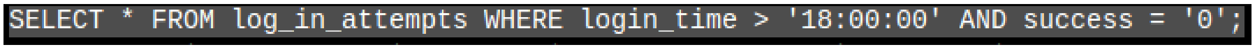
Apply filters to SQL queries

Project description

You are a security professional at a large organization. Part of your job is to investigate security issues to help keep the system secure. You recently discovered some potential security issues that involve login attempts and employee machines.
Your task is to examine the organization's data in their **`employees`** and **`log_in_attempts`** tables. You'll need to use SQL filters to retrieve records from different datasets and investigate the potential security issues.

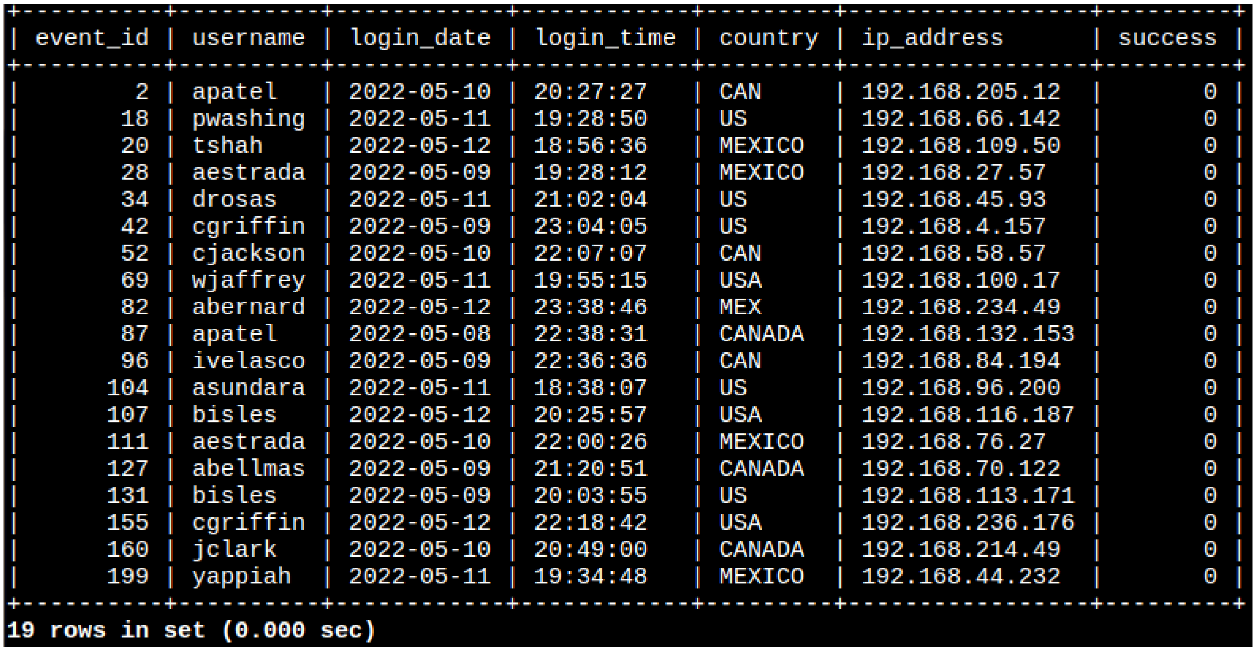
Retrieve after hours failed login attempts

The first task is to retrieve failed login attempts that took place after '**18:00**'. To accomplish this we will use the following command:



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

This tells SQL that we are pulling all columns from the **'log_in_attempts'** failed login attempts that happened after **'18:00:00'** which gives us the result:

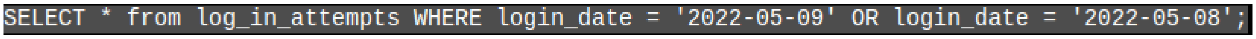


```
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        2 | apatel   | 2022-05-10 | 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 (0.000 sec)
```

As you can see, we have 19 failed login attempts that occurred after 18:00.

