Apply filters to SQL queries

Project description

In this project, an organization needed to obtain specific information about employees, their machines, and the departments they belong to from the database. My team needs data to investigate potential security issues, apply updates to employees' computers, and improve the overall security posture of my organization.

First, I retrieved all failed login attempts after business hours. Next, I retrieved all login attempts that occurred on specific dates. Third, I retrieved logins that didn't originate in Mexico. After that, I retrieved information about certain employees in the Marketing department. Fifth, I retrieved information about employees in the Finance or the Sales department. Finally, I obtain information about employees who are not in the Information Technology department.

Retrieve after hours failed login attempts

The login_time column in the log_in_attempts table contains information on when login attempts were made. Office hours end at '18:00'.

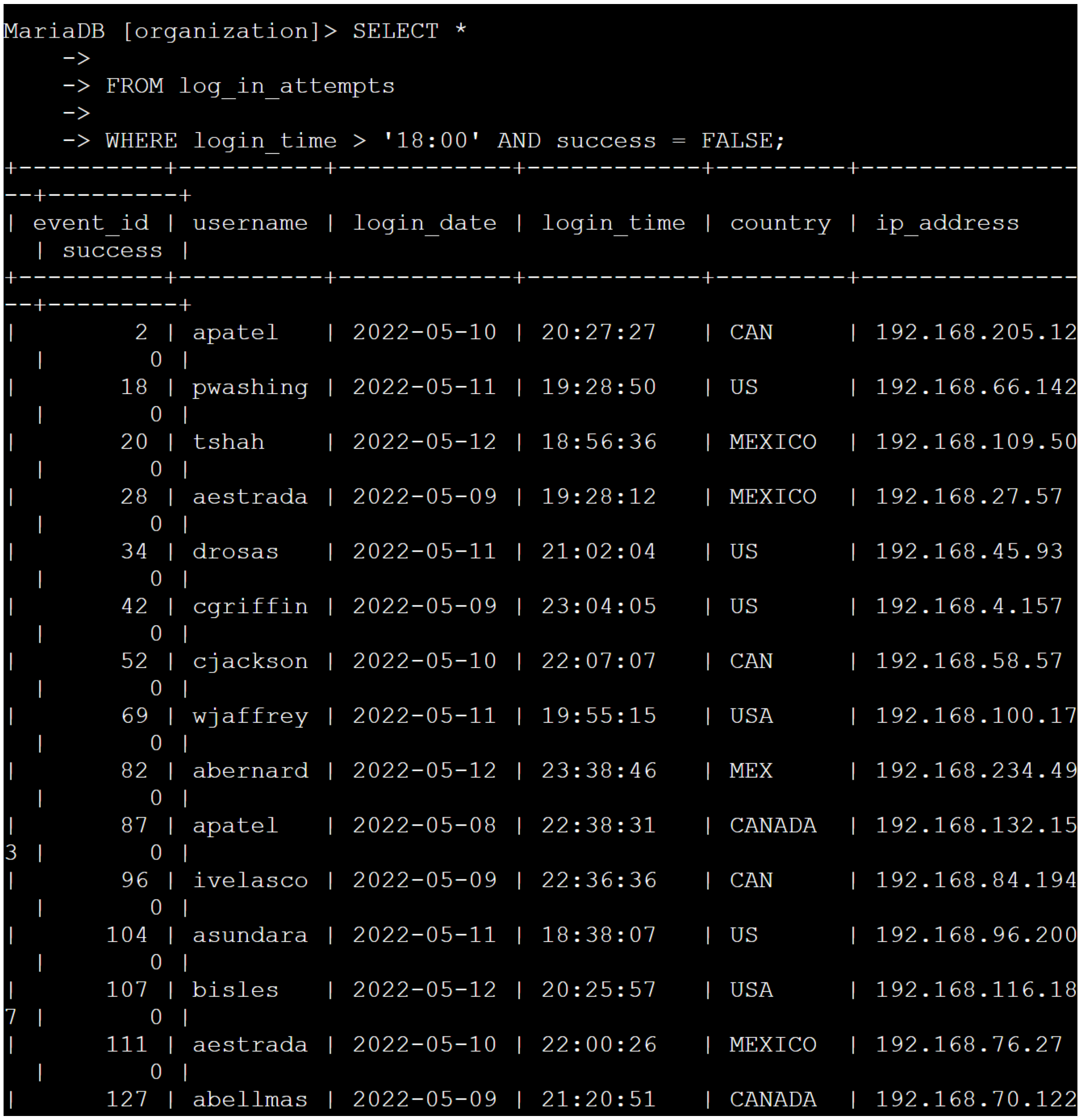
The success column in the log_in_attempts table contains values of TRUE or FALSE to indicate whether the login was successful. MySQL stores Boolean values as 1 for TRUE,
and 0 for FALSE. This means that TRUE is represented as 1, and FALSE represented as 0 in the success column.

