Apply filters to SQL queries

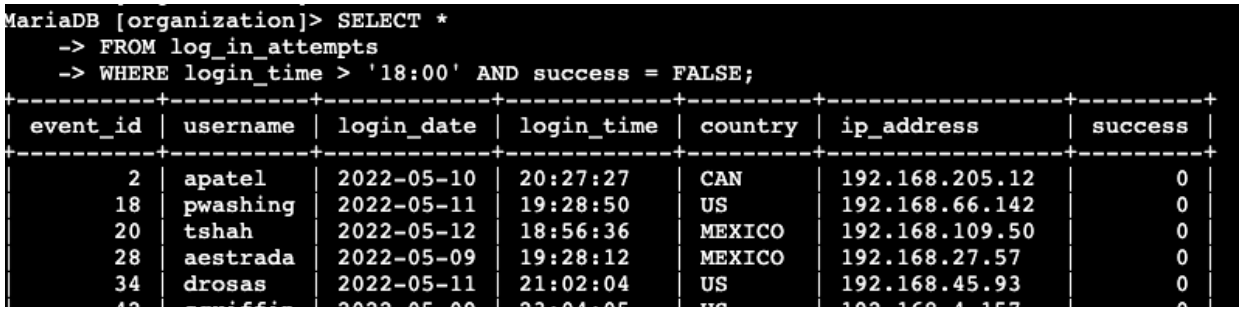
Project description

I am working as part of the security team of a large company. My current assignment is to investigate some suspicious login attempts. Furthermore, I was asked to identify employees belonging to specified department who require their machines to be updated.
The task required to query the company's database using SQL in order to retrieve the information needed and complete the investigation.

Retrieve after hours failed login attempts

In this example, I was asked to investigate any failed login attempts that happened after working hours (later than 18:00).

In order to carry out the investigation, I queried the log using the following syntax:



```
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 |
```

The first part of the screenshot includes a query that filters for failed login attempts occurred after 18:00:

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

I started by selecting all data belonging to the `log_in_attempts` table. Then I used a `WHERE` clause with a first condition `login_time > '18:00'`, requesting to filter for the login attempts that occurred after 18:00. This is followed by an `AND` operator with the second condition `success = FALSE` which asks to filter for failed attempts only.
The second part of the screenshot includes a quick view of the output obtained.

Retrieve login attempts on specific dates

I was asked to investigate any login activity that happened on 2022-05-09 or on the day before due to a suspicious event that took place on 2022-05-09.

```
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 |
```

In order to examine any suspicious login attempt involving two specific dates I used the following syntax (also visible in the first part of the screenshot):

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

I started by selecting all data belonging to the `log_in_attempts` table. Then I asked SQL to find login dates that occurred either on `'2022-05-09'` or on `'2022-05-08'`.
To do this, I used a `WHERE` clause with a first condition `login_date = '2022-05-09'`, which filters for logins on 2022-05-09. Then I employed the operator `OR` followed by the second condition `login_date = '2022-05-08'`, to filter for any login attempts that occurred on 2022-05-08.
The second part of the screenshot includes a quick view of the output obtained.

Retrieve login attempts outside of Mexico

Some suspicious login activities were detected. However, the login attempts are almost certainly not coming from Mexico. For this reason, I was asked to investigate login attempts from locations that are not 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 |
```

To complete the investigation, I used the below query (also visible in the first part of the above screenshot):

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

Firstly, I asked SQL to select all data belonging to the `log_in_attempts` table. The rest of the query retrieves any login attempts that were made outside of Mexico. To do this, I used a `WHERE` clause in conjunction with the `NOT` operator to look for countries (included in the `country` column) other than Mexico. This is followed by `LIKE`, which is the operator that allows to look for patterns when combined with wildcards such as `%`. Since the `country` column contains rows including both `Mexico` and `Mex` I felt it was appropriate to use the `%` wildcard following `MEX` in order to exclude both types of results.
The second part of the screenshot shows a snippet of the output.

Retrieve employees in Marketing

My team wanted to perform a security updates on specific machines in the Marketing department. I was responsible for getting information on the machines to update.

