Apply filters to SQL queries

Project description

In this scenario, the team is tasked with investigating and addressing security concerns related to employee login attempts and implementing specific updates on employee machines. The goal is to use SQL queries to extract relevant information from the database and filter it based on different conditions such as department, login times, and office locations. By applying SQL filters, we aim to identify and address security issues, streamline updates, and enhance overall system security.

Retrieve after hours failed login attempts

```
SELECT *

FROM log_in_attempts

WHERE login_time > '18:00:00' AND success = 0;
```

This SQL query checks the log_in_attempts table to find all the records where the login attempt happened after 6:00 PM and wasn't successful (marked by a 0 in the success column). The AND operator connects these two conditions, making sure both have to be true for a record to be included in the results

Retrieve login attempts on specific dates

```
SELECT *

FROM log_in_attempts

WHERE login_date IN ('2022-05-08', '2022-05-09');
```

This SQL query retrieves all rows from the log_in_attempts table where the login_date is either '2022-05-08' or '2022-05-09'. The IN operator is used to specify a list of values for the login_date column, and the SELECT * statement retrieves all columns for the matching rows.

Retrieve login attempts outside of Mexico

```
SELECT *

FROM log_in_attempts

WHERE country NOT LIKE 'MEX%';
```

this query will include login attempts where the country is anything other than 'MEX' or 'MEXICO'.

Retrieve employees in Marketing

```
SELECT *

FROM employees

WHERE department = 'Marketing' AND office LIKE 'East%';
```

This query picks out employees who work in the Marketing department and are in the East building. It looks at the "employees" table, only keeping rows where the department is 'Marketing' and the office starts with 'East'. The AND operator ensures both conditions must be true for a row to be in the result. The LIKE condition with 'East%' is like a shortcut saying, "Find offices that start with 'East', and whatever comes after that is okay." So, it captures offices like 'East-170', 'East-320', and so on.

Retrieve employees in Finance or Sales

```
SELECT *

FROM employees

WHERE department IN ('Sales', 'Finance');
```

Query retrieves all rows from the employees table where the department is either 'Sales' or 'Finance'. The IN operator is used to specify a list of values for the department column, and the SELECT * statement retrieves all columns for the matching rows.

Retrieve all employees not in IT

```
SELECT *

FROM employees

WHERE department <> 'Information Technology';
```

This SQL query retrieves all rows from the employees table where the department is not equal to 'Information Technology'. The <> is a symbol for "not equal to," and the SELECT * statement retrieves all columns for the matching rows.

Summary

Throughout the tasks, we utilized SQL queries to address specific security concerns and update employee machines in a fictional scenario. We retrieved information on failed login attempts after 6:00 PM, identified employees in the Marketing department in the East building, selected employees in Sales or Finance departments, and excluded those in the Information Technology department. Screenshots and descriptions were provided for each query, showcasing the use of LIKE for pattern matching, filtering dates and times, combining conditions with AND and OR, and using NOT in filters. These tasks collectively demonstrate

the versatility of SQL in extracting targeted data, aiding security investigations, and facilitating updates for specific user groups.