RELATIONAL DATABASE & SQL

WHY DO WE NEED DATABASES?

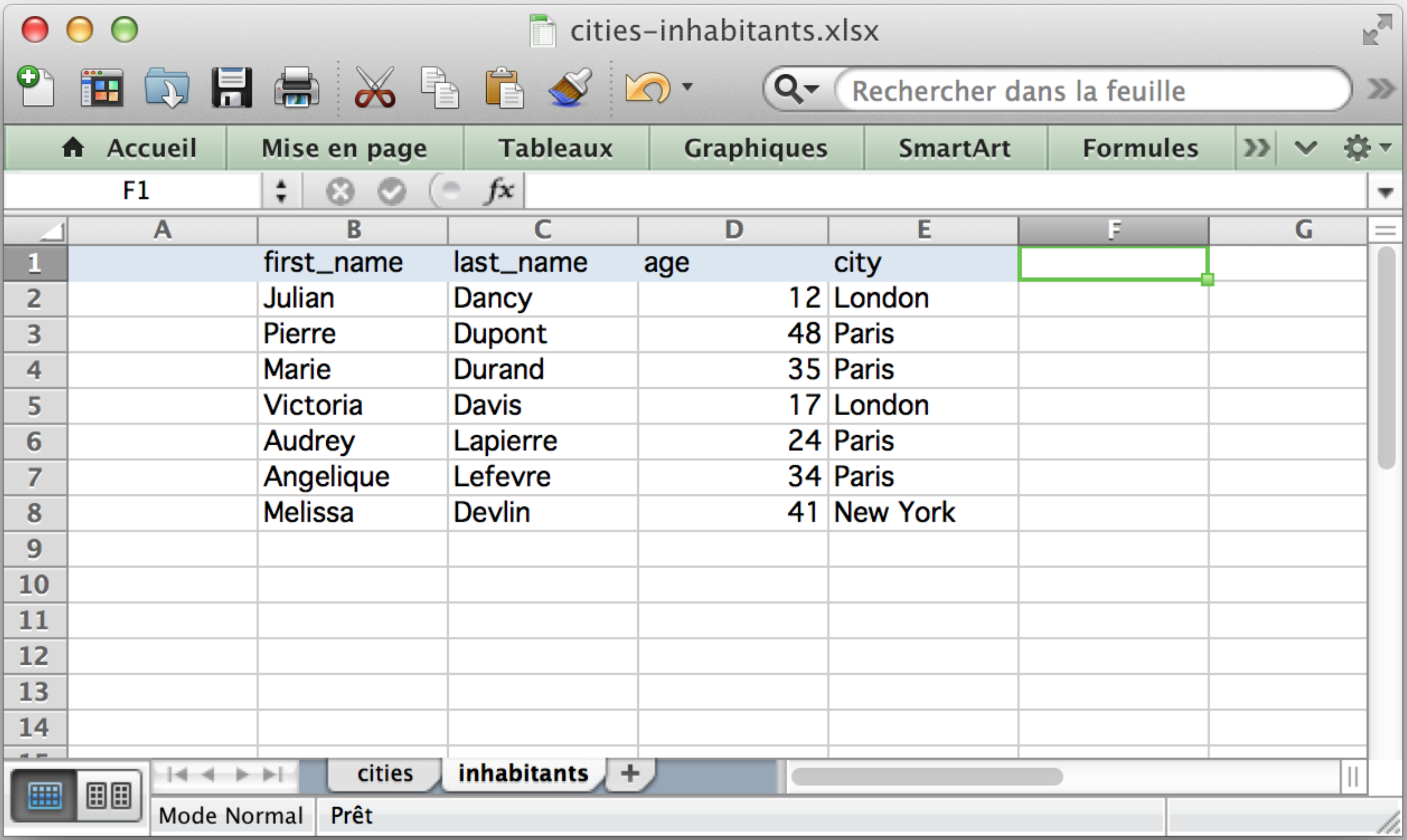
Store data. Persistently.

