Apply filters to SQL queries

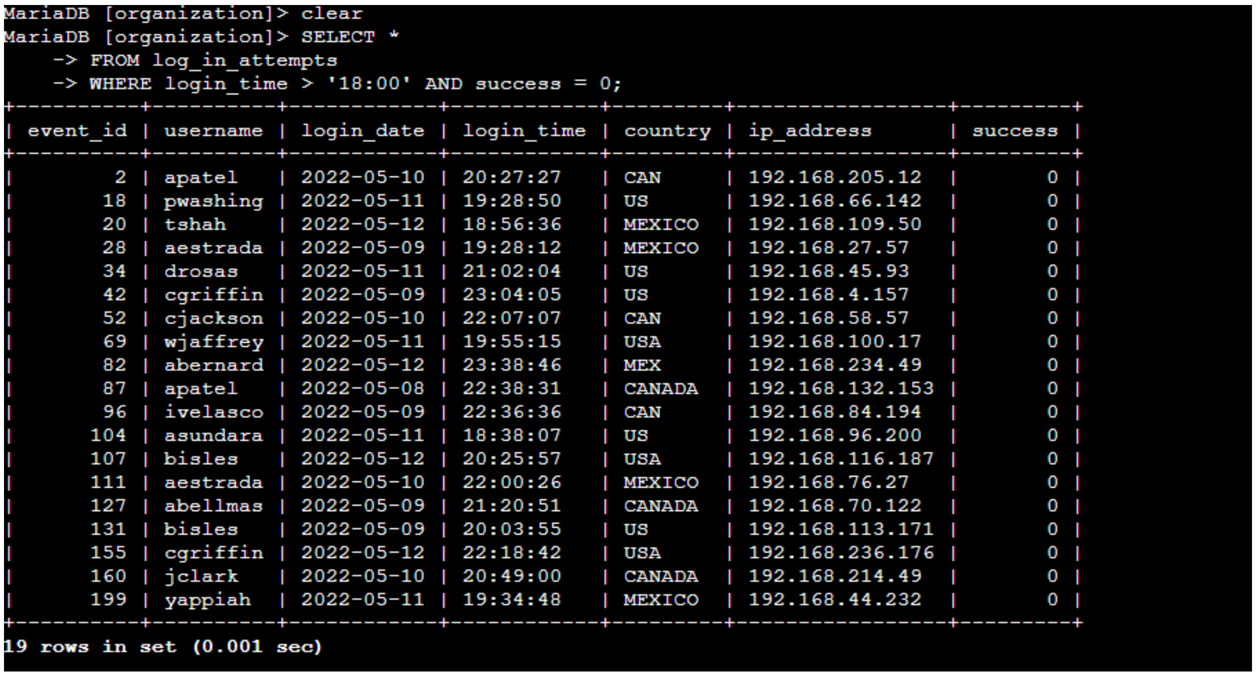
Project description

I am responsible for enhancing the security of my organization's system. My role involves ensuring the system's safety, examining potential security concerns, and updating employee computers as required. The following examples will demonstrate the process of how I used SQL with filters to perform security-related tasks.

Retrieve after hours failed login attempts

There was a potential security incident that occurred after business hours (18:00). The login times after business hours need to be investigated.

The following code illustrates my process of creating an SQL query to filter for failed login attempts happening after regular business hours.



```
MariaDB [organization]> clear
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 |
|      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 (0.001 sec)
```

