Apply filters to SQL queries

Project description

I was tasked by the organization I work with to investigate security issues in order to keep the system secure. In doing so I encountered potential security threats involving login attempts and employee machines. I will examine the SQL database to retrieve the data necessary to investigate these issues.

Retrieve after hours failed login attempts

After discovering a potential security issue involving failed login attempts after hours, my first step was to check the database using SQL to see the time and location of all failed login attempts that occurred after 18:00.

```
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 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57  |       0 |
|       34 | drosas   | 2022-05-11 | 21:02:04   | US      | 192.168.45.93  |       0 |
```

The top of the image is the query that was used, and below that is the output from the database. The code above demonstrates how I used a SQL query to retrieve all failed login attempts after the organization closed. The first line uses the **SELECT** keyword to retrieve the desired information from the designated database. In this case, we retrieved ***** (**all**) columns from the **log_in_attempts** database. The line after uses the **WHERE** clause followed by the **AND** clause. These statements allow me to set restrictions on the data retrieved from the database to only those whose **login_time** is later than **18:00 AND** whose success was a **0** (or **FALSE**), indicating a failed attempt.

Retrieve login attempts on specific dates

The potential security issue occurred on **May 9th, 2022**. After retrieving the failed login attempts, my next step was to analyze all login attempts that occurred the day of, and the day before the incident. The code below shows the query that was used in order to retrieve the desired information.

```
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 |
```

The top of the image is the query that was used, and below that is the output from the database. As mentioned previously, the first line uses **SELECT** to retrieve ***** (or everything) from the **log_in_attempts** database. Following this line is the **WHERE** and **OR** statements. The first condition **WHERE login_date = '2022-05-09** allows me to filter the login attempts to the day of the incident. The following condition **OR login_date = '2022-05-08'** allows me to expand that filter slightly further to also include the day before the incident, May 8th 2022.

Retrieve login attempts outside of Mexico

After some investigating, it has been determined that the security incident did not originate in Mexico. The image below demonstrates the query used in order to analyze the login attempts that occurred outside of Mexico.

```
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 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        5 | jrafael  | 2022-05-11 | 03:05:59   | CANADA  | 192.168.86.232  |       0 |
```

The top of the image is the query that was used, and below that is the output from the database. The first part of my code indicates I am **SELECT**-ing everything from the **log_in_attempts** database. The line after, **WHERE NOT**, allows me to filter the output by hiding an unwanted piece of data. In this case, we don't want anything where the country is **MEX** or **MEXICO**. I achieved this by using the **LIKE** clause. By adding **WHERE NOT country LIKE 'MEX%'**, the output will skip over any **row** where the **country** column starts with **MEX**.

Retrieve employees in Marketing

My team and I would like to perform security updates on specific machines within the **marketing** department. In order to do this, I need to query the database for all employees in the **marketing**

department for all offices in the **east** building. The image below demonstrates the code used to achieve this.

```
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 |
|        1088 | k8651965m233 | rgosh    | Marketing  | East-157 |
|        1103 | NULL         | randerss | Marketing  | East-460 |
|        1156 | a184b775c707 | dellery  | Marketing  | East-417 |
|        1163 | h679i515j339 | cwilliam | Marketing  | East-216 |
+-------------+-------------+----------+------------+----------+
7 rows in set (0.001 sec)
```

The top of the image is the query that was used, and below that is the output from the database. After selecting everything from the **employees** table, I included a **WHERE** clause to limit the **department** to **Marketing**. However, I needed the output to only include the employees located in the east building. I achieved this by adding **AND office LIKE 'East%'**. The **AND** clause expands my filter to include the specified **office** parameters, and the **LIKE** clause indicates to include everything starting with '**East**' as indicated by the **%** after the word **East**.

Retrieve employees in Finance or Sales

We now need to perform a different security update on every machine for employees in the **Sales** and **Finance** departments. The image below demonstrates the SQL query used in order to retrieve the machines from our database.

```
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   |
```

The top of the image is the query that was used, and below that is the output from the database. The first line in my code uses **SELECT * FROM employees** indicating that we are selecting every column (*****) from the **employees** table. The line that follows this uses a **WHERE** and **OR** clause. The statement **WHERE department = 'Sales'** limits the query to retrieve only the employees from the **Sales** department. The statement **OR department = 'Finance'** allows us to further expand that limitation to also include the employees from the **Finance** department.

Retrieve all employees not in IT

Our last update is for every other employee who is not in the **IT** department. The image below demonstrates the code used to query the database in order to find the desired machines.

```
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   |
```

The top of the image is the query that was used, and below that is the output from the database. The first line in my code uses **SELECT * FROM employees** indicating that we are selecting every column (*****) from the **employees** table. The statement that appears after, **WHERE NOT department = 'Information Technology'** is a filter that will skip over every employee in the IT department, and will output every other employee outside of the IT department.

Summary

I applied various filters to SQL queries in order to retrieve the desired information of login attempts and employee machines in need of updates. I achieved this by analyzing the data in the **log_in_attempts** and **employees** tables while using the **AND, OR,** and **NOT** keywords in order to filter the desired output..