SQL Queries: Techniques and Wildcards

Project description

I was tasked with conducting security investigations and updates within a fictional organization. Using SQL with filters, I performed various security-related tasks on the organization's database.

```
|      190 | jsoto    | 2022-05-09 | 05:09:21   | USA     | 192.168.25.60   |        0 |
|      191 | cjackson | 2022-05-08 | 06:46:07   | CANADA  | 192.168.7.187   |        0 |
|      192 | bisles   | 2022-05-10 | 08:32:03   | USA     | 192.168.201.40  |        1 |
|      193 | lrodriqu | 2022-05-08 | 07:11:29   | US      | 192.168.125.240 |        0 |
|      194 | jclark   | 2022-05-12 | 14:11:04   | CAN     | 192.168.197.247 |        0 |
|      195 | alevitsk | 2022-05-11 | 06:59:13   | CANADA  | 192.168.236.78  |        1 |
|      196 | acook    | 2022-05-10 | 09:56:48   | CAN     | 192.168.52.90   |        0 |
|      197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |        0 |
|      198 | yappiah  | 2022-05-12 | 10:37:22   | MEXICO  | 192.168.103.106 |        1 |
|      199 | yappiah  | 2022-05-11 | 19:34:48   | MEXICO  | 192.168.44.232  |        0 |
|      200 | jclark   | 2022-05-12 | 01:11:45   | CANADA  | 192.168.91.103  |        1 |
+----------+----------+------------+------------+---------+-----------------+----------+
200 rows in set (0.001 sec)

MariaDB [organization]> SELECT *
    ->
    -> FROM log_in_attempts
    ->
    -> ORDER BY login_date;
+----------+----------+------------+------------+---------+-----------------+----------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+----------+
|      145 | ivelasco | 2022-05-08 | 09:06:02   | CANADA  | 192.168.39.196  |        1 |
|      163 | tmitchel | 2022-05-08 | 09:21:16   | MEX     | 192.168.119.29  |        0 |
|       36 | asundara | 2022-05-08 | 09:00:42   | US      | 192.168.78.151  |        1 |
|      165 | jreckley | 2022-05-08 | 15:28:43   | MEXICO  | 192.168.34.193  |        0 |
|      168 | jlansky  | 2022-05-08 | 13:25:42   | USA     | 192.168.210.94  |        1 |
|      169 | alevitsk | 2022-05-08 | 08:10:43   | CANADA  | 192.168.210.228 |        0 |
|       72 | alevitsk | 2022-05-08 | 12:09:10   | CANADA  | 192.168.139.176 |        1 |
|      101 | sbaelish | 2022-05-08 | 12:01:22   | US      | 192.168.145.158 |        0 |
|      172 | mabadi   | 2022-05-08 | 08:06:50   | US      | 192.168.180.41  |        1 |
|      150 | nmason   | 2022-05-08 | 14:40:02   | CAN     | 192.168.204.124 |        0 |
|       68 | mrah     | 2022-05-08 | 17:16:13   | US      | 192.168.42.248  |        1 |
|       66 | aestrada | 2022-05-08 | 21:58:32   | MEX     | 192.168.67.223  |        1 |
|       53 | nmason   | 2022-05-08 | 11:51:38   | CAN     | 192.168.133.188 |        1 |
|      147 | yappiah  | 2022-05-08 | 06:04:34   | MEX     | 192.168.65.245  |        0 |
|      148 | daquino  | 2022-05-08 | 06:15:55   | CANADA  | 192.168.135.6   |        1 |
|       49 | asundara | 2022-05-08 | 14:00:01   | US      | 192.168.173.213 |        0 |
|       47 | dkot     | 2022-05-08 | 05:06:45   | US      | 192.168.233.24  |        1 |
|       44 | daquino  | 2022-05-08 | 07:02:35   | CANADA  | 192.168.168.144 |        0 |
|       43 | mcouliba | 2022-05-08 | 02:35:34   | CANADA  | 192.168.16.208  |        0 |
|       56 | acook    | 2022-05-08 | 04:56:30   | CAN     | 192.168.209.130 |        1 |
|       80 | cjackson | 2022-05-08 | 02:18:10   | CANADA  | 192.168.33.140  |        1 |
```

