# Apply filters to SQL queries

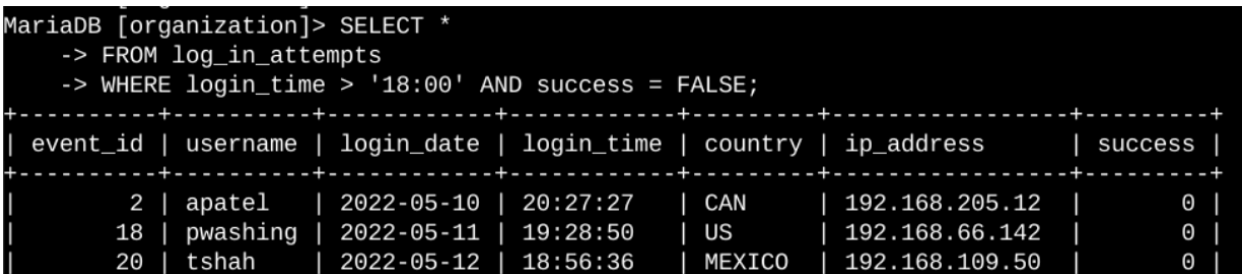
## Project description

The organization I work for is making their system more secure. It is my job to make sure that the system is secure, check out all potential security issues, and keep employee computers up to date. The following are examples of steps that I have taken to perform security-related tasks using SQL filters.

## Retrieve after hours failed login attempts

I noticed that a potential security incident could have occurred after 18:00 (ie. business hours). Therefore, all login attempts that were made after closing hours and failed need to be investigated.

The following is a screenshot showing a SQL query that filters for failed login attempts occurring after 18:00 (see first 3 lines) and a section of the table that gets outputted.



```
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 |
```

I started this SQL query by first selecting all of the data from the `log_in_attempts` table. Then, I filtered the results using a `WHERE` clause with an `AND` operator in order to only output the  unsuccessful login attempts that occurred after 18:00. The first condition in the aforementioned clause was `login_time > '18:00'`, which filtered for login attempts after 18:00. The second condition was `success = FALSE`, which filtered for the failed login attempts.

## Retrieve login attempts on specific dates

Based on the findings from investigating failed after-hour logging attempts, I noticed that a suspicious event occurred on 2022-05-09. Therefore, any login activity that happened either on 2022-05-09 or the day before needs to be investigated.

The following is a screenshot showing a SQL query that filters for certain login attempts that occurred on either 2022-05-09 or 2022-05-08 (the dates of interest). The first part of the screenshot was my query, and the second part was a portion of the returned output, which was all of the login attempts that occurred on 2022-05-09 or 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 |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

I started this SQL query by first selecting all of the data from the `log_in_attempts` table. Then, I filtered the results using a `WHERE` clause along with an `OR` operator in order to only output the  previously stated `login_date` parameters.

## Retrieve login attempts outside of Mexico

Based on the findings from investigating the failed after-hour logging attempts on the previously specified dates, I noticed that there was specifically an issue with the login attempts occurring in countries that are not Mexico. Therefore, these login attempts need to be investigated.

The following is a screenshot showing a SQL query that filters for certain login attempts that occurred outside of Mexico. The first part of the screenshot was my query while the second part was a portion of the returned output, which was all of the login attempts, except for the ones that happened in 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 |       0 |
|        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 |       0 |
```

I started this SQL query by first selecting all of the data from the `log_in_attempts` table. Then, I filtered the results using a `WHERE` clause along with a `NOT` operator in order to only output the data entries that did not arise in Mexico. Since the dataset represents

Mexico as `MEX` and `MEXICO`, I used `LIKE` with `MEX%` as the string pattern to match. The addition of the percentage sign (`%`) in this condition allows for the selection of any number of unspecified characters so long as the desired string pattern is present.

## Retrieve employees in Marketing

My team has noticed that certain employees in the Marketing department need to have their computers updated. As a result, we need to identify all of the employee machines in the Marketing department that need to be updated.

The following is a screenshot showing a SQL query that filters for certain employee machines that belong to the Marketing department in the East building. The first part of the screenshot was my query while the second part was a portion of the returned output, which was all of the employees that work in the East building's Marketing department.

```
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 |
```

I started this SQL query by first selecting all of the data from the `employees` table. Then, I filtered the results using a `WHERE` clause along with the `AND` operator in order to only output the data entries that matched two conditions. The first condition filters for the employees in the Marketing department using the `department = "Marketing"` section of the query. The second condition filters for the employees that are located in the East building using the `LIKE East%` section of the query. The reason why `East%` is used as the pattern to match is because the data in the office column represents the East building with a specific office number.

## Retrieve employees in Finance or Sales

My team has also noticed that certain employees in the Finance and Sales department need to have a separate update conducted on their computers. As a result, we need to

identify all of the employee machines in specifically these two departments that need to be updated.

The following is a screenshot showing a SQL query that filters for certain employee machines that belong to either the Finance or Sales departments. The first part of the screenshot was my query while the second part was a portion of the returned output, which was all of the data from the `employees` table.

```
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   |
```

I started this SQL query by first selecting all of the data from the `employees` table. Then, I filtered the results using a `WHERE` clause along with the `OR` operator in order to only output the workers from either department. The first condition filters for the employees in the Finance department using the `department = “Finance”` section of the query. The second condition filters for the employees in the Sales department using the `department = “Sales”` section of the query.

## Retrieve all employees not in IT

An additional security update needs to be conducted for employee computers that are not in the Information Technology (IT) department. As a result, we need to identify all of the employee machines in non-IT departments that need to be updated.

The following is a screenshot showing a SQL query that filters for the employee machines that aren’t in the IT department. The first part of the screenshot was my query while the second part was a portion of the returned output, which was all of the data from the `employees` table that wasn’t associated with the IT 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   |
```

I started this SQL query by first selecting all of the data from the `employees` table. Then, I filtered the results using a `WHERE` clause along with the `NOT` operator in order to only output the workers that weren't in the IT department.

## Summary

In conclusion, I used SQL filter queries in order to obtain certain information on login attempts and identify the employee machines that require updates, which I applied to the `log_in_attempts` and `employees` tables, respectively. To make these queries possible, I applied such operators as `AND`, `OR`, and `NOT` in order to filter for the information needed in that given task. When I needed to filter for certain string patterns, I applied the `LIKE` condition along with the percentage sign (`%`), which acted as the wildcard at the end of the desired pattern to be matched.