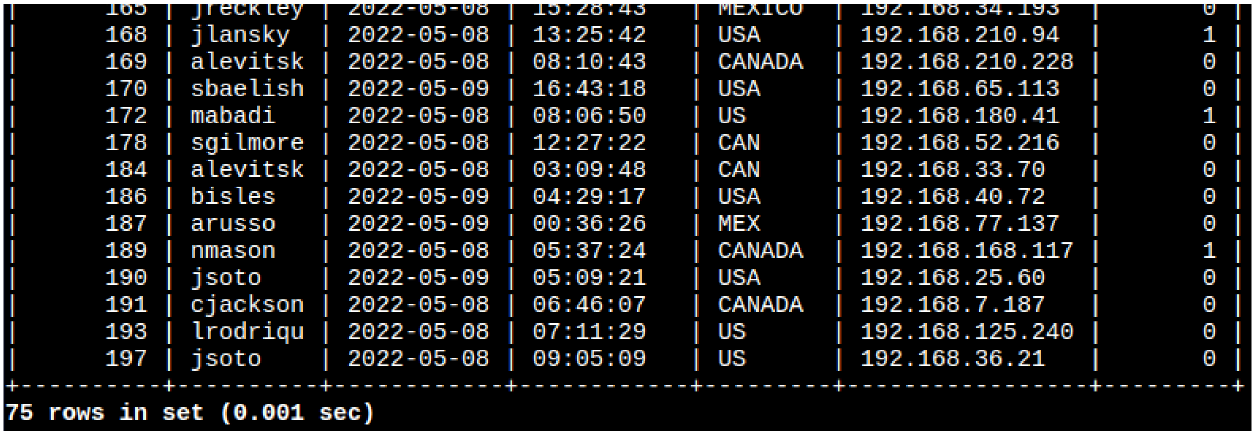
Retrieve login attempts on specific dates

For the second task, we must investigate a suspicious event that occurred on a specific date. The event took place on 2022-05-09, so we want to know what took place on that day, and the day before that for good measure. So we need to pull login attempts on **‘2022-05-09’** and **‘2022-05-08’**. We’re going to be using the following query with filters to get the information we need:



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

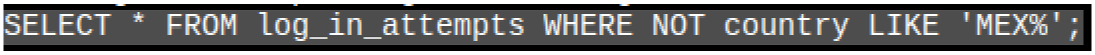
What this is saying is that we’re pulling all columns from the **‘log_in_attempts’**, but we’re filtering the results to only display information from the 2 specific dates that we need. It returns with 75 rows of information:



```
|        165 | jreckley | 2022-05-08 | 15:28:43   | MEXICO  | 192.168.34.193  |       0 |
|        168 | jlansky  | 2022-05-08 | 13:25:42   | USA     | 192.168.210.94  |       1 |
|        169 | alevitsk | 2022-05-08 | 08:10:43   | CANADA  | 192.168.210.228 |       0 |
|        170 | sbaelish | 2022-05-09 | 16:43:18   | USA     | 192.168.65.113  |       0 |
|        172 | mabadi   | 2022-05-08 | 08:06:50   | US      | 192.168.180.41  |       1 |
|        178 | sgilmore | 2022-05-08 | 12:27:22   | CAN     | 192.168.52.216  |       0 |
|        184 | alevitsk | 2022-05-08 | 03:09:48   | CAN     | 192.168.33.70   |       0 |
|        186 | bisles   | 2022-05-09 | 04:29:17   | USA     | 192.168.40.72   |       0 |
|        187 | arusso   | 2022-05-09 | 00:36:26   | MEX     | 192.168.77.137  |       0 |
|        189 | nmason   | 2022-05-08 | 05:37:24   | CANADA  | 192.168.168.117 |       1 |
|        190 | jsoto    | 2022-05-09 | 05:09:21   | USA     | 192.168.25.60   |       0 |
|        191 | cjackson | 2022-05-08 | 06:46:07   | CANADA  | 192.168.7.187   |       0 |
|        193 | lrodriqu | 2022-05-08 | 07:11:29   | US      | 192.168.125.240 |       0 |
|        197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |       0 |
+----------+----------+------------+------------+---------+-----------------+---------+
75 rows in set (0.001 sec)
```

