Apply filters to SQL queries

Project description

In this project, I will demonstrate and explain the different commands that can be used for SQL.

SQL is primarily used to organize data, so most of these queries will involve filtering a large amount of data.

Retrieve after hours failed login attempts

```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_time > '18:00' AND
success = FALSE;
+----------+----------+------------+------------+---------+-----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address      | success
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0
 |
|       18 | pwashing | 2022-05-11 | 19:28:50   | US      | 192.168.66.142  |       0
 |
|       20 | tshah    | 2022-05-12 | 18:56:36   | MEXICO  | 192.168.109.50  |       0
 |
```

In order to retrieve failed login attempts after work hours, I used the command at the top of the screenshot.

The `SELECT` command will select data. The `FROM` command will indicate where the data will be selected from, which is `log_in_attempts` in this case. I am selecting all (`*` means all) data from `log_in_attempts`. `WHERE` asks for any conditions for filtering, which we want to find failed login attempts after office hours. Therefore, we list our conditions. The first condition asks for any data that is more recent than `'18:00'` (6PM), since that is when office hours end. The second condition asks for failed login attempts. Combine them together with the `AND` operator, this command will only return login attempts after 6PM that failed.

Retrieve login attempts on specific dates

```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_date = '2022-05-08'
 OR login_date = '2022-05-09';
+----------+----------+------------+------------+---------+-----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address      | success
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1
 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1
 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0
 |
```

Next, I was tasked with retrieving login attempts from specific dates. These dates are the 8th and 9th of May 2022. Here is what my command does (top of screenshot).

I selected all the data from the `log_in_attempts` table again, however my conditions have changed. This time I only want the rows where the `login_date` is either `'2022-05-08'` or `'2022-05-09'`.

Retrieve login attempts outside of Mexico

```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE NOT country LIKE 'MEX%';
+----------+----------+------------+------------+---------+-----------------+--------
-+
| event_id | username | login_date | login_time | country | ip_address      | success
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1
 |
|        2 | apatel   | 2022-05-10 | 20:27:27   | CAN     | 192.168.205.12  |       0
 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1
 |
```

After that, I retrieved all login attempts outside of Mexico.

Again, the first part is mostly straightforward, I am selecting all the data from `log_in_attempts` that correspond to the condition. However, if you look at `'MEX%'`, notice how there is a percent symbol. This symbol substitutes for any characters that trail after `'MEX'`, such as `'MEX12'` or `'MEXHEX'`. This works because the only countries in our system are `'CAN'` (Canada), `'USA'` (United States of America), and `'MEX'` (Mexico). In this case, the computer will find all login_attempts from a country that has a symbol starting with `'MEX'`. However, notice the command has a `NOT` condition. This negates it, which tells the computer to return all the login attempts that do not come from Mexico.

Retrieve employees in Marketing

```
MariaDB [organization]> SELECT * FROM employees WHERE department = 'Marketing' AND of
fice LIKE 'East-%';
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson  | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa  | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing  | East-267 |
|        1088 | k8651965m233 | rgosh    | Marketing  | East-157 |
|        1103 | NULL         | randerss | Marketing  | East-460 |
|        1156 | a184b775c707 | dellery  | Marketing  | East-417 |
|        1163 | h679i515j339 | cwilliam | Marketing  | East-216 |
+-------------+-------------+----------+------------+----------+
7 rows in set (0.002 sec)
```

In this command, I will retrieve all employees from the `'Marketing'` department and from the east office building.

The first part is still the same, however the conditions are different. The first one is directly asking for any employee whose department is exactly `'Marketing'`. The second condition asks for employees who are located in the east building. The condition asks for any office that starts with `'East-'` only. Notice how I use the `LIKE` operator in this case, since it enables the percent symbol to have an effect. If I were to use `=` instead, the data will only return employees from the `'East-%'` office, which are none in this case. Combine the two conditions asking for marketing employees and employees in the east building with an `AND` statement, and the computer will output employees that fulfill both conditions.

Retrieve employees in Finance or Sales

```
MariaDB [organization]> SELECT * FROM employees WHERE department = 'Finance' OR depar
tment = 'Sales';
+-------------+-------------+----------+------------+-------------+
| employee_id | device_id   | username | department | office      |
+-------------+-------------+----------+------------+-------------+
|        1003 | d394e816f943 | sgilmore | Finance    | South-153   |
|        1007 | h174i497j413 | wjaffrey | Finance    | North-406   |
|        1008 | i858j583k571 | abernard | Finance    | South-170   |
|        1009 | NULL         | lrodriqu | Sales      | South-134   |
```

Next up, I will retrieve all employees from the finance or sales department.

I selected all the data from the `employees` file, then I set two conditions asking for employees who are in the `'Finance'` department or the `'Sales'` department. I combine them with the `OR` operator, which will output all employees that fulfill any one of these conditions.

Retrieve all employees not in IT

```
MariaDB [organization]> SELECT * FROM employees WHERE department <> 'Information Tech
nology';
+-------------+--------------+----------+-----------------+-------------+
| employee_id | device_id    | username | department      | office      |
+-------------+--------------+----------+-----------------+-------------+
|        1000 | a320b137c219 | elarson  | Marketing       | East-170    |
|        1001 | b239c825d303 | bmoreno  | Marketing       | Central-276 |
|        1002 | c116d593e558 | tshah    | Human Resources | North-434   |
|        1003 | d394e816f943 | sgilmore | Finance         | South-153   |
|        1004 | e218f877g788 | eraab    | Human Resources | South-127   |
```

For my final task, I had to retrieve all employees that were not in the informational technology department.

Notice how I used the `<>` operator instead of `WHERE NOT`. This is because the `<>` operator does the same thing. In order to fulfill that condition, any department that is not equal to `'Informational Technology'` will make it true. Since that is also my only condition, it will return all employees not in IT.

Summary

These are all the important commands you can use to filter data on SQL. Hopefully, these examples and explanations can teach you more about the language, and hopefully I've demonstrated my ability to manage data on SQL.