Apply filters to SQL queries

Project description

I am responsible for helping my organization improve system security. My role involves making sure the system is protected, investigating any potential security issues, and updating employee computers when necessary. The steps below demonstrate how I applied SQL queries with filters to carry out security-related tasks effectively.

Retrieve after hours failed login attempts

During my security review, I noticed unusual activity after business hours (after 18:00). To verify this, I needed to check all login attempts that failed outside of working hours. I wrote the following SQL query to filter the results:

```
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 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157   |       0 |
|       52 | cjackson | 2022-05-10 | 22:07:07   | CAN     | 192.168.58.57   |       0 |
|       69 | wjaffrey | 2022-05-11 | 19:55:15   | USA     | 192.168.100.17  |       0 |
|       82 | abernard | 2022-05-12 | 23:38:46   | MEX     | 192.168.234.49  |       0 |
|       87 | apatel   | 2022-05-08 | 22:38:31   | CANADA  | 192.168.132.153 |       0 |
|       96 | ivelasco | 2022-05-09 | 22:36:36   | CAN     | 192.168.84.194  |       0 |
|      104 | asundara | 2022-05-11 | 18:38:07   | US      | 192.168.96.200  |       0 |
|      107 | bisles   | 2022-05-12 | 20:25:57   | USA     | 192.168.116.187 |       0 |
|      111 | aestrada | 2022-05-10 | 22:00:26   | MEXICO  | 192.168.76.27   |       0 |
```

First, I selected all data from the **log_in_attempts** table ,Then I added a **WHERE** clause with an **AND** operator to narrow down the results,The first condition `login_time > '18:00'`

returns login attempts that occurred after 6:00 PM.The second condition `success = 0` includes only failed attempts.

With this query, I was able to view all failed login attempts that happened after business hours. This helps me identify suspicious activity or potential unauthorized access outside of normal working times.

Retrieve login attempts on specific dates

To investigate the suspicious event that occurred on **2022-05-09**, I needed to review all login activity from that date as well as the previous day.

I used the following SQL query:

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

This query retrieves only the login attempts that occurred on 2022-05-08 and 2022-05-09. By focusing on these two days, I can narrow the results to the exact timeframe related to the incident. This makes it easier to analyze login behavior and identify any unusual activity connected to the event.

Retrieve login attempts outside of Mexico

While reviewing login activity, I noticed some login attempts that did not originate from Mexico. To focus on these attempts, I created the following SQL query:

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

This query retrieves all login attempts from countries other than Mexico,I selected all data from the log_in_attempts table,Then I used a WHERE clause with NOT LIKE to exclude any record where the country column contains “MEXICO”, The % wildcard allows the query to catch both “MEX” and “MEXICO” in the dataset.
By running this query, I can focus on login attempts from outside Mexico and investigate any unusual or suspicious activity more efficiently.

Retrieve employees in Marketing

To identify which employee machines need to be updated, I created an SQL query to retrieve employees who work in the **Marketing department** and are located in the **East building**.

The query is written as follows:

```
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 |
```

The query retrieves only the relevant columns instead of selecting all data, such as the employee ID, name, machine ID, department, and office. The data is taken from the **employees** table. A `WHERE` clause is used to filter the results so that only employees in the **Marketing** department are returned. In addition, the condition `office LIKE 'East%'` ensures that only employees located in the **East building** are included, since the office field contains both the building name and number.

As a result, the query returns the subset of employees in the Marketing department whose offices are in the East building. From this output, we can clearly identify the machines that require updates.

Retrieve employees in Finance or Sales

The screenshot below shows the SQL query that I created to retrieve employee information for the Finance and 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   |
|        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   |
```

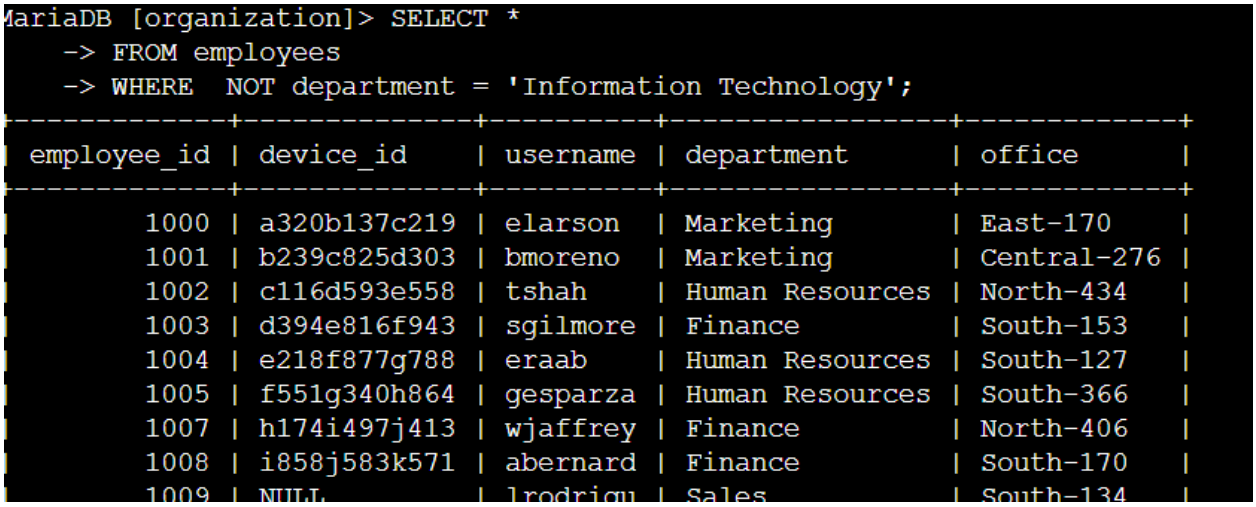
The query uses a `WHERE` clause with the `OR` operator to include employees from either department. This ensures that all employees in **Finance** or **Sales** are returned.

The next screenshot displays a portion of the output. As shown, the results list all employees who belong to the Finance or Sales departments. This allows me to identify the machines that require the appropriate security update for these two groups.

Retrieve all employees not in IT

My team needs to apply one more security update for employees who are not in the Information Technology department. Since the IT department has already received the update, I created an SQL query to filter out those employees and return only the ones in other departments.

The screenshot below shows the SQL query and a portion of its output:



```
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   |
|        1007 | h174i497j413 | wjaffrey | Finance         | North-406   |
|        1008 | i858j583k571 | abernard | Finance         | South-170   |
|        1009 | NULL         | lrodrigu | Sales           | South-134   |
```

The query uses the `NOT` operator in the `WHERE` clause to exclude employees from the Information Technology department. As shown in the screenshot, the results include only employees outside of the IT department, making it clear which machines still require the security update.

Summary

filtered SQL queries to retrieve targeted information about login attempts and employee machines. I worked with two separate tables: **log_in_attempts** and **employees**. To narrow down the results for each task, I applied the `AND`, `OR`, and `NOT` operators. Additionally, I used the `LIKE` operator along with the `%` wildcard to match specific patterns in the data.