Apply filters to SQL queries

Project description

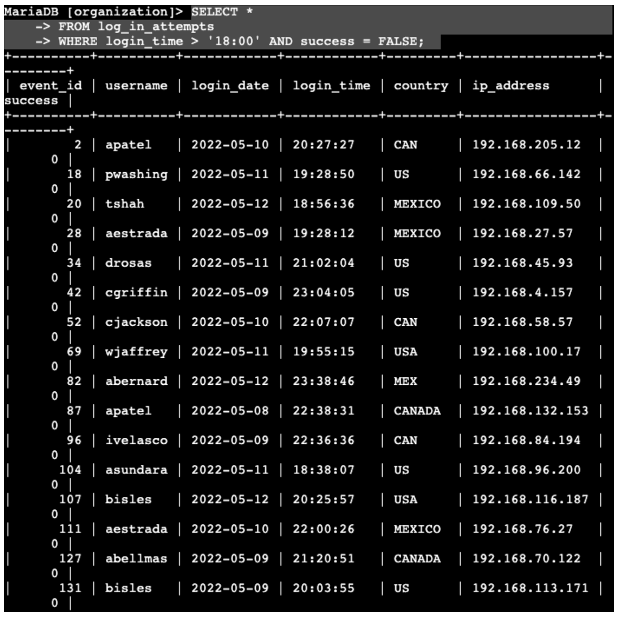
Using SQL you can retrieve data from tables by writing queries. SQL helps you retrieve logs within an organization's system.

<u>Retrieve after hours failed login attempts</u>

You recently discovered a potential security incident that occurred after business hours. To investigate this, you need to query the **`log_in_attempts`** table and review after hours login activity. Use filters in SQL to create a query that identifies all failed login attempts that occurred after 18:00. (The time of the login attempt is found in the **`login_time`** column. The **`success`** column contains a value of `0` when a login attempt failed; you can use either a value of `0` or **`FALSE`** in your query to identify failed login attempts.)

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

I selected all the contents of the after hours log in attempts from the log_in_attempts table. I input login_time after the WHERE clause because that is where I would find the login attempts made after 18:00 as requested.



```
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 |
|       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 |
|      127 | abellmas | 2022-05-09 | 21:20:51   | CANADA  | 192.168.70.122  |
      0 |
|      131 | bisles   | 2022-05-09 | 20:03:55   | US      | 192.168.113.171 |
      0 |
```

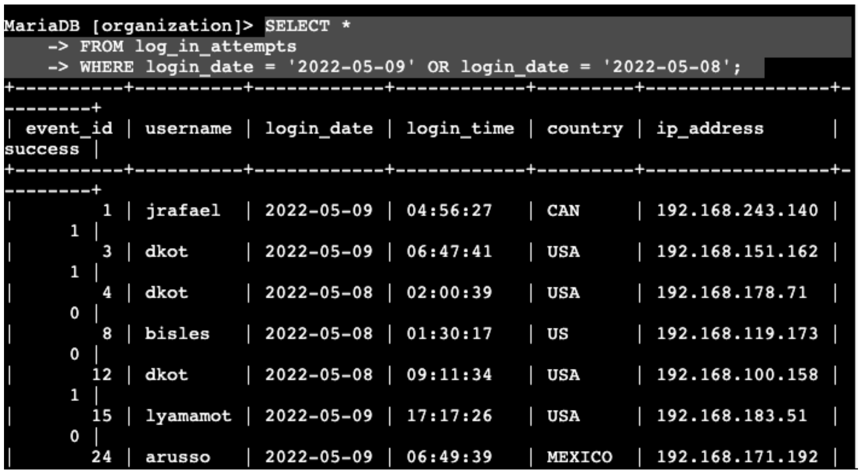
Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. To investigate this event, you want to review all login attempts which occurred on this day and the day before. Use filters in SQL to create a query that identifies all login attempts that occurred on 2022-05-09 or 2022-05-08. (The date of the login attempt is found in the `login_date` column.

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

View the screenshot from the lab below.

