Project description

My organization tasked me with ensuring the system was safe, investigating potential security issues, and updating employee computers as necessary. The following are examples of how I used SQL to filter through records of various tables and datasets to accomplish my tasks.

Retrieve after hours failed login attempts

There was a potential security incident that occurred that required all login attempts after business hours to be investigated. The following is how I queried SQL to filter for failed login attempts after 18:00.

```
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
```

In the first line you will see that I queried SQL to return all columns from the `log_in_attempts` table using the select all command (`select *`). However, I only wanted attempts made after business hours since that is when the incident occurred. I used the `where` clause and the `>` (greater than) operator; this ensured that the only results returned were those that occurred after 18:00 or 6PM. To return only failed attempts, I used the `and` logical operator to create another condition in addition to the time and that condition was that the `success` column must `=` (equal) 0 (which is representative for failure). The `and` operator returns only those results that include both of my conditions.

Retrieve login attempts on specific dates

A suspicious event occurred on May 9, 2022. To investigate this event, I wanted to review all login attempts which occurred on this day and the day before. The following demonstrates how I queried SQL to filter for login attempts that occurred on specific dates.

```
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
 |
+----------+----------+------------+------------+---------+----------------+--------
```

My query returns only those results that occurred on 2022-05-08 and 2022-05-09. I achieved this using the `where` clause with the `or` operator using the 2 dates as my conditions. The `or` operator returns data that meets either of my conditions. My first

condition was `login_date = '2022-05-08'` which filters for logins on 2022-05-08 and my second condition was `login_date = '2022-05-09'` which filters for logins on 2022-05-09.

Retrieve login attempts outside of Mexico

The team determined that suspicious login activity occurred that was not in Mexico, so I used SQL to filter out login attempts that occurred there.

```
MariaDB [organization]> select * from log_in_attempts where not country like 'MEX%';
+----------+----------+------------+------------+---------+-----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address      | success
 |
+----------+----------+------------+------------+---------+-----------------+--------
```

After querying for all data from the `log_in_attempts` table, I filtered for countries other than Mexico by using the `where` clause with the `not` operator. The `not` operator negates the condition of the country being Mexico. I also used the `like` operator with the `%` wildcard to target countries that start with "MEX" but could include any number of characters afterwards. This is because sometimes Mexico is rendered as `MEX` and `MEXICO`.

Retrieve employees in Marketing

My team wanted to perform security updates on specific employee machines in the Marketing department. I was responsible for getting information on these employee machines.

```
MariaDB [organization]> select * from employees where department = 'Marketing' and of
fice like 'EAST%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
```

I queried the `employees` table to retrieve records of only those Marketing employees with offices in the East building. I used the `where` clause to create the condition that the department must `=` (equal) `'Marketing'` and used the `and` operator to ensure the other condition of the office was also met. I used the `like` operator with the `%` wildcard to return all results that began with "EAST."

Retrieve employees in Finance or Sales

My team then needed to perform a different security update on machines for employees in the Sales and Finance departments.

```
MariaDB [organization]> select * from employees where department = 'Finance' or depar
tment = 'Sales';
+-------------+--------------+----------+------------+-------------+
| employee_id | device_id    | username | department | office      |
+-------------+--------------+----------+------------+-------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153   |
```

I queried SQL by returning all (`select *`) columns from the employees table and used the `where` clause with the `or` operator to filter for employees in the Finance and Sales departments. I used `or` instead of `and` because I wanted employees who were in either department.

Retrieve all employees not in IT

My team needed to make one more update to employee machines. The employees who were in the Information Technology department already had this update, but employees in all other departments needed it.

```
MariaDB [organization]> select * from employees where not department = 'Information T
echnology';
+-------------+--------------+----------+----------------+-------------+
| employee_id | device_id    | username | department     | office      |
+-------------+--------------+----------+----------------+-------------+
```

Therefore, I queried SQL to return all employees except those in the IT department by using the `where` clause and using the `not` operator to negate those in the Information Technology department.

Summary

I used SQL to make many queries that involved me using various clauses, operators, and wildcards to filter the returns of those queries. This was due to the specific nature of the investigations and reports I had to retrieve. Some required filtering for certain times, filtering out certain employees, filtering to include 2 conditions, or filtering to negate a condition. These proficiencies are necessary when dealing with the large quantities of the datasets I was working with.