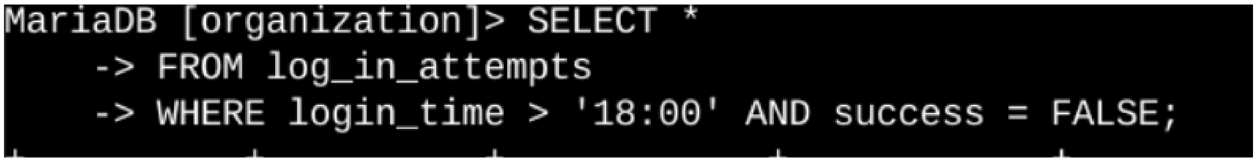
Apply filters to SQL queries

Project Description:
In order to enhance system security, I have been assigned the task of ensuring system safety, investigating potential security issues, and updating employee computers as necessary. Below are examples of how I utilized SQL filters to perform these security-related tasks.

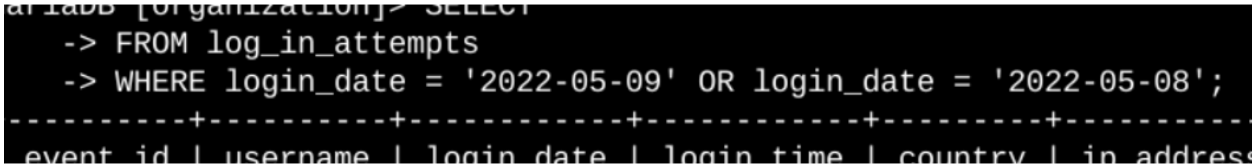
1. Retrieve after-hours failed login attempts:



```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00' AND success = FALSE;
```

To investigate potential security incidents that occurred after business hours (after 18:00), I created a SQL query to filter for failed login attempts that happened during this time. The query selects data from the `log_in_attempts` table, filtering for login attempts after 18:00 and unsuccessful attempts using the conditions `login_time > '18:00'` and `success = FALSE`, respectively.

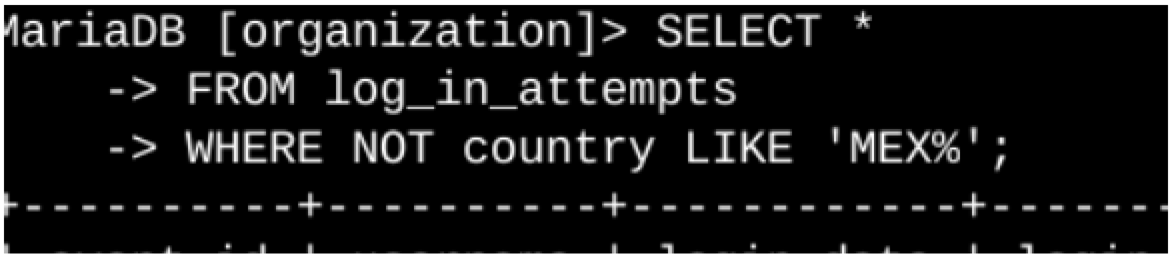
2. Retrieve login attempts on specific dates:



```
ariaDB [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
```

To investigate a suspicious event on a specific date (e.g., 2022-05-09), I created a SQL query to filter for login attempts on that date or the day before. The query selects data from the `log_in_attempts` table, filtering for login attempts on either 2022-05-09 or 2022-05-08 using the conditions `login_date = '2022-05-09'` and `login_date = '2022-05-08'`.

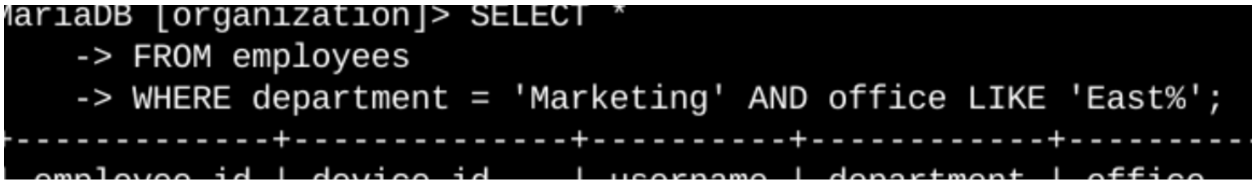
3. Retrieve login attempts outside of Mexico:



```
lariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------
```

After analyzing the organization's login attempt data, I discovered potential issues with attempts outside of Mexico. To investigate further, I created a SQL query to filter for login attempts occurring in countries other than Mexico. The query selects data from the `log_in_attempts` table, using the condition `NOT (country LIKE 'MEX%')` to exclude login attempts associated with Mexico.

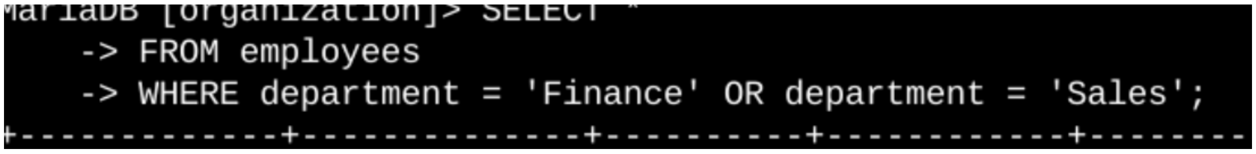
4. Retrieve employees in the Marketing department:



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

To identify employee machines requiring updates within the Marketing department located in the East building, I created a SQL query to filter for relevant employee machines. The query selects data from the `employees` table, filtering for employees in the Marketing department and the East building using the conditions `department = 'Marketing'` and `office LIKE 'East%'`.

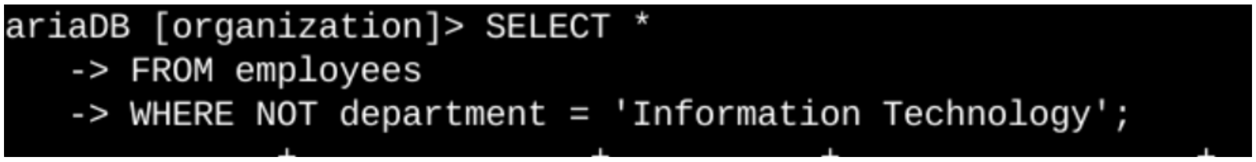
5. Retrieve employees in the Finance or Sales departments:



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

To obtain information about employees in the Finance or Sales departments, whose machines require separate security updates, I created a SQL query to filter for these employees. The query selects data from the `employees` table, filtering for employees in either the Finance or Sales departments using the conditions `department = 'Finance'` and `department = 'Sales'`.

6. Retrieve all employees not in IT:



```
ariaDB [organization]> SELECT *
   -> FROM employees
   -> WHERE NOT department = 'Information Technology';
```

To identify employees for a final security update, specifically those not belonging to the Information Technology department, I created a SQL query to filter for relevant employee machines. The query selects data from the `employees` table, filtering for employees not in the Information Technology department using the condition `department <> 'Information Technology'`.

Summary:

By effectively utilizing SQL filters, I successfully obtained specific information on login attempts and employee machines. The queries involved filtering data from the `log_in_attempts` and `employees` tables, utilizing operators such as AND, OR, and NOT to refine the results. Additionally, the LIKE operator with the percentage sign (%) wildcard was used to match patterns when necessary.