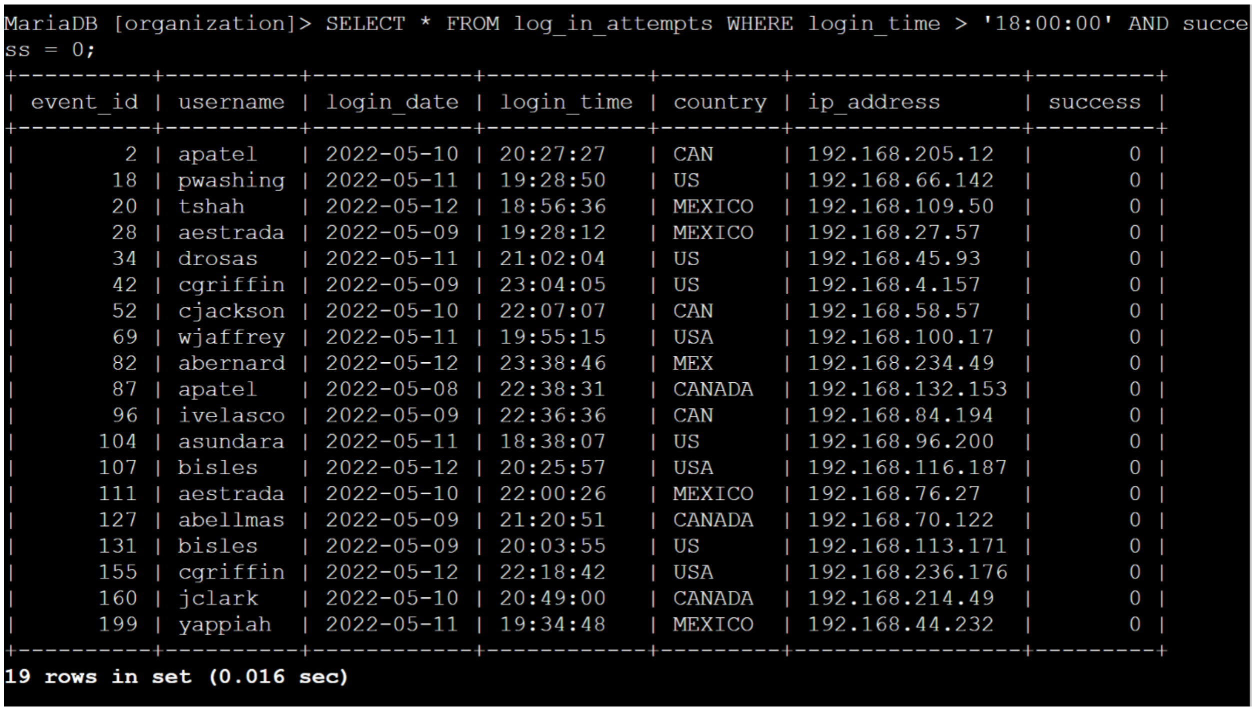
Apply filters to SQL queries

Project description

I was tasked by my organisation to investigate potential security issues involving login attempts on employee machines. I Examined the organisation's data, held in the `employees` and `log_in_attempts` tables, using various SQL filters in order to retrieve records relevant to the investigation.

Retrieve after hours failed login attempts

My organisation recently discovered a potential security incident that occurred after business hours. I have been tasked with investigating all failed login attempts that occurred after 1800. This data is held withing the `log_in_attempts` table. The data I need from the table is found in the `login_time` column and the success column (a `0` indicates a failed login attempt). The query I used can be seen in the screenshot below:



```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_time > '18:00:00' AND succe
ss = 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 (0.016 sec)
```

The query I used was:

