
Apply filters to SQL queries

Project description

My organization needed to investigate potential security issues, so I analyzed login attempts and employee data using SQL queries. I filtered information from the `log_in_attempts` and `employees` tables to identify failed login attempts, track down suspicious activity patterns, and figure out which employee machines needed security updates.

Retrieve after hours failed login attempts

A potential security incident happened after business hours (after 18:00), so I needed to pull all the failed login attempts from that timeframe.

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

This query grabs failed login attempts that happened after 18:00. I started by selecting everything from the `log_in_attempts` table, then added a `WHERE` clause with `AND` to narrow it down. The first part (`login_time > '18:00'`) pulls attempts after business hours. The second part (`success = FALSE`) filters for failed attempts only. Both conditions need to be true for a record to show up.

Retrieve login attempts on specific dates

Something suspicious went down on 2022-05-09, so I needed to check all login attempts from that day and the day before.

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

This query pulls login attempts from both May 9th and May 8th. I went with `OR` in the `WHERE` clause because I needed records from either date. The first condition grabs May 9th attempts, the second grabs May 8th. Any record matching either date shows up in the results.

Retrieve login attempts outside of Mexico

Our security team figured out the suspicious activity didn't come from Mexico, so I needed to investigate attempts from everywhere else.

```
SELECT *
FROM log_in_attempts
```

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

This query returns login attempts from any country except Mexico. I went with NOT in the WHERE clause to exclude Mexico entries. The tricky part here is that the country column has both 'MEX' and 'MEXICO' for Mexico, so I needed LIKE with 'MEX%' to catch both versions. The percentage sign matches any characters after 'MEX'. Then NOT flips it to exclude those matches and give me everything else.

Retrieve employees in Marketing

The security team wanted to update computers for Marketing employees in the East building, so I pulled that employee data.

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

This query gets all Marketing employees working in the East building. I combined two conditions with AND because both had to be true. First condition (department = 'Marketing') filters for Marketing staff. Second condition (office LIKE 'East%') filters for East building offices. I needed LIKE here because office values include the building name plus room numbers like East-170 or East-320. The % wildcard handles whatever comes after 'East'.

Retrieve employees in Finance or Sales

The team had a different security update for Finance and Sales employees, so I needed to pull just those departments.

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

This query gets employees from Finance and Sales. I went with OR because I needed people from either department, not both. First condition pulls Finance employees, second pulls Sales employees. Anyone matching either condition shows up.

Retrieve all employees not in IT

The last update was for everyone except IT, since they already got their update.

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

This query pulls all employees outside IT. I added NOT to the WHERE clause to exclude the Information Technology department. This gave me everyone else who still needed the update.

Summary

I ran SQL queries with different filters to investigate security issues and target the right employee machines for updates. First, I queried the `log_in_attempts` table to spot suspicious login patterns and narrow down when and where they happened. Then I queried the `employees` table to identify specific departments and locations that needed updates. Throughout these queries, I combined conditions with `AND` when I needed multiple criteria met at once, `OR` when I needed records matching any of several options, and `NOT` when I needed to exclude specific values. I also leaned on `LIKE` with the `%` wildcard to match partial patterns in the data, which came in handy for things like office locations and country names with multiple formats.