

## Using SQL joins to obtain the precise data

### Project description

The leadership team at my organization tasked me with assessing potential security risks and implementing necessary updates on employee computers. As a Linux administrator, I utilized SQL queries with filters and JOIN techniques to perform security-related tasks. These efforts involved analysing and integrating data from two interconnected tables, demonstrating my expertise in relational database management and security analysis.

### Match Employees to their Machines

To link employees with their assigned machines, I wrote an SQL query on MariaDB to join two tables: `machines` and `employees`. The query focused on the **intersection of both tables**, ensuring that only rows with matching values were included.

```
clear
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.3.39-MariaDB-0+deb10u2 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [organization]> clear
MariaDB [organization]> clear
MariaDB [organization]> SELECT *
->
-> FROM machines;
```

device_id	operating_system	email_client	OS_patch_date	employee_id
a184b775c707	OS 1	Email Client 1	2021-09-01	1156
a192b174c940	OS 2	Email Client 1	2021-06-01	1052
a305b818c708	OS 3	Email Client 2	2021-06-01	1182
a317b635c465	OS 1	Email Client 2	2021-03-01	1130
a320b137c219	OS 2	Email Client 2	2021-03-01	1000
a398b471c573	OS 3	Email Client 2	2021-12-01	0
a667b270c984	OS 1	Email Client 1	2021-03-01	1078
a821b452c176	OS 2	Email Client 2	2021-12-01	1104
a998b568c863	OS 3	Email Client 1	2021-12-01	1026
b157c491d493	OS 2	Email Client 1	2021-03-01	0

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

employee_id	device_id	username	department	office
1000	a320b137c219	elarsen	Marketing	East-170
1001	b239c825d303	hmoreno	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	lrodriq	Sales	South-134
1010	k242l212m542	jlansky	Finance	South-109
1011	l748m120n401	drosas	Sales	South-292
1012	m756n668o146	nmason	Information Technology	North-160
1013	n205o559p243	zbernal	Information Technology	South-229
1014	NULL	asundara	Information Technology	West-219
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
1019	t815u205v470	mcouliba	Information Technology	North-108
1020	u899v381w363	arutley	Marketing	South-351
1021	v200w121x977	smartell	Information Technology	South-138
1022	w237x430y567	arusso	Finance	West-465
1023	x253y759z103	aalonso	Information Technology	West-393

The SQL query retrieved the following data:

- Username (from the relevant table)
- Operating system (from the relevant table)
- Employee ID (from the `employees` table)

To avoid ambiguity for shared column names, I used the `table.column` format.

The result included 185 rows, each representing a unique username, operating system, and device ID, providing a comprehensive view of the assigned machines.

```
200 rows in set (0.089 sec)
MariaDB [organization]> SELECT *
->
-> FROM machines
->
-> INNER JOIN employees ON machines.device_id = employees.device_id;
```

