Apply filters to SQL queries

Project description

In this activity, we are going to investigate specific security issues, utilizing SQL. We need to obtain information on employees, the machines they use and their departments. Our primary objective is to filter out relevant data from the `organization` database and perform further analysis. Based on the scenarios, we will execute complex SQL queries on the `log_in_attempts` and `employees` tables to investigate potential security issues.

The data structure and samples of either tables is shown below for better understanding and clarity of the project

`Log_in_attempts`

Data		Data Type
`event_id`	:	String
`username`	:	String
`login_date`	:	Date
`login_time`	:	Time
`country`	:	String
`ip_address`	:	String
`success`	:	Boolean

`employees`

Data		Data Type
`Employee_id`	:	String
`device_id`	:	String
`Username`	:	String
`Department`	:	String
`Office`	:	String

Retrieve after hours failed login attempts

Scenario: The team is investigating failed login attempts that were made after business hours. We want to retrieve this information from the login activity by identifying all unsuccessful attempts after 18:00.

We will create a query to identify all failed login attempts in the `log_in_attempts` table by selecting all columns for a clarified visibility on the information.

```
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 |
|       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 |
|       52 | cjackson | 2022-05-10 | 22:07:07   | CAN     | 192.168.58.57   |       0 |
|       69 | wjaffrey | 2022-05-11 | 19:55:15   | USA     | 192.168.100.17  |       0 |
|       82 | abernard | 2022-05-12 | 23:38:46   | MEX     | 192.168.234.49  |       0 |
|       87 | apatel   | 2022-05-08 | 22:38:31   | CANADA  | 192.168.132.153 |       0 |
|       96 | ivelasco | 2022-05-09 | 22:36:36   | CAN     | 192.168.84.194  |       0 |
|      104 | asundara | 2022-05-11 | 18:38:07   | US      | 192.168.96.200  |       0 |
|      107 | bisles   | 2022-05-12 | 20:25:57   | USA     | 192.168.116.187 |       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 (0.103 sec)
```

- `SELECT *` : Select every column within the specified table
- `FROM` : Specifies which table to select some or all columns
- `WHERE` : A conditional clause that returns records of data if the condition is fulfilled.
- `AND` : An operator that is used with `WHERE` clause to indicate that both conditions, before and after the `AND` operator must be fulfilled.

Retrieve login attempts on specific dates

Scenario: The team wants to investigate a suspicious event that occurred on '2022-05-09'. We want to retrieve all login attempts that occurred on this day and the day before ('2022-05-08')

The `OR` operator is used in the scenario to return login attempts occurred on either dates as shown below in the screenshot. The query shown in the screenshot 'selects' all columns from the `log_in_attempts` table fulfilling the condition that the login date is either on '2022-05-09' or ('2022-05-08'), and is returned to us as output.

```
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 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       24 | arusso   | 2022-05-09 | 06:49:39   | MEXICO  | 192.168.171.192 |       1 |
|       25 | sbaelish | 2022-05-09 | 07:04:02   | US      | 192.168.33.137  |       1 |
|       26 | apatel   | 2022-05-08 | 17:27:00   | CANADA  | 192.168.123.105 |       1 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57   |       0 |
|       30 | yappiah  | 2022-05-09 | 03:22:22   | MEX     | 192.168.124.48  |       1 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|       36 | asundara | 2022-05-08 | 09:00:42   | US      | 192.168.78.151  |       1 |
|       38 | sbaelish | 2022-05-09 | 14:40:01   | USA     | 192.168.60.42   |       1 |
|       39 | yappiah  | 2022-05-09 | 07:56:40   | MEXICO  | 192.168.57.115  |       1 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157   |       0 |
|       43 | mcouliba | 2022-05-08 | 02:35:34   | CANADA  | 192.168.16.208  |       0 |
|       44 | daquino  | 2022-05-08 | 07:02:35   | CANADA  | 192.168.168.144 |       0 |
|       47 | dkot     | 2022-05-08 | 05:06:45   | US      | 192.168.233.24  |       1 |
|       49 | asundara | 2022-05-08 | 14:00:01   | US      | 192.168.173.213 |       0 |
|       53 | nmason   | 2022-05-08 | 11:51:38   | CAN     | 192.168.133.188 |       1 |
|       56 | acook    | 2022-05-08 | 04:56:30   | CAN     | 192.168.209.130 |       1 |
|       58 | ivelasco | 2022-05-09 | 17:20:54   | CAN     | 192.168.57.162  |       0 |
|       61 | dtanaka  | 2022-05-09 | 09:45:18   | USA     | 192.168.98.221  |       1 |
|       65 | aalonso  | 2022-05-09 | 23:42:12   | MEX     | 192.168.52.37   |       1 |
```

