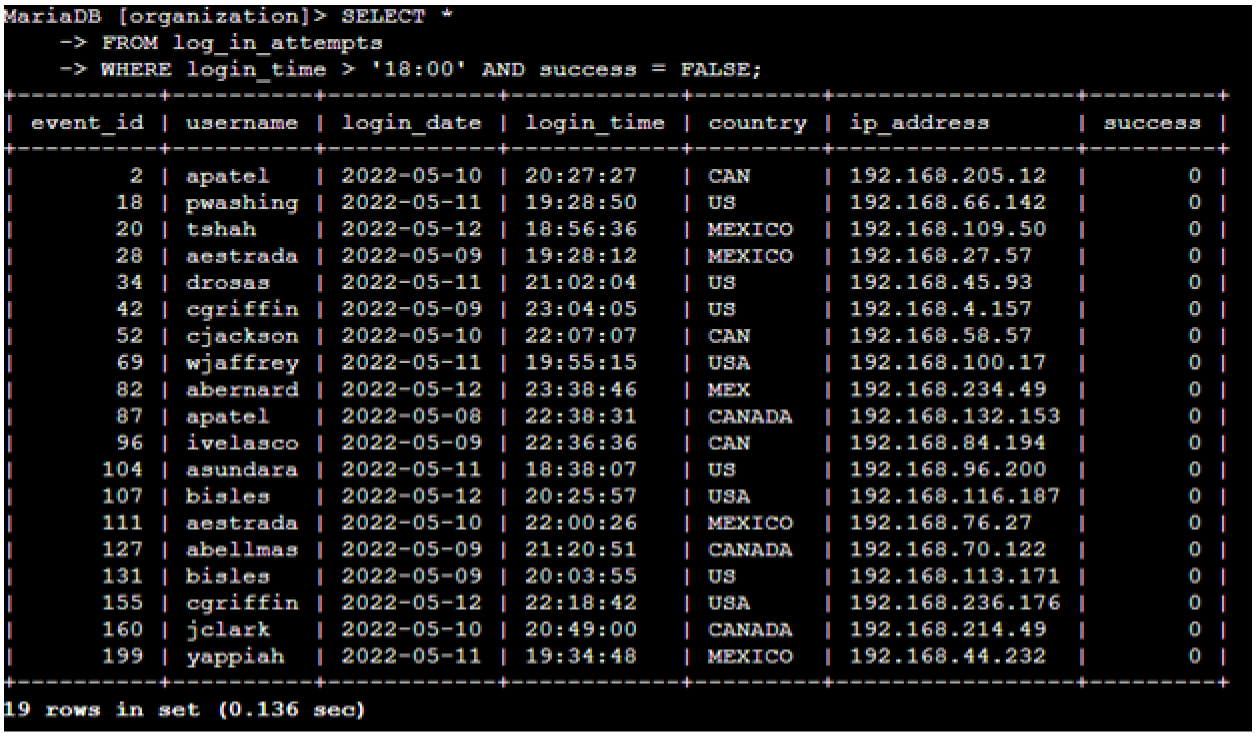
Apply filters to SQL queries

Project description

My organization is working to strengthen its security systems, and it is my job to ensure the system is safe by investigating potential security issues and updating employee computers as needed. The following steps outline how I used SQL with filters to perform security-related tasks involving login attempts and employee machine data.

Retrieve after hours failed login attempts

A potential security incident was identified with login attempts occurring after business hours (after 18:00 PM). To investigate, I needed to gather all failed login attempts that happened during this time frame. I wrote a SQL query to filter for login attempts that occurred after 18:00 PM and were unsuccessful.



