Apply filters to SQL queries

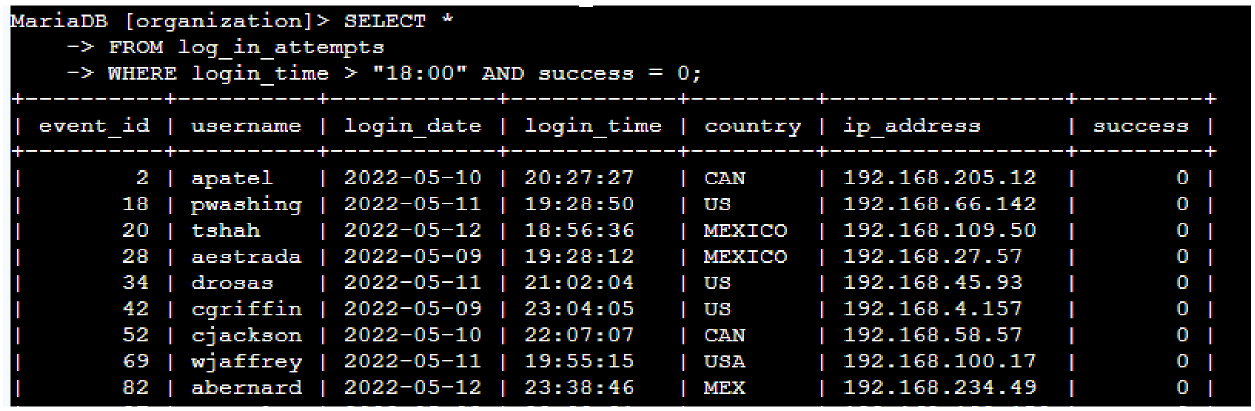
Project description

My organization is working to increase the security of their system. It is my job to ensure the system is safe, investigate any potential security issues and update employee computers as needed. The following steps provide examples of how I used SQL to filter data and perform security related tasks.

Retrieve after hours failed login attempts

There was a potential security incident that occurred outside business hours, all after hours login attempts that failed need to be investigated.

The following code demonstrates how I created a SQL query to filter data for failed login attempts that occurred outside business hours. To query the `log_in_attempts` table for after hours login activity I created a query with: `SELECT * FROM log_in_attempts WHERE login_time > “18:00” AND success = 0;`.



```
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 |
```

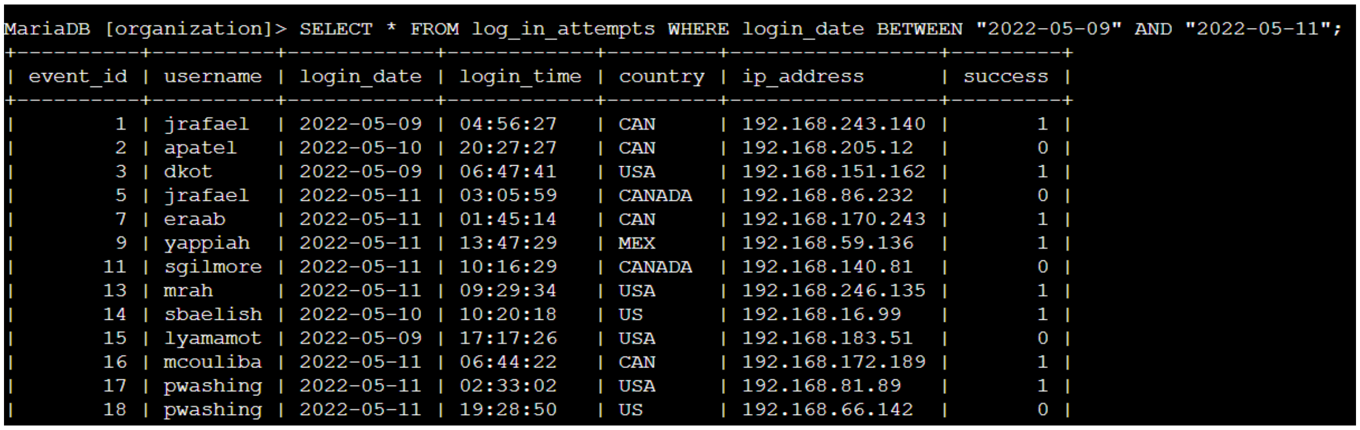
The first part of this screenshot is my query and can be broken down into three parts:

1. The first line of the query I started with the `SELECT` operator which selects all the columns in the `log_in_attempts` table, using the * (asterisk) operator as the standard input ensures all columns are selected.
2. The second line of the query uses the `FROM` operator to select which table the query will be retrieving data from, by supplying the table name as an argument after the `FROM` operator. This can be seen by the `log_in_attempts` input after the `FROM` operator.

