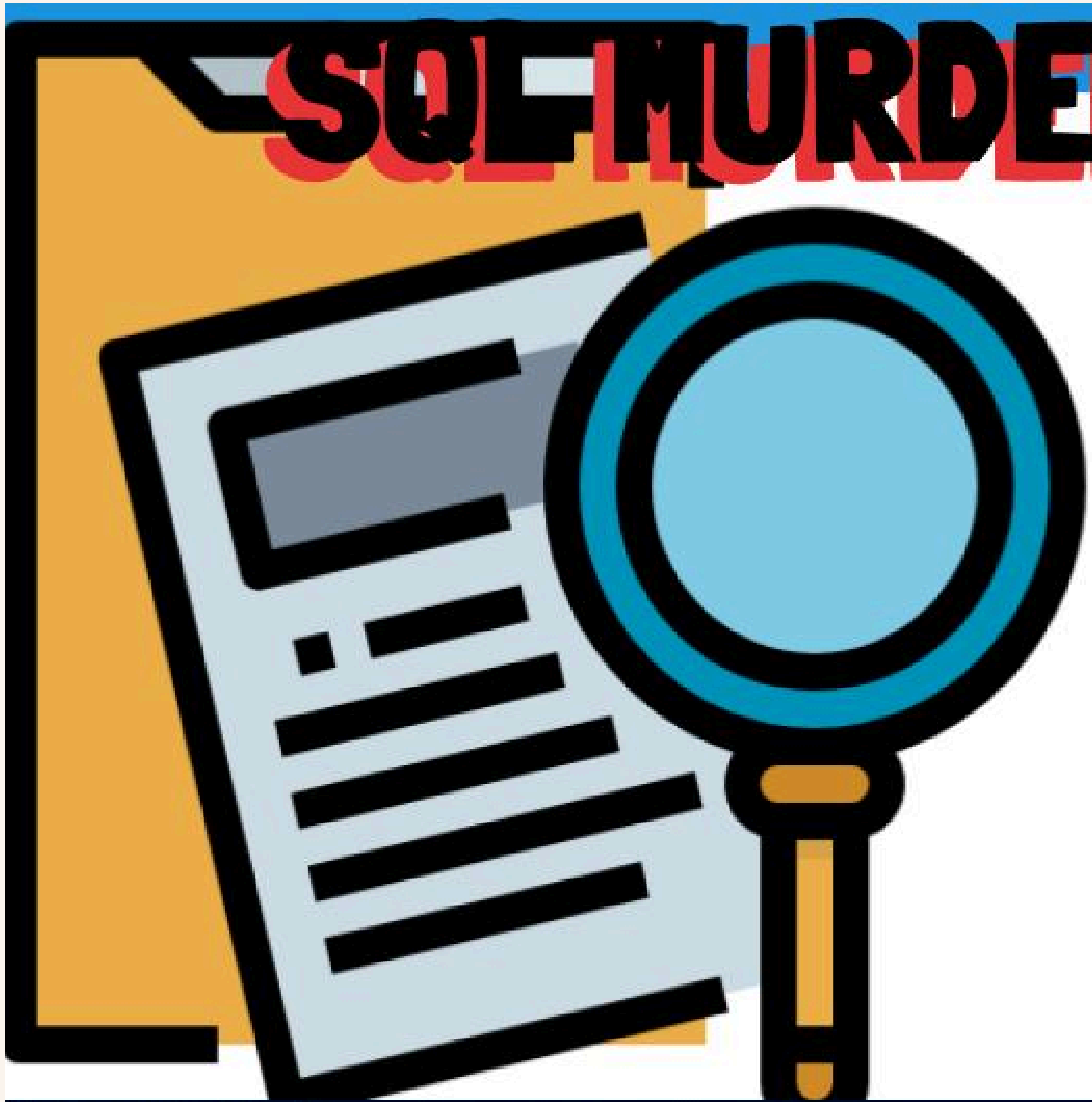
Basic SQL

SQL MURDER MYSTERY

SILVER