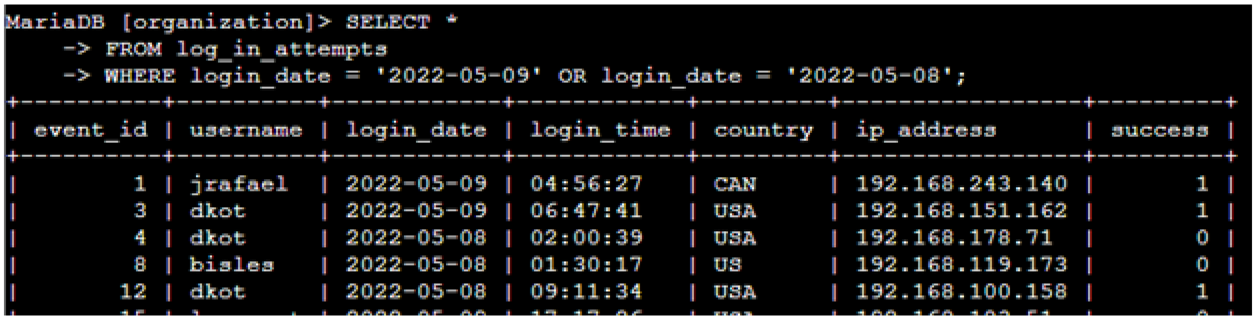
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
|      155 | cgriffin | 2022-05-12 | 22:18:42   | USA     | 192.168.236.176 |       0 |
|      160 | jclark   | 2022-05-10 | 20:49:00   | CANADA  | 192.168.214.49  |       0 |
|      199 | yappiah  | 2022-05-11 | 19:34:48   | MEXICO  | 192.168.44.232  |       0 |
+----------+----------+------------+------------+---------+-----------------+---------+
19 rows in set (0.136 sec)
```

This command retrieves all columns from the `log_in_attempts` table. The `WHERE` clause filters the results based on two conditions: the first condition, `login_time > '18:00'`, ensures that only login attempts occurring after 6:00 PM are included. The `AND` operator is used to combine this condition with the second condition, `success = FALSE`, which ensures that only failed login attempts are returned.

As shown in the screenshot, the query outputs only the failed login attempts that took place after 6:00 PM.

Retrieve login attempts on specific dates

A suspicious event took place on 2022-05-09, and I needed to investigate any login activity on 2022-05-09 or 2022-05-08 (the day before).



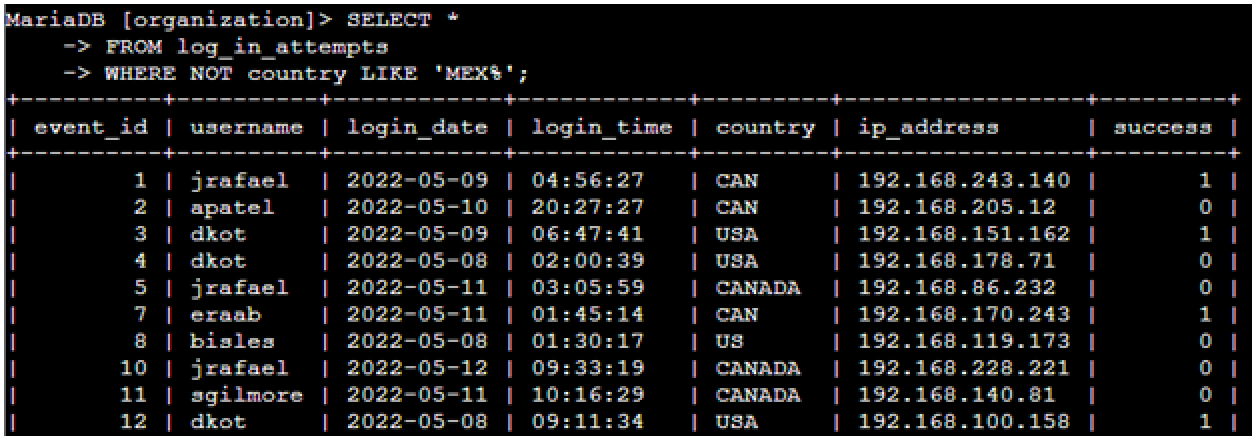
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
```

This query selects all data from the `log_in_attempts` table, with a `WHERE` clause that filters records where the `login_date` is either `2022-05-09` or `2022-05-08`. By using the `OR` operator, this query ensures that login attempts from both the specific day of the suspicious event and the day before are captured. This allows for an investigation of any activity that may have been linked to the event.

As shown in the screenshot, the results return login attempts from the two specific dates: 2022-05-09 and 2022-05-08.

Retrieve login attempts outside of Mexico

During my investigation, I suspected that login attempts from outside of Mexico might pose a security threat. I needed to identify and investigate login attempts that originated outside of Mexico.



