Apply filters to SQL queries

Project description

In this scenario, I assist with investigating security issues for a large organization. I discovered potential issues involving employee login attempts and office machines. I investigated data within the `employees` and `log_in_attempts` tables. Using SQL filters, I retrieved records from different data sets ranging from after hour failed login attempts, to login attempts in other countries, and identified employee machines in specific offices/departments that required security updates.

Retrieve after hours failed login attempts

A potential security incident occurred after normal business hours. To investigate, I used a SQL query to filter all failed login attempts that occurred after 18:00:00 (6pm).

The following shows the SQL query I created:

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

The (`*`) dictates I want to select all available data columns. `FROM log_in_attempts` identifies what table I want to search from. For my `WHERE` command, `>` indicates that I want time values greater than 18:00:00 (i.e after hours) Because I also want to simultaneously filter only failed logins, I add the `AND` filter to the success column for data that equals 0 (0 representing FALSE in the success column, generating only failed login attempts).

Retrieve login attempts on specific dates

After a suspicious event occurred on 2022-05-09, I needed to write a query to pull up all login attempts that occurred both on the day and the day before.

The following shows the SQL query I created:

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

For our `WHERE` filter, we use an `OR` operator to search login attempts that occurred on either date. An important note here is that, even though both dates are being found from the `login_date` column, SQL format requires that I reiterate the same column again for the second date after `OR`.

Retrieve login attempts outside of Mexico

The security team concludes that the incident did not originate in the country of Mexico, so I need to use a SQL query that will negate Mexico from the search and only pull logins that originated elsewhere.

The following shows the SQL query I created:

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

The `NOT` operator excludes the specified value(s) from the search. In the `country` column, Mexico is represented by multiple values (`'MEX'` and `'MEXICO'`). I use a `LIKE` operator so that my `WHERE` command searches for patterns of characters in the column; then, I use the keyword `'MEX%'` because `%` finds any number of characters following MEX (which will generate both Mexico values).

Retrieve employees in Marketing

The next step in the scenario requires me to now search the `employees` table for machines of employees both in the Marketing department and in the East Building office that need security updates.

The following shows the SQL query I created:

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

Under the `office` column, values for employee machines are identified by the building they are located in, followed by a hyphen '-' and number (example: `North-160`). Using a `LIKE` operator to identify patterns, I filter for `'East%'` in order to retrieve all the east building machines (`%` will find any following characters, numerical or hyphens). Because we simultaneously need employees in the Marketing department, I use `AND` before my

`department = 'Marketing'` condition. `=` indicates a search for the condition keyword under the department column.

Retrieve employees in Finance or Sales

Now I have to do the same thing for employees in either the Finance or Sales department so that the security team can perform a different update on their machines.

The following shows the SQL query I created:

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

Since we need to update employee machines in both departments, I use `OR` to filter either Finance or Sales under the `department` column, which SQL requires me to reiterate again after `OR` despite the column remaining the same.

Retrieve all employees not in IT

The team needs to perform one more update for all employees; however, employees in the Information Technology department already have the update and need to be excluded from my search.

The following shows the SQL query I created:

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

This one is straightforward. Select all data columns with (`*`), search from the `employees` table, and a `NOT` operator to my `WHERE` command in order to negate the Information Technology department from my search.

Summary

My SQL queries for the `log_in_attempts` table helped narrow down my search of potential users that could've been involved with the security incident, along with determining the exact time and location the incident occurred from. My SQL queries pertaining to the `employees` table helped organize all employees within the organization and determine which departments/offices needed which security updates.