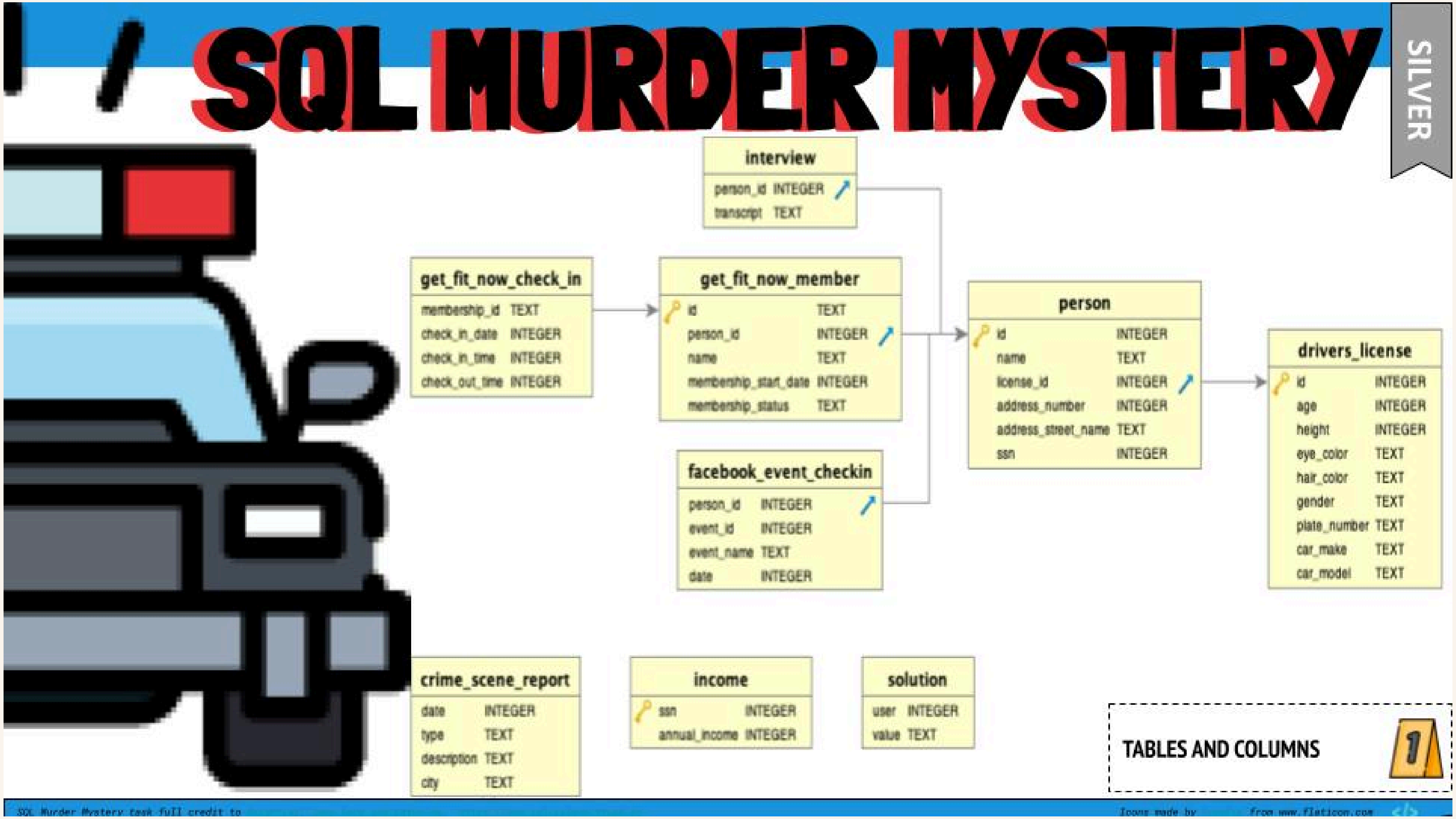


FIRST CLUE...

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.