3. Lastly, I used the `WHERE` operator to filter the results of the query. Here I specified that I wanted the data that shows any unsuccessful login attempts made after 6pm. I achieved this by filtering the login time column with `WHERE login_time > “18:00”`, this tells the query I want to see all data where the login time is greater than 18:00 (6pm). Next I added a second argument to the filter to show only unsuccessful attempts with `AND success = 0;`, the `success` column holds boolean values (true or false) and these are shown by 0's (false) and 1's (true) so by specifying to the query to only show data with a success value of 0, only unsuccessful attempts will be shown in the table. The use of the `AND` operator to connect these arguments ensures only data that passes both filter arguments is shown.

Retrieve login attempts on specific dates

Next I needed to narrow the focus of the search by filtering for login attempts made between 2022-05-09 & 2022-05-11. To achieve this I used the query `SELECT * FROM log_in_attempts WHERE login_date BETWEEN “2022-05-09” AND “2022-05-11”;`

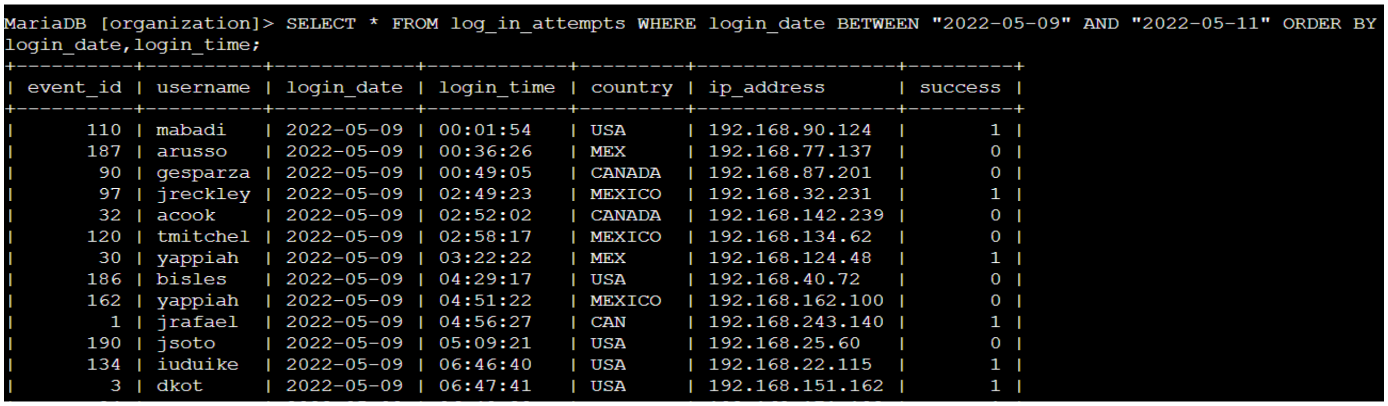


```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_date BETWEEN "2022-05-09" AND "2022-05-11";
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
|        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 |
|        9 | yappiah  | 2022-05-11 | 13:47:29   | MEX     | 192.168.59.136  |       1 |
|       11 | sgilmore | 2022-05-11 | 10:16:29   | CANADA  | 192.168.140.81  |       0 |
|       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 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       16 | mcouliba | 2022-05-11 | 06:44:22   | CAN     | 192.168.172.189 |       1 |
|       17 | pwashing | 2022-05-11 | 02:33:02   | USA     | 192.168.81.89   |       1 |
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0 |
```

The first 2 parts of this query are the same as the previous query, I used the `SELECT` operator with the * (asterisk) character to select all columns and the `FROM` operator coupled with the input `log_in_attempts` to request the data from the appropriate table.

Next I used the `WHERE` operator to apply a filter to my query. I requested data that had values for the `login_date` column that was between the dates of 2022-05-09 & 2022-05-11. I achieved this by using the `BETWEEN` operator which will filter the data by only showing values that are between the specified values given as the input. This can be seen by the `BETWEEN “2022-05-09” AND “2022-05-11”` section of the query.