..

..

```
|      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 SQL query must return any login attempt data that occurred on the specified dates

- `OR` : This operator within the `WHERE` clause will return any record of data that fulfilled either of the conditions mentioned in the `WHERE` clause

Retrieve login attempts outside of Mexico

Scenario: So there's been suspicious activity with login attempts, however the team determined that there is no suspicious activity that can possibly originate from Mexico. Now, to reduce amount of data that needs to be analyzed, we need to investigate login attempts that occurred outside of Mexico only

We will create a query that excludes all records of data where the country is Mexico. We also know that the word 'Mexico' is inserted to the table as Mex, Mexico or Mexx. We need to exclude all those records as well.

```
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 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        5 | jrafael  | 2022-05-11 | 03:05:59   | CANADA  | 192.168.86.232  |       0 |
|        7 | eraab    | 2022-05-11 | 01:45:14   | CAN     | 192.168.170.243 |       1 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       10 | jrafael  | 2022-05-12 | 09:33:19   | CANADA  | 192.168.228.221 |       0 |
|       11 | sgilmore | 2022-05-11 | 10:16:29   | CANADA  | 192.168.140.81  |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
```

..

..

```
|      193 | lrodriqu | 2022-05-08 | 07:11:29   | US      | 192.168.125.240 |       0 |
|      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)
```

As the screenshot depicts, there are no entries where the country is Mexico or Mex.

- **`NOT`** : This operator negates the condition mentioned in the **`WHERE`** clause. If the condition is to list entries where the country is USA, adding the **`NOT`** operator will list entries where the country is actually not USA.
- **`LIKE`** : This is used within **`WHERE`** clause when we use wildcards. In this scenario we want to list entries with countries starting with 'Mex'. Any no. of unknown characters or nothing can follow the word 'Mex'. This is depicted as '**`%`**' following Mex. The **`%`** sign is a wildcard. **`LIKE`** is used when a wildcard is used.

Retrieve employees in Marketing

Scenario: The team wants to perform security updates on specific employee machines that identify all employees in the Marketing department for all offices in the East building. Being responsible for getting information on these employee machines, we will need to query the employees table.

We need to query specific information, that includes the employee's machine ID and offices only. We don't need other information like employee ID's from the `employees` table.

```
MariaDB [organization]> SELECT device_id, office
    -> FROM employees
    -> WHERE department = 'Marketing' AND office LIKE 'East%';
+--------------+----------+
| device_id    | office   |
+--------------+----------+
| a320b137c219 | East-170 |
| a192b174c940 | East-195 |
| x573y883z772 | East-267 |
| k8651965m233 | East-157 |
| NULL         | East-460 |
| a184b775c707 | East-417 |
| h679i515j339 | East-216 |
+--------------+----------+
7 rows in set (0.001 sec)
```

Notice how only `device_id` and `office` is returned. In this scenario, we selected specific columns to obtain machine information in different East locations

- `AND` : Because we need both info on employees from marketing and device offices, we used `AND` operator to include both conditions
- `LIKE` : Since we don't know all and how many values follow after the word 'East', we used wildcard `%`, hence we used `LIKE`.

Retrieve employees in Finance or Sales

Scenario: Now, the team needs to perform a different update to the computers of all employees in the Finance or the Sales department, and we need to gather information on these employees.

The query must gather information about the device to be updated, name of the employee that uses the device, office they are located and the department to make sure no other departments are included. The screenshot shows this information.

