

# Apply filters to SQL queries

Arthur Sahertian

## Project description

As part of my Google Cybersecurity Professional Certificate, a fictional company has tasked our security team with investigating a potential security as well as update their systems. We are required to filter through their database for information that is essential for the completion of these tasks. The following document demonstrates how I utilized specific filters within SQL such as **AND**, **LIKE**, **OR** and **NOT** to refine queries to perform certain security related tasks.

## Retrieve after hours failed login attempts

The company reported there was a potential security breach that may have occurred after business hours. We are tasked with investigating any suspicious activity that may have occurred after work hours. We had to search the database for logins that occurred after work hours, but without SQL filtering, it would be quite difficult. We first selected the **login\_time** column from the **log\_in\_attempts** table. Business hours for the day concludes at approximately 6 PM, therefore we filtered for login attempts that were attempted after 6PM (**> 18:00**) with the **WHERE** operator **AND** also filter for login attempts that failed (**success = 0**). The output is shown in the following screenshot:

```
MariaDB [organization]> clear
MariaDB [organization]> SELECT login_time FROM log_in_attempts WHERE login_time > '18:00' AND success = 0;
+-----+
| login_time |
+-----+
| 20:27:27   |
| 19:28:50   |
| 18:56:36   |
| 19:28:12   |
| 21:02:04   |
| 23:04:05   |
| 22:07:07   |
| 19:55:15   |
| 23:38:46   |
| 22:38:31   |
| 22:36:36   |
| 18:38:07   |
| 20:25:57   |
| 22:00:26   |
| 21:20:51   |
| 20:03:55   |
| 22:18:42   |
| 20:49:00   |
| 19:34:48   |
+-----+
19 rows in set (0.146 sec)
```

The screenshot shows the number of failed login attempts that occurred after business hours. This helps us narrow down when the attack or suspicious behavior may have occurred.

## Retrieve login attempts on specific dates

The organization wanted to investigate suspicious that occurred on the dates of 2022-05-08 and 2022-05-09. They tasked our team with filtering the data in their `log_in_attempts` table and isolating these specific dates in their database. To accomplish this task, I first utilized `SELECT *` to select all columns `FROM` the `log_in_attempts` table. I then used the `WHERE` operator to filter my results and specified that the `login_date = '2022-05-08' OR '2022-05-09'`. With this operator, the results of the output will only show data that was recorded on the specified dates. After executing the command the following output was displayed which is shown in the screenshot below:

```
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

As shown in the screenshot, only entries that occurred on the specific dates are being displayed. This helps tremendously with pinpointing the suspicious behavior that we would like to investigate.

## Retrieve login attempts outside of Mexico

The organization have been receiving complaints from users outside of Mexico regarding the login process. The organization tasked us with investigating this issue. To start, we once again must access the `log_in_attempts` table. There is a slight inconvenience however, since some users who are located in Mexico differed in their input with some putting down MEX or MEXICO. This causes issues when we are trying to find all the users because we cannot use the `=` because it will only display results for either or. To address this, when we use the `WHERE` operator, we will also use `NOT country LIKE 'MEX%'`. The following filter means that the output should display all results that has MEX in the beginning. The `(%)` operator indicates a wildcard, which means it can be any possible letters or numbers. This accounts for both inputs that are MEX and MEXICO. After executing the following command, the output that was displayed is shown in the screenshot below:

```
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

The table only displays entries that are from Canada and US which is exactly the data we need to investigate the login problems.

## Retrieve employees in Marketing

The organization also tasked us with completing security updates for employees, specifically in the marketing department. They also specified that the employees that require the security update will be those who work in the east office. To accomplish this I first have to discover which employees are currently working in the marketing department on the east office by filtering the database. I used `SELECT` with the `*` and drew `FROM` the `employees` table, to see what columns are present in this specific table that could potentially be used to refine my search. After I executed my command, the following output was displayed:

```
MariaDB [organization]> SELECT *
->
-> FROM employees
-> ;
```

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
1006	g329h357i597	alevitsk	Information Technology	East-320
1007	h174i497j413	wjaffrey	Finance	North-406
1008	i858j583k571	abernard	Finance	South-170
1009	NULL	lrodriqu	Sales	South-134
1010	k242l212m542	jlansky	Finance	South-109
1011	l748m120n401	drosas	Sales	South-292

All the employees data showed up which is not ideal for our purpose. However, I noticed that a `department` column that classified what specific department each employees works in. In addition, there is also an `office` column which specifies which office each employee works at. I decided these 2 columns would be the best column to filter since we wanted to isolate employees in the marketing department and specifically the east office. I did this by using the `WHERE` operator and specified that `department = 'Marketing'` then I added `AND office LIKE 'East%'`. After executing the command the following that was displayed is shown in the screenshot below:

```
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 | k865l965m233 | 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)
```

By adding the `AND` operator, I specified that I wanted results that not only had Marketing from the `department` column, but it also must have East in the beginning of their input in the `office` column.

## Retrieve employees in Finance or Sales

The organization informed us that we were required to perform security updates for employee devices that work in finance and sales. To accomplish this, I once again used `SELECT` with the `*` and drew `FROM` the `employees` table. This time when I utilized the `WHERE` operator I specified that `department = 'Finance' OR department = 'Sales'`. This essentially means that the output will only show results that has Finance or Sales as the input for the `department` column. After executing the command, the following output was displayed, shown in the screenshot below:

```

MariaDB [organization]> clear
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	k242l212m542	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	i236k303l245	bisles	Sales	South-171

The output only shows results for employees who are currently working in the finance and sales department. This allows us to efficiently sift through the employees that will require the security update.

## Retrieve all employees not in IT

The final task I was given was to complete a security update for employees who are not in the IT department. To do this, I had to first filter the employees table to only display employees who are not currently working in the Information Technology department. To accomplish this, I once again used `SELECT` with the `*` and drew `FROM` the `employees` table. However, this time I utilized the `NOT` operator and specified that any employees with `department = 'Information Technology'` will not be displayed in the following table. After the command was processed, the output that was displayed is shown in the screenshot below:

```

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	k242l212m542	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

This table shows all employees who are not in the IT department, making it much easier to see which employees are not in the IT department and will need a security update.

## Summary

I gathered information from the `log_in_attempts` and `employees` table and I utilized a variety of different filtering operators such as `WHERE`, `AND`, `OR`, to help me perform my analysis much more efficiently. I also utilized operators such as `LIKE` and `(%)` wildcard to search for specific patterns that appear in the tables.