Lastly I also applied a sorting function to the query to better display the data returned, this is shown by the `ORDER BY login_date,login_time` section of the query. `ORDER BY` are operators that will take a standard input as an argument to sort the returned data, here I specified `ORDER BY` to sort the data firstly by `login_date` to show the login attempts in order of the date of the attempt. I then added a second argument `login_time` to the sorting filter to order the data by the time of the login attempt after it had been ordered by login date. This results in the login attempts being displayed in date order, with each group of attempts displayed by the times of the login attempts.

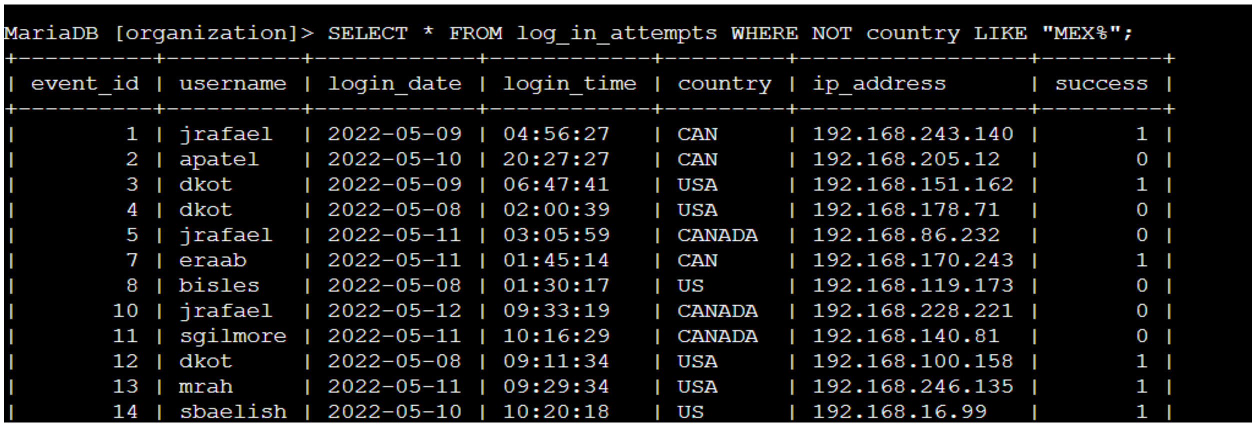


```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_date BETWEEN "2022-05-09" AND "2022-05-11" ORDER BY
login_date,login_time;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|      110 | mabadi   | 2022-05-09 | 00:01:54   | USA     | 192.168.90.124  |       1 |
|      187 | arusso   | 2022-05-09 | 00:36:26   | MEX     | 192.168.77.137  |       0 |
|       90 | gesparza | 2022-05-09 | 00:49:05   | CANADA  | 192.168.87.201  |       0 |
|       97 | jreckley | 2022-05-09 | 02:49:23   | MEXICO  | 192.168.32.231  |       1 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|      120 | tmitchel | 2022-05-09 | 02:58:17   | MEXICO  | 192.168.134.62  |       0 |
|       30 | yappiah  | 2022-05-09 | 03:22:22   | MEX     | 192.168.124.48  |       1 |
|      186 | bisles   | 2022-05-09 | 04:29:17   | USA     | 192.168.40.72   |       0 |
|      162 | yappiah  | 2022-05-09 | 04:51:22   | MEXICO  | 192.168.162.100 |       0 |
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|      190 | jsoto    | 2022-05-09 | 05:09:21   | USA     | 192.168.25.60   |       0 |
|      134 | iuduike  | 2022-05-09 | 06:46:40   | USA     | 192.168.22.115  |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1 |
```

Retrieve login attempts outside of Mexico

After reviewing the organization's data on the login attempts. I determined the suspicious login activity did not originate from Mexico, I needed to investigate all login attempts that occurred outside of Mexico.

The following code demonstrates how I created a SQL query to filter for login attempts that occurred outside of Mexico. To achieve this I used the query `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     | 192.168.243.140 |       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 first two parts of this query use the `SELECT` operator with the `*` (asterisk) character to specify to the query to return all columns, and the `FROM` operator with `log_in_attempts` to pull data from the login attempts table.