The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

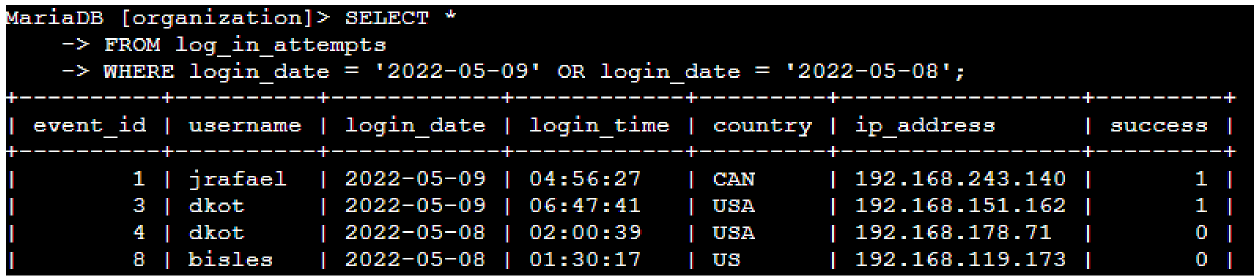
In order to filter for failed login attempts that happened after 18:00, I began by selecting all the data from the `log_in_attempts` table. In order to achieve this, I used the following code: SELECT * FROM log_in_attempts.

In order to display specific columns from the table, I can utilize the SELECT command. To extract all the data from the table log_in_attempts, I will use `*` as a wildcard symbol that selects all the columns. To indicate the source of the data, I will include FROM log_in_attempts in the query.

Subsequently, I employed a **`WHERE`** clause with an **`AND`** operator to further narrow down the results. This ensured that only login attempts occurring after 18:00 and being unsuccessful were displayed.
To achieve this, the first condition utilized was **`login_time > '18:00'`**. This condition filters for login attempts occurring strictly after 18:00. The second condition, `success = FALSE`, was used to filter for failed login attempts.

Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09; all login attempts that occurred on the day and the day before need investigation.



```
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 |
```

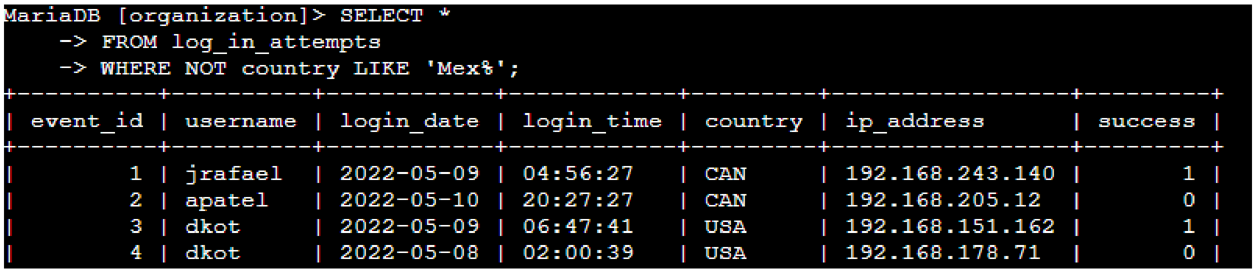
The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

The above query will return all login attempts from 2022-05-08 or 2022-05-09. The first thing I did was select all of the data from the **`log_in_attempts`** table using SELECT * FROM log_in_attempts. Afterward, I utilized a **`WHERE`** clause with an **`OR`** operator to filter my output to display login attempts on 2022-05-08 or 2022-05-09. The first condition is **`login date = ‘2022-05-09’`**. This will filter the logins for 2022-05-09. Then, the second condition input is **`login_date = ‘2022-05-08’`** which will filter the logins for 2022-05-08.

Retrieve login attempts outside of Mexico

After a careful review of the login attempts via the data, I have determined that the activity did not originate in Mexico. I will now have to filter my searches for login attempts outside of Mexico.

The following code was used to filter out any 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 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