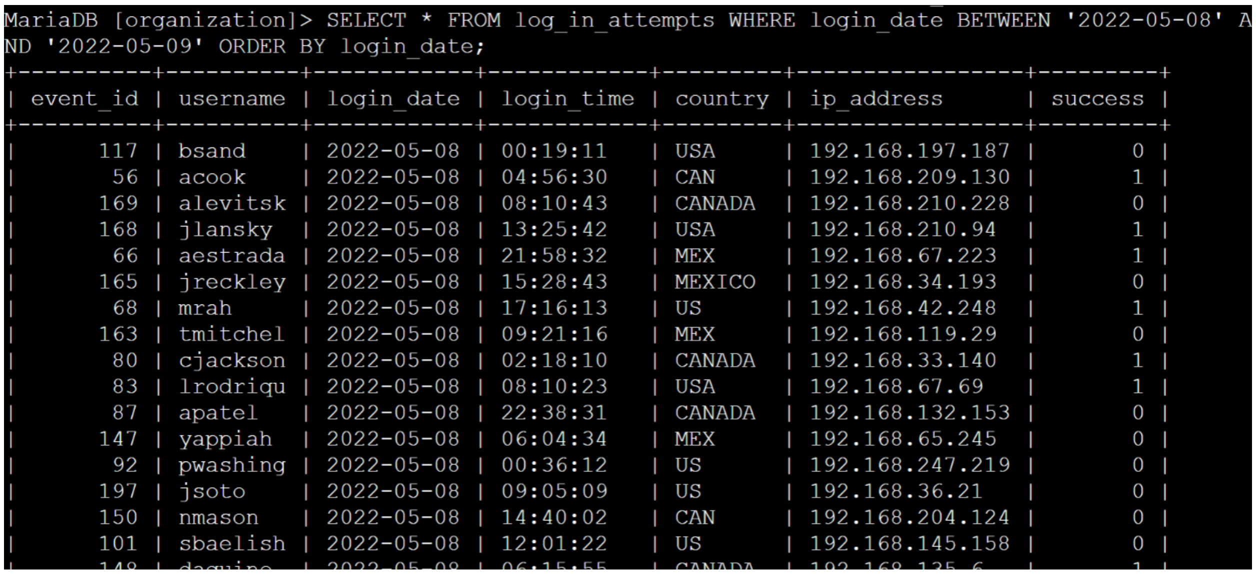
```
SELECT * FROM log_in_attempts WHERE login_time > ’18:00:00’ AND
success = 0
```

I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)
- `FROM log_in_attempts` (This directs the query to the `log_in_attempts` table)
- `WHERE login_time > ’18:00:00’` (This searches the `login_time` column for login times that are greater than 1800)
- `AND success = 0` (This searches the success column for login attempts that are equal to `0`)

Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. To review this, I needed to investigate all login attempts which occurred on this day and the day before. The query I used for this can be seen in the screenshot below:



```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_date BETWEEN '2022-05-08' A
ND '2022-05-09' ORDER BY login_date;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|      117 | bsand    | 2022-05-08 | 00:19:11   | USA     | 192.168.197.187 |       0 |
|       56 | acook    | 2022-05-08 | 04:56:30   | CAN     | 192.168.209.130 |       1 |
|      169 | alevitsk | 2022-05-08 | 08:10:43   | CANADA  | 192.168.210.228 |       0 |
|      168 | jlansky  | 2022-05-08 | 13:25:42   | USA     | 192.168.210.94  |       1 |
|       66 | aestrada | 2022-05-08 | 21:58:32   | MEX     | 192.168.67.223  |       1 |
|      165 | jreckley | 2022-05-08 | 15:28:43   | MEXICO  | 192.168.34.193  |       0 |
|       68 | mrah     | 2022-05-08 | 17:16:13   | US      | 192.168.42.248  |       1 |
|      163 | tmitchel | 2022-05-08 | 09:21:16   | MEX     | 192.168.119.29  |       0 |
|       80 | cjackson | 2022-05-08 | 02:18:10   | CANADA  | 192.168.33.140  |       1 |
|       83 | lrodriqu | 2022-05-08 | 08:10:23   | USA     | 192.168.67.69   |       1 |
|       87 | apatel   | 2022-05-08 | 22:38:31   | CANADA  | 192.168.132.153 |       0 |
|      147 | yappiah  | 2022-05-08 | 06:04:34   | MEX     | 192.168.65.245  |       0 |
|       92 | pwashing | 2022-05-08 | 00:36:12   | US      | 192.168.247.219 |       0 |
|      197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |       0 |
|      150 | nmason   | 2022-05-08 | 14:40:02   | CAN     | 192.168.204.124 |       0 |
|      101 | sbaelish | 2022-05-08 | 12:01:22   | US      | 192.168.145.158 |       0 |
|      148 | daquino  | 2022-05-08 | 06:15:55   | CANADA  | 192.168.135.6   |       1 |
```