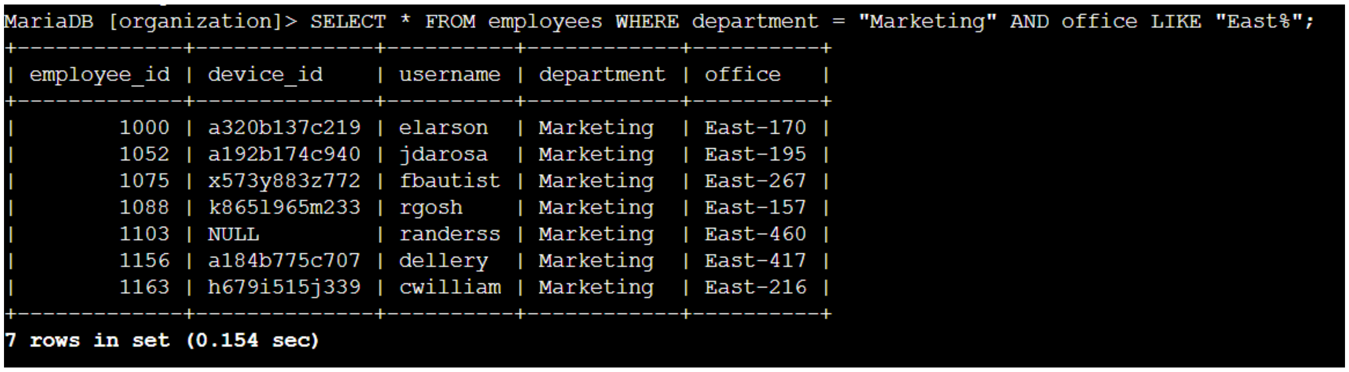
To filter out all login attempts that did not originate in Mexico, I used the `WHERE` and `NOT` operator together. The `NOT` operator is used to specify that data should only be returned if it does not meet the criteria of the argument that will be provided to it. After the `NOT` operator I provided the arguments of `country` and `LIKE "MEX%"`, the argument country tells the query that I am filtering by the country column of the table and the `LIKE "MEX%"` instructs the query to only return data where the value in the country column does not contain Mexico or Mex. The `LIKE` operator is used in conjunction with the `WHERE` operator to search for a pattern in a column, the % (percentage) character is a wildcard symbol and is used to substitute any number of characters and is used here because the values for countries in the countries column are not all following the same naming convention (e.g. you will see USA also represented as US, Mexico also as Mex etc).

Retrieve employees in Marketing

My team is tasked with performing security updates on specific employee machines in the marketing department. I was responsible for identifying all employees from the marketing department for all offices in the east building.

The following code shows how I created an SQL query to filter for employee machines from employees in the Marketing department in the East building . I used the query `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 (0.154 sec)
```

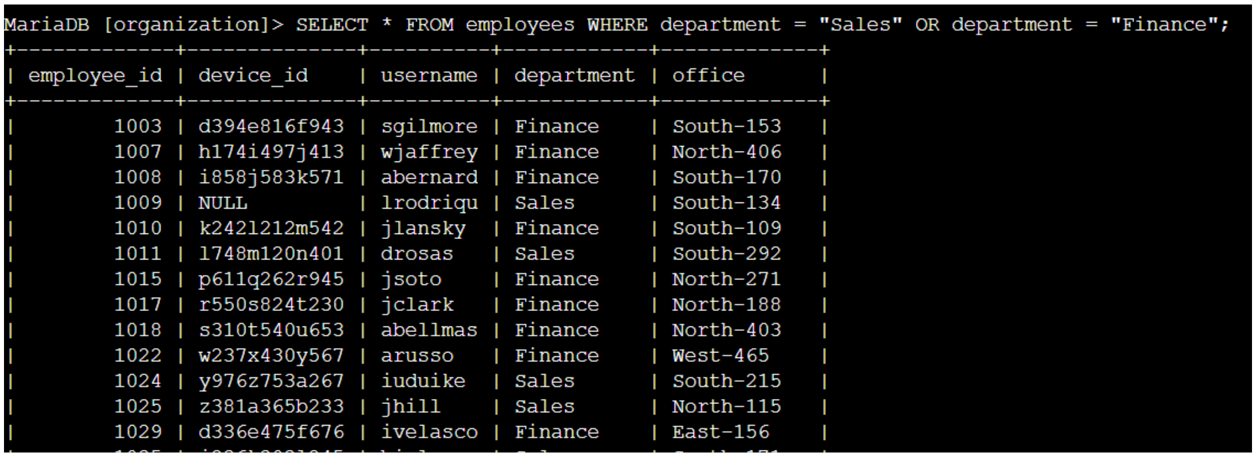
To break this query down, I used the `SELECT` operator with the * (asterisk) character to query all columns, and used the `FROM` operator with `employees` to request data from the employees table similar to my queries in the previous sections.

Next I used `WHERE` to apply two filters to my query with the use of the `AND` operator to specify that both arguments need to return true for the data to be returned. The two arguments I provided to the `WHERE` filter were `department = “Marketing”` which tells the query to return data that has a value of Marketing for the department column of the table, and `office LIKE “East%”` which instructs the query to return data that has a value in the office column beginning with East, as well as any characters following as indicated by the use of the `%` (wildcard symbol) character. If data meets both of these conditions then it is returned by the query.

Retrieve employees in Finance or Sales

The machines for the employees in the Finance and Sales departments also need to be updated. Since a different security update is needed, I have to get information on employees only from these two departments.

The following code demonstrates how I created a SQL query to filter for employee machines from employees in the Finance or Sales departments. I used the query `SELECT * FROM employees WHERE department = “Marketing” OR department = “Sales”`



