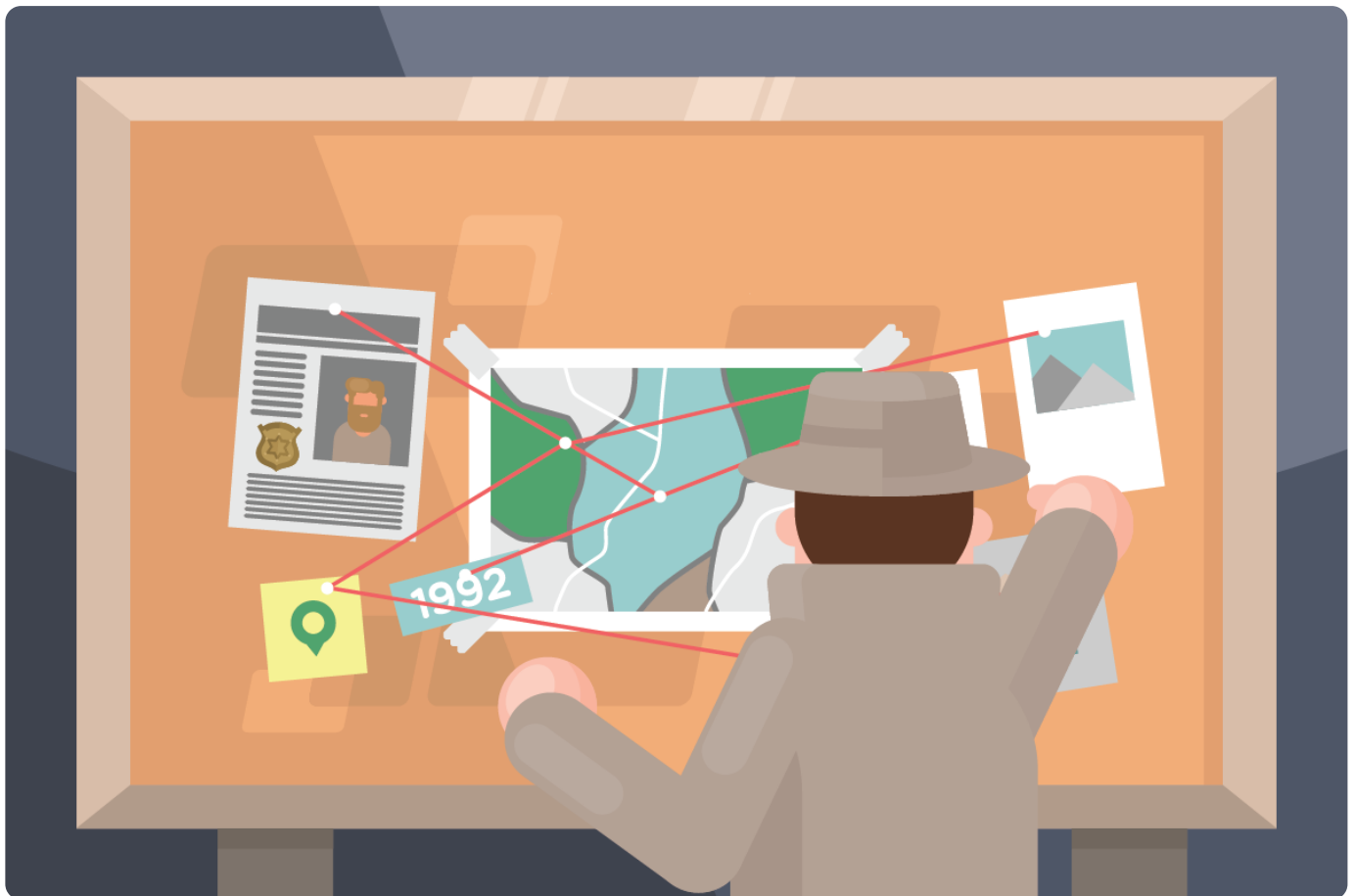# SQL Murder Mystery

Can you find out whodunnit?

There's been a Murder in SQL City! The SQL Murder Mystery is designed to be both a self-directed lesson to learn SQL concepts and commands and a fun game for experienced SQL users to solve an intriguing crime.

New to SQL?

This exercise is meant more as a way to practice SQL skills than a full tutorial. If you've never used SQL at all, try the walkthrough. If you really want to learn a lot about SQL, you may prefer a complete tutorial like Select Star SQL.

If you're comfortable with SQL, you can dive in below!

Experienced SQL sleuths start here

A crime has taken place and the detective needs your help. The detective gave you the crime scene report, but you somehow lost it. You vaguely remember that the crime was a **murder** that occurred sometime on **Jan.15, 2018** and that it took place in **SQL City**. Start by retrieving the corresponding crime scene report from the police department's database.

Exploring the Database Structure

Experienced SQL users can often use database queries to infer the structure of a database. But each database system has different ways of managing this information. The SQL Murder Mystery is built using SQLite. Use this SQL command to find the tables in the Murder Mystery database.

Run this query to find the names of the tables in this database.

This command is specific to SQLite. For other databases, you'll have to learn their specific syntax.

```
1  SELECT name
2    FROM sqlite_master
3   where type = 'table'
```

RUN ⇓   RESET

Besides knowing the table names, you need to know how each table is structured. The way this works is also dependent upon which database technology you use. Here's how you do it with SQLite.