The query I used was:

```
SELECT * FROM log_in_attempts WHERE login_date BETWEEN '2022-05-08'
AND '2022-05-09' ORDER BY login_date;
```

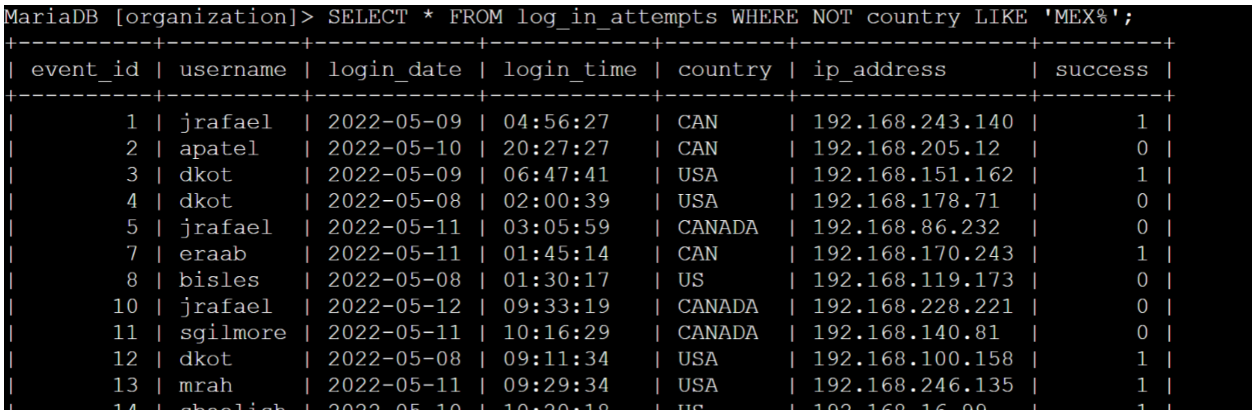
I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)

- `FROM log_in_attempts` (This directs the query to the `log_in_attempts` table)
- `WHERE login_date BETWEEN '2022-05-08' AND '2022-05-09'` (This will return all login attempts between, and including, 2022-05-08 and 2022-05-09
- `ORDER BY login_date` (This will order the data by login date, ascending)

