Apply filters to SQL queries

Project description

In this project I am demonstrating some of the ways SQL is useful in analyzing information such as login attempts in an organization.

Retrieve after hours failed login attempts

In order to retrieve failed login attempts after 6pm, I select all from the log_in_attempts table and return the attempts that match the condition that the time was greater than 18:00 and login success equals false.

```
MariaDB [organization]> SELECT *
    -> FROM log_in_attempts
    -> WHERE login_time > '18:00:00' AND success = 0 ;
```

Retrieve login attempts on specific dates

To look at login attempts on specific dates I can use the OR operator.

```
iaDB [organization]> SELECT *
 -> FROM log_in_attempts
 -> WHERE login_date = '2022-05-09' OR login_date = '2022-05-08' ;
```

Retrieve login attempts outside of Mexico

To filter any attempts made outside of Mexico I can use the NOT operator. Additionally, I can use % after MEX in case any data shows ‘MEX’ or ‘Mexico’. This includes any country starting with MEX.

```
iaDB [organization]> SELECT *
 -> FROM log_in_attempts
 -> WHERE NOT country LIKE 'MEX%' ;
```

Retrieve employees in Marketing

I need to find any employee that works both in Marketing and is located in an East office. By using the AND and LIKE operators I can specify both of these conditions.

```
iaDB [organization]> SELECT *
 -> FROM employees
 -> WHERE department = 'Marketing' AND office LIKE 'East%' ;
```

Retrieve employees in Finance or Sales

When returning employees in specific departments such as Finance or Sales, I can use the OR operator. Even though the departments are listed in the same column, I must use '`department =`' twice.

```
aDB [organization]> SELECT *
-> FROM employees
-> WHERE department = 'Sales' OR department = 'Finance' ;
```

Retrieve all employees not in IT

When finding all employees excluding ones in IT, I can use the NOT operator. This is more efficient than selecting each department I need. This would also be the same as using a not equal to operator such as `<>` or `!=`.

```
iaDB [organization]> SELECT *
 -> FROM employees
 -> WHERE NOT department = 'Information Technology' ;
```

Summary

In this project I was able to explore different ways of filtering information quickly and efficiently, while leaving out excess and/or unnecessary information. Using the right operators can be crucial to finding the correct data.