Apply filters to SQL queries

Project description

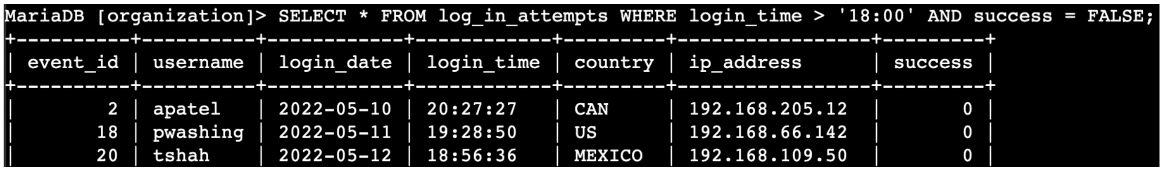
I am tasked with extracting specific employee and login data from the database to support a security investigation and system updates.

The responsibilities include:

1. Retrieving failed login attempts outside business hours
2. Filtering logins by specific dates
3. Identifying logins from outside Mexico
4. Getting data on selected Marketing employees
5. Listing employees in Finance or Sales
6. Excluding employees from the IT department

Retrieve after hours failed login attempts

The team is investigating failed login attempts made **after business hours (after 18:00)**.
All **unsuccessful logins** that happened **after 18:00** need to be investigated.



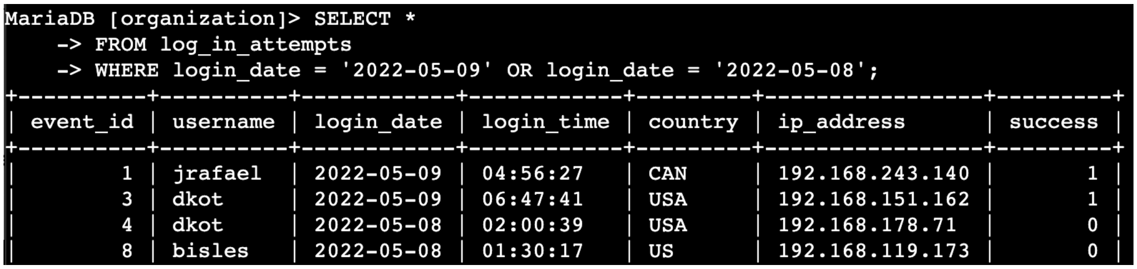
```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_time > '18:00' AND success = FALSE;
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        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 |
```

This SQL query retrieves **failed login attempts** made **after 18:00** from the `log_in_attempts` table. It uses a `WHERE` clause with two conditions:

- `login_time > '18:00'` filters for login attempts after business hours.
- `success = 0` ensures only **unsuccessful** attempts are returned.
- The `AND` operator ensures both conditions are true for each result.

Retrieve login attempts on specific dates

Suspicious event occurred on **‘2022-05-09’**. Every login attempt that occurred on this day and the day before **‘2022-05-08’** needs to be investigated.



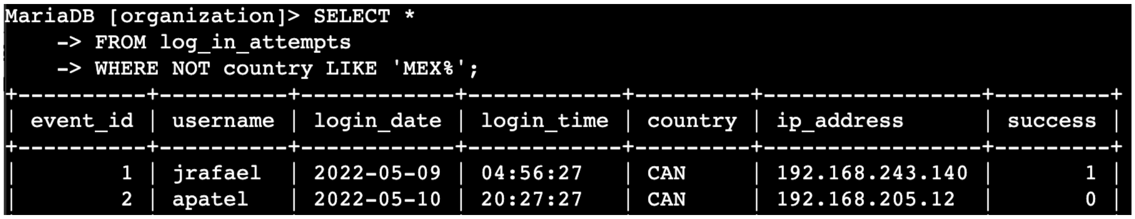
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|        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 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
```

This SQL query retrieves all login attempts that occurred on **May 8 or May 9, 2022**, from the `log_in_attempts` table. It uses a `WHERE` clause with an `OR` operator to match either date:

- `login_date = '2022-05-09'` filters for logins on May 9, 2022.
- `login_date = '2022-05-08'` filters for logins on May 8, 2022.
- The `OR` operator returns records matching **either** date.

Retrieve login attempts outside of Mexico

After retrieving dates for login attempts, the team is now investigating login attempts outside of Mexico.



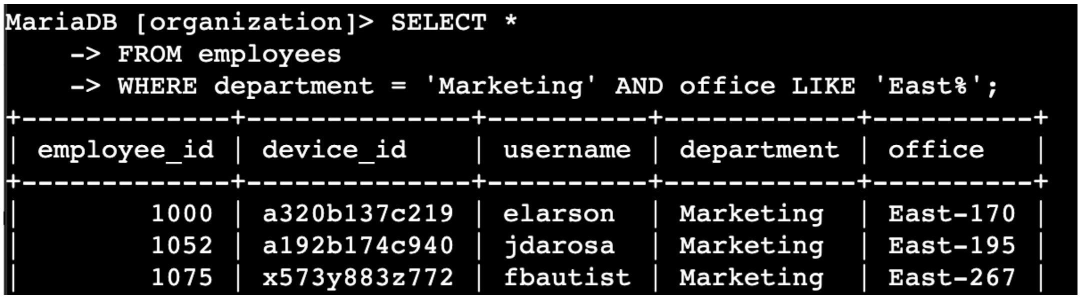
```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0 |
```

This SQL query retrieves all login attempts from **countries other than Mexico**. The `log_in_attempts` table is filtered using a `WHERE` clause with `NOT` and `LIKE` to exclude entries where the country starts with "MEX":

- `NOT LIKE 'MEX%'` excludes records where the country is labeled as "MEX" or "MEXICO".
- The `%` wildcard matches any characters that follow "MEX".
- This ensures only login attempts from **non-Mexican sources** are returned.

Retrieve employees in Marketing

The team wanted to update the employee machines in the Marketing department. I need to retrieve the information about employees in the ‘Marketing’ department who are located in all offices in the East building (such as **'East-170'** or **'East-320'**).



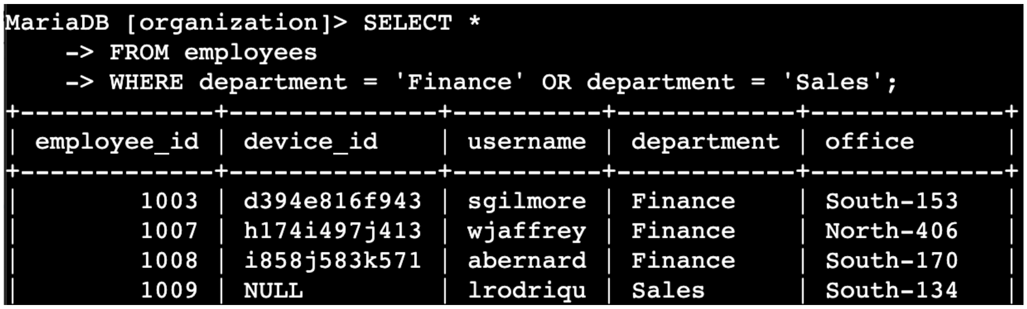
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+-------------+--------------+----------+------------+----------+
| employee_id | device_id    | username | department | office   |
+-------------+--------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa  | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing  | East-267 |
```

