Apply filters to SQL queries

Project description

In my role as a security professional at a large organization, my responsibilities include investigating potential security issues and ensuring the safety of our system. To achieve this goal, I have been instructed to investigate potential security issues and take necessary actions to ensure the system's safety. Recently we have experienced indications of compromise (IoC) relating to login attempts and employee machines. In the following paragraphs, I will provide examples of how I examined the organization's data stored in the `employees` and `log_in_attempts` tables using SQL filters. The investigations performed are listed below:

- Investigate failed login attempts that were made after business hours, after 6p.m. (18:00).
- Investigate a suspicious event that occurred on 2022-05-09.
- Investigate logins that did not originate in Mexico.
- Obtain information about employees in the Marketing department who are located in all offices in the East building to assist the team in updating employee machines.
- Obtain information about employees in the Finance and Sales departments.
- Obtain information about employees in the Information Technology department.

Retrieve after hours failed login attempts

After normal business hours (office hours end at 18:00), there was a security incident that occurred. For this reason, all failed after hours login attempts must be reviewed to ensure no malicious activity is occurring. The `log_in_attempts` table contains information on when login attempts were made, the `login_time` column. This table also contains the `success` column, which holds values of `TRUE` or `FALSE` to indicate whether the login was successful. It is worth noting that MySQL stores Boolean values as `1` for `TRUE`, and `0` for `FALSE`.

Given the information I have, I began to query the `log_in_attempts` database by selecting all rows of data (`*`) from the `log_in_attempts` database and modifying that result with the `WHERE` clause to only return the `login_time` column, supplemented by the `AND` operator to include all failed login attempts (`FALSE` or `0`) after 18:00. The `WHERE` clause is the filter that will parse through the selected data. The `AND` operator includes data that meets the specified conditions. I specified this parameter with a condition, `login_time > '18:00'`. The last condition I used was used to filter for the failed login attempts, `success = 0`. Since I like numbers and they are less characters than `TRUE` or `FALSE`, I chose to use `0` in my filter. Please refer to the following code and screenshot for reference:

```
SELECT *
FROM log_in_attempts
WHERE login_time > '18:00' AND success = 0;
```

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

During the investigation a suspicious login attempt was noticed on 2022-05-09. I began to investigate the day as well as one day prior. As stated earlier, the `login_date` column in the `log_in_attempts` table contains information on the dates when login attempts were made.

I began with selecting all data from the `log_in_attempts` table. Next, I used the `WHERE` clause and `OR` operator to filter for login attempts on 2022-05-09 or 2022-05-08. Filtering for this requires two conditions, `login_date = '2022-05-09'` and `login_date = '2022-05-08'` with the `OR` operator in between the two conditions. For refence, please refer to the code and screenshot below:

```
SELECT *
FROM log_in_attempts
WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
```

```
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 |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       24 | arusso   | 2022-05-09 | 06:49:39   | MEXICO  | 192.168.171.192 |       1 |
```

Retrieve login attempts outside of Mexico

After initial investigation, I determined that the login attempts occurred outside of Mexico. To assist my team in investigating the attempts, I ran another query.

I started the query by selecting all login attempts from the `log_in_attempts table`. The `log_in_attempts` table includes a country field and includes entries with '`MEX`' and '`MEXICO`'. Using the `WHERE` clause and `NOT` operator, I filtered for login attempts that occurred outside of Mexico. To avoid missing data, I used the `NOT` and `LIKE` operators and matched the query to the pattern '`MEX%`'. The `NOT` operator selects all data except the specified value. The `LIKE` operator selects data that matches a specified pattern. Taking advantage of the percentage sign (`%`) we can look for all countries that start with the first three letters of Mexico and return anything that matches. Please refer to the code and screenshot below for reference:

```
SELECT *
FROM log_in_attempts
WHERE NOT country LIKE '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 |       1 |
|        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 |       1 |
```

Retrieve employees in Marketing

I then assisted my team in updating computers for specific employees in the Marketing department to address the incident. The request was to select only employees who are in the Marketing department and in the East buildings.

I began by selecting all data from the `employees` table. This gave me a list of all the data in the table to help me to determine how I will enter the next query. Please refer to the code and screenshot for reference:

```
SELECT *
FROM employees;
```

```
MariaDB [organization]> SELECT *
    -> FROM employees;
+-------------+--------------+----------+-----------------------+-------------+
| 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   |
```

I noted that the office syntax was as such: '`East-170`','`East-320`', etc.). Now that I understand the columns better, I began to query once more. I started by selecting all data from the `employees` table. Next, I used the `WHERE` clause with `AND` operator to filter for the employees who work in Marketing and in the East buildings. The first condition that I used was `department = 'Marketing'` and the second was `office LIKE 'East%'`. Combining these conditions with the `AND` operator returned Marketing employees from all East buildings. Please refer to the code and screenshot below for reference:

```
SELECT *
FROM employees
WHERE department = 'Marketing' AND office LIKE 'East%';
```

```
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

After helping to identify the Marketing department machines requiring an update, I then assisted my team in applying a different update to the computers of all employees in the Finance or the Sales department. Once again, I used data to obtain the data for this request.

I began this by first selecting all data from the `employees` table. Then, I used the `WHERE` clause and the `OR` operator to filter for the two departments. I did not use the `AND` operator because it would only include employees who work in both departments and not employees from either department. The first condition that I wrote was the `department = 'Sales'`, the second condition I wrote was `department = 'Finance'`. With the `OR` operator

separating the two conditions, I obtained the data I needed to help my team. Please refer to the code and screenshot below for reference.

```
SELECT *
FROM employees
WHERE department = ‘Sales’ OR department = 'Finance';
```

```
MariaDB [organization]> SELECT *  FROM employees  WHERE department = 'Sales' OR department = 'Finance';
+-------------+--------------+----------+------------+-------------+
| 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   |
|        1009 | NULL         | lrodriqu | Sales      | South-134   |
|        1010 | k2421212m542 | jlansky  | Finance    | South-109   |
|        1011 | l748m120n401 | drosas   | Sales      | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance    | North-271   |
|        1017 | r550s824t230 | jclark   | Finance    | North-188   |
```

Retrieve all employees not in IT

The final task essential to securing the systems at our organization was to apply a separate update to employee computers who are not in the Information Technology department. This update was already made to employee computers in the Information Technology department and must be deployed to the others.

I began the first step in this query by selecting all data from the `employees` table. I then used the `WHERE` clause and `NOT` operator. I filtered for all employees that are not listed in the Information Technology column/department. The `NOT` operator allowed this as it selects all data except the specified value. Please refer to the following code and screenshot for reference.

```
SELECT *
FROM employees
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   |
|        1003 | d394e816f943 | sgilmore | Finance         | South-153   |
|        1004 | e218f877g788 | eraab    | Human Resources | South-127   |
|        1005 | f551g340h864 | gesparza | Human Resources | South-366   |
```

Summary

I efficiently obtained precise data regarding login attempts and employee machines from the `log_in_attempts` and `employees tables` by applying filters to SQL queries. Utilizing the power of `AND`, `OR`, and `NOT` operators, I tailored the filters to extract the exact information required for each task to help my team. Additionally, I harnessed the `LIKE` keyword with the percentage sign (`%`) wildcard to filter for specific patterns in the data. These efforts enable me to conduct in-depth investigations into potential security incidents and address security-related challenges within our system.