Screen capture of filtered organization's database

Retrieve after hours failed login attempts

There was a potential security incident that occurred after business hours (after 18:00). To investigate potential security incidents that occurred after business hours, I created a SQL query to filter for failed login attempts during that time. By selecting data from the log_in_attempts table and using a WHERE clause with an AND operator, I filtered the results to show login attempts that happened after 18:00 and were unsuccessful.

```
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 |
```

Retrieve login attempts on specific dates

A suspicious event took place on 2022-05-09, and I needed to investigate login activity on that day and the previous day. Using a SQL query, I filtered for login attempts that occurred on specific dates by selecting data from the log_in_attempts table and using a WHERE clause with an OR operator to filter for login attempts on either 2022-05-09 or 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     | 192.168.243.140 |       0 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       0 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
```

Retrieve login attempts outside of Mexico

Upon analyzing login attempt data, I noticed a potential issue with attempts originating outside of Mexico. To investigate further, I created a SQL query to filter for login attempts from countries other than Mexico. By joining the log_in_attempts table with the employees table and using a WHERE clause with the NOT operator and LIKE pattern matching (%), I retrieved login attempts made outside of Mexico.

```
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 |       0 |
|        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 |       0 |
```

Retrieve employees in Marketing

For the purpose of updating computers for employees in the Marketing department, I needed to gather information on their machines. To achieve this, I crafted a SQL query to filter for employee machines in the Marketing department located in the East building. By selecting data from the employees table and using a WHERE clause with the department = 'Marketing' condition and an office LIKE 'East%' pattern match, I retrieved the relevant employee machines.

```
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 |
```

Retrieve employees in Finance or Sales

To update machines for employees in the Finance and Sales departments, I needed specific information on employees from these departments. Using a SQL query, I filtered for employee machines in either the Finance or Sales department by selecting data from the employees table and using a WHERE clause with the OR operator to filter for employees in either department.

```
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   |
```

Retrieve all employees not in IT

To perform a final security update on employees not belonging to the Information Technology department, I gathered information on these employees. By creating a SQL query, I filtered for employee machines from employees not in the IT department using a WHERE clause with the NOT operator.

```
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   |
```

Summary

During this practice exam for my Google Cybersecurity Professional Certification, I applied SQL filters to conduct security investigations and updates in a fictional organization's database. By retrieving after-hours failed login attempts, login attempts on specific dates, login attempts outside of Mexico, employees in specific departments, and employees not in the IT department, I demonstrated proficiency in utilizing SQL commands and operators to retrieve specific information and address potential security issues.

In addition to the basic filters and searches, I also gained proficiency in utilizing more advanced SQL tools during the course to analyze and manipulate data effectively. Some of the advanced techniques and tools I learned include:

1. **Aggregate Functions**: I explored the use of aggregate functions such as COUNT, SUM, AVG, MIN, and MAX. These functions allowed me to perform calculations and obtain aggregated results from large datasets. For example, I could count the number of login attempts, calculate the average login time, or determine the total failed login attempts.
2. **Joins**: I learned about different types of joins, including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN. Joins enabled me to combine data from multiple tables based on common columns, allowing for more comprehensive analysis and insights. For instance, I could join the log_in_attempts table with the employees table to retrieve login information along with employee details.
3. **Subqueries**: I explored the concept of subqueries, which involve nesting one query within another. Subqueries allowed me to retrieve data from one query and use it as a filter or condition in another query. This technique provided more flexibility and precision in retrieving specific information based on complex conditions.
4. **Sorting and Ordering**: I utilized the ORDER BY clause to sort the retrieved data in ascending or descending order based on one or more columns. This feature helped me organize and analyze data in a more structured and meaningful way, facilitating easier identification of patterns or anomalies.
5. **Grouping** and Aggregating Data: I learned about the GROUP BY clause, which enabled me to group data based on specific columns. Combined with aggregate functions, this allowed me to summarize and analyze data at different levels of granularity. For example, I could group login attempts by date or department and calculate the total number of login attempts or average login time for each group.

Additionally, I learned about the power of wildcards, such as the percentage sign (%) wildcard with the LIKE operator, which enables pattern matching and flexible search capabilities. By leveraging these advanced SQL techniques, I expanded my capabilities in conducting in-depth analysis, creating complex queries, and generating meaningful insights from the organization's database. These tools provided me with a solid foundation for performing advanced security investigations and addressing potential vulnerabilities more effectively.