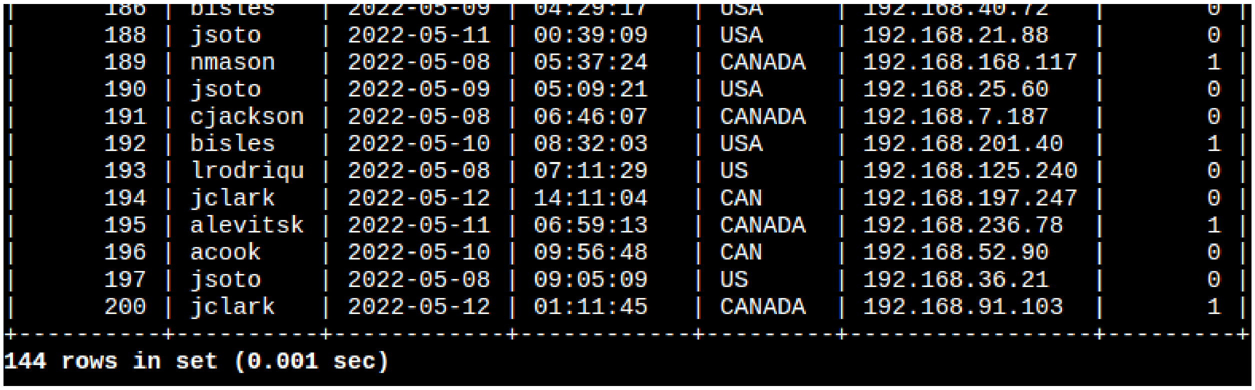
Retrieve login attempts outside of Mexico

We’ve detected suspicious login attempts, but the team has determined that it didn’t originate in Mexico, so we need rows of information that do not include mexico. Mexico is listed in 2 different ways, one as **‘MEX’** and also as **‘MEXICO’** so we need to make sure that we filter out results from both of them. This is a simple matter and can be achieved with use of the wildcard:



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

By entering the country filter as **‘MEX%’** we’re saying that we do not want to include rows where the country value begins with **‘MEX’**. It produces 144 rows of information that do not include entries from **‘MEX’** or **‘MEXICO’**.



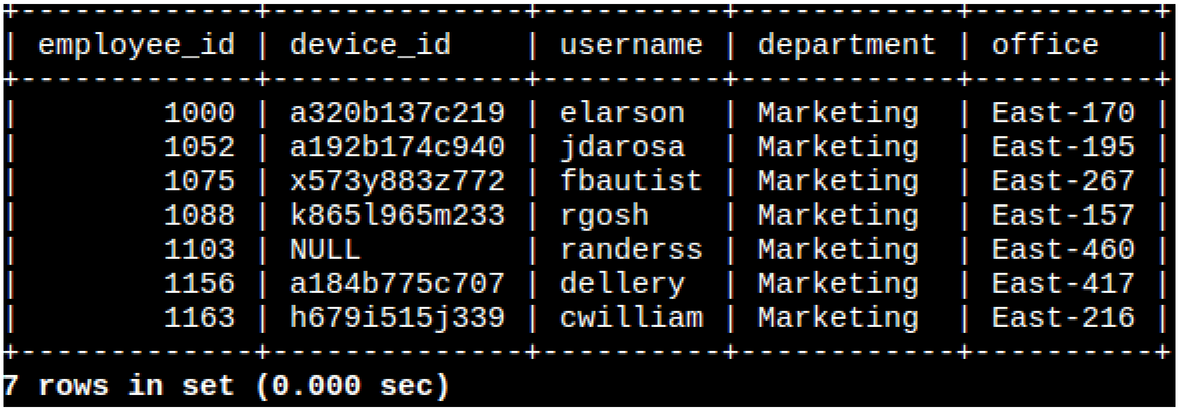
```
|       186 | bisles   | 2022-05-09 | 04:29:17   | USA     | 192.168.40.72   |        0 |
|       188 | jsoto    | 2022-05-11 | 00:39:09   | USA     | 192.168.21.88   |        0 |
|       189 | nmason   | 2022-05-08 | 05:37:24   | CANADA  | 192.168.168.117 |        1 |
|       190 | jsoto    | 2022-05-09 | 05:09:21   | USA     | 192.168.25.60   |        0 |
|       191 | cjackson | 2022-05-08 | 06:46:07   | CANADA  | 192.168.7.187   |        0 |
|       192 | bisles   | 2022-05-10 | 08:32:03   | USA     | 192.168.201.40  |        1 |
|       193 | lrodriqu | 2022-05-08 | 07:11:29   | US      | 192.168.125.240 |        0 |
|       194 | jclark   | 2022-05-12 | 14:11:04   | CAN     | 192.168.197.247 |        0 |
|       195 | alevitsk | 2022-05-11 | 06:59:13   | CANADA  | 192.168.236.78  |        1 |
|       196 | acook    | 2022-05-10 | 09:56:48   | CAN     | 192.168.52.90   |        0 |
|       197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |        0 |
|       200 | jclark   | 2022-05-12 | 01:11:45   | CANADA  | 192.168.91.103  |        1 |
+----------+----------+------------+------------+---------+-----------------+---------+
144 rows in set (0.001 sec)
```

