Apply filters to SQL queries

Project description

My organization is investigating security issues to help keep our systems secure. I found some potential security issues that involve login attempts and employee machines. To retrieve records and perform the tasks needed from different datasets and investigate the potential security issues was applied SQL filters.

Retrieve after hours failed login attempts

I recently discovered a potential security incident that occurred after business hours. To investigate this I needed to query the log_in_attempts table and review after hours login activity to identify all failed login attempts that occurred after 18:00.

The following code demonstrates how I created a SQL query to filter for failed login attempts that occurred after business hours.

```
SELECT *
FROM log_in_attempts
WHERE log_in_time > '18:00' AND success = FALSE;
```

First, I started selecting all data from the `log_in_attempts` table. Then, I used the clause `WHERE` with an `AND` operator to filter for all failed attempts that occurred after business hours (18:00) and were unsuccessful. The first condition `log_in_time > '18:00'` filters the time and the second condition `success = FALSE` is used to filter the failed login attempts.

Retrieve login attempts on specific dates

A suspicious event occurred on 2022-05-09. To investigate this event was needed to review all login attempts that occurred on this day and the day before.

The following code demonstrates how I created a SQL query to retrieve for specific dates.

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

First, I selected all data from the `login_attempts` table. Then, I used the clause `WHERE` with an `OR` operator to output all login attempts that occurred on 2022-05-09 or 2022-05-08. The first condition `login_date = '2022-05-09'` filters the specific date 2022-05-09 that needs to retrieve the login attempts. Same as the first condition, the second condition `login_date = '2022-05-08` filters for logins on 2022-05-08.

Retrieve login attempts outside of Mexico

The team has determined there's been suspicious activity with login attempts that didn't originate in Mexico. To complete this task I had to investigate login attempts that occurred outside Mexico.
The code below shows the SQL query that I created to identify all login attempts that occurred outside Mexico.

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

This query returns all login attempts that occurred in countries other than Mexico. First, I selected all data from the `login_attempts` table. Then, I used the clause `WHERE` with `NOT` operator to filter for countries other than Mexico. Afterwards, I used `LIKE` with `MEX%` as a pattern because the dataset represents Mexico as MEX and MEXICO. The percentage (`%`) with `LIKE` allows searching for data that matches a specific pattern rather than an exact value.

Retrieve employees in Marketing

The team wants to update the machines in the Marketing department. Is my responsibility to get information on these employee machines. To complete this task, it needs to perform filters in SQL to create a query that identifies all employees in the Marketing department for all offices in the East building.
The following code shows how this task was executed.

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

First, I selected all data from the `employees` table. Then, I used the clause `WHERE` with an `AND` operator to output all the employees in the Marketing department that are located and in the East building. Afterwards, I used `LIKE` with `East%` as a pattern because the data in the column `office` represents the East building with a specific office number. The first condition `department = 'Marketing'` filters the employees in the Marketing department and the second condition `office LIKE = 'East%'` was applied to filter the offices located at the East building.

Retrieve employees in Finance or Sales

A different security update needs to be performed on the machines for the employees in the Sales and Finance departments. For the update it is necessary to identify all employees that work in those departments.
The following code demonstrates how this task was executed:

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

This query returns all employees in the departments of Sales and Finance. First, I selected all data from the `employees` table. Then, I used the clause `WHERE` with an `OR` operator to output all the employees in the departments of Sales and Finance. The first condition `department = 'Sales'` filters the employees in the Sales

department. The second condition `department = 'Finance'` filters the employees in the Finance department.

Retrieve all employees not in IT

The team needs to perform one more update to the employee machines. All the employees need this update except the employees of the Information Technology department which already have this update done.
The code below shows how I created a query to filter for employee machines from all employees except the Information Technology department.

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

This query returns all employees that are not in the Information Technology department. First, I selected all data from the `employees` table. Then, I used the clause `WHERE` with a `NOT` operator to output all the employees that are not in the Information Technology department.

Summary

To help keep the organization systems secured I applied filters to SQL queries to get specific information on login attempts and employees machines. I used two different tables to get the specific tasks done, the `employees` table and the `login_attempts` table. I used `AND`, `OR` and `NOT` operators to filter information needed for each task. I also applied `LIKE` and the percentage sign (`%`) to search for a specific pattern.