Retrieve login attempts outside of Mexico

The organisation determined that the suspicious activity originated from outside of Mexico. I now had to create a query to return all login attempts that occurred outside of Mexico. The query I used can be seen 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 |
```

The query I used was:

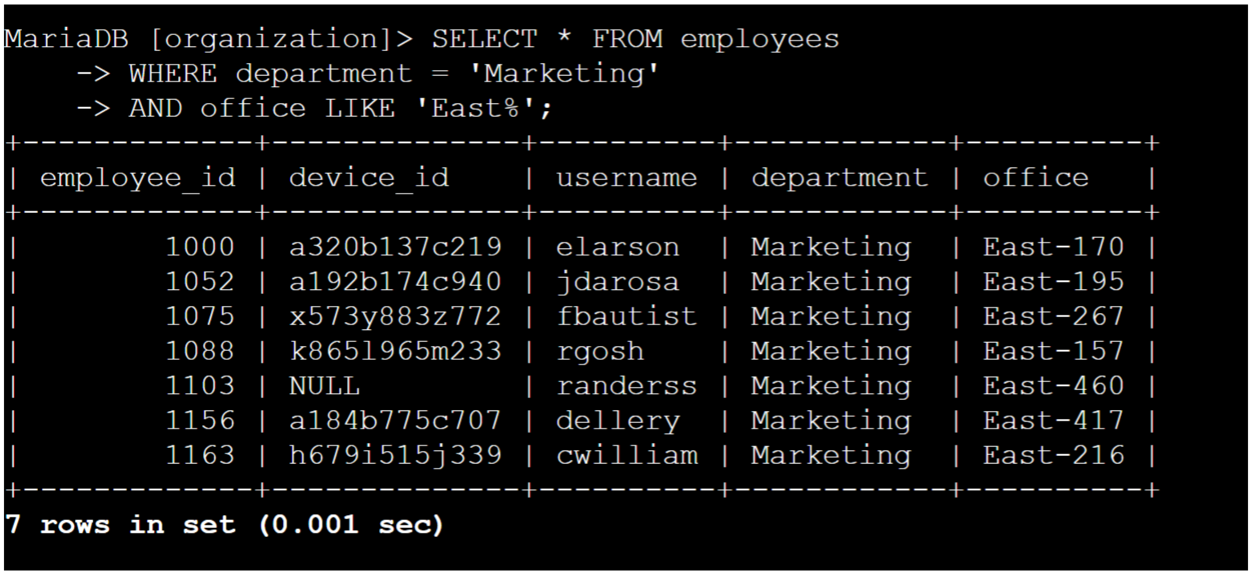
`SELECT * FROM log_in_attempts WHERE NOT country LIKE 'MEX%';`

I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)
- `FROM log_in_attempts` (This directs the query to the `log_in_attempts` table)
- `WHERE NOT country LIKE 'MEX%'` (This will return all instances where the country is `NOT` Mexico. I used a `%` wildcard here since Mexico can be written as 'Mexico' or 'Mex')

Retrieve employees in Marketing

My organisation wanted to perform security updates on specific employee machines in the Marketing department. I needed to find the relevant information on these employee machines. I needed to create a query that returned all employees in the Marketing department for all offices in the East building. The query I used can be seen 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 | 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)
```

The query I used was:

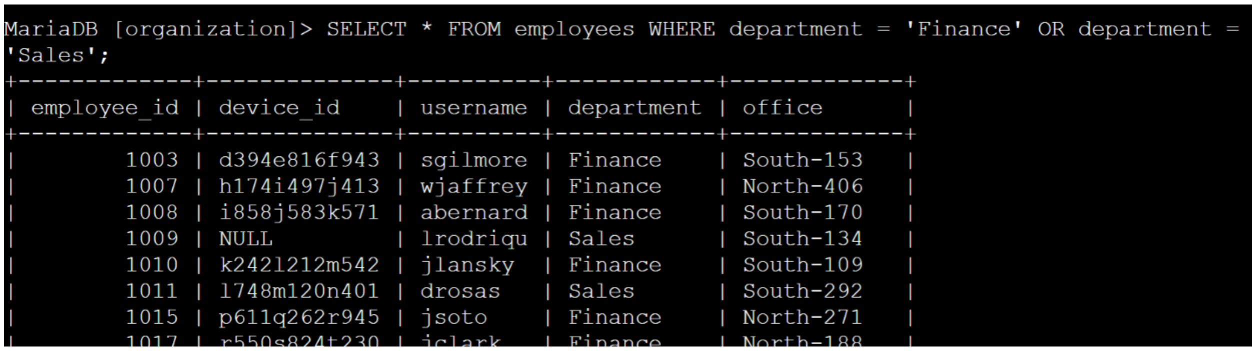
`SELECT * FROM employees WHERE department = 'Marketing' AND office LIKE 'East%';`