```
MariaDB [organization]> SELECT device_id, username, department, office
    -> FROM employees
    -> WHERE department = 'Finance' OR department = 'Sales';
+--------------+----------+------------+-------------+
| device_id    | username | department | office      |
+--------------+----------+------------+-------------+
| d394e816f943 | sgilmore | Finance    | South-153   |
| h174i497j413 | wjaffrey | Finance    | North-406   |
| i858j583k571 | abernard | Finance    | South-170   |
| NULL         | lrodriqu | Sales      | South-134   |
| k2421212m542 | jlansky  | Finance    | South-109   |
| l748m120n401 | drosas   | Sales      | South-292   |
| p611q262r945 | jsoto    | Finance    | North-271   |
| r550s824t230 | jclark   | Finance    | North-188   |
| s310t540u653 | abellmas | Finance    | North-403   |
| w237x430y567 | arusso   | Finance    | West-465    |
| y976z753a267 | iuduike  | Sales      | South-215   |
```

..

..

```
| t959u687v394 | jclark2  | Finance    | North-194   |
| u849v569w521 | nliu     | Sales      | West-220    |
| z803a233b718 | sessa    | Finance    | South-207   |
| d790e839f461 | revens   | Sales      | North-330   |
| e281f433g404 | sacosta  | Sales      | North-460   |
| f963g637h851 | bbode    | Finance    | East-351    |
| g164h566i795 | noshiro  | Finance    | West-252    |
| n516o853p957 | orainier | Finance    | East-346    |
+--------------+----------+------------+-------------+
71 rows in set (0.001 sec)
```

- `OR` : The department of both finance and sales is included in the output. Even if we query the same column (department), the values are different, and the column name must be repeated.

Retrieve all employees not in IT

Scenario: The team needs to make one more update. However, the update was already made to employee computers in the Information Technology department. We need information about employees who are not in that department. It is important that we do not include info on employees or their devices, since it is already made

The query in this case must include employee id, device id and office names of all employees except the IT department. We can do this using the `NOT` operator that negates the clause `WHERE department = 'Information Technology'`. Since this was already

demonstrated, we will execute a different SQL query using a different approach.

```
MariaDB [organization]> SELECT device_id, employee_id, office
    -> FROM employees
    -> WHERE department != 'Information Technology';
+--------------+-------------+-------------+
| device_id    | employee_id | office      |
+--------------+-------------+-------------+
| a320b137c219 |        1000 | East-170    |
| b239c825d303 |        1001 | Central-276 |
| c116d593e558 |        1002 | North-434   |
| d394e816f943 |        1003 | South-153   |
| e218f877g788 |        1004 | South-127   |
| f551g340h864 |        1005 | South-366   |
| h174i497j413 |        1007 | North-406   |
| i858j583k571 |        1008 | South-170   |
| NULL         |        1009 | South-134   |
| k242l212m542 |        1010 | South-109   |
| l748m120n401 |        1011 | South-292   |
| p611q262r945 |        1015 | North-271   |
```

..

..

```
| h784i120j837 |        1189 | West-342    |
| NULL         |        1190 | Central-270 |
| NULL         |        1191 | Central-366 |
| m340n287o441 |        1194 | West-212    |
| n516o853p957 |        1195 | East-346    |
| q308r573s459 |        1198 | South-117   |
| r520s571t459 |        1199 | East-100    |
+--------------+-------------+-------------+
161 rows in set (0.001 sec)
```

- `!=`: This operator works just the same as the **`NOT`** operator. Only difference is the syntax and is mentioned after the column name

Summary

In this scenario-based mini project on SQL, we successfully executed complex SQL queries that demonstrate primarily the usage of operators like NOT, AND, OR, WHERE and LIKE. We successfully filtered and retrieved specific data required to perform security investigations and other need-to-know basis information. This activity enabled me to demonstrate my SQL skills and technical capabilities.