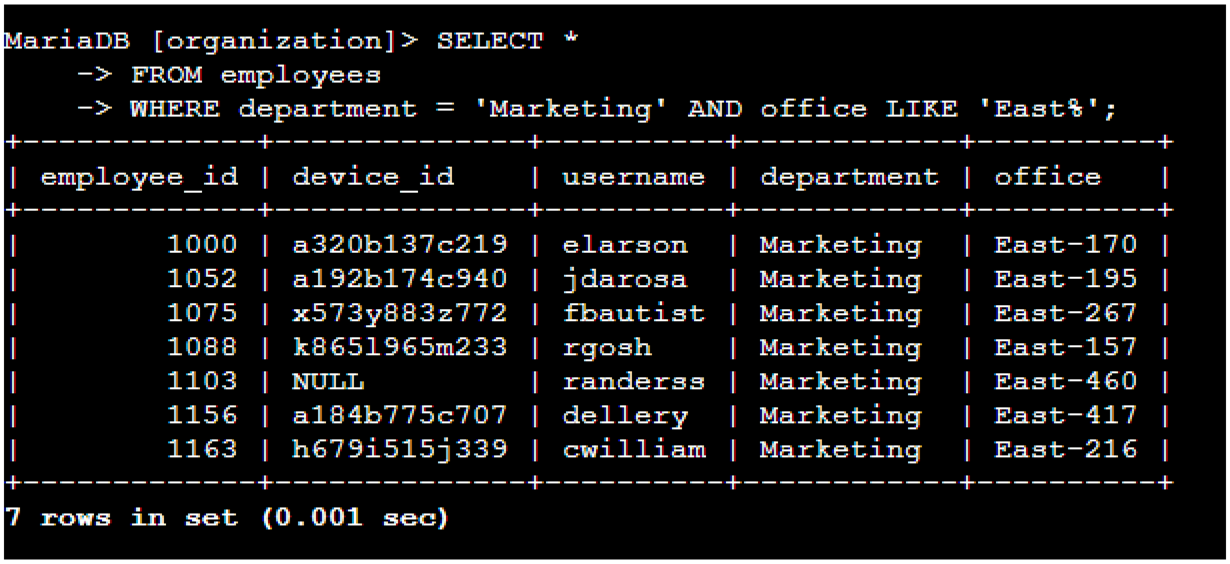
The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

The query that I have entered will return an output of all the login attempts from countries outside of Mexico. In order to do this, I first started by selecting all the data from the `log_in_attempts` table using `SELECT * FROM log_in_attempts`. The next step I did was to use a `WHERE` clause with a `NOT` filter, allowing me to filter out countries that are not Mexico. Utilizing `LIKE` with `MEX%`, I am able to search for both `MEX` and `MEXICO` because both can be represented in the dataset. The percentage sign (`%`) after `MEX` is a wildcard that represents any number of unspecified characters when used with `LIKE`.

Retrieve employees in Marketing

My team wants to perform security updates on specific employee machines in the Marketing department.