device_id	operating_system	email_client	OS_patch_date	employee_id	employee_id	device_id	username	department
a320b137c219	OS 2	Email Client 2	2021-03-01	1000	1000	a320b137c219	elarsen	Marketing
East-170	OS 1	Email Client 1	2021-03-01	1001	1001	b239c825d303	hmoreno	Marketing
Central-276	OS 3	Email Client 1	2021-09-01	1002	1002	c116d593e558	tshah	Human Resources
North-434	OS 3	Email Client 2	2021-03-01	1003	1003	d394e816f943	sgilmore	Finance
South-153	OS 2	Email Client 1	2021-09-01	1004	1004	e218f877g788	eraab	Human Resources
South-127	OS 3	Email Client 2	2021-12-01	1005	1005	f551g340h864	gesparza	Human Resources
South-366	OS 1	Email Client 2	2021-06-01	1006	1006	g329h357i597	alevitsk	Information Technology
East-320	OS 2	Email Client 1	2021-03-01	1007	1007	h174i497j413	wjaffrey	Finance
North-406	OS 2	Email Client 2	2021-06-01	1008	1008	i858j583k571	abernard	Finance

```

Central-247 |
d790e839f461 | OS 1 | Email Client 1 | 2021-06-01 | 1185 | 1185 | d790e839f461 | revens | Sales
North-330 |
e281f433g404 | OS 1 | Email Client 2 | 2021-12-01 | 1186 | 1186 | e281f433g404 | sacosta | Sales
North-460 |
f963g637h851 | OS 1 | Email Client 1 | 2021-06-01 | 1187 | 1187 | f963g637h851 | bbode | Finance
East-351 |
q164h566l795 | OS 1 | Email Client 1 | 2021-09-01 | 1188 | 1188 | q164h566l795 | noshiro | Finance
West-252 |
h784i120j837 | OS 3 | Email Client 2 | 2021-12-01 | 1189 | 1189 | h784i120j837 | slefkowi | Human Resources
West-342 |
k570l183m949 | OS 3 | Email Client 1 | 2021-12-01 | 1192 | 1192 | k570l183m949 | rlaghari | Information Technology
East-138 |
l186m618n319 | OS 1 | Email Client 2 | 2021-12-01 | 1193 | 1193 | l186m618n319 | esantiaq | Information Technology
Central-300 |
m340n287o441 | OS 2 | Email Client 2 | 2021-09-01 | 1194 | 1194 | m340n287o441 | zwarren | Human Resources
West-212 |
n516o853p957 | OS 1 | Email Client 1 | 2021-09-01 | 1195 | 1195 | n516o853p957 | orainier | Finance
East-346 |
o225p357q829 | OS 3 | Email Client 1 | 2021-12-01 | 1196 | 1196 | o225p357q829 | sshah2 | Information Technology
South-385 |
p791q114r509 | OS 2 | Email Client 1 | 2021-09-01 | 1197 | 1197 | p791q114r509 | aabara | Information Technology
North-159 |
q308r573s459 | OS 3 | Email Client 1 | 2021-03-01 | 1198 | 1198 | q308r573s459 | jmartine | Marketing
South-117 |
r520s571t459 | OS 2 | Email Client 2 | 2021-03-01 | 1199 | 1199 | r520s571t459 | areyes | Human Resources
East-100 |

```

185 rows in set (0.100 sec)

## Return more data

### LEFT JOIN:

- Ensured that all rows from the `machines` table were included, regardless of whether they had matching entries in the `employees` table.
- Used the `device_id` column as the link between the tables.
- Highlighted all machines, including those not assigned to an employee, by populating unmatched rows from the `employees` table with `NULL` values.

```

200 rows in set (0.003 sec)
MariaDB [organization]> SELECT *
->
-> FROM machines
->
-> LEFT JOIN employees ON machines.device_id = employees.device_id;

```

device_id	office	operating_system	email_client	OS_patch_date	employee_id	employee_id	device_id	username	department
a320b137c219	East-170	OS 2	Email Client 2	2021-03-01	1000	1000	a320b137c219	el Larson	Marketing
b239c825d303	Central-276	OS 1	Email Client 1	2021-03-01	1001	1001	b239c825d303	bmoreno	Marketing
c116d593e558	North-434	OS 3	Email Client 1	2021-09-01	1002	1002	c116d593e558	tshah	Human Resources
d394e816f943	South-153	OS 3	Email Client 2	2021-03-01	1003	1003	d394e816f943	sgillmore	Finance
e218f877g788	South-127	OS 2	Email Client 1	2021-09-01	1004	1004	e218f877g788	eraab	Human Resources
f551g340h864	South-366	OS 3	Email Client 2	2021-12-01	1005	1005	f551g340h864	gesparza	Human Resources
g329h357i597	East-320	OS 1	Email Client 2	2021-06-01	1006	1006	g329h357i597	alevitsk	Information Technology
h174i497j413	North-406	OS 2	Email Client 1	2021-03-01	1007	1007	h174i497j413	wjaffrey	Finance
i858j583k571		OS 2	Email Client 2	2021-06-01	1008	1008	i858j583k571	abernard	Finance

### 1. RIGHT JOIN:

- Ensured that all rows from the `employees` table were included, even if they lacked matching entries in the `machines` table.

