Apply filters to SQL queries

Project description

In this SQL project, I perform data analysis to investigate potential security issues related to login attempts and employee information. By using SQL filters, I retrieve specific records from the log_in_attempts and employees tables to identify suspicious activity and gather data about employees in different departments and locations.

Retrieve after-hours failed login attempts

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

To investigate potential security incidents occurring after business hours, I query the log_in_attempts table for failed login attempts that happened after 18:00. The SQL query retrieves records where the login_time is greater than '18:00' and the success column equals 0 (indicating a failed attempt).

```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_time > '18:00' AND
success = 0;
+----------+----------+------------+------------+---------+-----------------+--------
-+
| 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
 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57   |       0
 |
|       34 | drosas   | 2022-05-11 | 21:02:04   | US      | 192.168.45.93   |       0
 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157   |       0
 |
|      111 | aestrada | 2022-05-10 | 22:00:26   | MEXICO  | 192.168.76.27   |       0
 |
|      127 | abellmas | 2022-05-09 | 21:20:51   | CANADA  | 192.168.70.122  |       0
 |
|      131 | bisles   | 2022-05-09 | 20:03:55   | US      | 192.168.113.171 |       0
 |
|      155 | cgriffin | 2022-05-12 | 22:18:42   | USA     | 192.168.236.176 |       0
 |
|      160 | jclark   | 2022-05-10 | 20:49:00   | CANADA  | 192.168.214.49  |       0
 |
|      199 | yappiah  | 2022-05-11 | 19:34:48   | MEXICO  | 192.168.44.232  |       0
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
19 rows in set (0.044 sec)
```

The initial screenshot displays my query along with its corresponding output, while the next screenshot highlights a segment of the output indicating a total of 19 rows in the dataset. This query is designed to filter failed login attempts that took place after 18:00. I extracted all data from the `log_in_attempts` table and then applied a `WHERE` clause featuring an `AND` operator to narrow down the results. This clause ensures that only unsuccessful login attempts occurring after 18:00 are included. The conditions used are "`login_time > '18:00`" to filter post-18:00 attempts and "`success = FALSE`" to filter failed attempts.

Retrieve login attempts on specific dates

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

To analyze suspicious events on specific dates, I query the log_in_attempts table for login attempts that occurred on 2022-05-09 or 2022-05-08. The SQL query retrieves records where the login_date matches either '2022-05-09' or '2022-05-08'.

```
MariaDB [organization]> SELECT * FROM log_in_attempts WHERE login_date = '2022-05-09'
 OR login_date = '2022-05-08';
+----------+----------+------------+------------+---------+-----------------+--------
-+
| 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
 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0
 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1
 |
|      186 | bisles   | 2022-05-09 | 04:29:17   | USA     | 192.168.40.72   |       0
 |
|      187 | arusso   | 2022-05-09 | 00:36:26   | MEX     | 192.168.77.137  |       0
 |
|      189 | nmason   | 2022-05-08 | 05:37:24   | CANADA  | 192.168.168.117 |       1
 |
|      190 | jsoto    | 2022-05-09 | 05:09:21   | USA     | 192.168.25.60   |       0
 |
|      191 | cjackson | 2022-05-08 | 06:46:07   | CANADA  | 192.168.7.187   |       0
 |
|      193 | lrodriqu | 2022-05-08 | 07:11:29   | US      | 192.168.125.240 |       0
 |
|      197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |       0
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
75 rows in set (0.001 sec)
```

The initial screenshot displays my query along with its corresponding output, while the next screenshot highlights a segment of the output indicating a total of 75 rows in the dataset. This query fetches all login attempts that occurred either on 2022-05-09 or 2022-05-08. To accomplish this, I began by selecting all data from the `log_in_attempts` table. I utilized a `WHERE` clause with an `OR` operator to refine the results, ensuring that only login attempts on either 2022-05-09 or 2022-05-08 are included. The first condition, "`login_date = '2022-05-09`'," filters logins on 2022-05-09, while the second condition, "`login_date = '2022-05-08`'," filters logins on 2022-05-08.

Retrieve login attempts outside of Mexico

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

To investigate suspicious login activity originating outside Mexico, I query the log_in_attempts table for login attempts not from Mexico. The SQL query retrieves records where the country column does not start with 'MEX'.

```
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
 |
|      194 | jclark   | 2022-05-12 | 14:11:04   | CAN     | 192.168.197.247 |       0
 |
|      195 | alevitsk | 2022-05-11 | 06:59:13   | CANADA  | 192.168.236.78  |       1
 |
|      196 | acook    | 2022-05-10 | 09:56:48   | CAN     | 192.168.52.90   |       0
 |
|      197 | jsoto    | 2022-05-08 | 09:05:09   | US      | 192.168.36.21   |       0
 |
|      200 | jclark   | 2022-05-12 | 01:11:45   | CANADA  | 192.168.91.103  |       1
 |
+----------+----------+------------+------------+---------+-----------------+--------
-+
144 rows in set (0.001 sec)
```

The initial screenshot displays my query along with its corresponding output, while the next screenshot highlights a segment of the output indicating a total of 144 rows in the dataset. This query retrieves all login attempts made in countries outside of Mexico. I selected all data from the `log_in_attempts` table. Next, I used a `WHERE` clause with `NOT` to exclude entries from Mexico. I used the `LIKE` operator with the pattern '`MEX%`' because the dataset uses both `'MEX'` and '`MEXICO`' to represent Mexico. The '`%`' symbol in the pattern matches any number of unspecified characters when combined with `LIKE`.

