**Apply Filters to SQL Queries**

**Project Description**

My organization is working to make its system more secure. I am responsible for ensuring the system is safe, investigating all potential security issues, and updating employee computers as needed. The following steps show how I used SQL with filters to carry out security-related tasks.

**1. Retrieve After-Hours Failed Login Attempts**

A possible security incident happened after working hours (after 18:00). I was tasked with investigating all failed login attempts outside regular hours.

I created the following SQL query to identify failed login attempts after 18:00:

**SELECT \* FROM log\_in\_attempts**

**WHERE login\_time > '18:00' AND success = FALSE;**

The first condition **(`login\_time > '18:00'`)** filters attempts made after 6:00 PM, and the second condition **(`success = FALSE`)** filters out unsuccessful attempts. This query helped me identify potential security breaches occurring after working hours.

**2. Retrieve Login Attempts on Specific Dates**

A suspicious event was recorded on May 9, 2024, and I needed to investigate any login activity on that date and the day before.

I wrote the following SQL query to filter login attempts on May 8 or May 9, 2024:

**SELECT \* FROM log\_in\_attempts**

**WHERE login\_date = '2024-05-08' OR login\_date = '2024-05-09';**

This query returned login attempts made on May 8 or 9, 2024, allowing me to focus my investigation on those dates.

**3. Retrieve Login Attempts Outside of Mexico**

After reviewing the data, I discovered potential issues with login attempts originating from outside Mexico. These attempts required further investigation.

I wrote a query to filter login attempts that occurred outside Mexico:

**SELECT \* FROM log\_in\_attempts**

**WHERE location NOT LIKE 'MEX%';**

This query retrieved all login attempts made from locations other than Mexico. The use of `LIKE` with the pattern `MEX%` captured variations in how Mexico was recorded, including 'MEX' and 'MEXICO.'

**4. Retrieve Employees in Marketing**

My team needed to update employees' computers in the Marketing department, specifically those based in the East building.

I created a query to filter data for Marketing department employees located in the East building:

**SELECT \* FROM employees**

**WHERE department = 'Marketing' AND location = 'East';**

This query identified employees from the Marketing department in the East building, allowing the team to target the necessary security updates efficiently.

**5. Retrieve Employees in Finance or Sales**

We also needed to update computers for employees in the Finance and Sales departments, which required different security updates.

I wrote the following query to extract information on employees from these two departments:

**SELECT \* FROM employees**

**WHERE department = 'Finance' OR department = 'Sales';**

This query retrieved employees from the Finance and Sales departments, streamlining the process of applying the relevant updates.

**6. Retrieve All Employees Not in IT**

My team required data on employees outside the IT department to apply one final security update.

**I gathered this information using the following query:**

**SELECT \* FROM employees**

**WHERE department! = 'IT';**

This query helped us identify employees outside the IT department, ensuring the security update was properly applied to the correct group.

**Summary**

I used SQL queries with filters throughout this project to gather specific information about login attempts and employee computers. I worked with two tables, `log\_in\_attempts` and `employees, ` using operators like `AND,` `OR,` and `NOT` to refine the results. Additionally, I utilized the `LIKE` operator and the `%` wildcard to match patterns in the data where necessary, allowing me to address security concerns effectively and efficiently.