EXCEL

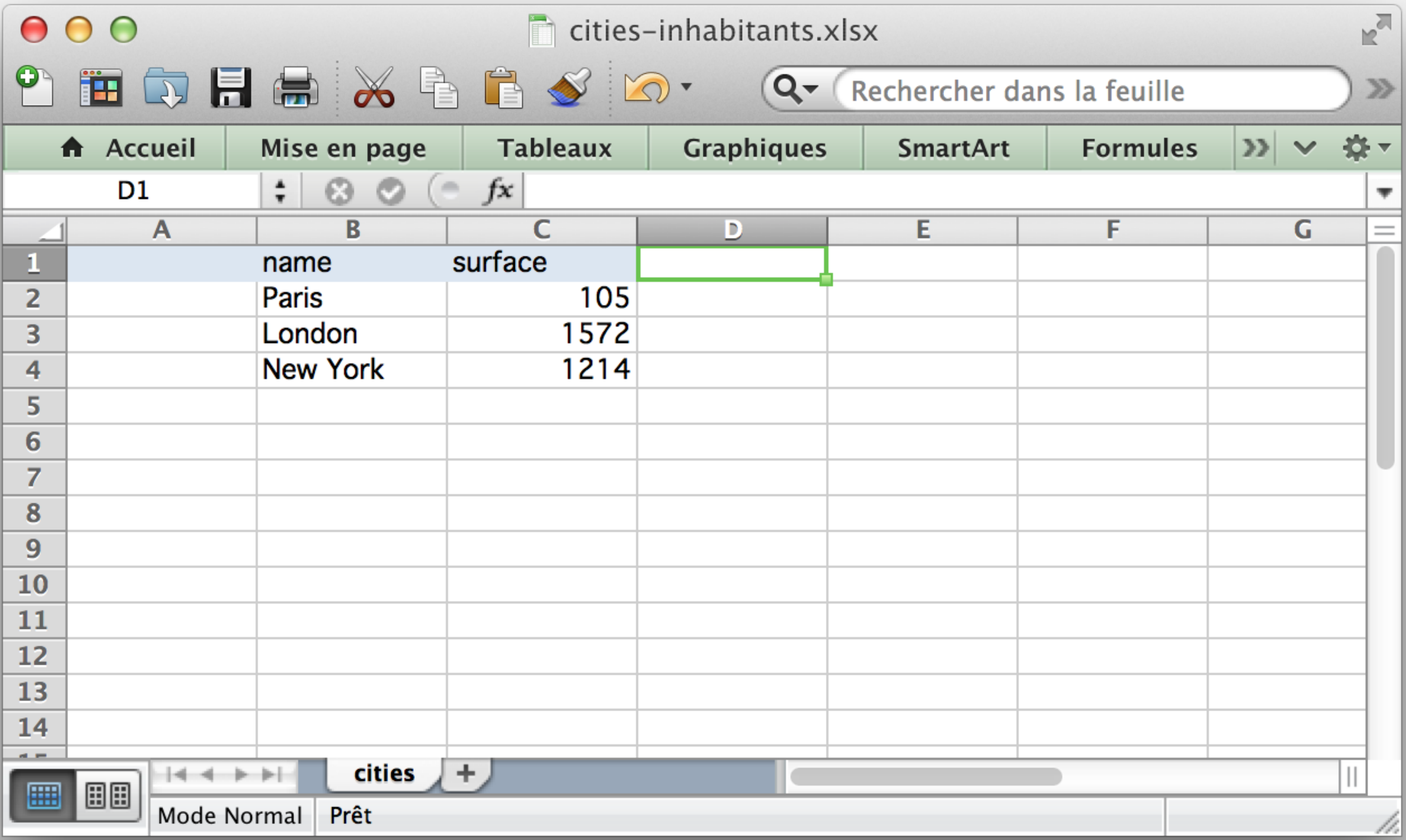
Let's start with something we all know

EXAMPLE

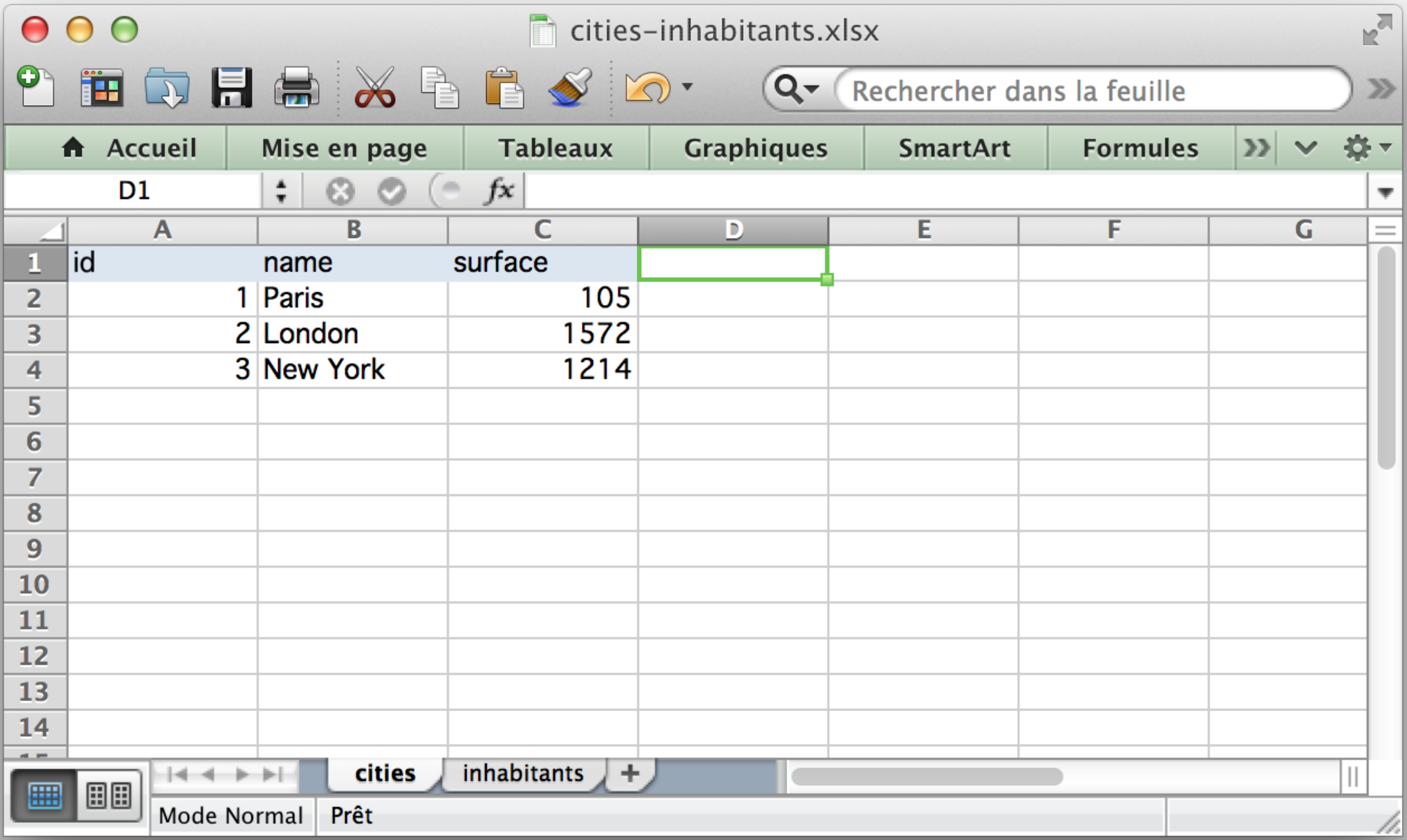
Let's store **cities** and their **inhabitants** using Excel. How would you do it?



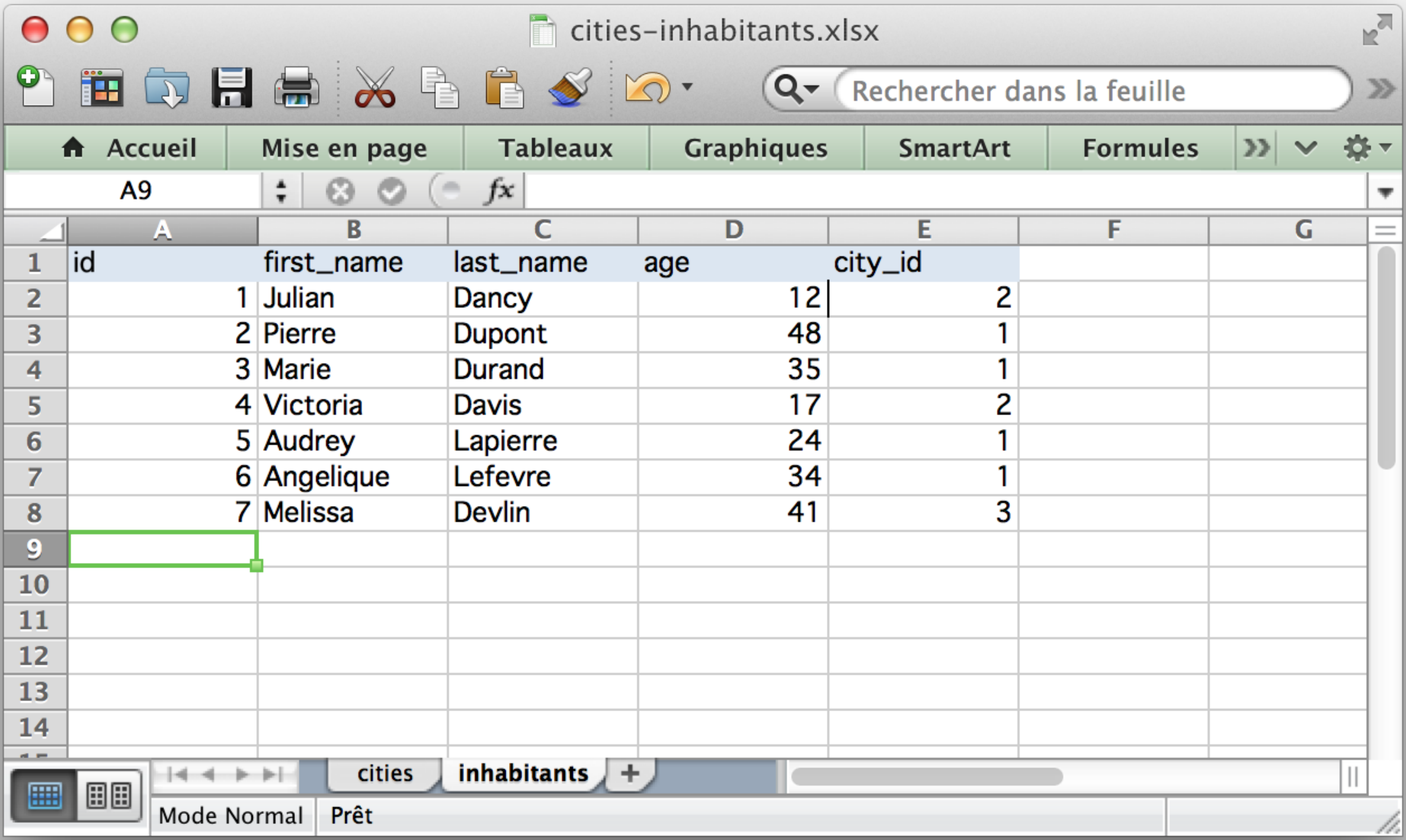
	A	B	C	D	E
1		first_name	last_name	age	city
2		Julian	Dancy	12	London
3		Pierre	Dupont	48	Paris
4		Marie	Durand	35	Paris
5		Victoria	Davis	17	London
6		Audrey	Lapierre	24	Paris
7		Angelique	Lefevre	34	Paris
8		Melissa	Devlin	41	New York