```
MariaDB [organization]> SELECT * FROM employees WHERE department = "Sales" OR department = "Finance";
+-------------+--------------+----------+------------+-------------+
| 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    |
```

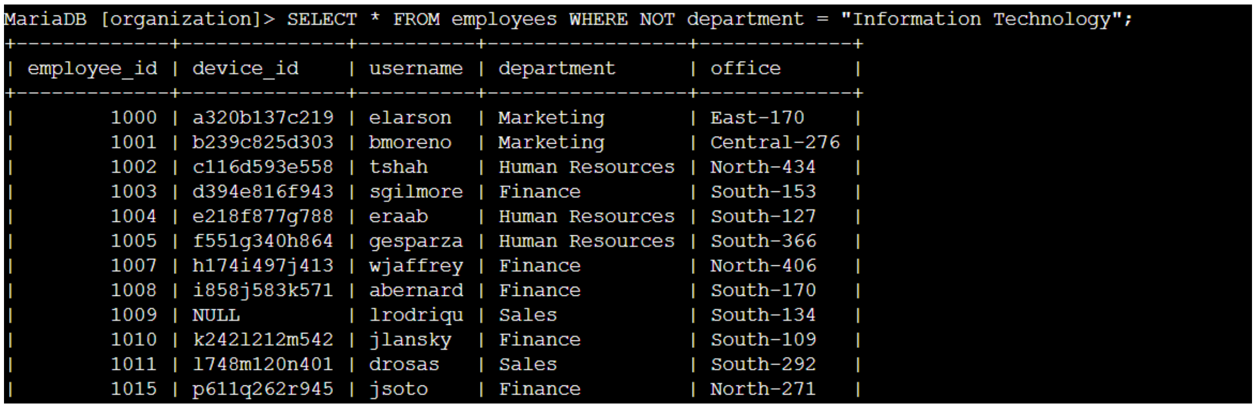
The first two parts of this query use the `SELECT` operator with the `*` (asterisk) character to specify to the query to return all columns, and the `FROM` operator with the argument `employees` to specify to the query to pull the data from the employees table.

Next I used the `WHERE` operator to specify a filter for the query and passed two arguments to the filter, `department = “Marketing”` to instruct the query to return any data that had the value Marketing in the department column of the table. The second argument passed to the filter was `department = “Sales”` which, similar to the first argument only returns data where the value for the department column is Sales. I used the `OR` operator instead of the `AND` operator for this query as the `OR` operator specifies to the query that either value being present in the department column would pass the criteria and be returned in the table.

Retrieve all employees not in IT

My team needed to make one more security update on employees who are not in the Information technology department. I had to get information on these employees to make the updates.

The following code demonstrates how I created a SQL query to filter for employee machines from the employees not in the Information Technology department. To achieve this I used the query `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   |
```

The first two parts of this query use the `SELECT` operator with the `*` (asterisk) character to specify to the query to return all columns, and the `FROM` operator with the argument `employees` to specify to the query to pull the data from the employees table.

The third part of the query specifies to only return data where the value for the department column is not Information Technology. I achieved this by using the `WHERE` operator to filter the data and provided the argument `NOT department = “Information Technology”,` to specify to the query to only return data which does not contain the value Information Technology in the department column of the employees table.

Summary

I applied filters to SQL queries to pull specific data on login attempts and employee machines/departments. I pulled this data from two different tables, `log_in_attempts` and `employees`. I filtered the data for specific information needed for the task using the `AND`, `OR`, `NOT` and `BETWEEN` operators. I also used `LIKE` and the percentage sign (`%`) wildcard to filter for patterns.