DSCI 6007 — Lab 7.1-Museum Mystery SQL

Museum Mystery PostgreSQL Lab includes:

- ✅ How to create the database
- ✅ How to import (\i) the SQL file
- ✅ Guided SQL practice activities
- ✅ Analytical challenges
- ✅ Advanced joins
- ✅ Mystery-style critical thinking questions
- ✅ All aligned with the schema in **museum.sql**

🏛️ **DSCI 6007 — Museum Mystery SQL Lab**

Relational Databases, Joins, Queries, and Data Analysis

Dataset: museum.sql (Visitors, CheckIns, Artifacts, SecurityLogs)
Database Name: MuseumMystery

✅ Part 1 — Create & Load Your PostgreSQL Database

Step 1 — Create a new database

In pgAdmin or psql:

Option A — pgAdmin GUI

1. Open **pgAdmin 4**
2. Right-click **Databases** → **Create** → **Database...**
3. Name it:
   MuseumMystery
4. Click **Save**

Option B — In psql

```
CREATE DATABASE MuseumMystery;
```

Step 2 — Load the dataset from your local machine

Move the SQL file into a simple path, like:

C:/psql_scripts/museum.sql

Then connect to your new DB in psql:

psql -U postgres -d MuseumMystery

Run:

\i 'C:/psql_scripts/museum.sql';

If you see:

CREATE TABLE

INSERT 0 5

...

Your database loaded successfully!

Step 3 — Verify tables

\dt;

Expected tables:

- **Visitors**
- **CheckIns**
- **Artifacts**
- **SecurityLogs**

🧪 Part 2 — Warm-Up SQL Queries

Q1 — View the first 5 visitors

SELECT * FROM Visitors LIMIT 5;

Q2 — Count the number of visitors

SELECT COUNT(*) FROM Visitors;

Q3 — List all artifacts with their status

```
SELECT artifact_name, status FROM Artifacts;
```

🔍 Part 3 — Filtering & Logic

Q4 — Find all visitors with Gold memberships

```
SELECT * FROM Visitors
WHERE membership_type = 'Gold';
```

Q5 — Which artifacts are missing?

```
SELECT * FROM Artifacts
WHERE status = 'Missing';
```

Q6 — Get all check-ins that occurred in the "Ancient Artifacts" gallery

```
SELECT *
FROM CheckIns
WHERE location = 'Ancient Artifacts';
```

🔗 Part 4 — Joins (Investigate the Mystery)

Q7 — List all Visitors and their Check-In locations

```
SELECT v.name, c.location, c.checkin_time
FROM Visitors v
JOIN CheckIns c ON v.visitor_id = c.visitor_id
ORDER BY c.checkin_time;
```

Q8 — Who was near the Lost Scepter before it went missing?

Artifact in dataset:

Lost Scepter — status: Missing

location: 'Main Exhibit'

Query:

```
SELECT v.name, c.checkin_time
FROM CheckIns c
JOIN Visitors v ON c.visitor_id = v.visitor_id
WHERE c.location = 'Main Exhibit'
ORDER BY c.checkin_time;
```

Q9 — Show all security camera logs around the time of the missing artifact

```
SELECT *
FROM SecurityLogs
WHERE time BETWEEN '2024-11-20 10:00:00' AND '2024-11-20 12:00:00'
ORDER BY time;
```

📊 Part 5 — Aggregations

Q10 — Total number of check-ins per gallery

```
SELECT location, COUNT(*) AS checkin_count
FROM CheckIns
GROUP BY location
ORDER BY checkin_count DESC;
```

Q11 — Count how many visitors checked in more than once

```
SELECT visitor_id, COUNT(*) AS visits
FROM CheckIns
GROUP BY visitor_id
HAVING COUNT(*) > 1;
```

Q12 — Latest known status by artifact

```
SELECT artifact_name, status
```

```
FROM Artifacts
ORDER BY artifact_name;
```

🧩 Part 6 — Multi-Table Analytical Challenges

Q13 — Match check-ins with security log events by approximate time

(± 20 minutes)

```
SELECT
  v.name,
  c.location,
  c.checkin_time,
  s.time AS security_time,
  s.description
FROM CheckIns c
JOIN Visitors v ON c.visitor_id = v.visitor_id
JOIN SecurityLogs s
  ON s.time BETWEEN c.checkin_time - INTERVAL '20 minutes'
        AND c.checkin_time + INTERVAL '20 minutes'
ORDER BY s.time;
```

Q14 — Who visited BOTH “Main Exhibit” and “Ancient Artifacts”?

```
SELECT visitor_id
FROM CheckIns
WHERE location IN ('Main Exhibit', 'Ancient Artifacts')
GROUP BY visitor_id
HAVING COUNT(DISTINCT location) = 2;
```

🦹 Part 7 — Solve the Museum Mystery

Use **Visitors**, **CheckIns**, **Artifacts**, and **SecurityLogs** to answer:

Q15 — Who was in the Main Exhibit closest to the time the Lost Scepter went missing?

Q16 — Which visitor's movements match suspicious activity captured in the security logs?

Q17 — List a timeline combining:

- Check-ins
- Camera activity
- Artifact status
- Visitor identities

(Use JOIN + ORDER BY time)

⭐ Part 8 — Stretch Questions (Optional)

Q18 — Create a view of "Suspect Movements"

```
CREATE VIEW SuspectMovements AS
SELECT v.name, c.location, c.checkin_time
FROM Visitors v
JOIN CheckIns c ON v.visitor_id = c.visitor_id;
```

Q19 — Add a new inserted security log

```
INSERT INTO SecurityLogs (time, camera_id, description)
VALUES ('2024-11-20 12:05:00', 'C3', 'Visitor running toward exit');
```

Q20 — Update artifact status after recovery

```
UPDATE Artifacts
SET status = 'Recovered'
WHERE artifact_name = 'Lost Scepter';
```