	name	surface
	Paris	105
	London	1572
	New York	1214



id	name	surface
1	Paris	105
2	London	1572
3	New York	1214



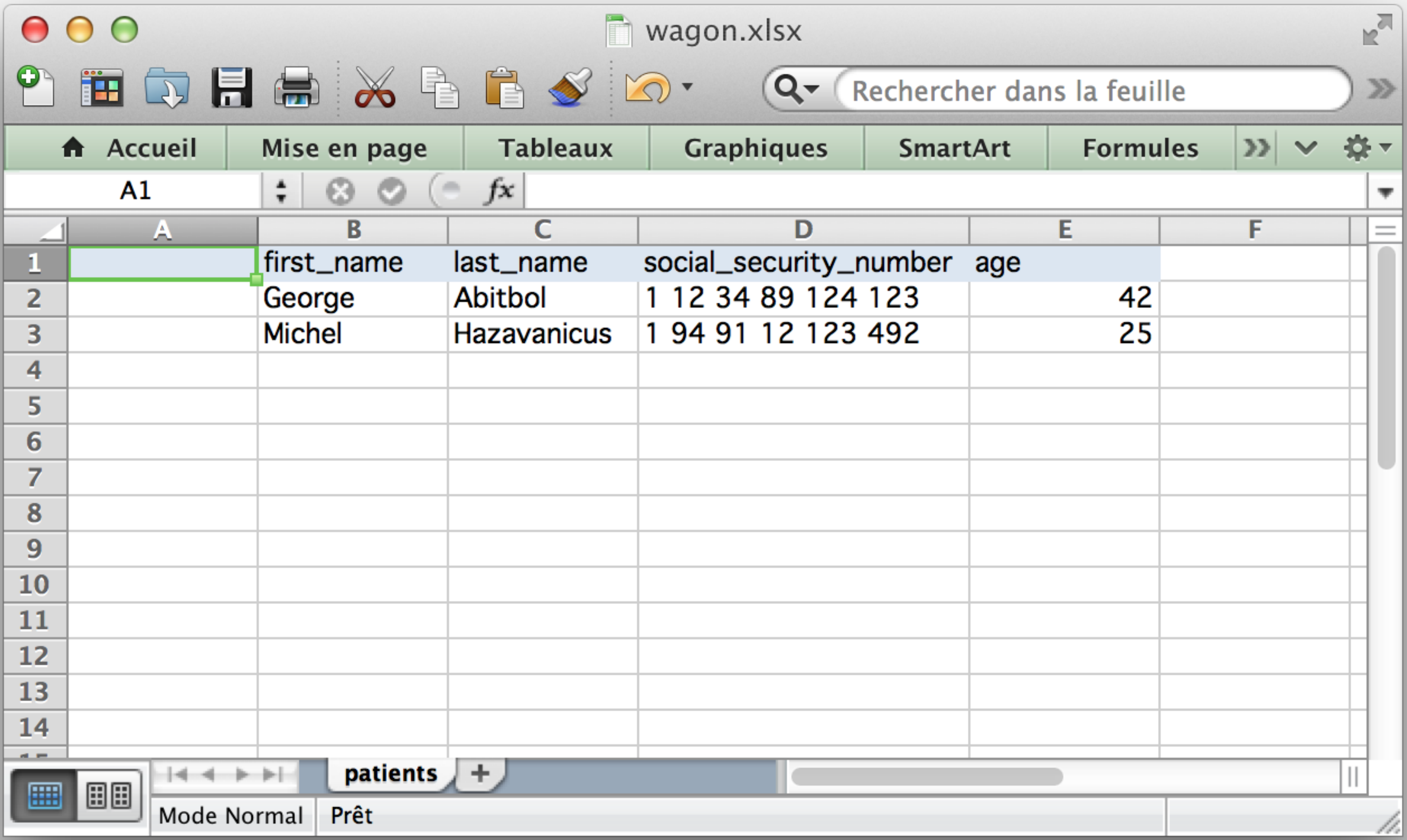
id	first_name	last_name	age	city_id
1	Julian	Dancy	12	2
2	Pierre	Dupont	48	1
3	Marie	Durand	35	1
4	Victoria	Davis	17	2
5	Audrey	Lapierre	24	1
6	Angelique	Lefevre	34	1
7	Melissa	Devlin	41	3

1:N RELATION (ONE TO MANY)

