Apply filters to SQL queries

Project description

The investigation team needs data to investigate potential security issues and update computers. This includes gathering specific information about employees, their machines, and the departments they belong to from the database.

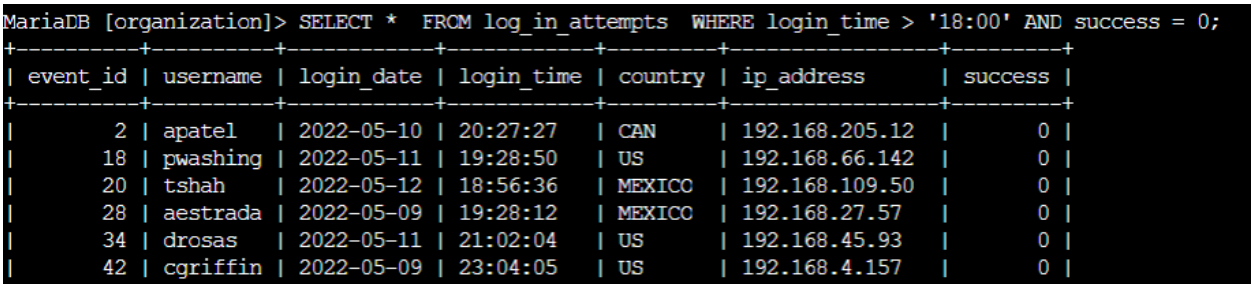
I am responsible for filtering the required information from the database, ensuring that only relevant and accurate data is retrieved. This involves designing and executing SQL queries to extract the necessary details.

Next, I provide examples 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 (after 18:00). All after-hours login attempts that failed needed to be investigated. The following code demonstrates how I created a SQL query to filter for failed login attempts that occurred after business hours.

```
SELECT * FROM log_in_attempts
WHERE login_time > "18:00" AND success = 0;
```



```
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 |
```

The first part of the screenshot is my query, and the second part is a portion of the output. This query filters for failed login attempts that occurred after 18:00.

First, I started by selecting all data from the `log_in_attempts` table. Then, I used a `WHERE` clause with an `AND` operator to filter my results to output only login attempts that occurred after 18:00 and were unsuccessful. The first condition is `login_time > '18:00'`, which filters for login attempts that occurred after 18:00. The second condition is `success = 0`, which filters for failed login attempts.

Retrieve login attempts on specific dates

An unusual event was detected on 2022-05-09. To thoroughly investigate this, it's necessary to examine any login activity that took place on 2022-05-09 as well as the day before.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred on specific dates.

```
SELECT * FROM log_in_attempts
WHERE login_date = '2022-05-09' OR login_date = '2022-05.08';
```

```
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 |       1 |
|       25 | sbaelish | 2022-05-09 | 07:04:02   | US      | 192.168.33.137  |       1 |
|       26 | apatel   | 2022-05-08 | 17:27:00   | CANADA  | 192.168.123.105 |       1 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57   |       0 |
|       30 | yappiah  | 2022-05-09 | 03:22:22   | MEX     | 192.168.124.48  |       1 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|       36 | asundara | 2022-05-08 | 09:00:42   | US      | 192.168.78.151  |       1 |
```

The first part of the screenshot shows my query, and the second part displays a portion of the output. This query retrieves all login attempts that took place on 2022-05-09 or 2022-05-08.

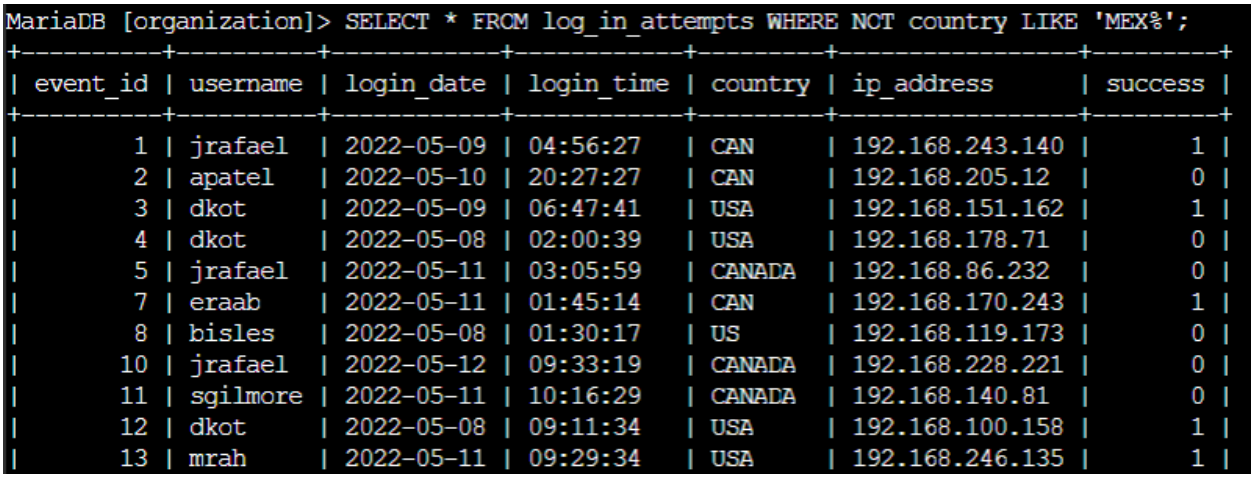
I began by selecting all data from the `log_in_attempts` table. Then, I applied a `WHERE` clause with an `OR` operator to filter the results, focusing only on login attempts that happened on either 2022-05-09 or 2022-05-08. The first condition, `login_date = '2022-05-09'`, isolates logins from 2022-05-09, while the second condition, `login_date = '2022-05-08'`, captures logins from 2022-05-08.