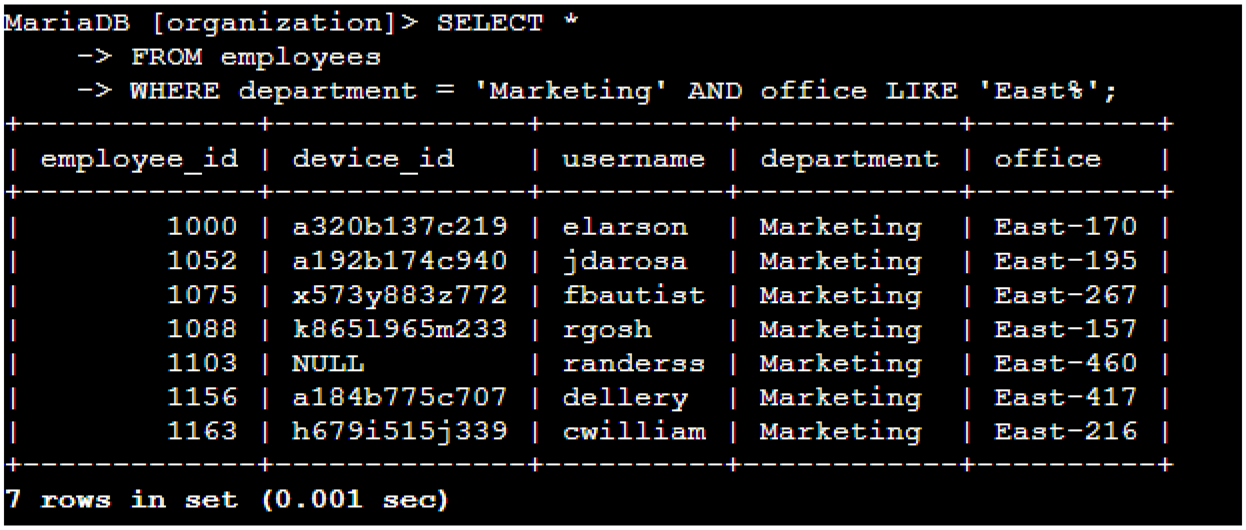
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
|        7 | eraab    | 2022-05-11 | 01:45:14   | CAN     | 192.168.170.243 |       1 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       10 | jrafael  | 2022-05-12 | 09:33:19   | CANADA  | 192.168.228.221 |       0 |
|       11 | sgilmore | 2022-05-11 | 10:16:29   | CANADA  | 192.168.140.81  |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
```

In this query, I again select all columns from the `log_in_attempts` table. The `WHERE` clause contains a condition that filters for login attempts from countries other than Mexico. Specifically, the `NOT LIKE 'MEX%'` condition excludes any login attempts where the country code starts with "MEX", which would represent Mexico (including variations like "Mexico" or "MEXICO"). The `%` wildcard allows for any characters after "MEX", helping to exclude all records related to Mexico. This query enables me to identify and investigate login attempts from other countries.

As shown in the screenshot, the query successfully filters out all login attempts originating from Mexico, leaving only those from other countries.

Retrieve employees in Marketing

My team needed to update the computers for employees in the Marketing department who work in the East building. To retrieve this information, I wrote a SQL query that selected employees from the Marketing department who also worked in the East building.



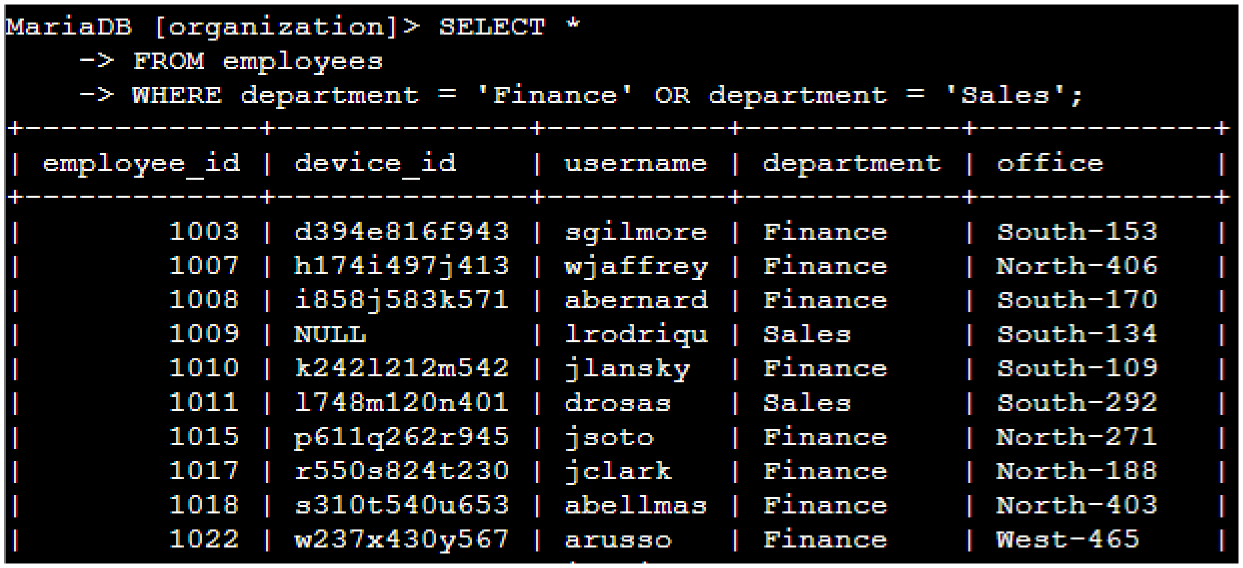
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+----------+
| employee_id | device_id     | username | department | office   |
+-------------+---------------+----------+------------+----------+
|        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)
```

In this query, I selected all columns from the `employees` table. The `WHERE` clause contains two conditions, combined using the `AND` operator. The first condition, `department = 'Marketing'`, filters for employees who work in the Marketing department. The second condition, `office LIKE 'East%'`, filters for employees whose office location begins with "East" (indicating they work in the East building). Both conditions must be true, meaning the query will return employees who are in the Marketing department and work in the East building. The `AND` operator ensures that both criteria are met simultaneously.