All the clues to this mystery are buried in a huge database, and you need to use SQL to navigate through this vast network of information. Your first step to solving the mystery is to retrieve the corresponding crime scene report from the police department's database.



interview

person_id	INTEGER
transcript	TEXT

get_fit_now_check_in

membership_id	TEXT
check_in_date	INTEGER
check_in_time	INTEGER
check_out_time	INTEGER

get_fit_now_member

id	TEXT
person_id	INTEGER
name	TEXT
membership_start_date	INTEGER
membership_status	TEXT

person

id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	INTEGER

drivers_license

id	INTEGER
age	INTEGER
height	INTEGER
eye_color	TEXT
hair_color	TEXT
gender	TEXT
plate_number	TEXT
car_make	TEXT
car_model	TEXT

facebook_event_checkin

person_id	INTEGER
event_id	INTEGER
event_name	TEXT
date	INTEGER

crime_scene_report

date	INTEGER
type	TEXT
description	TEXT
city	TEXT

income

ssn	INTEGER
annual_income	INTEGER

solution

user	INTEGER
value	TEXT

SQL Murder Mystery task full credit to

Icons made by from www.flaticon.com

To Write a Query:

1️⃣ Which `table`?

2️⃣ Which `columns`?

3️⃣ Which `rows`?

First clue

> **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.**

- 1️⃣ A murder that occurred on Jan.15, 2018 in SQL City.
- 2️⃣ `crime_scene_report` table

`SELECT` columns `FROM` a table `WHERE` conditions are true

Table: crime_scene_report

Columns: all

Rows: report type is murder

```
SELECT * FROM crime_scene_report WHERE type = 'murder';
```

`WHERE` clause

Filters records based on specified conditions

- **Comparison operators** (`=`, `!=`, `>=`, ...)
- **Logical operators** (`NOT`, `AND`, `OR`)
- **Membership operators** (`IN`, `NOT IN`)

SELECT all non-murder cases

```sql
-- using !=
SELECT * FROM crime_scene_report WHERE type != 'murder';
-- using not
SELECT * FROM crime_scene_report WHERE NOT type = 'murder';
```

Multiple matching criteria (`AND`, `OR`)

Table: crime_scene_report

Columns: all

Rows: a murder that occurred sometime on Jan.15, 2018 and that it took place in SQL City.

```sql
SELECT *
FROM crime_scene_report
WHERE type = 'murder'
AND date = 20180115
AND city = 'SQL City';
```

The Crime Report

```sql
SELECT *
FROM crime_scene_report
WHERE type = 'murder'
AND date = 20180115
AND city = 'SQL City';
```

> Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".

- 1️⃣ **Find the first witness who lives at the last house on "Northwestern Dr".**
- 2️⃣ Find the second witness named Annabel who lives on "Franklin Ave".

Matching strings

Table: person

Columns: all

Rows: lives at the last house on "Northwestern Dr"

```sql
SELECT *
FROM person
WHERE address_street_name = 'northWestern DR'
```

```sql
-- explicit case-insensitive matching
SELECT *
FROM person
WHERE lower(address_street_name) = 'northwestern dr'
```

`ORDER BY` address_number (`DESC`ending order)