Retrieve employees in Marketing

The team wants to perform updates on specific employee machines in the **'Marketing'** department. We will need a query that identifies all employees in the **'Marketing'** department for all offices in the East building. For this we're going to use the operators **AND** and **LIKE**:

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

This is specifying that we want information from the **'employees'** table, but only if the department matches **'Marketing'** and the office begins with **'East-'**.



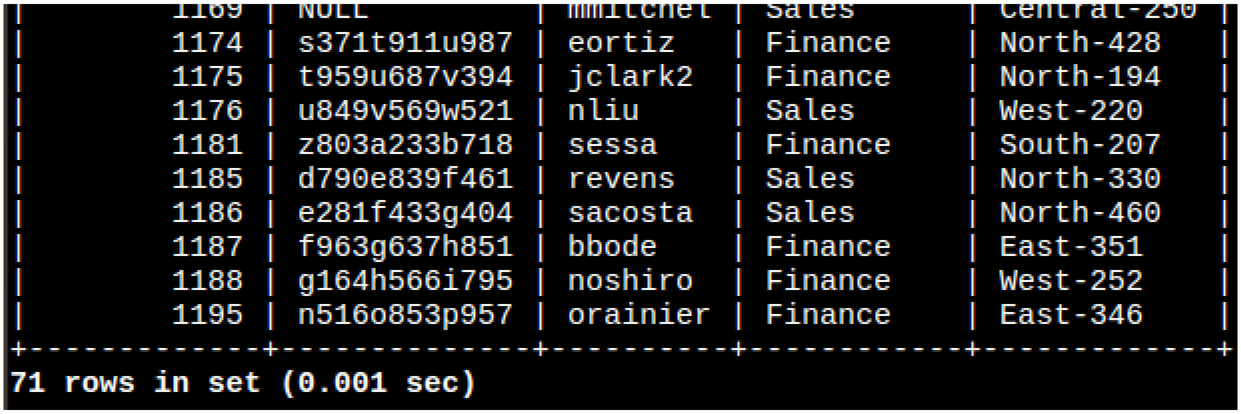
```
+-------------+---------------+----------+------------+----------+
| employee_id | device_id     | username | department | office   |
+-------------+---------------+----------+------------+----------+
|        1000 | a320b137c219  | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940  | 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 (0.000 sec)
```

Retrieve employees in Finance or Sales

Next, we have to perform a security update on machines for employees in the Sales and Finance departments. So we'll need to use filters to create a query that meets this objective for us. For this we're going to use:

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

This is saying that we want employees that are in the departments **'Sales'** OR **'Finance'**.
It results in the 71 rows of information we were looking for.