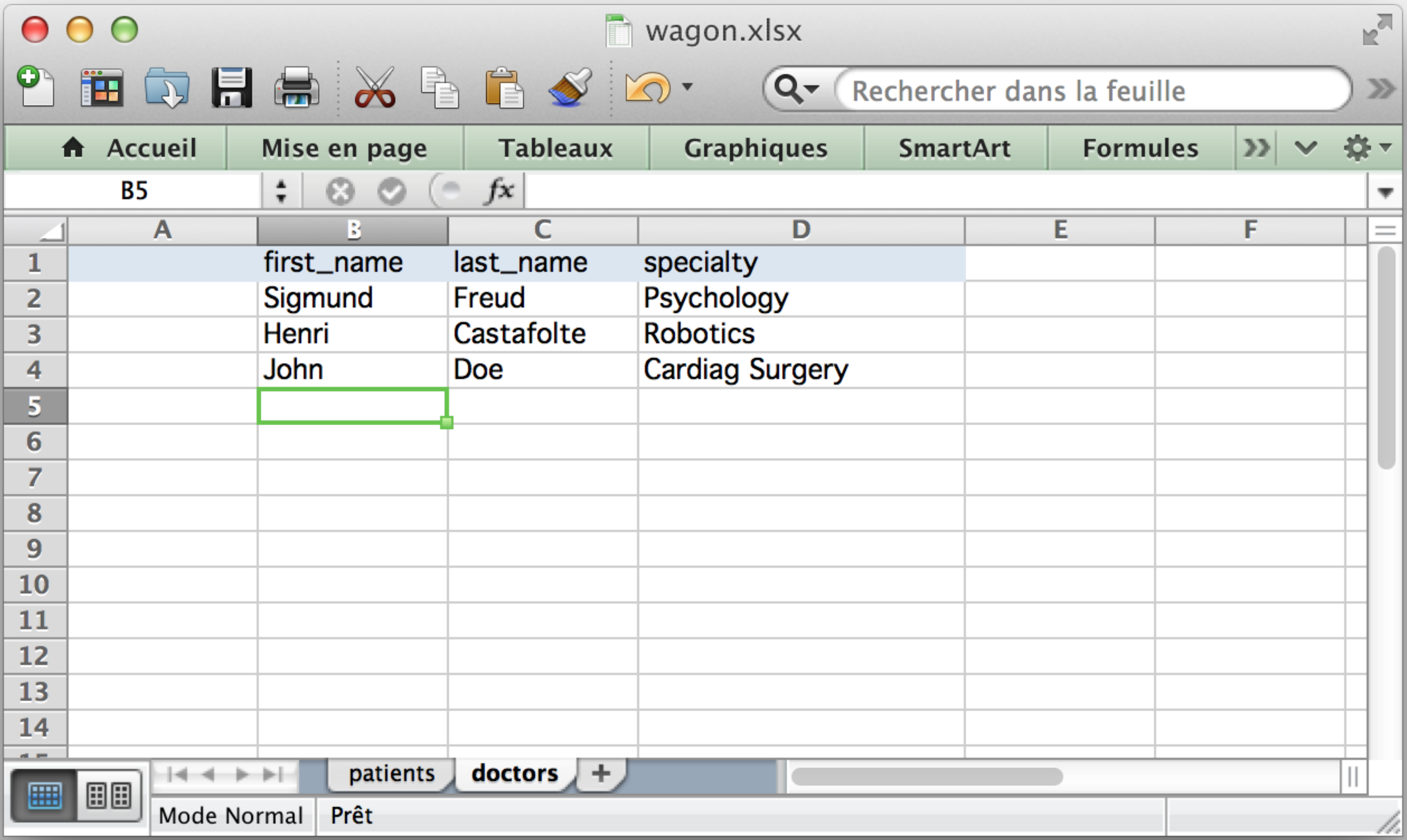
An inhabitant **belongs to** one city (or has one city)

EXCEL++

Let's go further



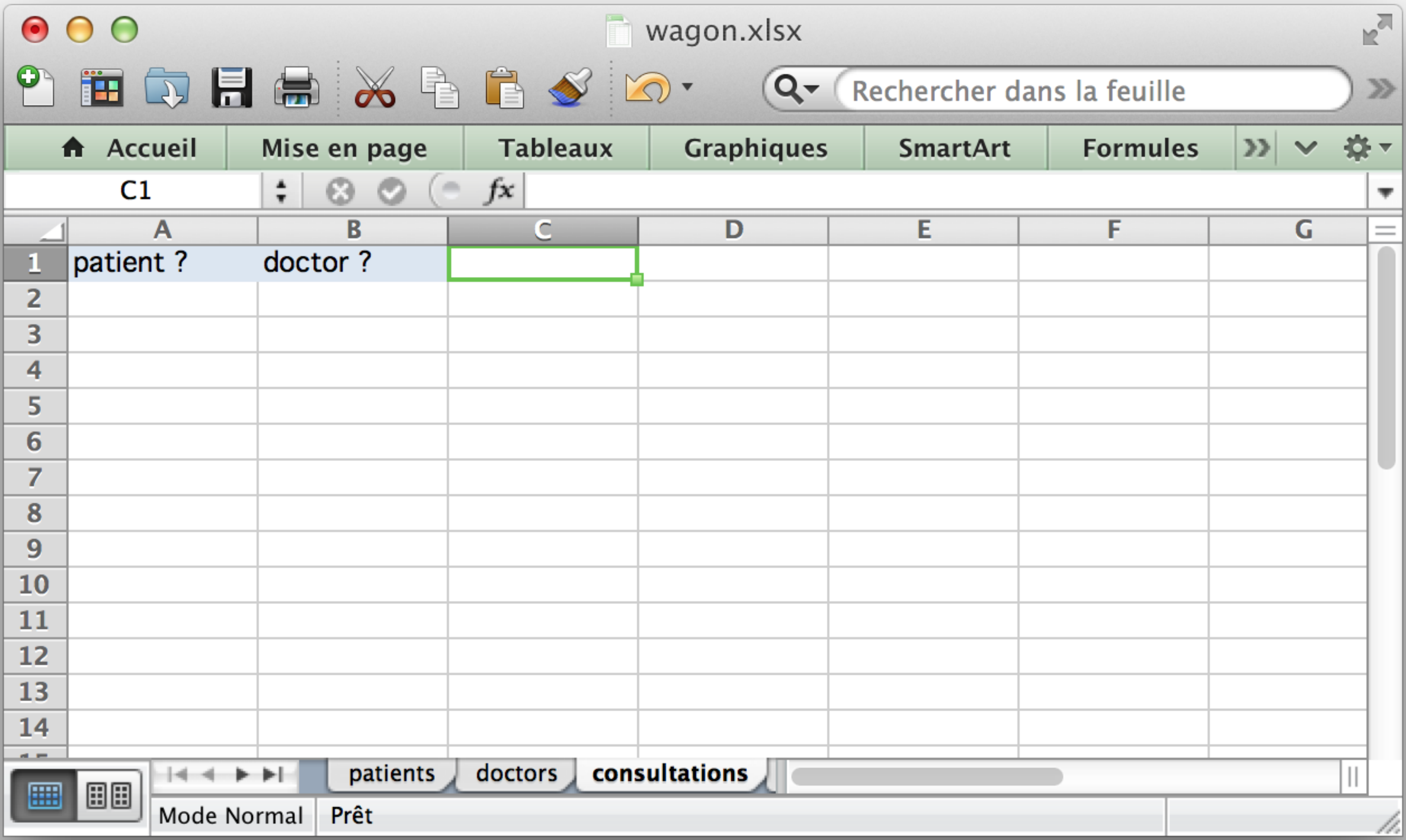
	A	B	C	D	E	F
1		first_name	last_name	social_security_number	age	
2		George	Abitbol	1 12 34 89 124 123	42	
3		Michel	Hazavanicus	1 94 91 12 123 492	25	



