Apply filters to SQL queries

Project description

As a security professional working at a large organization, I was given the task to investigate security issues that involve login attempts and employee machines to keep the system secure. I have recently discovered potential security issues that required my knowledge on SQL and SQL filters to retrieve records from different datasets. I then investigated and documented the events with screenshots and commands used throughout my process.

Retrieve after hours failed login attempts

I recently discovered a potential security incident that occurred after business hours. To investigate this, I needed to query the `log_in_attempts` table and review after hours login activity.

```
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 |
|       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 (0.004 sec)
```

As shown in the screenshot above, there are three lines of an SQL query that was used. `SELECT *` was used to select all columns FROM the `log_in_attempts` table WHERE the `login_time` was greater than 6PM (after hours) `AND` the `success` of each login attempt being 0 AKA false/failed. When inputting the SQL query, the output displays 19 rows of failed

login attempts that were made after hours. Each line contains corresponding details such as the username, login date and time, country of origin, and IP address.

Retrieve login attempts on specific dates

My security team has been informed of a suspicious event occurring on `2022-05-09`. To investigate this event, I would need to review all login attempts which occurred on this day and the day before.

```
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
38	sbaelish	2022-05-09	14:40:01	USA	192.168.60.42	1
39	yappiah	2022-05-09	07:56:40	MEXICO	192.168.57.115	1
42	cgriffin	2022-05-09	23:04:05	US	192.168.4.157	0
43	mcouliba	2022-05-08	02:35:34	CANADA	192.168.16.208	0
44	daquino	2022-05-08	07:02:35	CANADA	192.168.168.144	0
47	dkot	2022-05-08	05:06:45	US	192.168.233.24	1
49	asundara	2022-05-08	14:00:01	US	192.168.173.213	0
53	nmason	2022-05-08	11:51:38	CAN	192.168.133.188	1
56	acook	2022-05-08	04:56:30	CAN	192.168.209.130	1
58	ivelasco	2022-05-09	17:20:54	CAN	192.168.57.162	0
61	dtanaka	2022-05-09	09:45:18	USA	192.168.98.221	1
65	aalonso	2022-05-09	23:42:12	MEX	192.168.52.37	1
66	aestrada	2022-05-08	21:58:32	MEX	192.168.67.223	1
67	abernard	2022-05-09	11:53:41	MEX	192.168.118.29	1
68	mrah	2022-05-08	17:16:13	US	192.168.42.248	1
70	tmitchel	2022-05-09	10:55:17	MEXICO	192.168.87.199	1
71	mcouliba	2022-05-09	06:57:42	CAN	192.168.55.169	0
72	alevitsk	2022-05-08	12:09:10	CANADA	192.168.139.176	1
79	abernard	2022-05-09	11:41:15	MEX	192.168.158.170	0

While using filters in SQL, I was able to create a query that identifies all login attempts that occurred on `2022-05-09` or `2022-05-08`.In the screenshots above, I used the `SELECT *` to select all columns `FROM log_in_attempts`. I then used `WHERE` to filter the `login_date` to `2022-05-08 OR 2022-05-09`. The output has shown all

available results within the two dates and included all necessary information helping in this investigation.

Retrieve login attempts outside of Mexico

Moving forward, There has been suspicious activity with login attempts that didn't originate from Mexico. The team determined that the investigation must be continued. Now, you need to investigate login attempts that occurred outside of Mexico. For this investigation, I have used filters again in SQL to create a query that identifies all login attempts that occurred outside of Mexico.

```
MariaDB [organization]>     SELECT *
    ->
    ->     FROM log_in_attempts
    ->
    ->     WHERE NOT country LIKE 'MEX%';
```

The same 2 lines were used from previous queries (`SELECT *` & `FROM`) however, this time, `WHERE NOT` was used to identify all logins outside of Mexico. The `NOT` operator was used to display results that did not include the following operator and wildcard. By using the LIKE operator along with the `%` wildcard, we were able to mitigate the chances of the country column in the table to potentially have values `LIKE` "`MEX`" and "`MEXICO`". The addition of the `%` wildcard gives us the opportunity to identify multiple discrepancies within our results.

Retrieve employees in Marketing

The security team wanted to perform security updates on specific employee machines in the Marketing department. My responsibility included: Getting information on these employee machines and querying the **`employees`** table using filters in SQL. Additionally, I was to create a query that identifies all employees in the Marketing department for all offices in the East building.