I used the AND operator to retrieve the failed login attempts that occurred after business hours.

Note: *Values of TRUE and FALSE are not placed in single quotes because they are not string data. They are Boolean data, which is another data type.*

The command to complete this step:

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



```
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  | 192.168.132.15
3 |       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     | 192.168.116.18
7 |       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
```

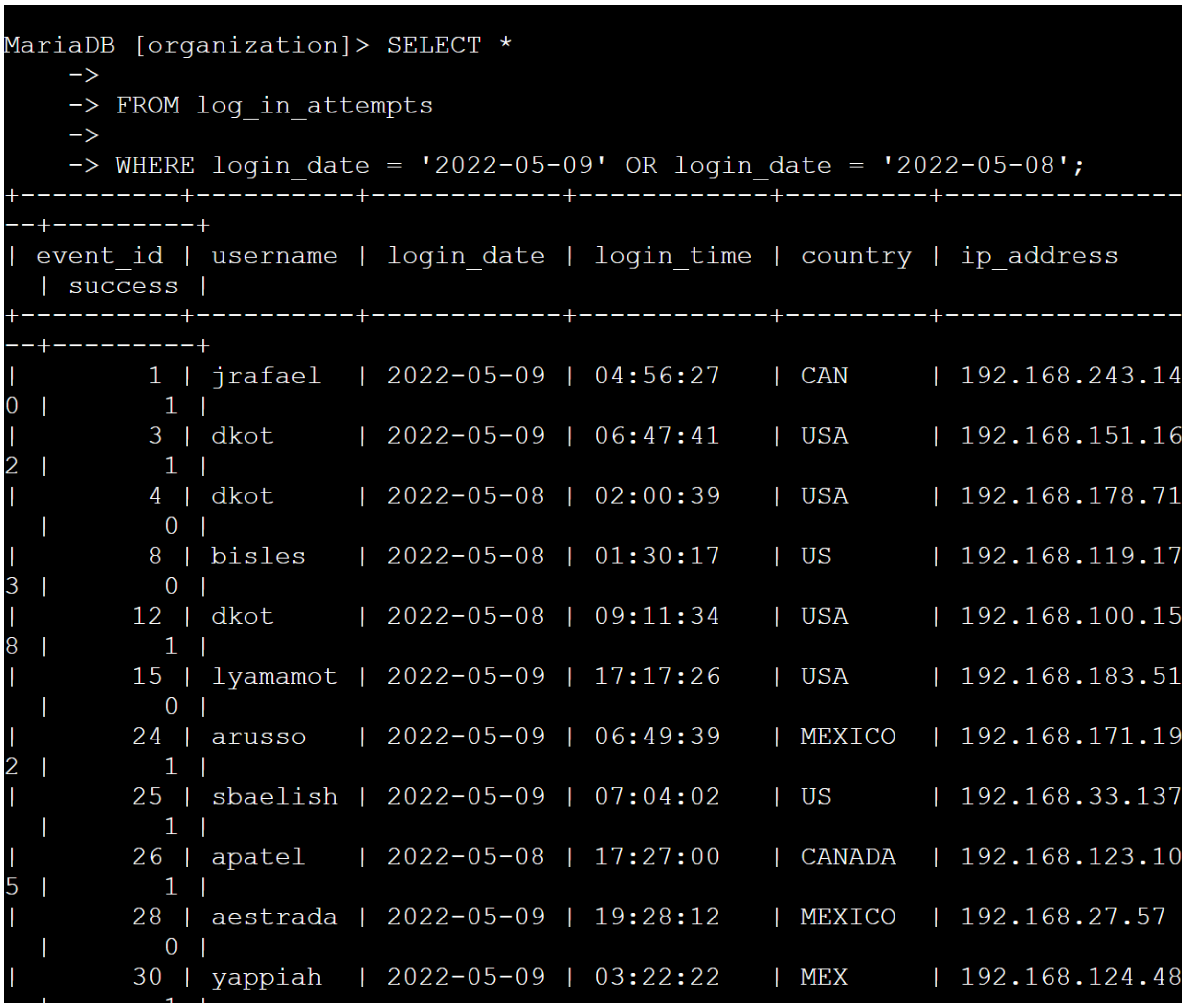
Retrieve login attempts on specific dates

My team was investigating a suspicious event that occurred on '2022-05-09'. I want to retrieve all login attempts that occurred on this day and the day before ('2022-05-08').