Table: person

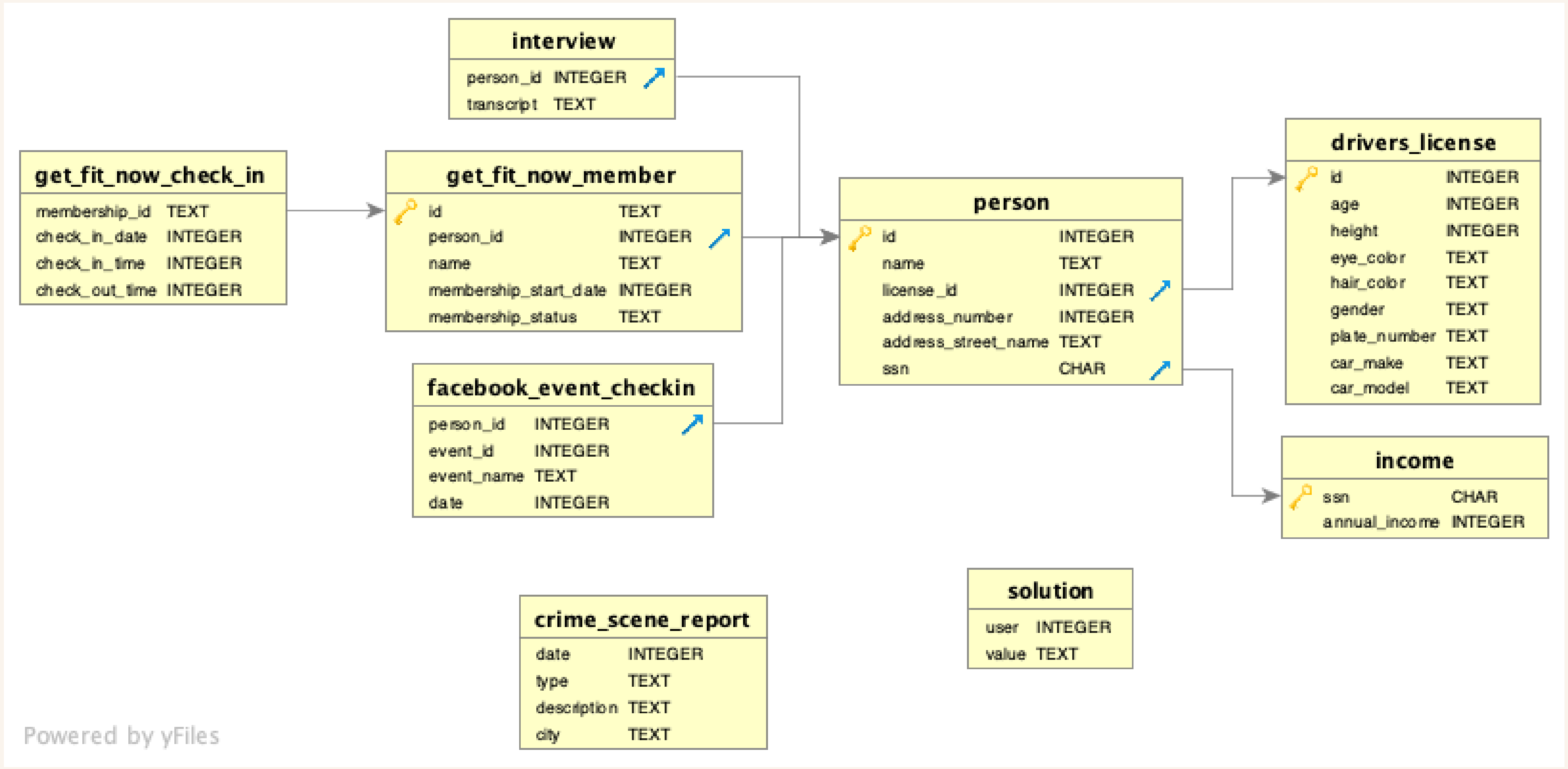
Columns: all

Rows: lives at ***the last house*** on "Northwestern Dr"

```
SELECT *
FROM person
WHERE address_street_name = 'Northwestern Dr'
ORDER BY address_number DESC
```

id	name	license_id	address_number	address_street_name	ssn
14887	Morty Schapiro	118009	4919	Northwestern Dr	1115649

Find the interview of the first witness



interview	
person_id	INTEGER
transcript	TEXT

get_fit_now_check_in	
membership_id	TEXT
check_in_date	INTEGER
check_in_time	INTEGER
check_out_time	INTEGER

get_fit_now_member	
id	TEXT
person_id	INTEGER
name	TEXT
membership_start_date	INTEGER
membership_status	TEXT

facebook_event_checkin	
person_id	INTEGER
event_id	INTEGER
event_name	TEXT
date	INTEGER

person	
id	INTEGER
name	TEXT
license_id	INTEGER
address_number	INTEGER
address_street_name	TEXT
ssn	CHAR

drivers_license	
id	INTEGER
age	INTEGER
height	INTEGER
eye_color	TEXT
hair_color	TEXT
gender	TEXT
plate_number	TEXT
car_make	TEXT
car_model	TEXT