Retrieve login attempts outside of Mexico

After analyzing the organization's data on login attempts, I identified a potential issue with attempts made from locations outside of Mexico. These login attempts warrant further investigation.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred outside of Mexico.

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



```
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 |
|       13 | mrah     | 2022-05-11 | 09:29:34   | USA     | 192.168.246.135 |       1 |
```

The first part of the screenshot shows my query, and the second part displays a portion of the output. This query retrieves all login attempts made in countries other than Mexico.

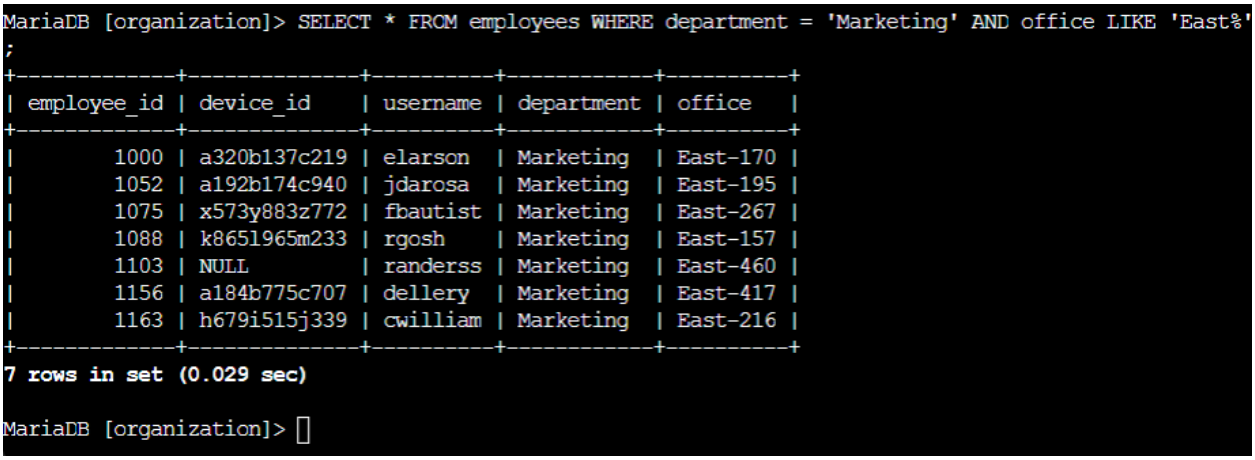
I began by selecting all data from the `log_in_attempts` table. Then, I applied a `WHERE` clause combined with `NOT` to exclude login attempts from Mexico. To ensure accurate filtering, I used `LIKE` with the pattern `MEX%`, as the dataset records Mexico

as both `MEX` and `MEXICO`. The percentage sign (%) acts as a wildcard, allowing the query to match any number of unspecified characters following "MEX."

Retrieve employees in Marketing

My team needs to update computers for employees working in the Marketing department, specifically those located in the East building. To determine which employee machines need updating, I created the following SQL query to retrieve the relevant information.

```
SELECT * FROM employees
WHERE department = 'Marketing' AND office LIKE 'East%';
```



```
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 (0.029 sec)

MariaDB [organization]>
```

The first part of the screenshot shows my query, and the second part displays a portion of the output. This query filters for all employees in the Marketing department who are stationed in the East building.

I began by selecting all records from the `employees` table. I then applied a `WHERE` clause with an `AND` operator to narrow down the results. The first condition, `department = 'Marketing'`, filters for employees in the Marketing department. The second condition, `office LIKE 'East%'`, targets those working in the East building. The `LIKE` operator with the pattern `East%` ensures that any office number starting with "East" is included, matching the data format used in the `office` column.

Retrieve employees in Finance or Sales

The machines for employees in the Finance and Sales departments also need to be updated. Since different security updates are required for these departments, I need to extract information about employees from both departments.

The following code demonstrates how I created a SQL query to filter for employee machines from the Finance or Sales departments.

```
SELECT * FROM employees
WHERE department = 'Finance' OR department = 'Sales';
```

```
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   |
|        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    |
```

The first part of the screenshot shows my query, and the second part displays a portion of the output. This query retrieves all employees from the Finance and Sales departments.