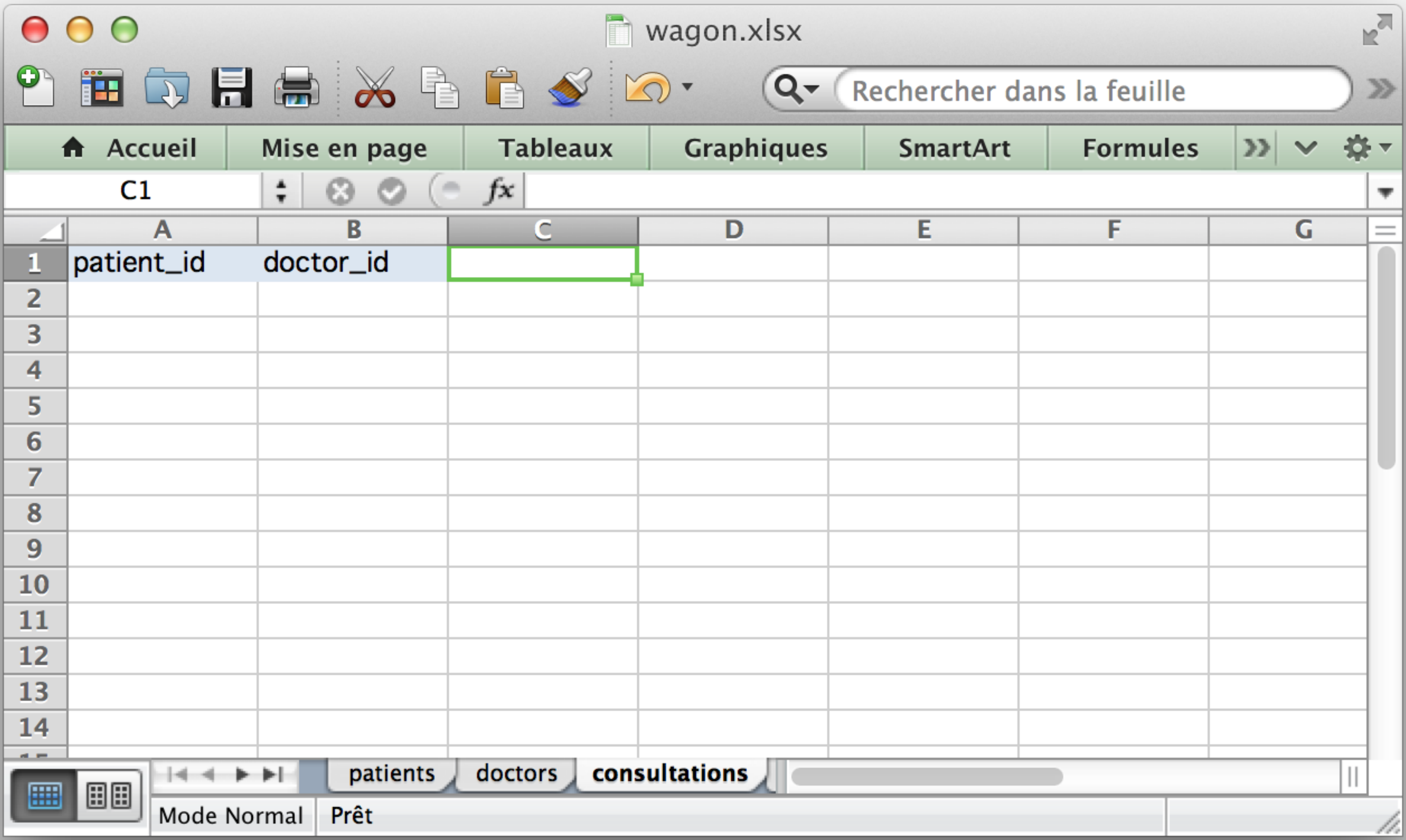
	A	B	C	D	E	F
1		first_name	last_name	specialty		
2		Sigmund	Freud	Psychology		
3		Henri	Castafolte	Robotics		
4		John	Doe	Cardiag Surgery		

CONSULTATIONS ?

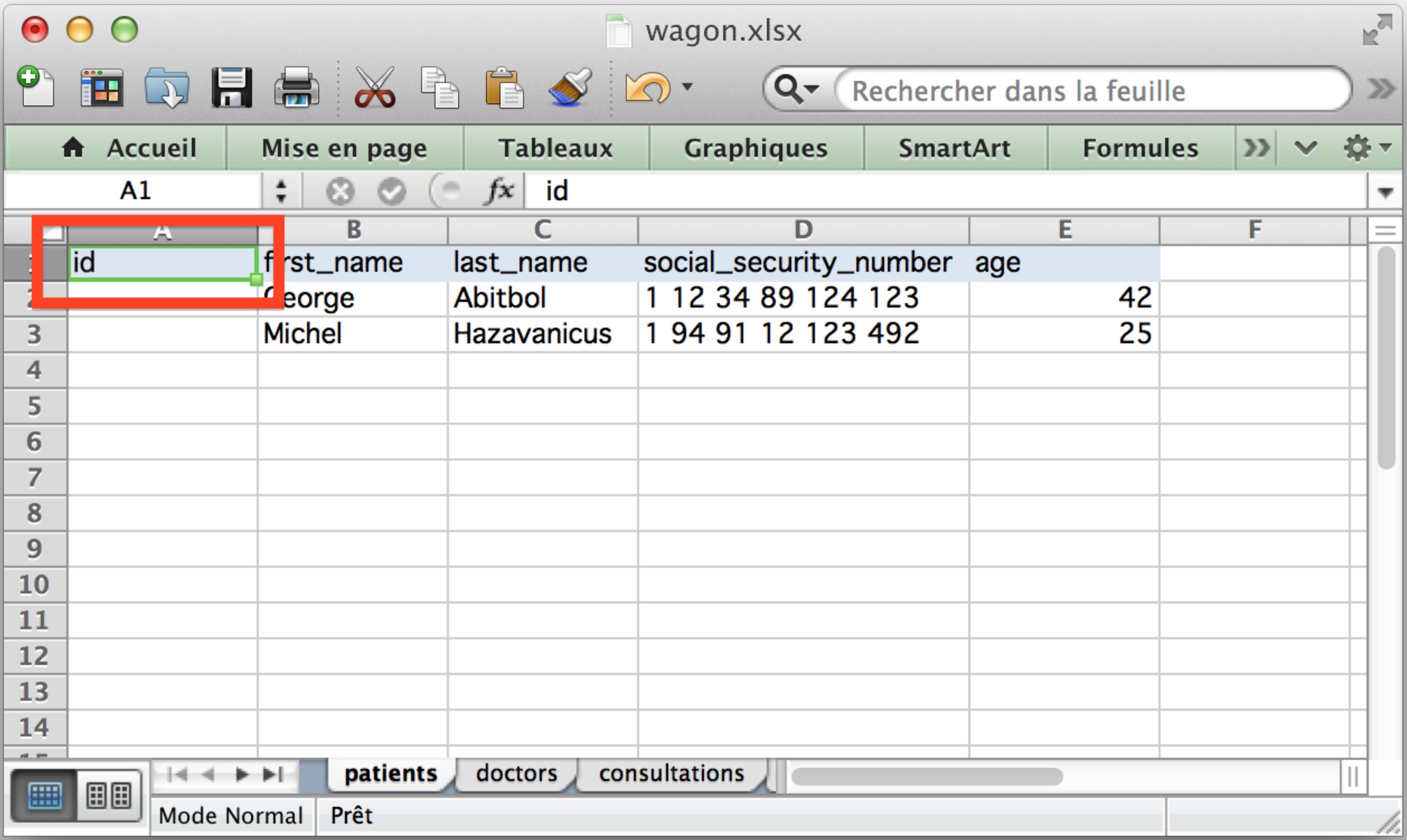
- **One** doctor can have **many** patients
- **One** patient can see **many** doctors