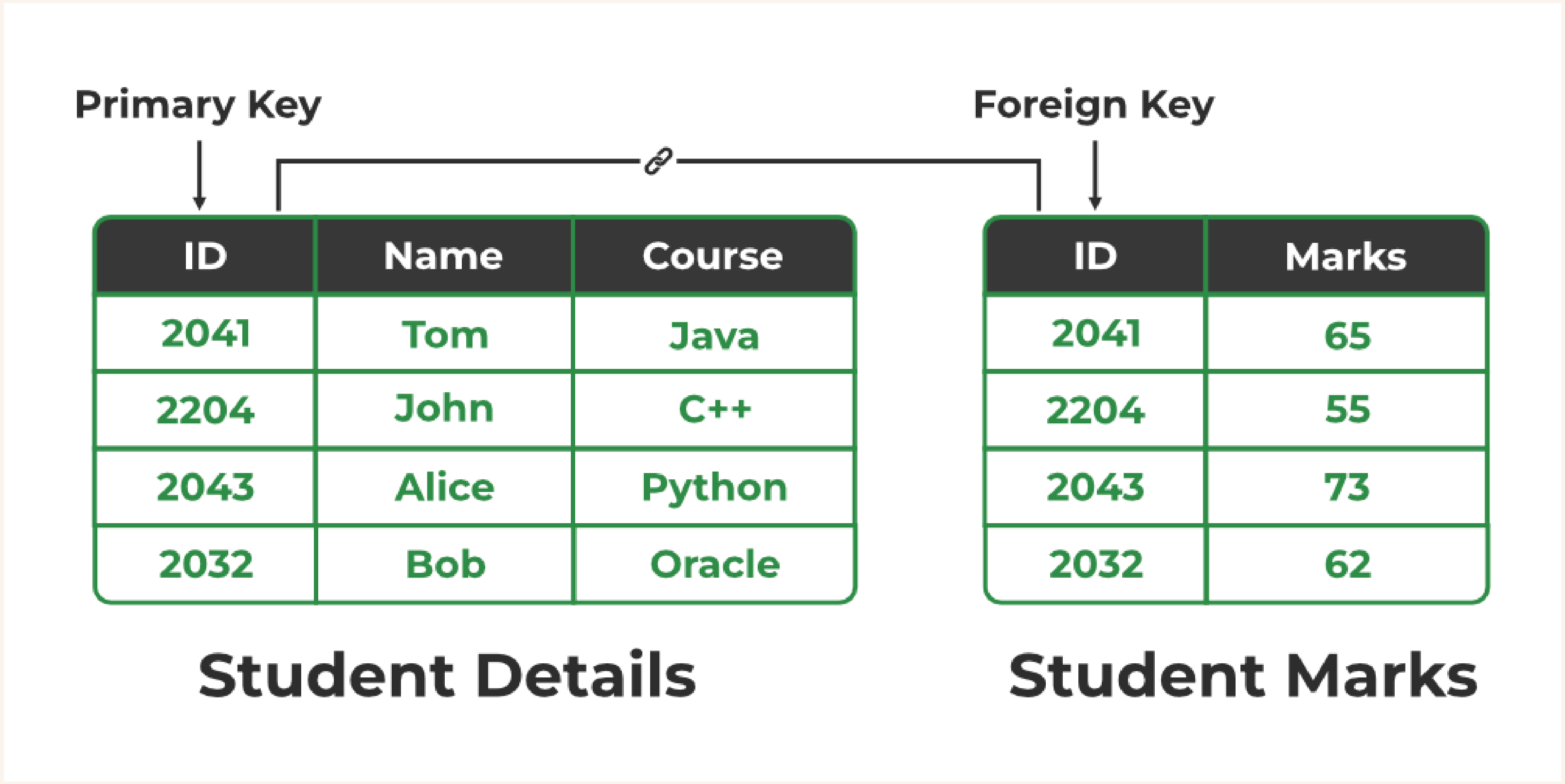
income	
ssn	CHAR
annual_income	INTEGER

solution	
user	INTEGER
value	TEXT

crime_scene_report	
date	INTEGER
type	TEXT
description	TEXT
city	TEXT

Primary key / Foreign key 🔑

- **PK:** unique ID of each record in a table
- **FK:** a column that references the PK of another table