The login_date column in the log_in_attempts table contains information on the dates when login attempts were made.

I use the OR operator to retrieve the failed login attempts on the specified days.

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



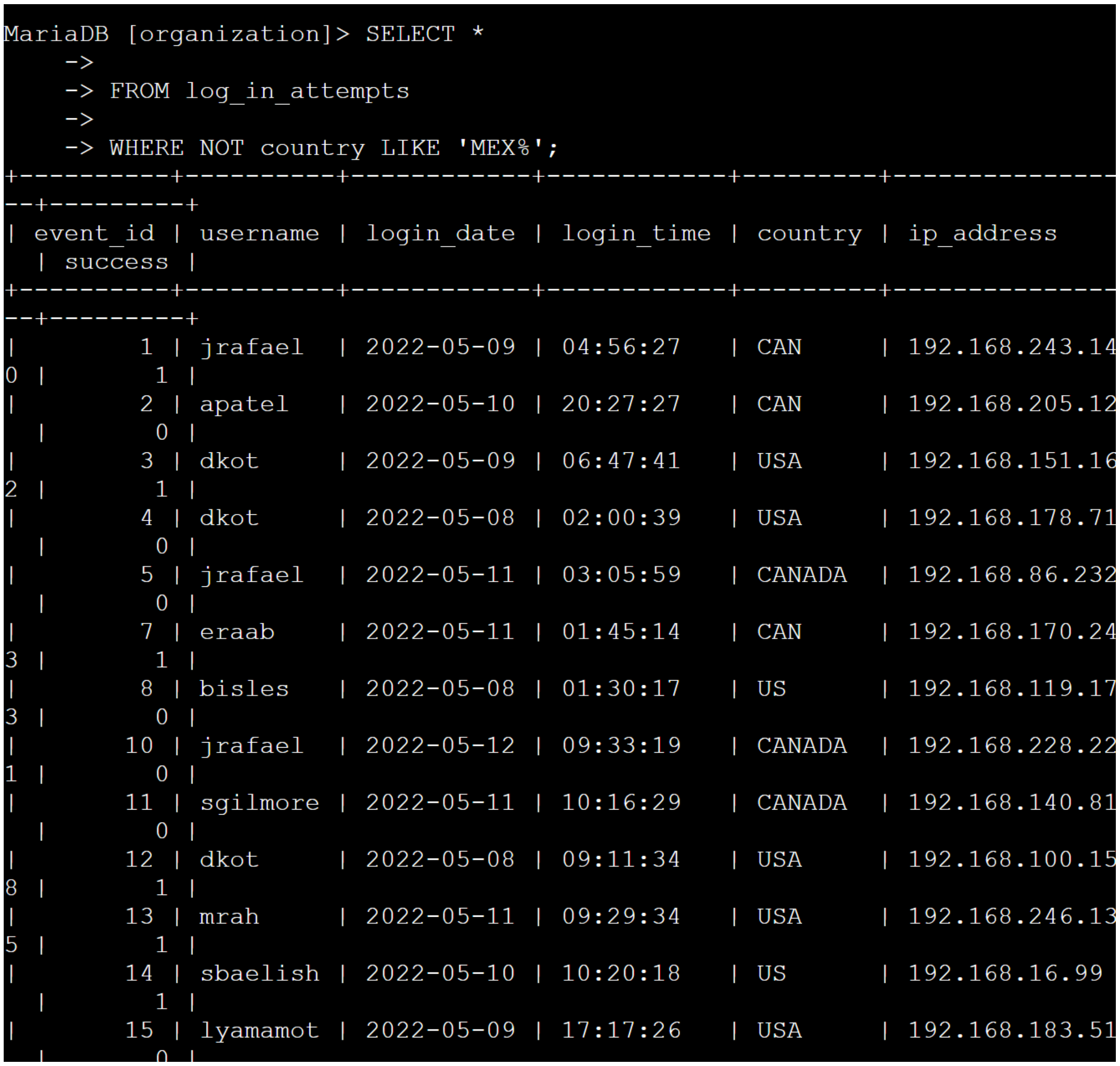
```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+---------------
--+---------+
| event_id | username | login_date | login_time | country | ip_address
  | success |
+----------+----------+------------+------------+---------+---------------
--+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.14
0 |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.16
2 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71
  |       0 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.17
3 |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.15
8 |       1 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51
  |       0 |
|       24 | arusso   | 2022-05-09 | 06:49:39   | MEXICO  | 192.168.171.19
2 |       1 |
|       25 | sbaelish | 2022-05-09 | 07:04:02   | US      | 192.168.33.137
  |       1 |
|       26 | apatel   | 2022-05-08 | 17:27:00   | CANADA  | 192.168.123.10
5 |       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
```