```
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)
```

Using the `AND` and `LIKE` operators, I was able to filter the employees table to find all employees that were assigned to the “Marketing” department and were in the east office building. The office could contain many different values such as. Because we have different parameters for `EAST` (such as: **`East-170`**, **`East-320`**) we would have to use `%`.

Retrieve employees in Finance or Sales

My team was tasked to perform a different security update on machines for employees in the Sales and Finance departments. In continuation, I was able to use 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      |
+-------------+-------------+----------+------------+-------------+
|        1009 | NULL        | lrodriqu | Sales      | South-134   |
|        1011 | l748m120n401 | drosas   | Sales      | South-292   |
|        1024 | y976z753a267 | iuduike  | Sales      | South-215   |
|        1025 | z381a365b233 | jhill    | Sales      | North-115   |
|        1035 | j236k3031245 | bisles   | Sales      | South-171   |
|        1039 | n253o917p623 | cjackson | Sales      | East-378    |
|        1041 | p929q222r778 | cgriffin | Sales      | North-208   |
|        1057 | f370g535h632 | mscott   | Sales      | South-270   |
|        1063 | l686m140n569 | lpope    | Sales      | East-226    |
|        1066 | o678p794q957 | ttyrell  | Sales      | Central-444 |
|        1071 | t244u829v723 | zdutchma | Sales      | West-348    |
|        1072 | u905v920w694 | esmith   | Sales      | East-421    |
|        1078 | a667b270c984 | sharley  | Sales      | North-418   |
|        1085 | h339i498j269 | cperez   | Sales      | East-325    |
|        1086 | i281j129k749 | lmajumda | Sales      | West-499    |
|        1089 | l358m929n154 | jpark2   | Sales      | West-251    |
|        1091 | n378o313p469 | rtran    | Sales      | Central-230 |
|        1092 | o391p779q935 | lpark    | Sales      | West-227    |
|        1098 | u671v146w618 | tarchamb | Sales      | North-423   |
|        1107 | d168e758f876 | akajwara | Sales      | North-471   |
|        1109 | f229g533h679 | nlocklea | Sales      | East-196    |
|        1110 | g567h376i314 | pchaudhu | Sales      | Central-428 |
|        1111 | h835i179j862 | jlee     | Sales      | West-309    |
|        1116 | m272n572o874 | nzhao    | Sales      | South-100   |
|        1117 | n683o758p820 | dahmad   | Sales      | West-405    |
|        1118 | o305p208q337 | jpark3   | Sales      | South-329   |
|        1119 | p164q780r999 | omubarak | Sales      | West-409    |
|        1121 | r628s557t397 | mrojas   | Sales      | East-288    |
|        1130 | a317b635c465 | tsnow    | Sales      | Central-451 |
|        1169 | NULL        | mmitchel | Sales      | Central-250 |
|        1176 | u849v569w521 | nliu     | Sales      | West-220    |
|        1185 | d790e839f461 | revens   | Sales      | North-330   |
|        1186 | e281f433g404 | sacosta  | Sales      | North-460   |
+-------------+-------------+----------+------------+-------------+
33 rows in set, 400 warnings (0.001 sec)
```

In the screenshot above, I was able to use the `OR` operator, in conjunction with the previous SQL commands, to filter the department column for all employees in `Finance` or `Sales`.

Retrieve all employees not in IT

Finally, my team needs to make one more update to employee machines. The employees who are in the Information Technology department already had an update, but employees in all other departments need it.

```
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

In the screenshot above, I was able to use a SQL query to filter the `department` using the `NOT` operator. This allowed me to find all rows where the department was not "`Information Technology`" which allows my team to update all corresponding employee machines.

Summary

All tasks performed demonstrated the use of SQL queries and filters to help complete given tasks. Using operators such as `WHERE, LIKE. AND, NOT` and `%`, I was able to filter through multiple records aiding me in identifying login attempts based on time and date, failed login attempts, logins from other countries, and employee machines from different departments. In this project, I was able to demonstrate the utilization of SQL and its filters to access records from a database efficiently and effectively.