Run this query to find the structure of the `crime_scene_report` table
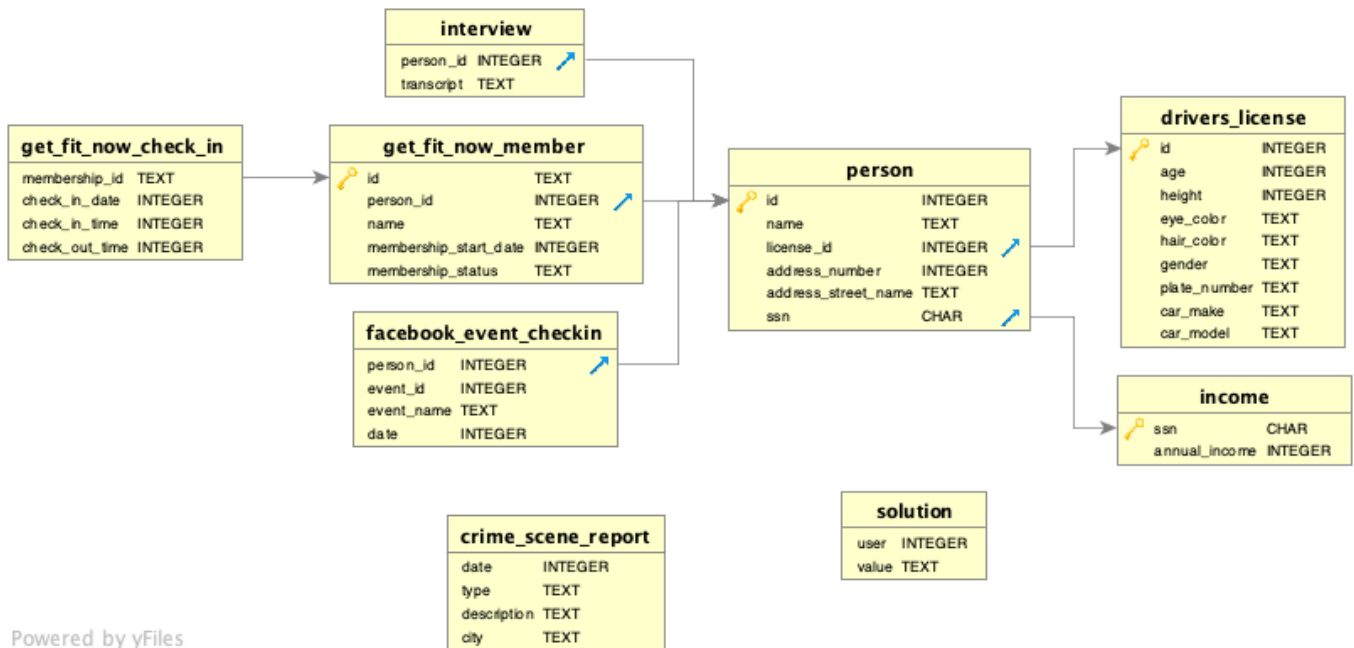
Change the value of 'name' to see the structure of the other tables you learned about with the previous query.

```
1  SELECT sql
2    FROM sqlite_master
3   where name = 'crime scene report'
```

RUN ⇓   RESET

The rest is up to you!

If you're really comfortable with SQL, you can probably get it from here.

But click here to show the schema diagram.

Powered by yFiles

And you can always go to the walkthrough.

Use your knowledge of the database schema and SQL commands to find out who committed the murder.

When you think you know the answer, go to the next section.

```
1  --Looking for the suspect with higher income
2  SELECT name FROM income as i
3  INNER JOIN person as p
4  ON i.ssn=p.ssn
5  WHERE license_id IN (SELECT id FROM drivers_license
6                         WHERE height IN (65,66,67)
7                         AND hair_color='red'
8                         AND car_make='Tesla'
9                         AND car_model='Model S')
10 ORDER BY annual_income DESC
11 LIMIT 1                        ;
12 --It must be Miranda Priestly because of the higher income
13
14
15
```

RUN ⇩   RESET

name
Miranda Priestly

Check your solution

Did you find the killer?

```
1  INSERT INTO solution VALUES (1, 'Miranda Priestly');
2
3          SELECT value FROM solution;
```

RUN ⇩   RESET

value
Congrats, you found the brains behind the murder! Everyone in SQL City hails you as the greatest SQL detective of all time. Time to break out the champagne!

Credits

The SQL Murder Mystery was created by Joon Park and Cathy He while they were Knight Lab fellows. See the GitHub repository for more information.

Adapted and produced for the web by Joe Germuska.

This mystery was inspired by a crime in the neighboring Terminal City.

Web-based SQL is made possible by SQL.js

SQL query custom web components created and released to the public domain by Zi Chong Kao, creator of Select Star SQL.

Detective illustration courtesy of Vectors by Vecteezy

Original code for this project is released under the MIT License

Original text and other content for this project is released under Creative Commons CC BY-SA 4.0

The Northwestern University Knight Lab is a team of technologists and journalists working at advancing news media innovation through exploration and experimentation.