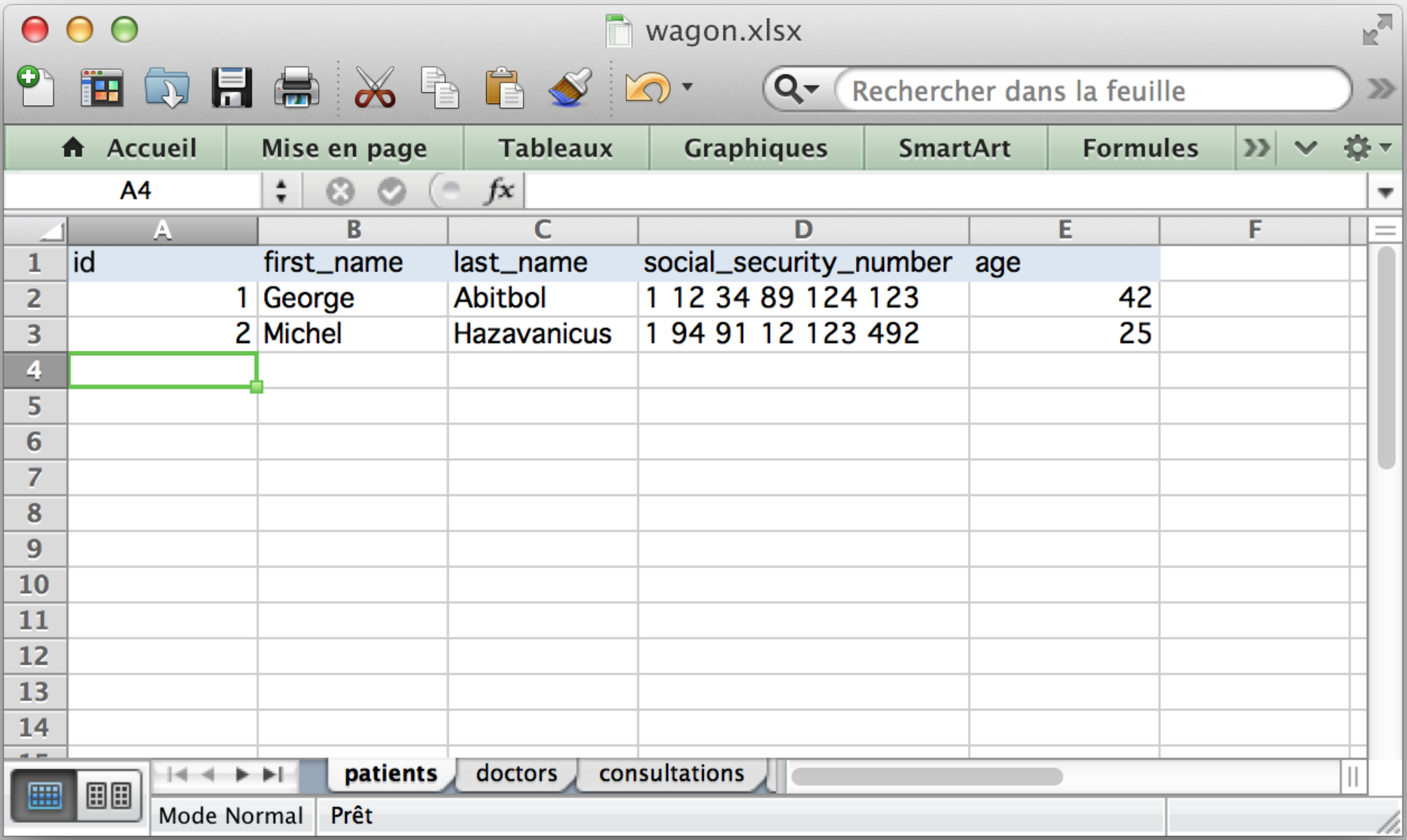
	A	B	C
1	patient ?	doctor ?	



