Apply filters to SQL queries

Project description

In this project, I investigated login activity and employee data using SQL to identify potential security risks and support system updates. The scenario involved a series of tasks that required filtering data from `log_in_attempts` and `employees` tables based on conditions like time of day, login success, department, office location, and country of origin.
By applying SQL filters using `AND, OR, NOT,` and `LIKE` operators, I was able to narrow down and retrieve relevant records to support cybersecurity decisions.

Retrieve after hours failed login attempts

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

This query retrieves all records from the `log_in_attempts` table where two conditions are met:

1. The `login_time` is after 18:00:00, which is considered outside of normal business hours.
2. The `success` value is `0`, which indicates that the login attempt failed.

The `AND` operator is used to ensure that both conditions must be true for a record to be included in the result. This helps the team focus specifically on login attempts that failed and happened after hours, a common indicator of suspicious activity.

Retrieve login attempts on specific dates

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

This SQL query retrieves all records from the `log_in_attempts` table where the `login_date` is either `2022-05-08` or `2022-05-09.`

The `OR` operator is used to check for either of the two specific dates. This allows us to view all login activity that took place on the day of the suspicious event (May 9) and the day before (May 8), which may help identify any unusual patterns or related login attempts leading up to the event.

Retrieve login attempts outside of Mexico

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

This SQL query retrieves all records from the `log_in_attempts` table where the value in the `country` column does **`not`** start with `'MEX'.`

The `LIKE 'MEX%'` condition matches any country value that starts with `'MEX',` including both `'MEX'` and `'MEXICO'`. By using `NOT` in front of it, we exclude those records and only return login attempts that occurred **outside of Mexico**.

This helps narrow the investigation to activity that didn't originate from within the country, which is useful for tracking suspicious foreign access.

Retrieve employees in Marketing

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

This SQL query retrieves all records from the `employees` table where two conditions are met:

- The employee works in the Marketing department `(department = 'Marketing')`
- The employee's office is in the **`East building,`** as indicated by the office name starting with `'East' (office LIKE 'East%')`

The `LIKE 'East%'` filter ensures that values such as `'East-170', 'East-320'`, and any other office starting with `'East'` are included. The `AND` operator combines both conditions, so only employees matching both criteria are returned.

This helps isolate employees who need the specific security updates for Marketing machines in the East building.

Retrieve employees in Finance or Sales

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

This SQL query pulls all records from the `employees` table where the employee's department is either **Finance** or **Sales**. It uses the `OR` operator to return employees who meet **at least one** of these conditions:

- `department = 'Finance'`
- `department = 'Sales'`

Each condition is fully written out to clearly specify the `department` column for both values, ensuring accuracy. This allows your team to target the right machines in both departments for the necessary security update.

Retrieve all employees not in IT

```
SELECT *
FROM employees
WHERE department != 'Information Technology';
```

This SQL query selects all records from the `employees` table where the `department` is **not** equal to `'Information Technology'`.

- The `!=` operator filters out any employee in the **Information Technology** department.
- This ensures that only employees in **other departments** are returned, since they still need the security update.

The `NOT` condition could also be written as:

```
WHERE NOT department = 'Information Technology'
```

Both forms will work, but `!=` is more common and concise in this context.

Summary

Throughout this project, I used SQL queries to filter and retrieve login attempts occurring after business hours, on specific dates, and outside of Mexico helping to isolate suspicious activity. I also pulled employee records based on department and office location to assist with targeted machine updates.
Each query helped simulate real world tasks a cybersecurity analyst or IT professional might face when monitoring systems or deploying updates. The project demonstrates my ability to apply conditional logic in SQL to investigate and support operational decisions.