

Apply Filters to SQL Queries

Cybersecurity Portfolio Project

Project Description

As a security professional at a large organization, I investigated potential security issues involving login attempts and employee machines. This project demonstrates my ability to use SQL queries with various filters (AND, OR, NOT, LIKE) to retrieve specific records from organizational databases. Through these queries, I identified failed login attempts after business hours, suspicious login activity on specific dates, and retrieved employee information for targeted security updates across different departments and locations.

Retrieve After Hours Failed Login Attempts

To investigate a potential security incident that occurred after business hours, I needed to query the log_in_attempts table to identify all failed login attempts after 18:00 (6:00 PM). The query uses two conditions combined with the AND operator: one to filter for login times after 18:00 and another to filter for failed attempts (where success = 0).

```
SELECT *  
FROM log_in_attempts  
WHERE login_time > '18:00' AND success = 0;
```

This query filters the log_in_attempts table to return only records where the login_time column contains a value greater than '18:00' AND the success column contains a value of 0 (indicating a failed login attempt). The AND operator ensures both conditions must be true for a record to be included in the results. This helps identify potential unauthorized access attempts occurring outside of normal business hours.

Retrieve Login Attempts on Specific Dates

A suspicious event occurred on 2022-05-09, requiring investigation of all login attempts on that day and the preceding day (2022-05-08). I used the OR operator to retrieve login attempts from either date, as we needed to examine activity from both days to understand the full scope of the incident.

```
SELECT *  
FROM log_in_attempts  
WHERE login_date = '2022-05-09' OR login_date = '2022-05-08';
```

This query uses the OR operator to filter the log_in_attempts table for records matching either of the two specified dates. The OR operator returns results where at least one of the conditions is true. This allows us to capture all login activity across both days in a single query, which is essential for investigating the timeline of the suspicious event and identifying any patterns or related activity.

Retrieve Login Attempts Outside of Mexico

After determining that suspicious login activity did not originate from Mexico, I needed to investigate login attempts from all other countries. The challenge was that Mexico appears in the database as both 'MEX' and 'MEXICO', requiring the use of the LIKE operator with the NOT operator to exclude both variations.

```
SELECT *
FROM log_in_attempts
WHERE NOT country LIKE 'MEX%';
```

This query uses the NOT operator combined with the LIKE operator and the wildcard pattern 'MEX%' to exclude all login attempts from Mexico. The LIKE operator allows pattern matching, where the percent sign (%) represents any number of characters. The pattern 'MEX%' matches both 'MEX' and 'MEXICO', ensuring both variations are excluded. The NOT operator negates this condition, returning only records where the country does NOT match the pattern, effectively showing login attempts from all countries except Mexico.

Retrieve Employees in Marketing

The security team needed to perform updates on employee machines in the Marketing department located specifically in the East building. This required querying the employees table with filters for both department and office location using pattern matching for the building identifier.

```
SELECT *
FROM employees
WHERE department = 'Marketing' AND office LIKE 'East%';
```

This query combines an exact match filter with a pattern match filter using the AND operator. The first condition (department = 'Marketing') performs an exact string match for the Marketing department. The second condition (office LIKE 'East%') uses the LIKE operator with the wildcard pattern 'East%' to match any office in the East building, regardless of the specific office number (such as East-170, East-320, etc.). Both conditions must be true for an employee record to be returned, ensuring we target only Marketing employees in East building offices.

Retrieve Employees in Finance or Sales

A different security update was required for machines belonging to employees in both the Sales and Finance departments. Unlike the previous query, this update applies to employees in either department regardless of their office location.

```
SELECT *  
FROM employees  
WHERE department = 'Sales' OR department = 'Finance';
```

This query uses the OR operator to retrieve records matching either of two conditions. An employee record is returned if the department column contains either 'Sales' OR 'Finance'. The OR operator is appropriate here because we need to include all employees from both departments in our results. This is more efficient than running two separate queries and allows the security team to process updates for both departments in a single operation.

Retrieve All Employees Not in IT

The final security update needed to be applied to all employee machines except those in the Information Technology department, as IT employees had already received this update. This required filtering out the IT department while including all other departments.

```
SELECT *  
FROM employees  
WHERE NOT department = 'Information Technology';
```

This query uses the NOT operator to exclude records matching a specific condition. By placing NOT before the condition (department = 'Information Technology'), the query returns all employee records where the department is NOT 'Information Technology'. This is more efficient than listing all other departments with multiple OR conditions, especially in an organization with many departments. The result includes employees from all departments except IT, allowing the security team to target the update to the appropriate machines.

Summary

Through this project, I demonstrated proficiency in using SQL to investigate security incidents and retrieve specific data for security operations. I successfully applied multiple SQL filtering techniques including time-based filters (login_time > '18:00'), date filters (login_date), pattern matching with LIKE and wildcards (%), and logical operators (AND, OR, NOT) to combine multiple conditions. These queries enabled me to identify failed after-hours login attempts, investigate suspicious activity on specific dates, analyze login attempts by geographic location, and target employee machines across different departments and office locations for security updates. The ability to construct precise SQL queries is essential for efficient security investigations and enables rapid response to potential threats while minimizing impact on

normal operations.