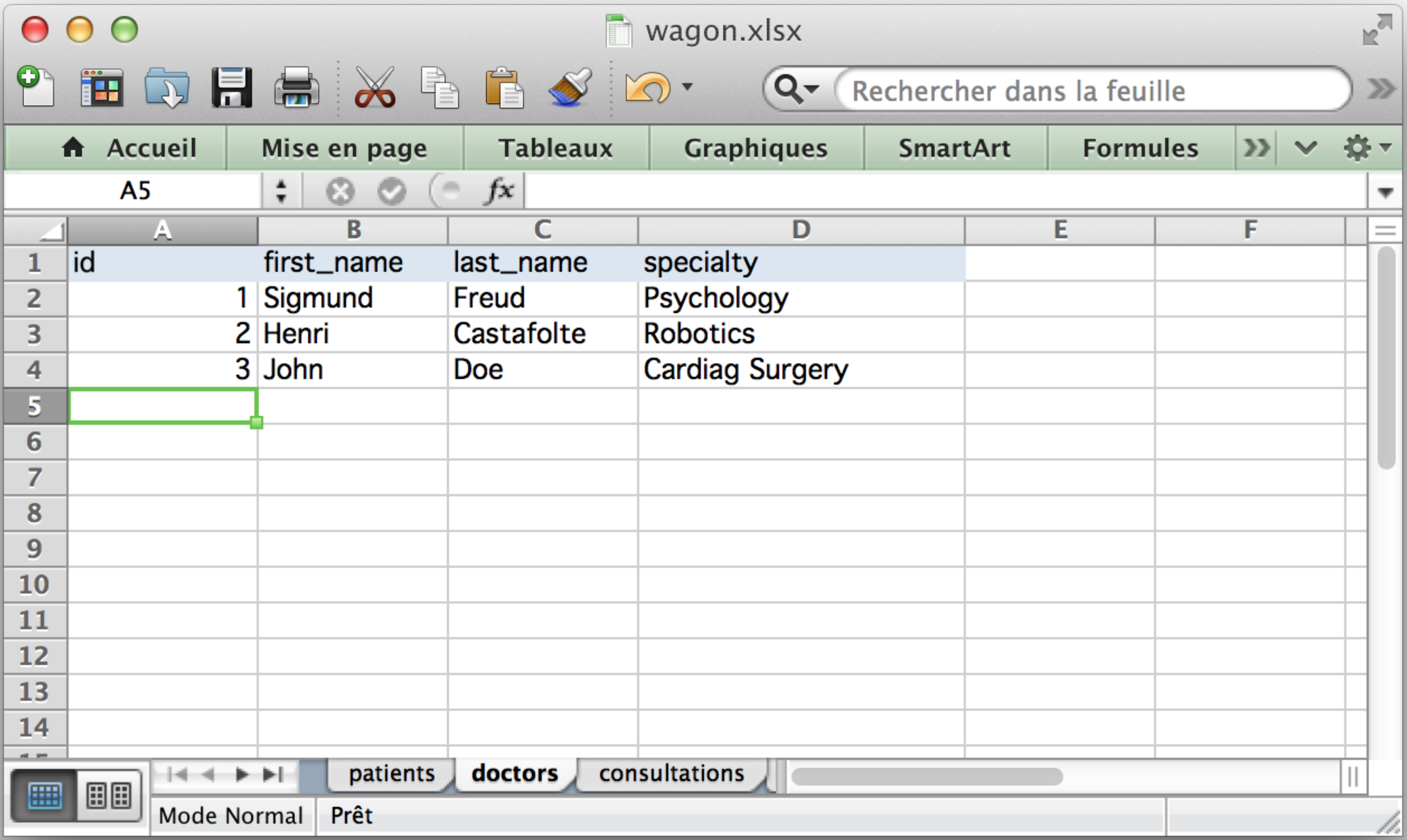
	A	B	C	D	E	F	G
1	patient_id	doctor_id					



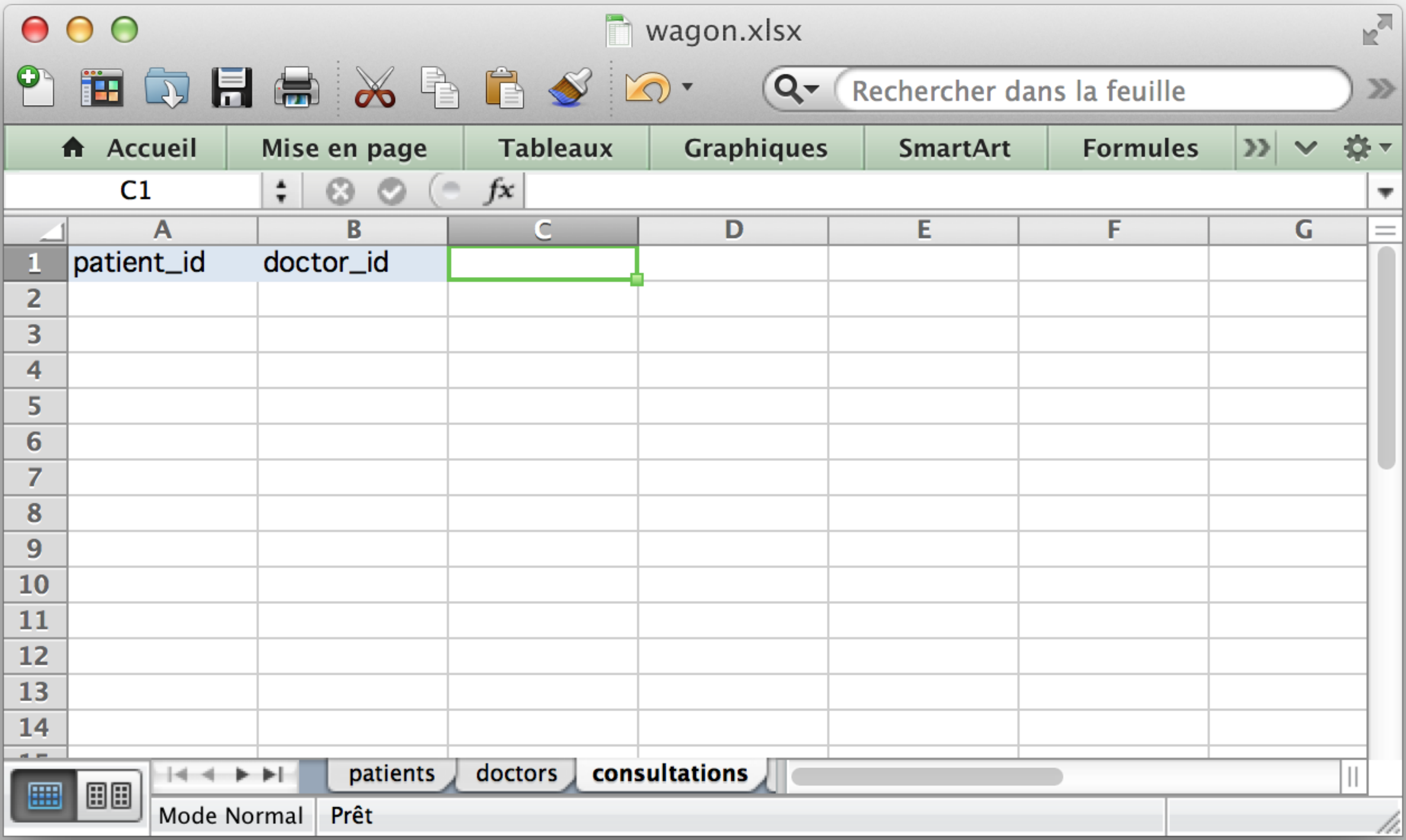
	A	B	C	D	E	F
1	id	first_name	last_name	social_security_number	age	
2		George	Abitbol	1 12 34 89 124 123	42	
3		Michel	Hazavanicus	1 94 91 12 123 492	25	



id	first_name	last_name	social_security_number	age
1	George	Abitbol	1 12 34 89 124 123	42
2	Michel	Hazavanicus	1 94 91 12 123 492	25

