Apply filters to SQL queries

Project Description

In this project, I took on the role of a security professional at a large organization. As part of my job, I investigated security issues in order to strengthen the system's security. My recent task was to explore potential security threats related to login attempts and employee machines.

As a security specialist, I examined the organization's data in the `employees` and `log_in_attempts` tables using SQL filters to retrieve records from different datasets and to identify potential security risks.

Retrieve After-Hours Failed Login Attempts

This time, my colleagues discovered suspicious activity based on login attempts that occurred outside of working hours. I wanted to collect information about every attempt made after 6:00 PM.

To retrieve details such as the number, time, location, and other important data related to employee login attempts in my company, I first used an SQL filter with the logical operator `AND`. This operator helps a security analyst define specific conditions in a query and obtain more accurate results from the log files.

```
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 (0.029 sec)
```

At the screenshot displayed the SQL command:

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

This command retrieves all records from the `log_in_attempts` table that contain information about login attempts made outside of working hours.
Typing `SELECT *` means that we are querying the database and the asterisk (`*`) indicates that all columns will be included in the result.
The `FROM` clause specifies the exact table — `log_in_attempts` — where the data to be analyzed is stored.
The next line starts with `WHERE`, which introduces a condition. I used the logical operator `AND`, which means that **both conditions must be true at the same time**. This operator connects two conditions:

- The first condition is `login_time > '18:00'`, which selects only the entries that occurred after 6:00 PM.
- The second condition is `success = FALSE`, which filters the results to include only unsuccessful login attempts.

It's also important to end the SQL statement with a semicolon (`;`).
As a result, the query returns a list of full information about who attempted to access the system, how many times, from which location, and at what time.

Retrieve Login Attempts on Specific Dates

On May 9, 2022, an event occurred that attracted the attention of my team. It involved suspicious activity that required closer examination.

```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE login_date = '2022-05-08' OR login_date = '2022-05-09';
+----------+----------+------------+------------+---------+-----------------+--------
-+
| 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
```

Using SQL commands, I retrieved a table with all entries from May 8, 2022 and May 9, 2022.
At this step, I used the following query:

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

The logical operator `OR` was useful here because I wanted to include all records that match either one condition or the other. It was not necessary for both conditions to be met at the same time.
The condition was applied to the `login_date` column in the `log_in_attempts` table, which stores information about the dates of all login attempts.
The logical operator `OR` combines two comparisons. With the help of the `=` operator, I specified the exact dates that should be included in the results.

Retrieve Login Attempts Outside of Mexico

I used the `NOT LIKE` condition to exclude all login attempts from Mexico, as the `country` field could

contain both `'MEX'` and `'MEXICO'`. The pattern `'MEX%'` allowed me to match both forms efficiently.

```
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
 |
|       14 | sbaelish | 2022-05-10 | 10:20:18   | US      | 192.168.16.99   |       1
 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0
 |
|       16 | mcouliba | 2022-05-11 | 06:44:22   | CAN     | 192.168.172.189 |       1
```

The screenshot above shows how I retrieved a table with all entries from the `log_in_attempts` table, focusing on the `country` column — while excluding entries that match the `'MEX'` or `'MEXICO'` patterns.

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

In the `WHERE` clause, I used the logical operators `NOT` and `LIKE` to define the condition for excluding specific patterns.
The condition is applied to the country column to ensure that entries from Mexico are not included in the results. The pattern `'MEX%'` is used to match any value starting with `'MEX'`, including both `'MEX'` and `'MEXICO'`.

Retrieve Employees in Marketing

Identifying all employees located in offices within the East building (e.g., `East-170`, `East-320`) is essential to ensure accurate updates of machines used by staff in the 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 |
|        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)

MariaDB [organization]>
```

The following SQL query was used to retrieve information from the `employees` table:

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

The code above selects all columns and applies filters to the `department` and `office` columns to return only the relevant records.

Retrieve Employees in Finance or Sales

To perform a separate update on the computers of all employees in the `'Finance'` or `'Sales'` department, I first needed to locate information about these employees.

To accomplish this, I wrote a SQL query to retrieve records for all employees whose department is either `'Finance'` or `'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    |
|        1035 | j236k3031245 | bisles   | Sales      | South-171   |
|        1039 | n253o917p623 | cjackson | Sales      | East-378    |
|        1041 | p929q222r778 | cgriffin | Sales      | North-208   |
|        1044 | s429t157u159 | tbarnes  | Finance    | West-415    |
|        1045 | t567u844v434 | pwashing | Finance    | East-115    |
|        1046 | u429v921w138 | daquino  | Finance    | West-280    |
|        1047 | v109w587x644 | cward    | Finance    | West-373    |
|        1048 | w167x592y375 | tmitchel | Finance    | South-288   |
|        1049 | NULL         | jreckley | Finance    | Central-295 |
|        1050 | y132z930a114 | csimmons | Finance    | North-468   |
|        1057 | f370g535h632 | mscott   | Sales      | South-270   |
|        1062 | k3671639m697 | redwards | Finance    | North-180   |
|        1063 | l686m140n569 | lpope    | Sales      | East-226    |
|        1066 | o678p794q957 | ttyrell  | Sales      | Central-444 |
|        1069 | NULL         | jpark    | Finance    | East-110    |
|        1071 | t244u829v723 | zdutchma | Sales      | West-348    |
```

The following code retrieves information from the `employees` table to help analyze the situation in the `'Finance'` and `'Sales'` departments:

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

The logical operator `OR`, along with the `=` operator, is used to set two conditions. This allows the query to return all records where the value in the `department` column is either `'Finance'` or `'Sales'`.

Retrieve All Employees Not in IT

There is one more update that my team needs to carry out on employee machines.
Previously, this work was already completed for employees in the IT department.

Now, I need to retrieve information about all employees except those in the Information Technology department.

```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE NOT department LIKE '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   |
|        1025 | z381a365b233 | jhill    | Sales           | North-115   |
|        1026 | a998b568c863 | apatel   | Human Resources | West-320    |
|        1027 | b806c503d354 | mrah     | Marketing       | West-246    |
|        1028 | c603d749e374 | aestrada | Human Resources | West-121    |
|        1029 | d336e475f676 | ivelasco | Finance         | East-156    |
|        1030 | e391f189g913 | mabadi   | Marketing       | West-375    |
|        1031 | f419g188h578 | dkot     | Marketing       | West-408    |
|        1034 | i679j565k940 | bsand    | Human Resources | East-484    |
|        1035 | j236k3031245 | bisles   | Sales           | South-171   |
|        1036 | k5501533m205 | rjensen  | Marketing       | Central-239 |
|        1038 | m873n636o225 | btang    | Human Resources | Central-260 |
|        1039 | n253o917p623 | cjackson | Sales           | East-378    |
```

This task can be completed using the `NOT` and `LIKE` operators.
In the screenshot above, the following query is shown:

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

It is important to remember that string values must be enclosed in quotation marks to avoid errors during code execution, whereas numeric values do not require such formatting.

Summary

In this task, I used logical operators such as `AND`, `OR`, and `NOT`, along with comparison operators like `=`, `>`, and `LIKE` (e.g., `'MEX%'`), to extract relevant data from the `log_in_attempts` and employees tables.
This analysis helped my team investigate suspicious activities and ensure that all employee machines remain up to date.
These measures are essential for proactively securing the company's systems and preventing them from being compromised.