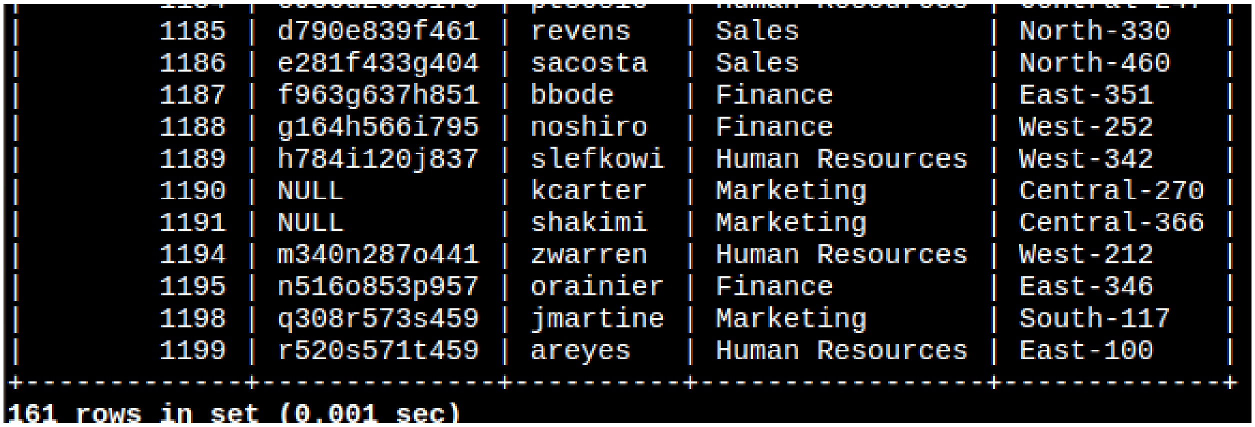
```
|        1169 | NULL         | mmitchel | Sales      | Central-250 |
|        1174 | s371t911u987 | eortiz   | Finance    | North-428   |
|        1175 | t959u687v394 | jclark2  | Finance    | North-194   |
|        1176 | u849v569w521 | nliu     | Sales      | West-220    |
|        1181 | z803a233b718 | sessa    | Finance    | South-207   |
|        1185 | d790e839f461 | revens   | Sales      | North-330   |
|        1186 | e281f433g404 | sacosta  | Sales      | North-460   |
|        1187 | f963g637h851 | bbode    | Finance    | East-351    |
|        1188 | g164h566i795 | noshiro  | Finance    | West-252    |
|        1195 | n516o853p957 | orainier | Finance    | East-346    |
+-------------+--------------+----------+------------+-------------+
71 rows in set (0.001 sec)
```

Retrieve all employees not in IT

In the next scenario, our team needs to make one more update to employee machines. The IT employees have already received the update, so we need to filter them out of the results. We need to pull all employees that are not in **'Information Technology'**. To do this we're going to use this command:

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

What we're telling SQL with this query is that we want all information on the **'employees'** table that is not in the department **'Information Technology'**.



```
|        1185 | d790e839f461 | revens   | Sales           | North-330   |
|        1186 | e281f433g404 | sacosta  | Sales           | North-460   |
|        1187 | f963g637h851 | bbode    | Finance         | East-351    |
|        1188 | g164h566i795 | noshiro  | Finance         | West-252    |
|        1189 | h784i120j837 | slefkowi | Human Resources | West-342    |
|        1190 | NULL         | kcarter  | Marketing       | Central-270 |
|        1191 | NULL         | shakimi  | Marketing       | Central-366 |
|        1194 | m340n287o441 | zwarren  | Human Resources | West-212    |
|        1195 | n516o853p957 | orainier | Finance         | East-346    |
|        1198 | q308r573s459 | jmartine | Marketing       | South-117   |
|        1199 | r520s571t459 | areyes   | Human Resources | East-100    |
+-------------+--------------+----------+-----------------+-------------+
161 rows in set (0.001 sec)
```

Summary

In this exercise we've used various filters and operators like **WHERE**, **AND**, **OR**, **LIKE**, and **NOT** to get all the information we need in a clean and simple format.