

Apply filters to SQL queries

Project description

My organization is focused on strengthening system security, and my role involves identifying potential vulnerabilities, investigating security issues, and maintaining secure configurations across employee systems. The following examples demonstrate how I leveraged SQL queries with filters to perform targeted, security-focused tasks, ensuring data integrity and system protection.

Retrieve after hours failed login attempts

A potential security incident occurred after business hours (after 18:00). I wrote a SQL query to filter for failed login attempts that happened after 18:00 so these could be investigated.

```
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
|      0 |          |            |            |          |             | 0
|     18 | pwashing | 2022-05-11 | 19:28:50 | US      | 192.168.66.142 | 0
|      0 |          |            |            |          |             | 0
|     20 | tshah    | 2022-05-12 | 18:56:36 | MEXICO  | 192.168.109.50 | 0
|      0 |          |            |            |          |             | 0
|     28 | aestrada | 2022-05-09 | 19:28:12 | MEXICO  | 192.168.27.57 | 0
|      0 |          |            |            |          |             | 0
```

The first part of my query selected all records from the `log_in_attempts` table. I then filtered the results to include only login attempts that occurred after 18:00 and were unsuccessful. Specifically, I used `login_time > '18:00'` to capture attempts after 6 PM, and `success = 0` to isolate failed logins. The screenshot shows the query I wrote along with a portion of the resulting output.

Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09, so any login activity from that day or the day before needs to be investigated. I wrote a SQL query to filter for login attempts on these 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 | su
ccess |
+-----+-----+-----+-----+-----+-----+
|      1 | jrafael | 2022-05-09 | 04:56:27 | CAN     | 192.168.243.140 |
|      1 |          |           |           |          |           |      1
|      3 | dkot    | 2022-05-09 | 06:47:41 | USA     | 192.168.151.162 |
|      1 |          |           |           |          |           |      1
|      4 | dkot    | 2022-05-08 | 02:00:39 | USA     | 192.168.178.71 |
|      0 |          |           |           |          |           |      0
```

The first part of my query selected all records from the `log_in_attempts` table. Then I filtered for login attempts that occurred on 2022-05-09 or 2022-05-08. I used a `WHERE` clause with an `OR` operator: `login_date = '2022-05-09'` for logins on May 9th, and `login_date = '2022-05-08'` for logins on May 8th. The second part of the screenshot shows a portion of the query output.

Retrieve login attempts outside of Mexico

After reviewing the organization's login data, I identified unusual login attempts occurring outside of Mexico that need further investigation. I wrote a SQL query to filter for these login attempts.

```
MariaDB [organization]> SELECT *      FROM log_in_attempts  WHERE NOT country LIKE
'MEX%';
+-----+-----+-----+-----+-----+-----+
| event_id | username | login_date | login_time | country | ip_address | su
ccess |
+-----+-----+-----+-----+-----+-----+
|      1 | jrafael | 2022-05-09 | 04:56:27 | CAN     | 192.168.243.140 |
|      1 |          |           |           |          |           |      1
|      2 | apatel   | 2022-05-10 | 20:27:27 | CAN     | 192.168.205.12  |
```

The first part of my query selected all records from the `log_in_attempts` table. Then I filtered for login attempts that occurred in countries other than Mexico. I used a `WHERE` clause with `NOT` and `LIKE 'MEX%'` to exclude Mexico, since the dataset represents the country as both `MEX` and `MEXICO`. The `%` wildcard matches any number of characters. The second part of the screenshot shows a portion of the query output.

Retrieve employees in Marketing

My team needed to update the computers for employees in the Marketing department. I wrote a SQL query to filter and get information on the machines for employees in Marketing who work 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 |
```

The first part of my query selected all records from the `employees` table. Then I filtered for employees who work in the Marketing department and in the East building. I used a `WHERE` clause with `AND` to apply both conditions: `department = 'Marketing'` to get employees in Marketing, and `office LIKE 'East%'` to capture employees in the East building, since the `office` column includes specific office numbers. The second part of the screenshot shows a portion of the query output.

Retrieve employees in Finance or Sales

The machines for employees in the Finance and Sales departments needed a different security update. I wrote a SQL query to filter and get information only for employees in these two 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 my query selected all records from the `employees` table. Then I filtered for employees in the Finance and Sales departments. I used a `WHERE` clause with `OR` to include employees from either department: `department = 'Finance'` to get Finance employees, and `department = 'Sales'` to get Sales employees. The second part of the screenshot shows a portion of the query output.

Retrieve all employees not in IT

My team needed to make a security update for employees who are not in the Information Technology department. I wrote a SQL query to filter and get information only for these 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 |
|     1003    | d394e816f943 | sgilmore | Finance    | South-153  |
|     1004    | e218f877g788 | eraab    | Human Resources | South-127  |
|     1005    | f551a340h864 | gesparza | Human Resources | South-366  |

```

The first part of my query selected all records from the `employees` table. Then I filtered for employees who are not in the Information Technology department using a `WHERE` clause with `NOT`. The second part of the screenshot shows a portion of the query output.

Summary

I applied filters to SQL queries to extract targeted information from the `log_in_attempts` and `employees` tables. Using operators like `AND`, `OR`, and `NOT`, I refined the results to meet specific task requirements. I also leveraged `LIKE` with the `%` wildcard to identify patterns, enabling precise analysis of login activity and employee systems.