As shown in the screenshot, the query results display only the employees in the Marketing department working in the East building.

Retrieve employees in Finance or Sales

I needed to identify employees in the Finance and Sales departments for a specific security update. Since these two departments required a different update than the others, I had to filter out all employees from the Finance and Sales departments specifically.



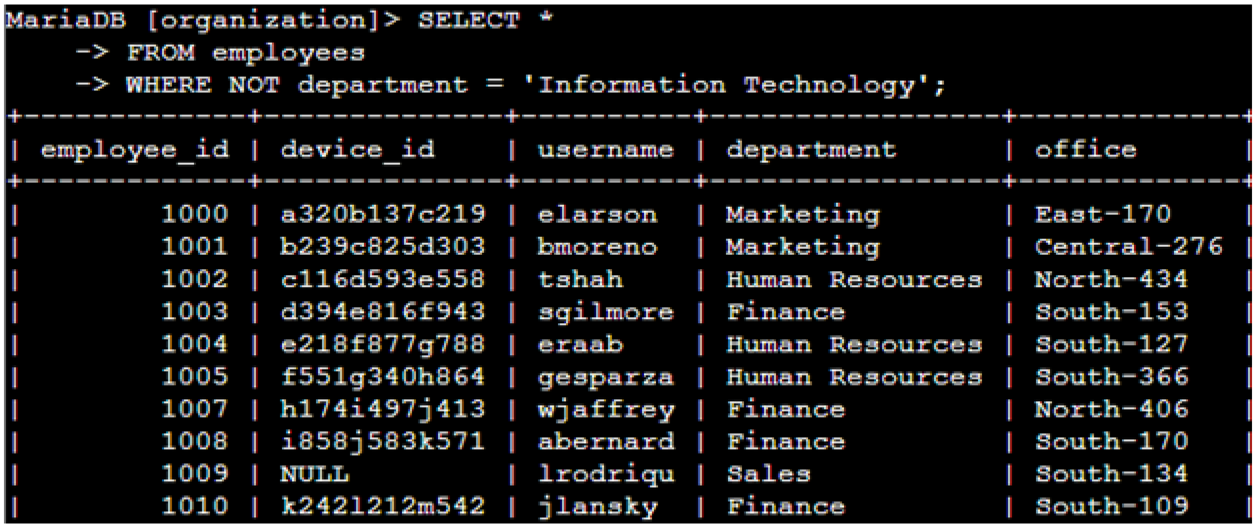
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+---------------+----------+------------+-------------+
| employee_id | device_id     | username | department | office      |
+-------------+---------------+----------+------------+-------------+
|        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    |
```

This query selects all data from the `employees` table. The WHERE clause contains two conditions: the first, `department = 'Finance'`, filters for employees in the Finance department, while the second, `department = 'Sales'`, filters for employees in the Sales department. The OR operator is used to ensure that employees from either department are included in the results.

As shown in the screenshot, the query returns employees from both the Finance and Sales departments.

Retrieve all employees not in IT

I needed to gather information about all employees not in the Information Technology (IT) department to apply a different security update.



```
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          | lrodriqu | Sales           | South-134   |
|        1010 | k2421212m542  | jlansky  | Finance         | South-109   |
```

In this query, I selected all columns from the `employees` table and used the `WHERE` clause to filter for employees whose department is not IT. The `NOT` operator ensures that employees in the IT department are excluded. While this query doesn't use the `AND` operator, it's important to understand that this filter is applied as a single condition, where the department must not be IT for the employee to be included in the result.

As shown in the screenshot, the query results exclude all employees in the IT department, returning only those from other departments.

Summary

In this project, I applied filters to SQL queries to get specific information on login attempts and employee machines. I used the `AND`, `OR`, and `NOT` operators to filter for specific information needed for each task. I also used `LIKE` and the percentage sign `(%)` wildcard to filter for patterns, such as employees in certain buildings or login attempts from specific countries. Using these SQL queries, I identified potential security issues and gathered the necessary data to secure the organization’s systems.