I selected all contents of the suspicious events that occurred on 2022-05-09 or 2022-05-08 from the log_in_attempts table.



```
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 |
```

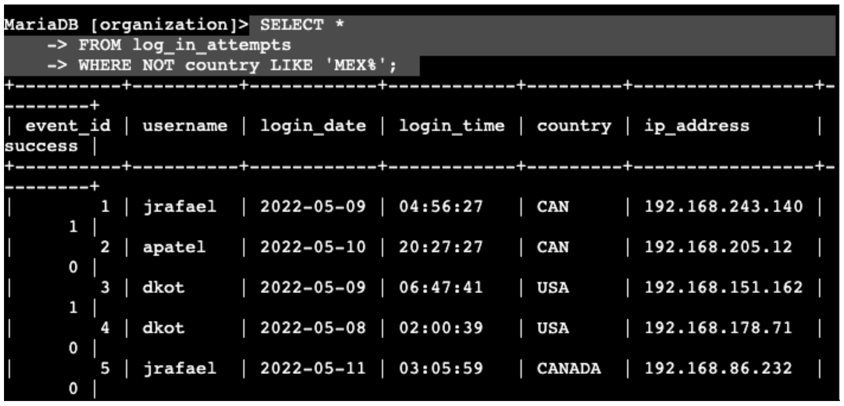
Retrieve login attempts outside of Mexico

There's been suspicious activity with login attempts, but the team has determined that this activity didn't originate in Mexico. Now, you need to investigate login attempts that occurred outside of Mexico. Use filters in SQL to create a query that identifies all login attempts that occurred outside of Mexico. (When referring to Mexico, the `country` column contains values of both `MEX` and `MEXICO`, and you need to use the `LIKE` keyword with `%` to make sure your query reflects this.)

SELECT *
FROM log_in_attempts
WHERE NOT country LIKE 'MEX%';

View the screenshot from the lab below.

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.



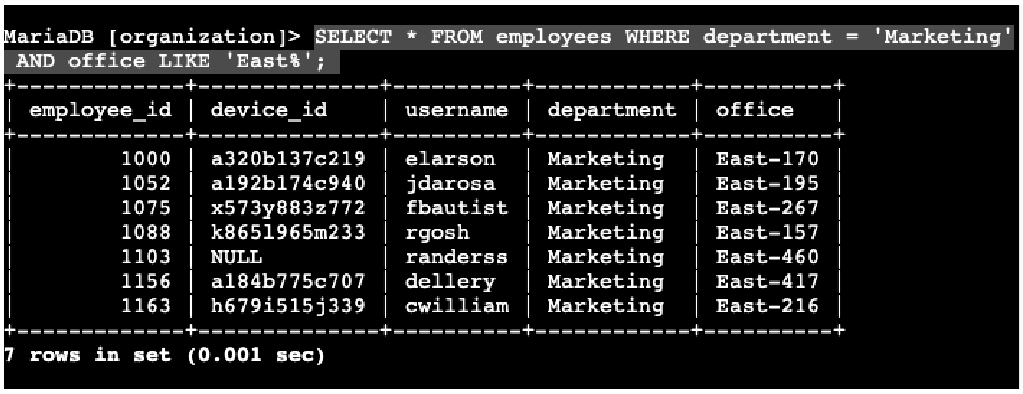
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+-
--------+
| event_id | username | login_date | login_time | country | ip_address      |
success |
+----------+----------+------------+------------+---------+-----------------+-
--------+
|        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 |
```

Retrieve employees in Marketing

Your team wants to perform security updates on specific employee machines in the Marketing department. You're responsible for getting information on these employee machines and will need to query the **`employees`** table. Use filters in SQL to create a query that identifies all employees in the Marketing department for all offices in the East building.
(The department of the employee is found in the **`department`** column, which contains values that include **`Marketing`**. The office is found in the office column. Some examples of values in this column are **`East-170`**, **`East-320`**, and **`North-434`**. You'll need to use the **`LIKE`** keyword with `%` to filter for the East building.)
SELECT *
FROM employees
WHERE department = 'Marketing' AND office LIKE 'East%';

View the screenshot from the lab below.

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.