ID	Name	Course
2041	Tom	Java
2204	John	C++
2043	Alice	Python
2032	Bob	Oracle

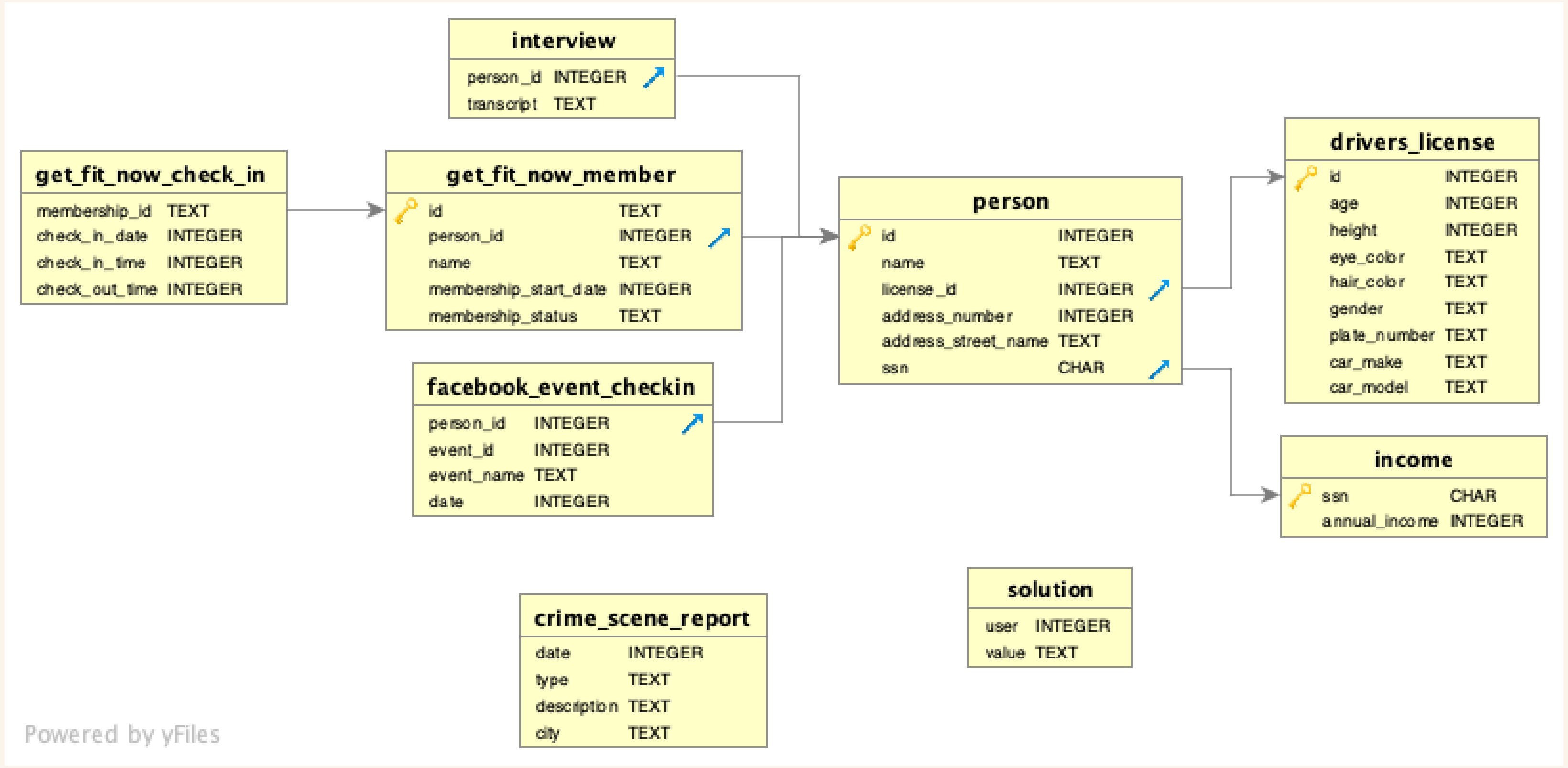
Student Details

ID	Marks
2041	65
2204	55
2043	73
2032	62

Student Marks

How do we connect `person` and `interview` tables?

