Apply filters to SQL queries

Project description

As part of the ongoing efforts to enhance the security of our organization's system, my responsibility involves ensuring the system's safety, investigating potential security issues, and updating employee computers as necessary. The ensuing steps illustrate instances where I utilized SQL with filters to carry out tasks directly related to security measures.

Retrieve after hours failed login attempts

At 18:00 there was a security incident involving failed login attempts that were after business hours. All attempts made after business hours are actively being investigated.

The following code demonstrates how I created a SQL query to filter for failed login attempts that occurred after business hours.

```
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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output.

This query filters for all failed login attempts that occurred after 18:00. First, I started by selecting all the data from the `log_in_attempts` table, Then I used a `WHERE` clause coupled with an `AND` operator to filter my results to output only login attempts that occurred after 18:00 and were unsuccessful. The first condition I used is `login_time > '18:00'`, which filters for the login attempts that occurred after 18:00. The second condition is `success = FALSE`, which filters for the failed login attempts

Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. Any login activity that happened on 2022-05-09 or on the day before needs to be investigated.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred on specific dates.

```
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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns al login attempts that occurred on 2022-05-09 or 2022-05-08. First I stared by selecting all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with an `OR` operator to filter my results to output only login attempts that occurred on either on 2022-05-09 or 2022-05-08. The first conditon is `login_date = '2022-05-09'`, which filters for logins on 2022-05-09. The second condition is `login_date = '2022-05-08'`, which filters for logins on 2022-05-08.

Retrieve login attempts outside of Mexico

Upon reviewing the organization's data concerning login attempts, it appears that there is a concern with those originating outside of Mexico. It is advisable to conduct an investigation into these specific login attempts.

The following code demonstrates how I created a SQL query to filter for 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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all login attempts that occurred in countries other than Mexico. First, I started by selecting all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with `NOT` to filter for countries other than Mexico. I used `LIKE` with `MEX%` as the pattern to match because the dataset represents Mexico as `MEX` and `MEXICO`. The percentage sign (`%`) represents any number of unspecified characters when used with `LIKE`.

Retrieve employees in Marketing

My team aims to perform updates on computers assigned to specific individuals within the Marketing department. To facilitate this, I need to gather details regarding the machines of the employees targeted for updates.

The following code demonstrates how I created a SQL query to filter for employee machines from employees in the Marketing department in the East building.

```
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 |
```

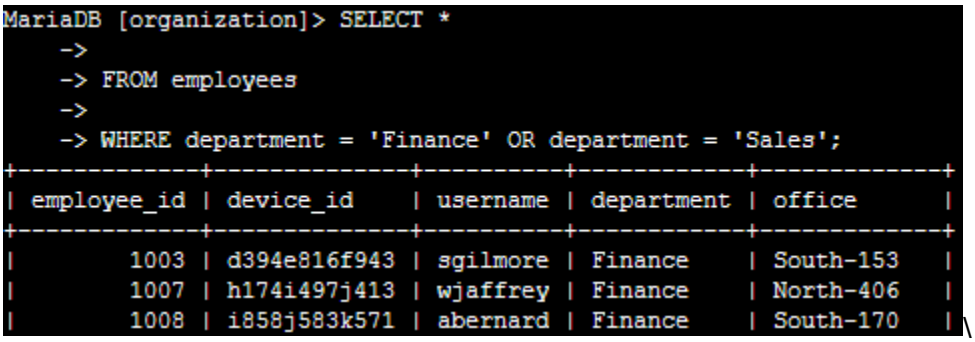
The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all employees in the Marketing department in the East building. First, I started by selecting all data from the `employees` table. Then, I used a `WHERE` clause with `AND` to filter for employees who work in the Marketing department and in the East building. I used `LIKE` with `East%` as the pattern to match because the data in the `office` column represents the East building with the specific office number. The first condition is the `department = 'Marketing'` portion, which filters for employees in the Marketing department. The second

condition is the `office LIKE 'East%'` portion, which filters for employees in the East building.

Retrieve employees in Finance or Sales

Updates are also required for machines used by employees in the Finance and Sales departments. Given the distinct security update needed for these departments, I must gather information exclusively on employees belonging to Finance and Sales.

The following code demonstrates how I created a SQL query to filter for employee machines from employees in the Finance or Sales departments.



```
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   |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query returns all employees in the Finance and Sales departments. First, I started by selecting all data from the `employees` table. Then, I used a `WHERE` clause with `OR` to filter for employees who are in the Finance and Sales departments. I used the `OR` operator instead of `AND` because I want all employees who are in either department. The first condition is `department = 'Finance'`, which filters for employees from the Finance department. The second condition is `department = 'Sales'`, which filters for employees from the Sales department.

Retrieve all employees not in IT

An additional security update is necessary for employees outside the Information Technology department. To initiate this update, the first step involves obtaining information on these specific employees.

```
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   |
```

The first part of the screenshot is my query, and the second part is a portion of the output. The query returns all employees not in the Information Technology department. First, I started by selecting all data from the `employees` table. Then, I used a `WHERE` clause with `NOT` to filter for employees not in this department.

Summary

I utilized filters in SQL queries to extract precise information regarding login attempts and employee machines. Employing two distinct tables, namely `log_in_attempts` and `employees`, I employed the `AND`, `OR`, and `NOT` operators to refine the queries for each task. Additionally, I incorporated the `LIKE` operator and the percentage sign (`%`) wildcard to filter for specific patterns as part of the filtering process. With practical experience, I have become proficient in utilizing SQL to run queries, extracting valuable information from databases. I can now apply AND, OR, and NOT operators effectively, allowing me to finely filter SQL queries to meet specific requirements.