Retrieve login attempts outside of Mexico

My team is investigating logins that did not originate in Mexico, and I need to find this information. Note that the country field includes entries with 'MEX' and 'MEXICO'.

I ran the following SQL query to retrieve login attempts that did not originate in Mexico.

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

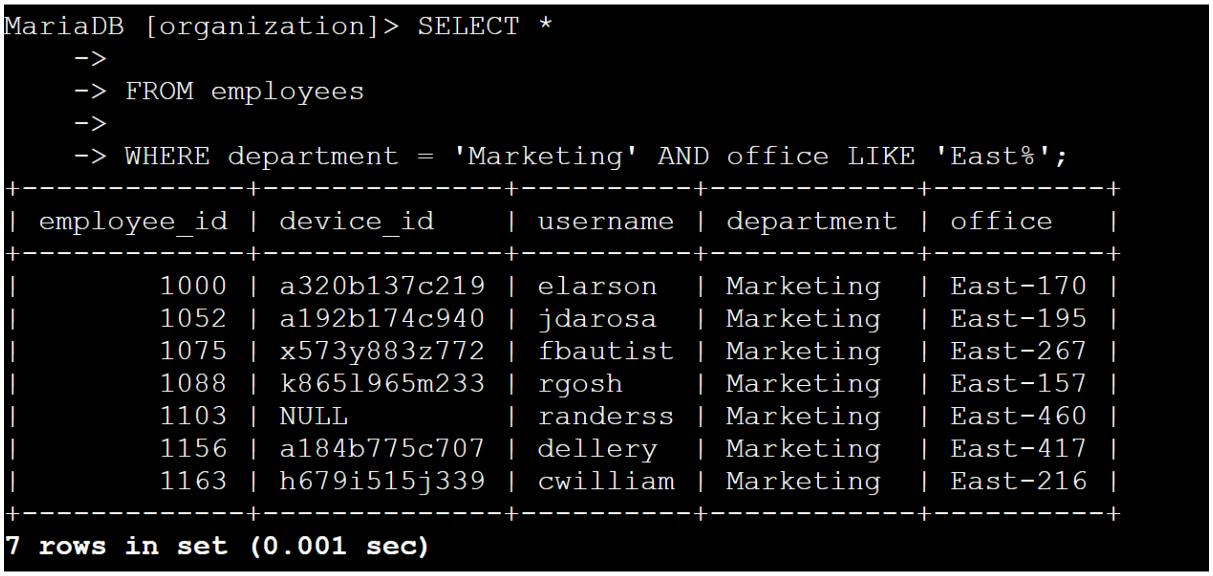


```
MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+---------------
--+---------+
| event_id | username | login_date | login_time | country | ip_address
  | success |
+----------+----------+------------+------------+---------+---------------
--+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.14
0 |       1 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12
  |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.16
2 |       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     | 192.168.170.24
3 |       1 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.17
3 |       0 |
|       10 | jrafael  | 2022-05-12 | 09:33:19   | CANADA  | 192.168.228.22
1 |       0 |
|       11 | sgilmore | 2022-05-11 | 10:16:29   | CANADA  | 192.168.140.81
  |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.15
8 |       1 |
|       13 | mrah     | 2022-05-11 | 09:29:34   | USA     | 192.168.246.13
5 |       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 |
```

Retrieve employees in Marketing

Here, my team needed to retrieve the information from the department and office columns in the employees table about employees in the Marketing department who are located in all offices in the East building (such as East-170 or East-320).
I ran the following SQL query to retrieve this information from the employees table. Note that I used AND and LIKE operators to satisfy both of the criteria.