🔑 Primary key ↗️ Foreign key



- ↗️ `person_id` in `interview` → 🔑 `id` in `person` table
- Morty's `person_id` is 14887

First witness's interview

```sql
SELECT * FROM interview WHERE person_id = 14887
```

> **I heard a gunshot and then saw a man run out. He had a "Get Fit Now Gym" bag. The membership number on the bag started with "48Z". Only gold members have those bags. The man got into a car with a plate that included "H42W".**

- 1️⃣ Gold member whose id starts with "48Z" from `get_fit_now_member`
- 2️⃣ Driver whose license plate includes "H42W" from `drivers_license`

Pattern matching `LIKE`

- `LIKE` operator for pattern matching in strings
- Wildcards:
    - `%`: Any number of characters
    - `_`: Only one character

Exact matching

```
-- exact matching
SELECT * FROM get_fit_now_member WHERE id = '48Z'
SELECT * FROM get_fit_now_member WHERE id LIKE '48Z'
```

Pattern matching

Starts with:

```
-- followed by any number of characters
SELECT * FROM get_fit_now_member WHERE id LIKE '48Z%'
-- followed by only one character
SELECT * FROM get_fit_now_member WHERE id LIKE '48Z_'
```

Ends with:

```
-- any number of preceeding characters
SELECT * FROM get_fit_now_member WHERE id LIKE '%48Z'
-- only one preceeding character
SELECT * FROM get_fit_now_member WHERE id LIKE '_48Z'
```

Includes:

```
SELECT * FROM get_fit_now_member WHERE id LIKE '%48Z%'
```

1️⃣ Gold member whose id starts with "48Z" from `get_fit_now_member`

```
SELECT * FROM get_fit_now_member
WHERE id LIKE '48Z%'
AND membership_status='gold'
```

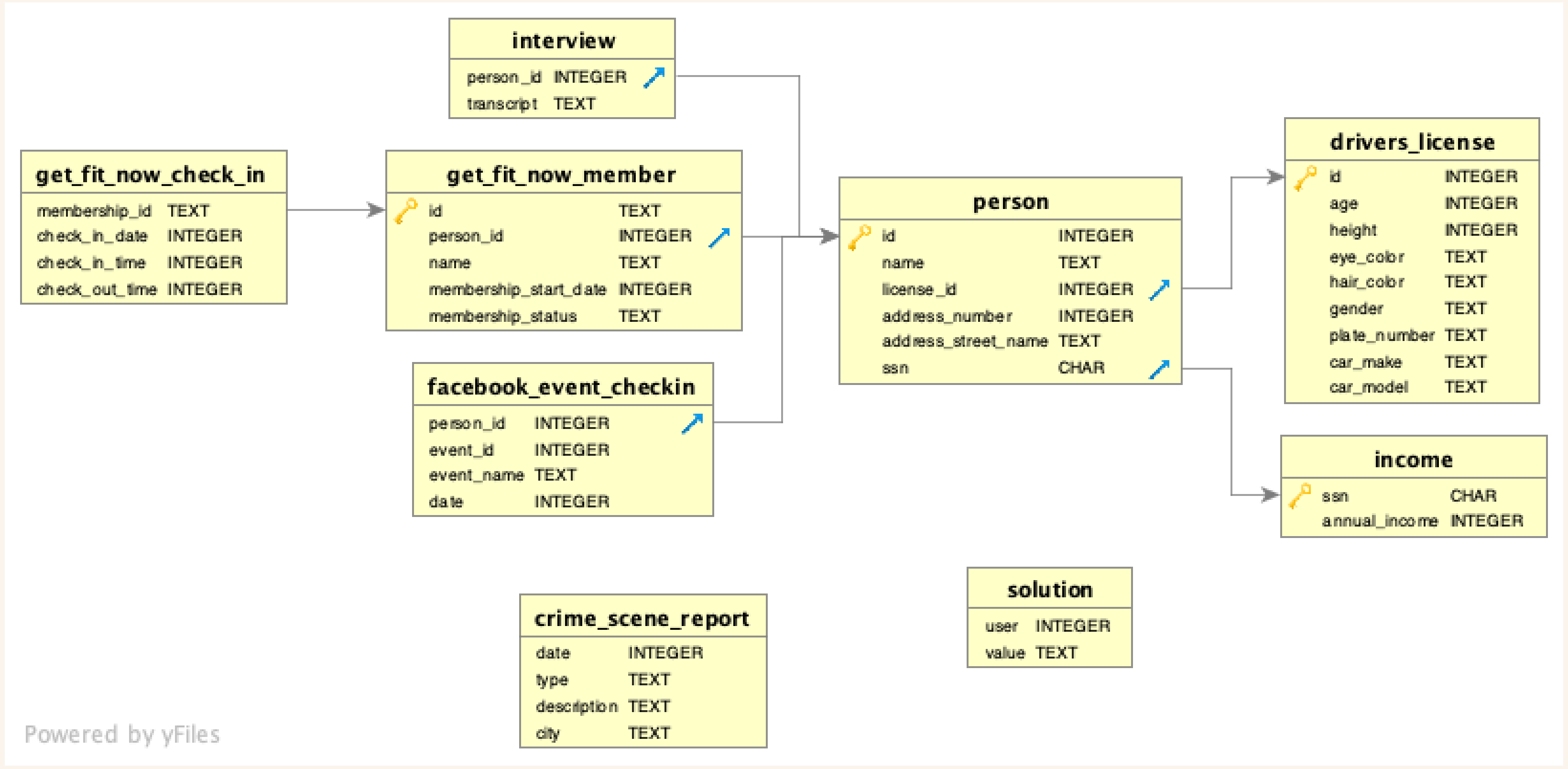
id	person_id	name	membership_start_date	membership_status
48Z55	67318	Jeremy Bowers	20160101	gold
48Z7A	28819	Joe Germuska	20160305	gold