I started by selecting all records from the `employees` table. Then, I used a `WHERE` clause with an `OR` operator to filter for employees who belong to either the Finance or Sales departments. The `OR` operator ensures that employees from both departments are included in the results. The first condition, `department = 'Finance'`, filters for employees in the Finance department. The second condition, `department = 'Sales'`, filters for those in the Sales department. This approach allows me to gather all relevant employee data for the required security updates.

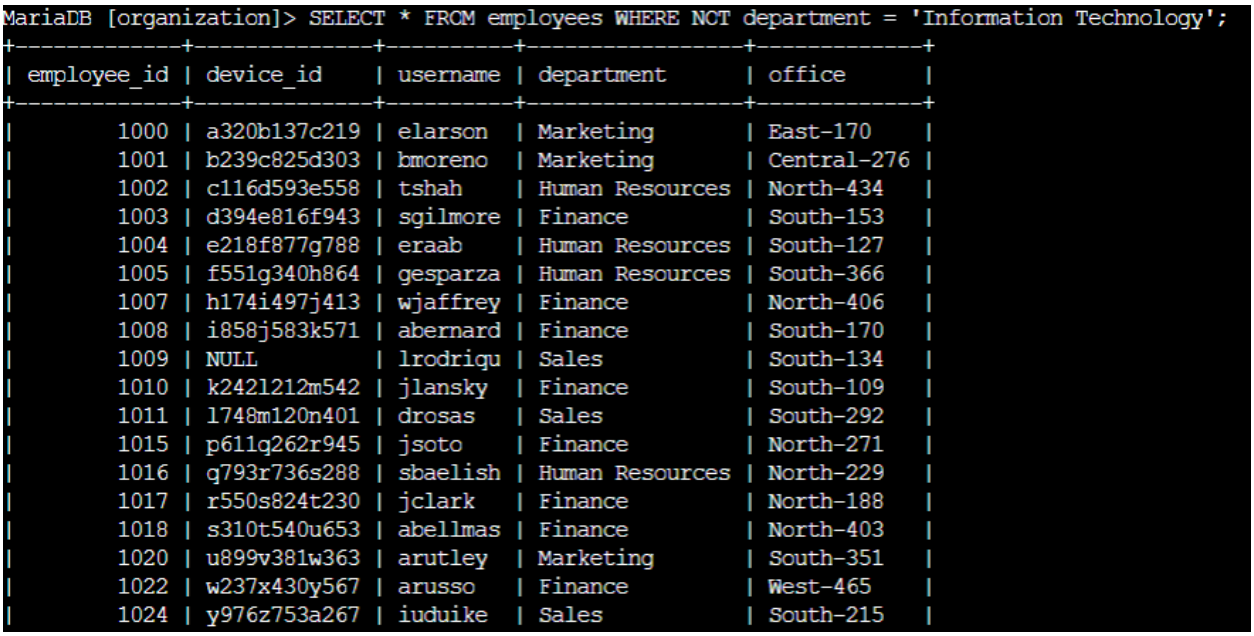
Retrieve all employees not in IT

My team needs to perform one final security update for employees who are not in the Information Technology department. To facilitate this update, I first need to

gather information on these employees.

The following code demonstrates how I created a SQL query to filter for employee machines from employees who are not in the Information Technology department.

```
SELECT * FROM employees
WHERE NOT department = 'Information Technology';
```



```
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   |
|        1011 | l748m120n401 | drosas   | Sales           | South-292   |
|        1015 | p611q262r945 | jsoto    | Finance         | North-271   |
|        1016 | q793r736s288 | sbaelish | Human Resources | North-229   |
|        1017 | r550s824t230 | jclark   | Finance         | North-188   |
|        1018 | s310t540u653 | abellmas | Finance         | North-403   |
|        1020 | u899v381w363 | arutley  | Marketing       | South-351   |
|        1022 | w237x430y567 | arusso   | Finance         | West-465    |
|        1024 | y976z753a267 | iuduike  | Sales           | South-215   |
```

The first part of the screenshot shows my query, and the second part displays a portion of the output. This query retrieves all employees who are not in the Information Technology department.

I began by selecting all records from the `employees` table. I then applied a `WHERE` clause with the `NOT` operator to exclude employees from the Information Technology department. This approach allows me to identify all relevant employees for the final security update.

Summary

I applied filters to SQL queries to extract specific information related to login attempts and employee machines. To achieve this, I utilized two different tables: `log_in_attempts` and `employees`.

I employed various SQL operators, including `AND`, `OR`, and `NOT`, to refine the queries and retrieve the exact information required for each task. Additionally, I used the `LIKE` operator along with the percentage sign (%) wildcard to filter data based on patterns and partial matches. This approach allowed me to efficiently gather and analyze data relevant to security and operational updates.