Exemplar: Apply more filters in SQL

Activity Overview

This exemplar provides a detailed walk through and solutions for the "Apply more filters in SQL" lab activity. As a security analyst, you'll often need to refine your data retrieval by filtering based on specific dates, times, and ranges. This exemplar will guide you through using SQL operators like **>, >=, <, <=, BETWEEN**, and filtering by specific IDs.

Scenario

Continuing your work with the organization database, your team requires more precise information from the login attempt logs. You need to retrieve records based on specific time frames and individual login identifiers for investigation and analysis.

Task 1. Filter for login attempts made after a certain date

Your team is interested in login attempts that occurred after January 15th, 2023, to investigate recent activity.

The **login_date** column in the **log_in_attempts** table stores the date of each attempt.

Use the > (greater than) operator to filter for records where the **login_date** is after the specified date. Replace **YYYY-MM-DD** with the correct date.

```
SELECT *
FROM log_in_attempts
WHERE login_date > 'YYYY-MM-DD';
```

The correct query to solve this step:

```
SELECT *
FROM log_in_attempts
WHERE login_date > '2023-01-15';
```

How many login attempts occurred after January 15th, 2023?

Answer: There were [Insert Example Answer Here, e.g., 112] login attempts after January 15th, 2023.

Task 2. Filter for login attempts made in a certain date range

Your team needs to analyze login activity during the first week of February 2023, specifically from February 1st to February 7th (inclusive).

The **login_date** column in the **log_in_attempts** table contains the date.

Use the **BETWEEN** operator to filter for records within the specified start and end dates. Replace **YYYY-MM-DD** with the correct dates.

```
SELECT *
FROM log_in_attempts
WHERE login_date BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD';
```

The correct query to solve this step:

```
SELECT *
FROM log_in_attempts
WHERE login_date BETWEEN '2023-02-01' AND '2023-02-07';
```

How many login attempts were made between February 1st and February 7th, 2023?

Answer: There were [Insert Example Answer Here, e.g., 88] login attempts in that date range.

Task 3. Filter for login attempts made at a certain time

Your team is investigating a potential issue that occurred around 09:30 AM. You need to retrieve all login attempts that happened at exactly this time.

The **login_time** column in the **log_in_attempts** table stores the time of each attempt.

Use the = (equals) operator to filter for records where the **login_time** matches the specific time. Replace **HH:MM:SS** with the correct time.

```
SELECT *
FROM log_in_attempts
WHERE login_time = 'HH:MM:SS';
```

The correct query to solve this step:

```sql
SELECT *
FROM log_in_attempts
WHERE login_time = '09:30:00';
```

How many login attempts occurred at 09:30:00?

Answer: There were [Insert Example Answer Here, e.g., 5] login attempts at 09:30:00.

Task 4. Filter for login attempts by ID

You need to retrieve the details of a specific login attempt with the **login_id** of **503**.

The **login_id** column in the **log_in_attempts** table uniquely identifies each login attempt.

Use the = (equals) operator to filter for the record where the **login_id** matches the specified value. Replace **ID_Value** with the correct ID.

```sql
SELECT *
FROM log_in_attempts
WHERE login_id = ID_Value;
```

The correct query to solve this step:

```sql
SELECT *
FROM log_in_attempts
WHERE login_id = 503;
```

What was the **login_date** for the login attempt with **login_id** 503?

Answer: The **login_date** for login ID 503 was [Insert Example Answer Here, e.g., 2023-02-10].