2️⃣ Driver whose license plate includes "H42W" from `drivers_license`

```sql
SELECT * FROM drivers_license
WHERE plate_number LIKE '%H42W%'
```

id	age	height	eye_color	hair_color	gender	plate_number	car_make
183779	21	65	blue	blonde	female	H42W0X	Toyota
423327	30	70	brown	brown	male	0H42W2	Chevrolet
664760	21	71	black	black	male	4H42WR	Nissan

No direct relationship between `get_fit_now_member` and `drivers_license`



- ↗ `person_id` in `get_fit_now_member` → 🔑 `id` in `person`
- ↗ `license_id` in `person` → `id` in 🔑 `drivers_license`

Cross-checking two tables

Using logical operators (`AND` , `OR` , `NOT`)

```sql
SELECT * FROM person
-- from get_fit_now_member
WHERE (id = 67318 OR id = 28819)
-- from drivers_license
AND (license_id = 183779 OR license_id = 423327 OR license_id = 664760)
```

Using membership operators (`IN` , `NOT IN`)

```sql
SELECT * FROM person
-- from get_fit_now_member
WHERE id IN (67318, 28819)
-- from drivers_license
AND license_id IN (183779, 423327, 664760)
```

💻 Find the second witness

```sql
SELECT *
FROM crime_scene_report
WHERE type = 'murder'
AND date = 20180115
AND city = 'SQL City';
```

> **Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".**

- 1️⃣ Find the first witness who lives at the last house on "Northwestern Dr".
- 2️⃣ **Find the second witness named Annabel who lives on "Franklin Ave".**

To Write a Query:

1️⃣ Which `table`?

2️⃣ Which `columns`?

3️⃣ Which `rows`?

💻 What's the person_id of the second witness?

> The second witness, named Annabel, lives somewhere on "Franklin Ave".

💻 What's on her interview?

Hint: Use the person_id to find her interview details.

💻 Figure out what time she was at the gym

Hint: find her `membership_id` first, then use it to find her check-in time.

💻 Who else was at the gym during the time she was there?

Hint: Anyone who left before she arrived or arrived after she left should be excluded.

`BETWEEN`, `NOT BETWEEN`

```
SELECT *
FROM get_fit_now_check_in
WHERE check_out_time BETWEEN 1600 AND 1700
```

```
SELECT *
FROM get_fit_now_check_in
WHERE check_out_time NOT BETWEEN 1600 AND 1700
```

💻 Who are they?

Hint: Use the membership IDs to find their names

Wrap up

- `SELECT` columns `FROM` a table `WHERE` conditions are true
  - `LIKE` pattern matching
  - `IN` membership operator
  - `BETWEEN` range operator
- `LIMIT` the number of records returned
- `ORDER BY` columns