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



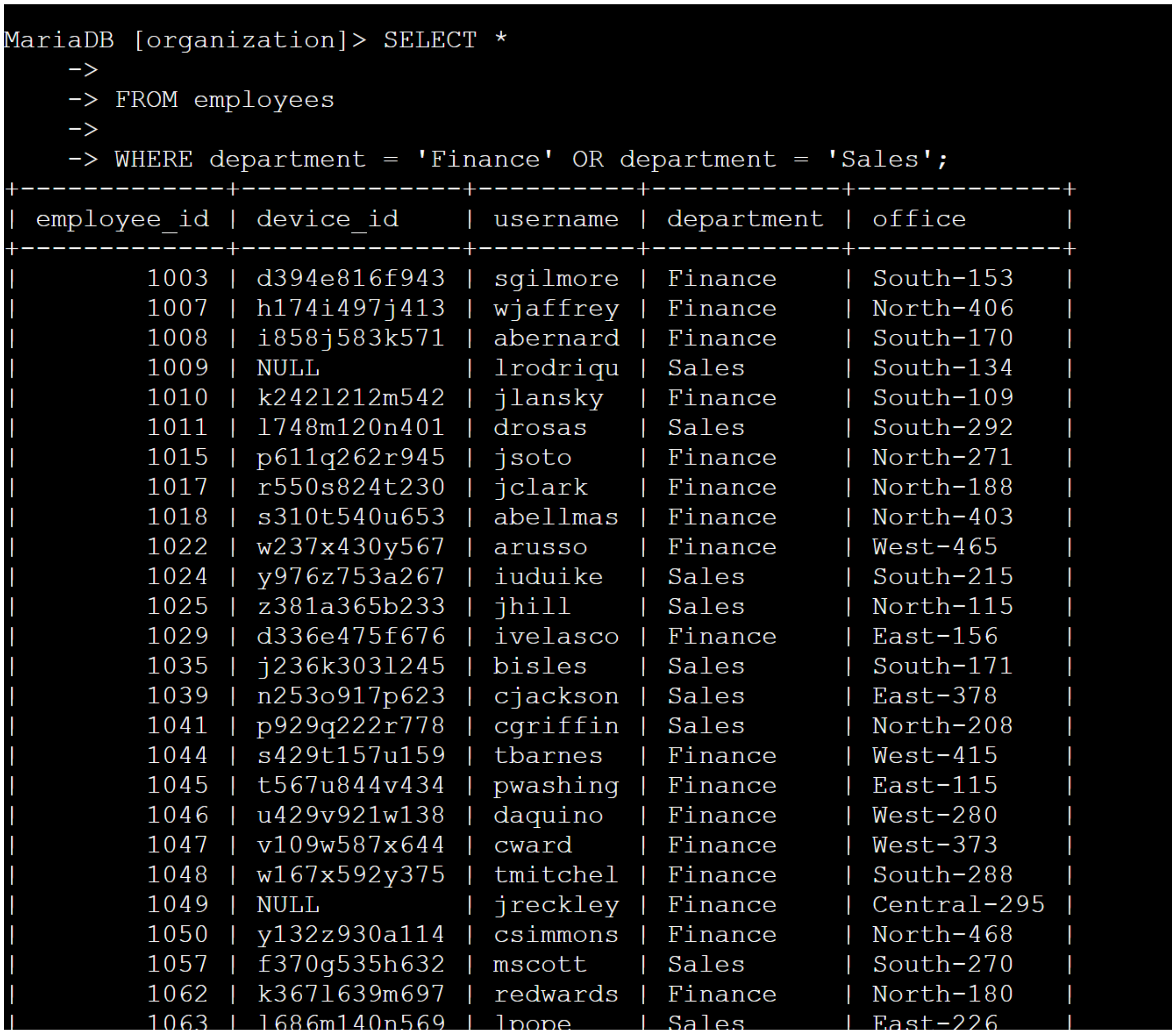
```
MariaDB [organization]> SELECT *
    ->
    -> FROM employees
    ->
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+---------------+----------+------------+----------+
| employee_id | device_id     | username | department | office   |
+-------------+---------------+----------+------------+----------+
|        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)
```

Retrieve employees in Finance or Sales

My team needed to perform a different update to the computers of all employees in the Finance or Sales department, and I was tasked to locate information on these employees.

I wrote the following SQL query to retrieve records for employees in the Finance or Sales department.

