts on specific dates

The next suspicious event I investigated occurred on `'2022-05-09'`, and the team wanted to retrieve all login attempts that occurred on that day and the day before, `'2022-05-08'`. Therefore, I select all from the `log_in_attempts` table again, but this time, I set the conditions to where the `login_date` column is either `'2022-05-08' OR '2022-05-09'`. This returns any

records where one or both conditions are met.

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

Retrieve login attempts outside of Mexico

The security team has identified more suspicious activity with login attempts, but they determined it did not originate in Mexico. As such, I needed to collect data for all login attempts outside of Mexico. The table has entries for Mexico that are both `'MEXICO'` and `'MEX'`, so I needed to check for a pattern when setting the criteria for `NOT`. So in the `WHERE` clause, I used `NOT country LIKE 'MEX%'` to specify the `country` column to be anything that was not a pattern that started with `'MEX'`.

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       0 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
```

Retrieve employees in Marketing

Next, the team wants to perform security updates on specific machines in the Marketing department for employees who are located in all offices in the East building. This time, I did `SELECT * FROM employees`, as we are now looking for employee and machine info instead of login attempts. Then I set the criteria for the columns to be `department = 'Marketing' AND office LIKE 'East%'` since there are multiple offices in the East building, and the column has entries of the form `'East-170'`. So, in order to find all the offices, I used the `LIKE` and pattern

matching with the wildcard character, `%`.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa  | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing  | East-267 |
```

Retrieve employees in Finance or Sales

Now, the team needs to perform a different update to the computers of all the Finance and Sales employees, so we needed to locate their info. This query remained similar to the last, except the `WHERE` conditions were `department = 'Finance' OR department = 'Sales'`. This returned both departments' employees for our use.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+-------------+----------+------------+-------------+
| employee_id | device_id   | username | department | office      |
+-------------+-------------+----------+------------+-------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153   |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406   |
|        1008 | i858j583k571 | abernard | Finance    | South-170   |
```

Retrieve all employees not in IT

The last step in the investigation and update process required one more update to all employee machines. However, employees in the IT department have already applied this update, so we only need to update employee machines who do not work in IT. Therefore, we queried the database for all employee records `WHERE NOT department = 'Information`

`Technology'`.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE NOT department = 'Information Technology';
+-------------+--------------+----------+-----------------+-------------+
| employee_id | device_id    | username | department      | office      |
+-------------+--------------+----------+-----------------+-------------+
|        1000 | a320b137c219 | elarson  | Marketing       | East-170    |
|        1001 | b239c825d303 | bmoreno  | Marketing       | Central-276 |
|        1002 | c116d593e558 | tshah    | Human Resources | North-434   |
```

Summary

Overall, in this project, we looked at querying and filtering results from a database to investigate potential security issues using SQL. We aggregated different login attempts associated with suspicious activity to evaluate potential threats and any compromised assets better. Upon completion of this, the security team decided to roll out new or missing updates to employee machines, which required finding where in the organization these employees worked, their machine numbers, and any machines that had yet to receive the new updates.