```
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)
```

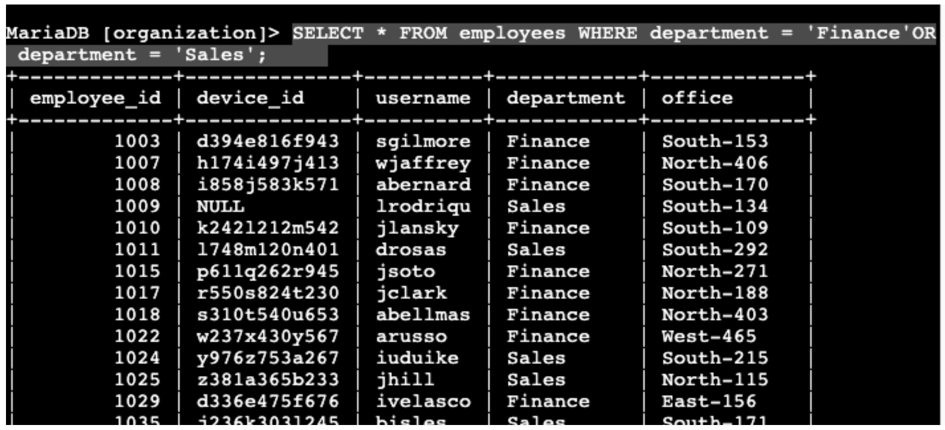
Retrieve employees in Finance or Sales

Your team now needs to perform a different security update on machines for employees in the Sales and Finance departments. Use filters in SQL to create a query that identifies all employees in the Sales or Finance departments. (The department of the employee is found in the **department** column, which contains values that include **Sales** and **Finance**.)

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

View the screenshot from the lab below.

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.



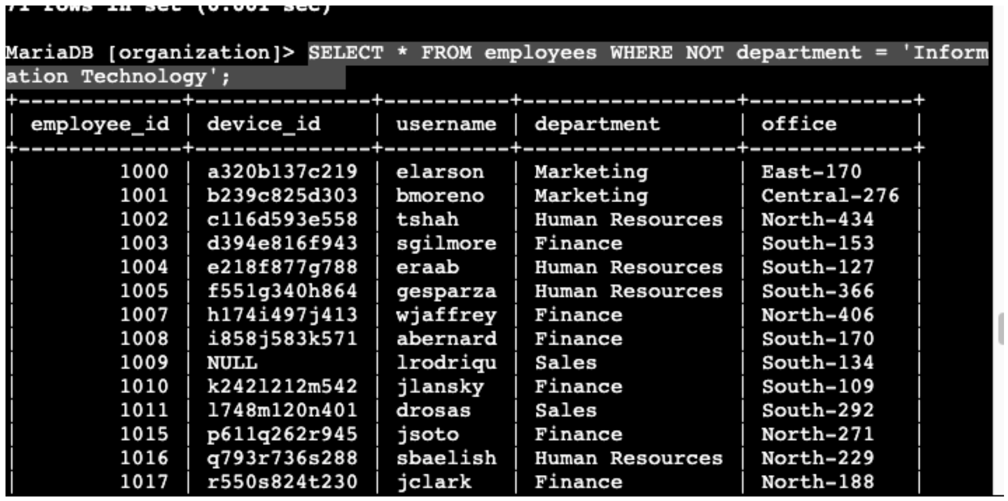
```
MariaDB [organization]> SELECT * FROM employees WHERE department = 'Finance'OR
 department = 'Sales';
+-------------+--------------+----------+------------+-------------+
| employee_id | device_id    | username | department | office      |
+-------------+--------------+----------+------------+-------------+
|        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 | 1748m120n401 | drosas   | Sales      | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance    | North-271   |
|        1017 | r550s824t230 | jclark   | Finance    | North-188   |
|        1018 | s310t540u653 | abellmas | Finance    | North-403   |
|        1022 | w237x430y567 | arusso   | Finance    | West-465    |
|        1024 | y976z753a267 | iuduike  | Sales      | South-215   |
|        1025 | z381a365b233 | jhill    | Sales      | North-115   |
|        1029 | d336e475f676 | ivelasco | Finance    | East-156    |
|        1035 | j236k3031245 | bisles   | Sales      | South-171   |
```

Retrieve all employees not in IT

Your team needs to make one more update to employee machines. The employees who are in the Information Technology department already had this update, but employees in all other departments need it. Use filters in SQL to create a query which identifies all employees not in the IT department. (The department of the employee is found in the **`department`** column, which contains values that include **`Information Technology`**.)

SELECT *
FROM employees
WHERE NOT department = ‘Information Technology’;

By using the NOT clause with WHERE, I was able to locate the employees that do not work in the IT department. View the screenshot from the lab below.
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.



```
71 rows in set (0.001 sec)

MariaDB [organization]> SELECT * FROM employees WHERE NOT department = 'Inform
ation 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         | lrodriqu | Sales           | South-134   |
|        1010 | k2421212m542 | jlansky  | Finance         | South-109   |
|        1011 | 1748m120n401 | drosas   | Sales           | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance         | North-271   |
|        1016 | q793r736s288 | sbaelish | Human Resources | North-229   |
|        1017 | r550s824t230 | jclark   | Finance         | North-188   |
```

Summary

In this scenario I was a security professional at a large organization. Part of my job was to investigate security issues to help keep the system secure. I recently discovered some potential security issues that involve login attempts and employee machines.
My task was to examine the organization’s data in their **`employees`** and **`log_in_attempts`** tables. I used SQL filters to retrieve records from different datasets and investigate the potential security issues.