```
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)
```

To achieve this I employed the following syntax, also visible in first part of the above screenshot:

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

Firstly, I asked SQL to select all data belonging to the `employees` table. The following line begins with `WHERE` followed by the condition `department = 'Marketing'`, which filters for employees belonging to the marketing department. Then, I included the `AND` operator followed by the condition `office LIKE 'East%'` to specify that I am looking for employees belonging to Marketing that are also working in any office located in the East building. Once again, I found appropriate to use the `%` wildcard in conjunction with `LIKE` because each East building offices are followed by specific identification numbers and I needed to include them all in the final output (visible in the second part of the screenshot).

Retrieve employees in Finance or Sales

My team needed to perform a different security update on machines for employees in the Sales and Finance departments. The below shows how I used filters in SQL to create a query that identifies all employees in the Sales or Finance departments.

```
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   |
```

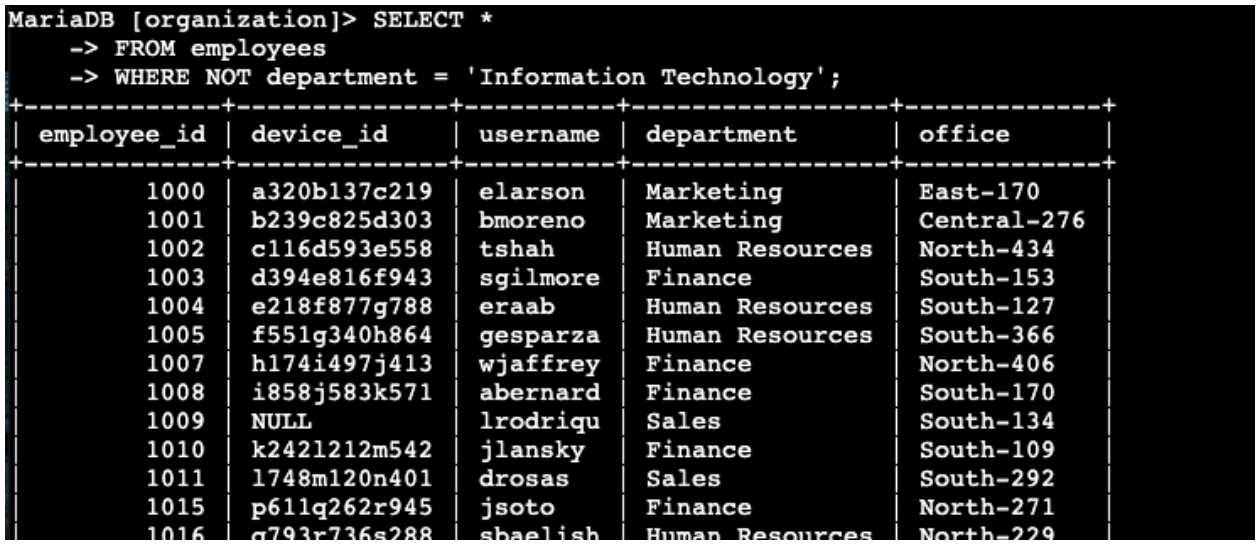
As shown in the screenshot, I completed this task by using the following syntax:

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

The query helped me to retrieve information about employees either belonging to the Finance or Sales team. I started by selecting all data belonging to the `employees` table. Then, I included the `WHERE` clause in conjunction with the `OR` operator to filter through the `department` column and retrieve the data needed. In this case, I user `OR` because I was interested in retrieving information about employees belonging to either department. The two conditions allowed me to be specific on the department that I needed to investigate.

Retrieve all employees not in IT

My team needed to perform a security update on machines who did not belong to employees from the Information Technology department. To make the update, I first had to get information on these employees.



```
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   |
```

The following demonstrates how I used SQL to retrieve information regarding employees in departments that were not Information Technology:

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

After selecting all data included in the `employees` table, I used `WHERE NOT` followed by the condition `department = 'Information Technology'` to get an inclusive list of employees not belonging to the Information Technology department. A snippet of the output can be seen in the second part of the above screenshot.

Project summary

This project has allowed me to showcase my skills with SQL, in particular the use of advanced filters. This involves the correct use of operators such as `AND`, `OR` and `NOT` based on the specific information that needs to be mined from the database. I also demonstrated how to use `LIKE` and the percentage sign (`%`) wildcard to filter for patterns.