The following code was used to SQL query filter for employee machines from employees within the marketing department that are 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)
```

The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

I start by selecting all the data from the employees table using SELECT * FROM employees. After that, I used a WHERE clause with an AND filter for any employee who works in the Marketing department and is located in the East building.

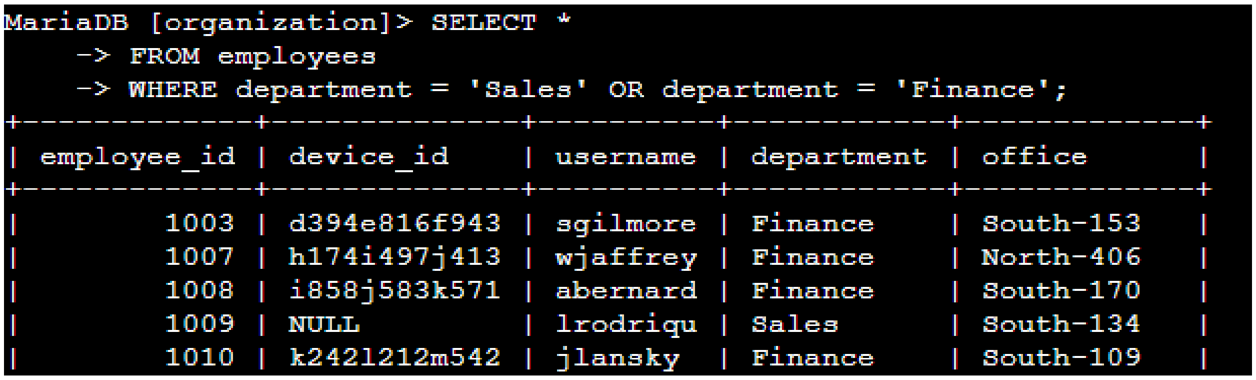
To filter for employees who are in the Marketing department as well as being located in the East building, I used the pattern East% with the LIKE operator. I used the wild card % to find any specific amount of characters after East because there are multiple East buildings within the office column.

In my first condition, using the WHERE clause with the AND filter, I search for employees in the Marketing department using department = ‘Marketing’. Then, in the second condition, I used office LIKE ‘East%’ to filter for employees in the East building.

Retrieve employees in Finance or Sales

Additionally, the employees in the Sales and Finance departments also require security updates. However, they require a different update. Therefore, I will need to obtain information on employees from only these two departments.

The following code shows how I filtered out employee machines from employees in the Marketing or Sales department.



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

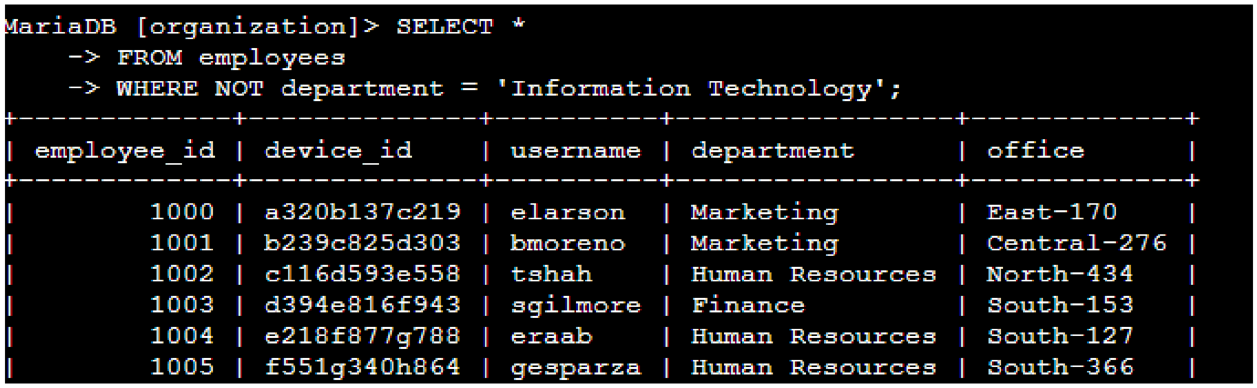
The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

First I selected all the data from the `employees` table using SELECT * FROM employees. After that, I used a **`WHERE`** clause with an **`OR`** to filter out employees who are in the Finance and Sales department. The **`OR`** operator was used instead of an **`AND`** because I wanted an output of those who are in either department. The first condition used was **`department = ‘sales’`**. This will filter the employees who are in the Sales department. In the next condition, I used **`department = 'Finance'`** to filter out employees who are in the Finance department.

Retrieve all employees not in IT

My team is required to make one more update on employee machines. This update is for every employee who is not part of the Information Technology department.

The following code demonstrates how I filtered for employee machines from employees who are not in the Information Technology department.



```
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   |
```

The query shown in the screenshot consists of two parts. The first part is my query, while the second part displays a section of the output.

First I selected all the data from the **`employees`** table using `SELECT * FROM employees`. The next step was using a **`WHERE`** clause with a **`NOT`** filter which allowed me to filter out employees who are not part of the Information Technology department.

Summary

To obtain specific details about login attempts and employee machines, I utilized filters in SQL queries. I worked with two tables, namely **`log_in_attempts`** and **`employees`**. By employing the operators **`AND`**, **`OR`**, and **`NOT`**, I could refine my search to extract the desired information for each task. Subsequently, I was able to use the asterisk (`*`) wildcard to select all data from columns within a table. Lastly, I made use of the **`LIKE`** operator along with the percentage sign (`%`) wildcard to uncover specific patterns within the data.