I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)
- `FROM employees` (This directs the query to the `employees` table)
- `WHERE department = ‘Marketing’ AND office LIKE ‘East%’;` (This will return all employees in the Marketing department who also work in offices in the East building, I used a `%` wildcard here as there are multiple different offices in the East building, e.g. East-216)

Retrieve employees in Finance or Sales

My organisation also wanted to perform a different security update on machines for employees in the Sales and Finance departments. I had to find the relevant information on these machines. The query I used for this can be seen in the screenshot below:



```
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   |
```

The query I used was:

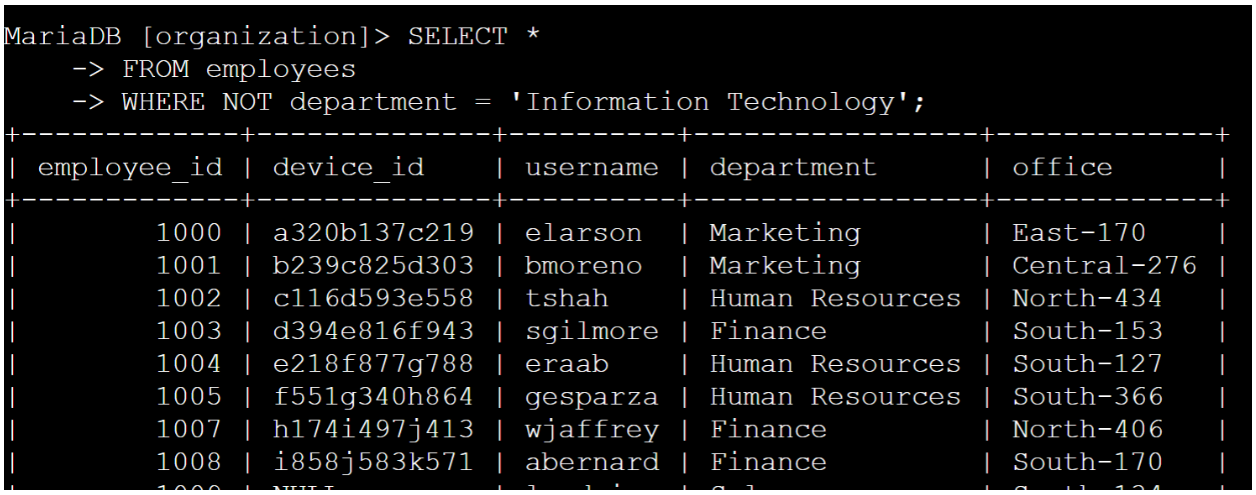
`SELECT * FROM employees WHERE department = 'Finance' OR department = 'Sales';`

I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)
- `FROM employees` (This directs the query to the `employees` table)
- `WHERE department = ‘Finance’ OR department = ‘Sales’;` (This will return employees who work within the Finance `OR` Sales departments)

Retrieve all employees not in IT

My organisation needed to perform one more update. Employees within the IT department already have it, however, employees in all the other department don’t. I need to get the information for all employee machines that are not within the IT department. The query I used for this can be seen 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   |
```

The query I used was:

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

I will now break this down and explain each part:

- `SELECT *` (This will return all columns in the table, signified by the `*`)
- `FROM employees` (This directs the query to the `employees` table)
- `WHERE NOT department = ‘Information Technology’;` (This will return all employees working in every department other than Information Technology)

Summary

I used filters within various SQL queries to get specific information on login attempts and employee machines for my organisation. I used two different tables, `log_in_attempts` and `employees`. I used the `AND`, `OR`, `NOT` and `ORDER BY` operators to filter for the specific information I needed to complete each task. I also used the `LIKE` and the `%` wildcard to filter for patterns.