Suspicious Logins Dashboard – Quick Guide

Suspicious Logins Dashboard – Quick Guide

What you have:
• schema.sql — defines the SQLite schema
and indexes for the logins table.
• data.sql — fake login events (100 rows) with realistic patterns,
 including:
    - multiple consecutive failed logins (eve)
    - “impossible travel” (charlie)
    -
 more than two unique IPs in one day (bob)
• logins.db — prebuilt SQLite database with schema + data
 loaded.

Prerequisites:
• macOS/Linux/WSL: sqlite3 installed (usually available by default).
•
Windows: install sqlite3 or use DB Browser for SQLite (optional).
• Python 3.x (optional, if you
want to script analysis).

Option A — Use the prebuilt database:
1) Unzip the project and cd into
the folder.
2) Open the database in sqlite3:
   sqlite3 logins.db
3) Try these sample queries:

```
--
Users with more than 3 consecutive failed logins (requires SQLite window functions)
WITH ordered AS
(
  SELECT username, timestamp, status,
         CASE WHEN status='failure' THEN 1 ELSE 0 END AS
is_fail,
         ROW_NUMBER() OVER (PARTITION BY username ORDER BY timestamp) AS rn1,

ROW_NUMBER() OVER (PARTITION BY username, status ORDER BY timestamp) AS rn2
  FROM logins
),
groups
AS (
  SELECT username, rn1 - rn2 AS grp
  FROM ordered
  WHERE status = 'failure'
)
SELECT
username, COUNT(*) AS fail_streak
FROM groups
GROUP BY username, grp
HAVING COUNT(*) >= 4;

-- Users
 with more than 2 unique IPs in the same day
SELECT username, date(timestamp) AS day, COUNT(DISTINCT
 ip_address) AS ip_count
FROM logins
GROUP BY username, day
HAVING ip_count > 2;

-- Most common IPs
 across all accounts
SELECT ip_address, COUNT(*) AS hits
FROM logins
GROUP BY ip_address
ORDER BY
hits DESC
LIMIT 5;
```

Option B — Build the DB yourself from the SQL files:
1) Create a new database
and load the schema:
   sqlite3 logins.db < schema.sql
2) Load the data:
   sqlite3 logins.db <
data.sql

Optional — Python starter snippet (sqlite3):

```
import sqlite3, pandas as pd
con = sqlite3.connect("logins.db")

q = "SELECT username,
date(timestamp) AS day, COUNT(DISTINCT ip_address) AS ip_count FROM logins GROUP BY username, day
HAVING ip_count > 2;"
df = pd.read_sql_query(q, con)
print(df)
```

Next steps / extensions:
• Add a
Python script to run all queries and export suspicious findings to CSV.
• Visualize failed logins
per user with matplotlib.
• Add a simple “watchlist” of IPs to flag.
• Expand the schema to include
user_agent or geo fields.