- o Used the `device_id` column as the link between the tables.
- o Highlighted all employees, including those without assigned machines, by populating unmatched rows from the `machines` table with `NULL` values.

```

200 rows in set (0.001 sec)
MariaDB [organization]> SELECT *
->
-> FROM machines
->
-> RIGHT JOIN employees ON machines.device_id = employees.device_id;

```

device_id	operating_system	email_client	OS_patch_date	employee_id	employee_id	device_id	username	department
office								
a320b137c219	OS 2	Email Client 2	2021-03-01	1000	1000	a320b137c219	el Larson	Marketing
East-170								
b239c825d303	OS 1	Email Client 1	2021-03-01	1001	1001	b239c825d303	bmoreno	Marketing
Central-276								
c116d593e558	OS 3	Email Client 1	2021-09-01	1002	1002	c116d593e558	tshah	Human Resources
North-434								
d394e816f943	OS 3	Email Client 2	2021-03-01	1003	1003	d394e816f943	sgilmore	Finance
South-153								
e218f877g788	OS 2	Email Client 1	2021-09-01	1004	1004	e218f877g788	eraab	Human Resources
South-127								
f551g340h864	OS 3	Email Client 2	2021-12-01	1005	1005	f551g340h864	gesparza	Human Resources
South-366								
g329h357i597	OS 1	Email Client 2	2021-06-01	1006	1006	g329h357i597	alevitsk	Information Technology
East-320								
h174i497j413	OS 2	Email Client 1	2021-03-01	1007	1007	h174i497j413	wjaffrey	Finance
North-406								

Both **JOIN** queries produced 200 rows each. However, some columns contained `NULL` values, reflecting the absence of corresponding data in the related table.

## Retrieve login attempt data

To further investigate the security incident, I retrieved details of all employees who had made login attempts. This involved performing an **INNER JOIN** on the `employees` and `log_in_attempts` tables, using the `username` column as the link. This query provided a focused dataset of relevant login activities, supporting a deeper analysis of potential security risks.

```

200 rows in set (0.003 sec)
MariaDB [organization]> SELECT *
->
-> FROM employees
->
-> INNER JOIN log_in_attempts ON employees.username = log_in_attempts.username;

```

employee_id	device_id	username	department	office	event_id	username	login_date	login_time	country	ip_address
1032	g773h303i639	jrafael	Information Technology	Central-309	1	jrafael	2022-05-09	04:56:27	CAN	192.168.243.140
1026	a998b568c863	apatel	Human Resources	West-320	2	apatel	2022-05-10	20:27:27	CAN	192.168.205.12
1031	f419g188h578	dkot	Marketing	West-408	3	dkot	2022-05-09	06:47:41	USA	192.168.151.162
1031	f419g188h578	dkot	Marketing	West-408	4	dkot	2022-05-08	02:00:39	USA	192.168.178.71
1032	g773h303i639	jrafael	Information Technology	Central-309	5	jrafael	2022-05-11	03:05:59	CANADA	192.168.86.232
1020	u899v381w363	arutley	Marketing	South-351	6	arutley	2022-05-12	17:00:59	MEXICO	192.168.3.24
1004	e218f877g788	eraab	Human Resources	South-127	7	eraab	2022-05-11	01:45:14	CAN	192.168.170.243
1035	j236k303i245	bisles	Sales	South-171	8	bisles	2022-05-08	01:30:17	US	192.168.103.103

## Summary

I developed SQL queries to join two tables, analysing three specific scenarios:

1. **Inner Join:** Retrieved only the rows with matching values in both tables, focusing on shared data.
2. **Left Join:** Included all rows from the `machines` table, with unmatched `employees` data represented as `NULL`.
3. **Right Join:** Included all rows from the `employees` table, with unmatched `machines` data represented as `NULL`.

Each query was tailored to explore different aspects of the data, providing critical insights for relational database management and security analysis. The investigation of login attempts via **INNER JOIN** further supported security risk assessments by linking employees to their activity logs.