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



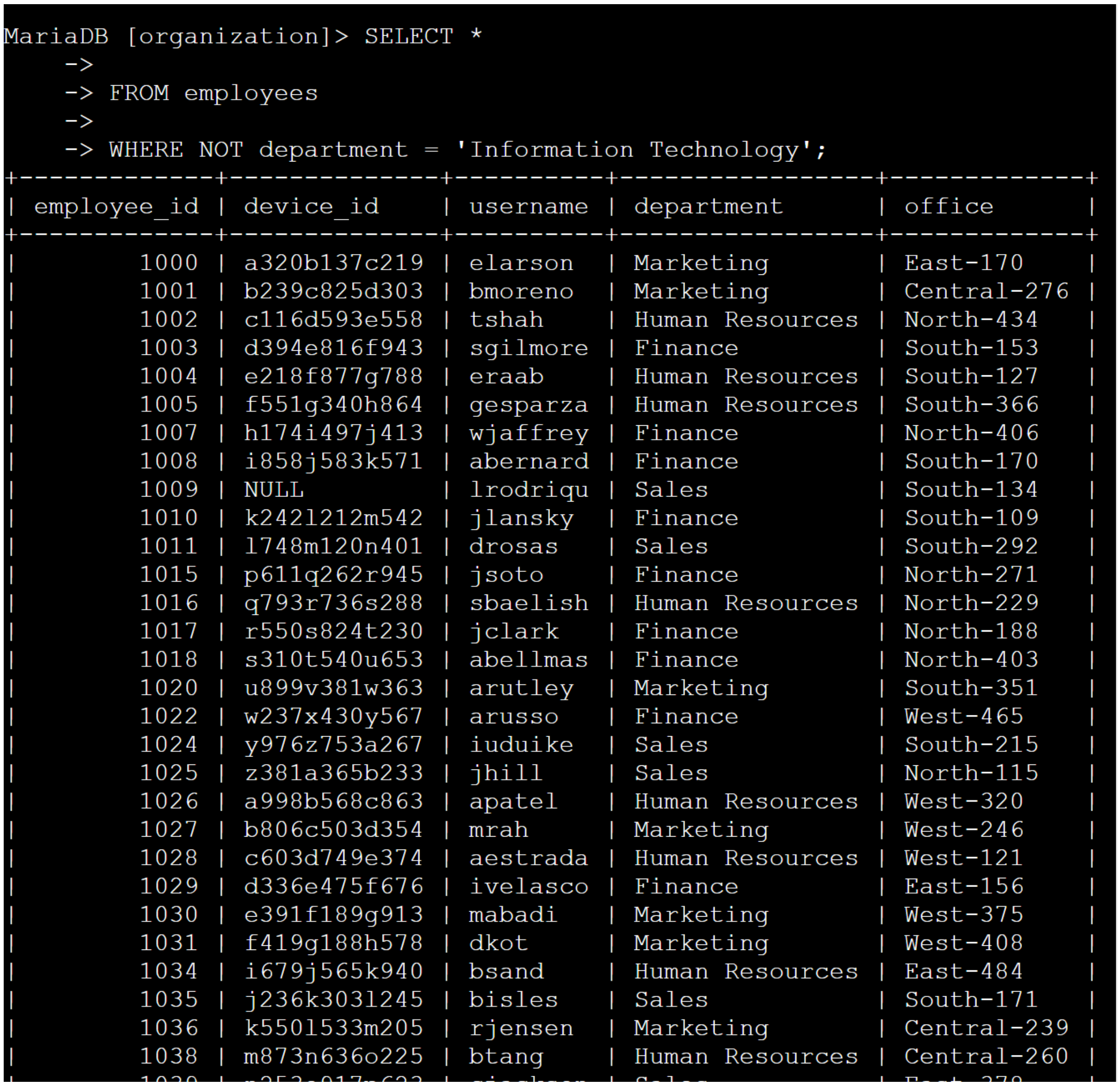
```
MariaDB [organization]> SELECT *
    ->
    -> FROM employees
    ->
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+---------------+----------+------------+-------------+
| employee_id | device_id     | username | department | office      |
+-------------+---------------+----------+------------+-------------+
|        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 | 1748m120n401  | 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    |
```

Retrieve all employees not in IT

The business need for this task was to replicate an update that had already been applied to employees' computers in the Information Technology department across the other departments. My team needed information about employees who are not in the Information Technology Department.

To fulfil this request I created the following SQL query using the NOT operator to identify the employees.

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



```
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 | 1748m120n401 | 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 |
```

Summary

By combining the use of standard query statements like SELECT and FROM with the WHERE filter, I was able to retrieve specific information from tables in a relational database. In addition, further filter options like using AND, OR, and NOT were crucial in combining certain logical conditions required. The LIKE operator (and it’s % wildcard) is especially handy when the exact strings being searched for is not completely known. These operators were applied on both numerical, string, and date and time data.