Apply filters to SQL queries

Project description

A detailed description of how I use SQL queries to review the potential security threats caused by login attempts and employee machines at the organisation where I work.

Retrieve after-hours failed login attempts

After business hours (after 18:00), there was a possible security issue. Investigations must be conducted into each failed attempt at logging in after hours.

The code below shows the SQL queries used to review the login attempts after work hours (after 18:00) or 6 p.m.

```
SELECT *
FROM log_in_attempts
WHERE login_time > '18:00' AND SUCCESS = 'FALSE';
```

This search restricts its results to unsuccessful login attempts that took place after `18:00`. I began by choosing all of the information from the `log_in_attempts` database.

Then, I filtered my data using a `WHERE` clause and an `AND` operator to report only failed login attempts that took place after `18:00`.

Login attempts made after `18:00` are filtered out by the first criteria, `login_time > '18:00'`. The second criterion, `success = 'FALSE'`, excludes unsuccessful login attempts.

Retrieve login attempts on specific dates

On 2022-05-09, an irregular occurrence took place so I deemed it necessary to look into any login activity that took place on 2022-05-09 or the day prior.

The code below demonstrates how I did so.

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

This search finds all attempts to log in that took place on either `2022-05-09` or `2022-05-08`. I began by choosing all of the information from the `log_in_attempts` database. Then, I filtered my findings using a `WHERE` clause and an `OR` operator to only show login attempts that happened on either `2022-05-09` or `2022-05-08`. Logins made on or after `2022-05-09` are excluded by the first criterion, `login_date = '2022-05-09'`. With the second condition, which reads `"login_date = '2022-05-08',"` only logins made on May 8, 2022 are considered.

Retrieve login attempts outside of Mexico

After some investigation, I discovered a problem with the login attempts logs that took place outside of Mexico after looking at the company's data on login attempts. These login attempts need to be looked into.

I used the following code to investigate the potential risks

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

My query is shown in the first half of the screenshot, and some of the output is shown in the second.
This search retrieves all attempts at login made outside of Mexico. I began by choosing all of the information from the `log_in_attempts` database.

I then applied a `WHERE` clause with a `NOT` to exclude all nations besides Mexico. Because the dataset refers to Mexico as `MEX` and `MEXICO`, I used `LIKE` with `MEX%` as the pattern to match. When used with `LIKE`, the percentage symbol (`%`) stands in for any amount of arbitrary characters.

Retrieve employees in Marketing

A few Marketing department employees computers need to be updated, according to my team. I need to find out which staff computers need updating in order to achieve this.

The code below demonstrates the SQL queries I used to figure out the necessary information

```
SELECT *
```

```
FROM employees
WHERE department = ‘Marketing’ AND office LIKE ‘East%’;
```

This specific query displays all employees in the marketing department in the East building. I began by choosing all of the information from the employee's table. Then, to find workers who are employed by the Marketing division and the East building, I utilised a `WHERE` clause with `AND` operators. Given that the information in the office column refers to the East building with the given office number, I used `LIKE` with `East%` as the pattern to match. The first criteria, `department = "Marketing,"` selects only those who work in the Marketing division. The office `LIKE 'East%'` component of the second condition filters for workers in the East building.

Retrieve employees in Finance or Sales

Additionally, the machines used by staff members in the sales and finance divisions need to be upgraded.
I can only receive personnel data from these two departments because I require a separate security upgrade.

The code below demonstrates how I used the SQL queries to filter the database to get the required information.

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

All personnel in the sales and finance divisions are returned by this query. I began by choosing all of the information from the `employees` table. I then used the `OR` and the `WHERE` clause to select workers in the finance and sales divisions. Because I wanted every employee in any department, I utilised the `OR` operator rather than the `AND` operator. Employees from the Finance department are filtered according to the first criteria, `department = "Finance". Department = 'Sales'` in the second criterion excludes personnel from the Sales department.

Retrieve all employees not in IT

My team still needs to change the security settings for those who work outside the information technology division. I must first gather information on these employees before I can make the upgrade.

The code below demonstrates how I used SQL queries to filter for the required information

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

All workers who are not in the information technology department are returned by the query. I began by choosing all of the information from the `employees` table. Then, to filter out workers who weren't in this department, I used a `WHERE` clause with `NOT` operators to do this.

Summary

To obtain detailed information on login attempts and employee workstations, I used SQL queries to filter out the required information. `employees` and `log_in_attempts` are the two tables I utilised. I filtered for the precise data required for each task using the `AND`, `OR`, and `NOT` operators. In order to search for trends, I also utilised `LIKE` and the wildcard percentage symbol `(%)`.