This SQL query retrieves all employees who work in the **Marketing department** and are located in the **East building**. It uses a `WHERE` clause with the `AND` operator to apply both filters:

- `department = 'Marketing'` filters employees by department.
- `office LIKE 'East%'` filters by building, matching any office that starts with "East" (e.g., "East101", "East-Wing").
- The `%` wildcard allows matching any characters after "East".

Retrieve employees in Finance or Sales

The machines for 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.



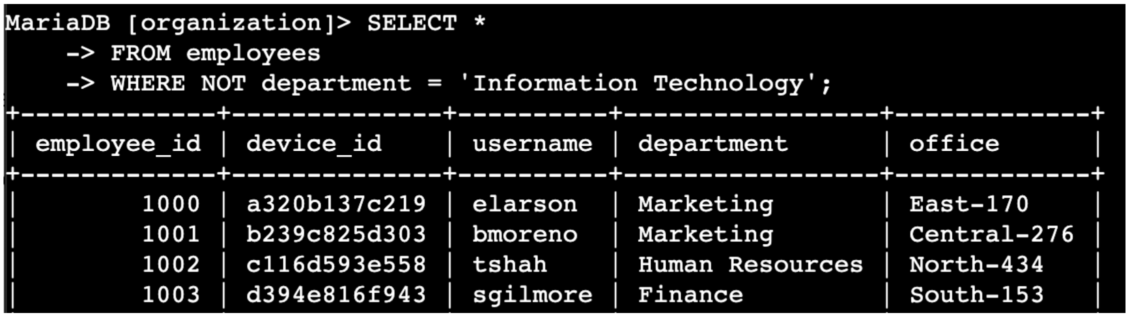
```
MariaDB [organization]> SELECT *
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+-------------+--------------+----------+------------+-------------+
| employee_id | device_id    | username | department | office      |
+-------------+--------------+----------+------------+-------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153   |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406   |
|        1008 | i858j583k571 | abernard | Finance    | South-170   |
|        1009 | NULL         | lrodriqu | Sales      | South-134   |
```

This SQL query retrieves all employees who belong to either the **Finance** or **Sales** departments. It uses a `WHERE` clause with the `OR` operator to include employees from both departments:

- `department = 'Finance'` filters for Finance employees.
- `department = 'Sales'` filters for Sales employees.
- The `OR` operator ensures records from **either** department are returned.

Retrieve all employees not in IT

The team needs to make one more update on employees who are not in the Information Technology department. To make the update, I first have to get information on these employees.



```
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   |
```

This SQL query retrieves all employees **not** in the **Information Technology (IT)** department. It uses a `WHERE` clause with the `NOT` operator to exclude IT employees:

- The `NOT` operator filters out employees in the IT department.
- This returns all employees from departments **other than IT**.

Summary

SQL queries were used to extract specific data from the `log_in_attempts` and `employees` tables. Filtering techniques included:

- Logical operators: `AND`, `OR`, and `NOT` to refine query conditions.
- Pattern matching: `LIKE` combined with `%` wildcard to filter by text patterns.

These methods are used to enable targeted retrieval of login attempts and employee information required for the tasks.