Retrieve employees in Marketing

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

To gather information about employees in the Marketing department located in the East building, I query the employees table for records matching department 'Marketing' and office starting with 'East'. The SQL query retrieves relevant employee data.

```
MariaDB [organization]> SELECT * FROM employees  WHERE department = 'Marketing' AND o
ffice  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 (0.001 sec)

MariaDB [organization]>
```

The initial screenshot displays my query along with its corresponding output and indicates a total of 7 rows in the dataset. This query fetches all employees in the Marketing department located in the East building. I retrieved all data from the `employees` table. Then, I used a `WHERE` clause with an `AND` operator to narrow down the results to employees in both the Marketing department and the East building. I used `LIKE` with '`East%`' as the pattern to match because the `office` column denotes the East building with specific office numbers. The `department = 'Marketing'` condition filters for Marketing department employees, while the `office LIKE 'East%'` condition filters for those in the East building.

Retrieve employees in Finance or Sales

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

To identify employees in the Finance or Sales departments, I query the employees table for records where the department is either 'Finance' or 'Sales'. The SQL query retrieves employee information for these departments.

```
MariaDB [organization]> SELECT * FROM employees  WHERE department = 'Sales' OR depart
ment = 'Finance';
+-------------+--------------+----------+------------+-------------+
| 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   |
|        1010 | k2421212m542 | jlansky  | Finance    | South-109   |
```

```
|        1181 | z803a233b718 | sessa    | Finance    | South-207   |
|        1185 | d790e839f461 | revens   | Sales      | North-330   |
|        1186 | e281f433g404 | sacosta  | Sales      | North-460   |
|        1187 | f963g637h851 | bbode    | Finance    | East-351    |
|        1188 | g164h566i795 | noshiro  | Finance    | West-252    |
|        1195 | n516o853p957 | orainier | Finance    | East-346    |
+-------------+--------------+----------+------------+-------------+
71 rows in set (0.001 sec)
```

The initial screenshot displays my query along with its corresponding output, while the next screenshot highlights a segment of the output indicating a total of 71 rows in the dataset. It retrieves all `employees` in the Finance and Sales departments. Initially, I extracted all data from the `employees` table. Subsequently, I applied a `WHERE` clause with an `OR` operator to filter for employees in either the Finance or Sales departments. I opted for the `OR` operator to include employees from both departments. The first condition, `department = 'Finance'`, selects employees from the Finance department, while the second condition, `department = 'Sales'`, selects employees from the Sales department.

Retrieve all employees not in IT

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

To target employees not in the IT department for specific updates, I query the employees table for records where the department is not 'Information Technology'. The SQL query retrieves data for employees in other departments.

```
MariaDB [organization]> SELECT * FROM employees  WHERE NOT  department = 'Information
 Technology';
+-------------+--------------+----------+-----------------+-------------+
| 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   |
|        1005 | f551g340h864 | gesparza | Human Resources | South-366   |
```

```
|        1185 | d790e839f461 | revens   | Sales           | North-330   |
|        1186 | e281f433g404 | sacosta  | Sales           | North-460   |
|        1187 | f963g637h851 | bbode    | Finance         | East-351    |
|        1188 | g164h566i795 | noshiro  | Finance         | West-252    |
|        1189 | h784i120j837 | slefkowi | Human Resources | West-342    |
|        1190 | NULL         | kcarter  | Marketing       | Central-270 |
|        1191 | NULL         | shakimi  | Marketing       | Central-366 |
|        1194 | m340n287o441 | zwarren  | Human Resources | West-212    |
|        1195 | n516o853p957 | orainier | Finance         | East-346    |
|        1198 | q308r573s459 | jmartine | Marketing       | South-117   |
|        1199 | r520s571t459 | areyes   | Human Resources | East-100    |
+-------------+--------------+----------+-----------------+-------------+
161 rows in set (0.001 sec)
```

The initial screenshot displays my query along with its corresponding output, while the next screenshot highlights a segment of the output indicating a total of 161 rows in the dataset. It retrieves employees who are not in the Information Technology department. I fetched all data from the `employees` table. Then, I applied a `WHERE` clause with `NOT` to filter out employees in the Information Technology department.

Summary

Through SQL queries with filters, I retrieved after-hours failed login attempts, login attempts on specific dates, login attempts outside of Mexico, employees in Marketing, employees in Finance or Sales, and employees not in the IT department. These queries helped me investigate security incidents, target specific employee groups for security updates, and ensure compliance with security protocols within the organization.