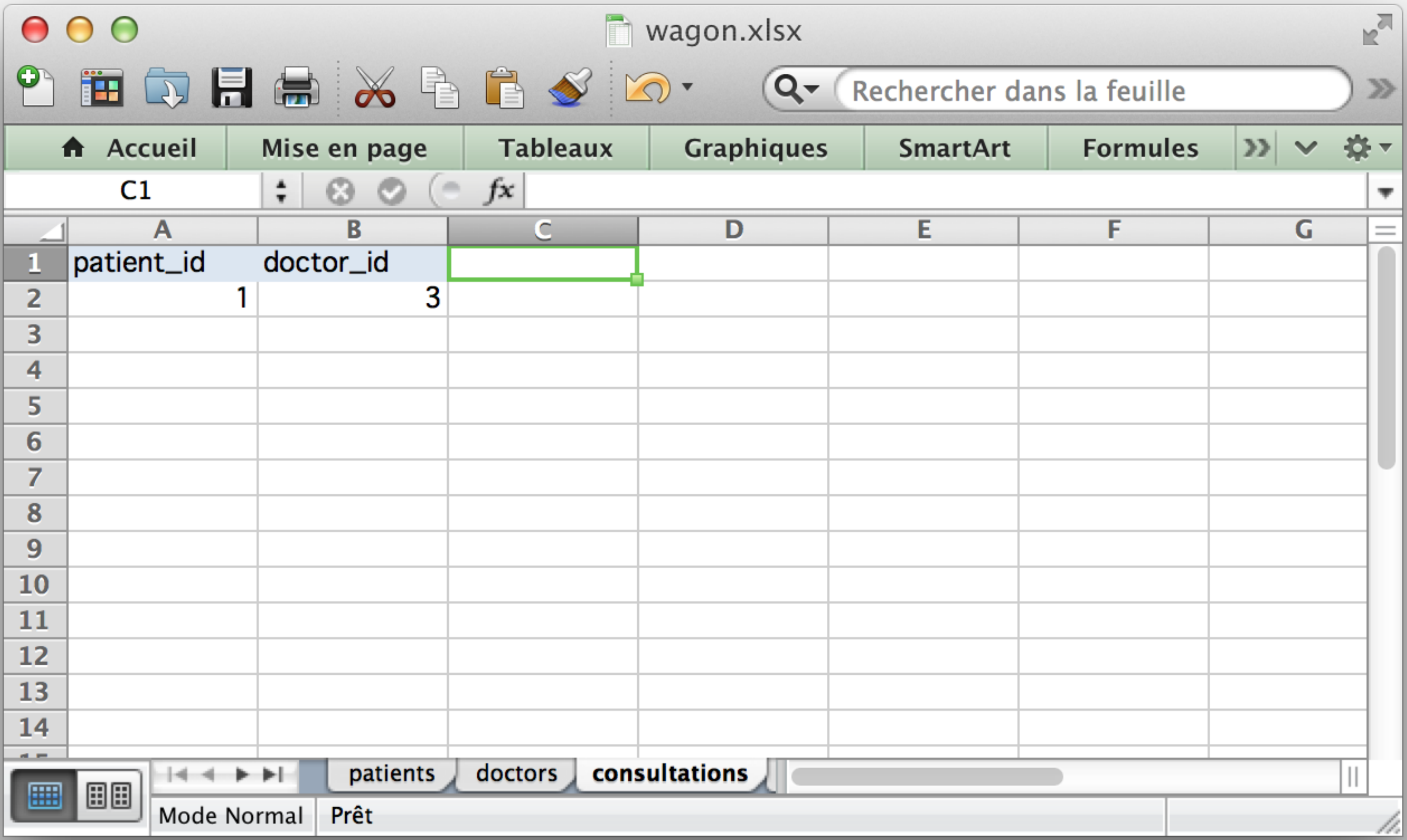


id	first_name	last_name	specialty
1	Sigmund	Freud	Psychology
2	Henri	Castafolte	Robotics
3	John	Doe	Cardiag Surgery



	A	B	C	D	E	F	G
1	patient_id	doctor_id					

George Abitbol (`id = 1`) has seen Doctor John Doe (`id = 3`)



	A	B
1	patient_id	doctor_id
2	1	3

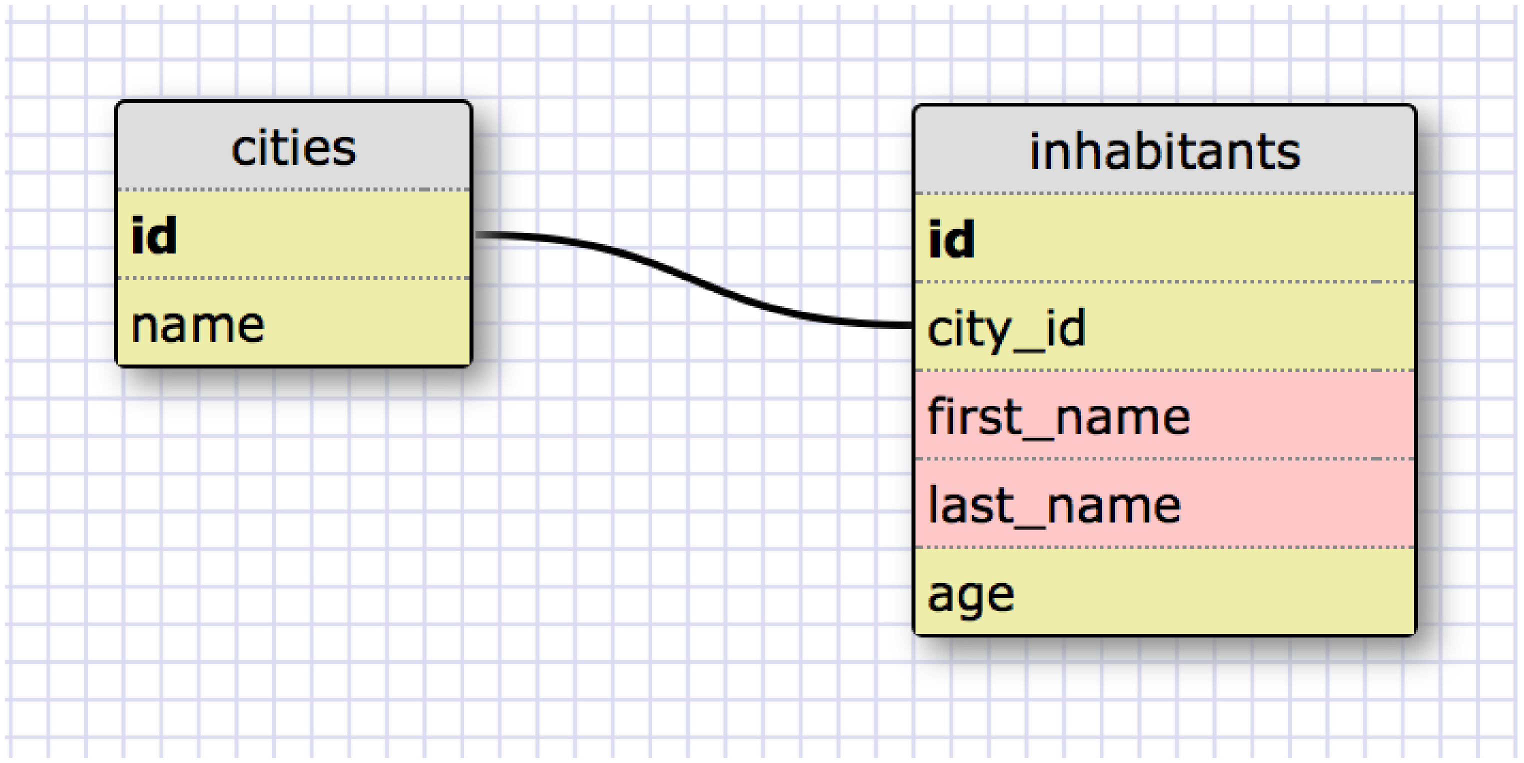
N:N RELATION (MANY TO MANY)

A patient **has many** doctors and a doctor **has many** patients.

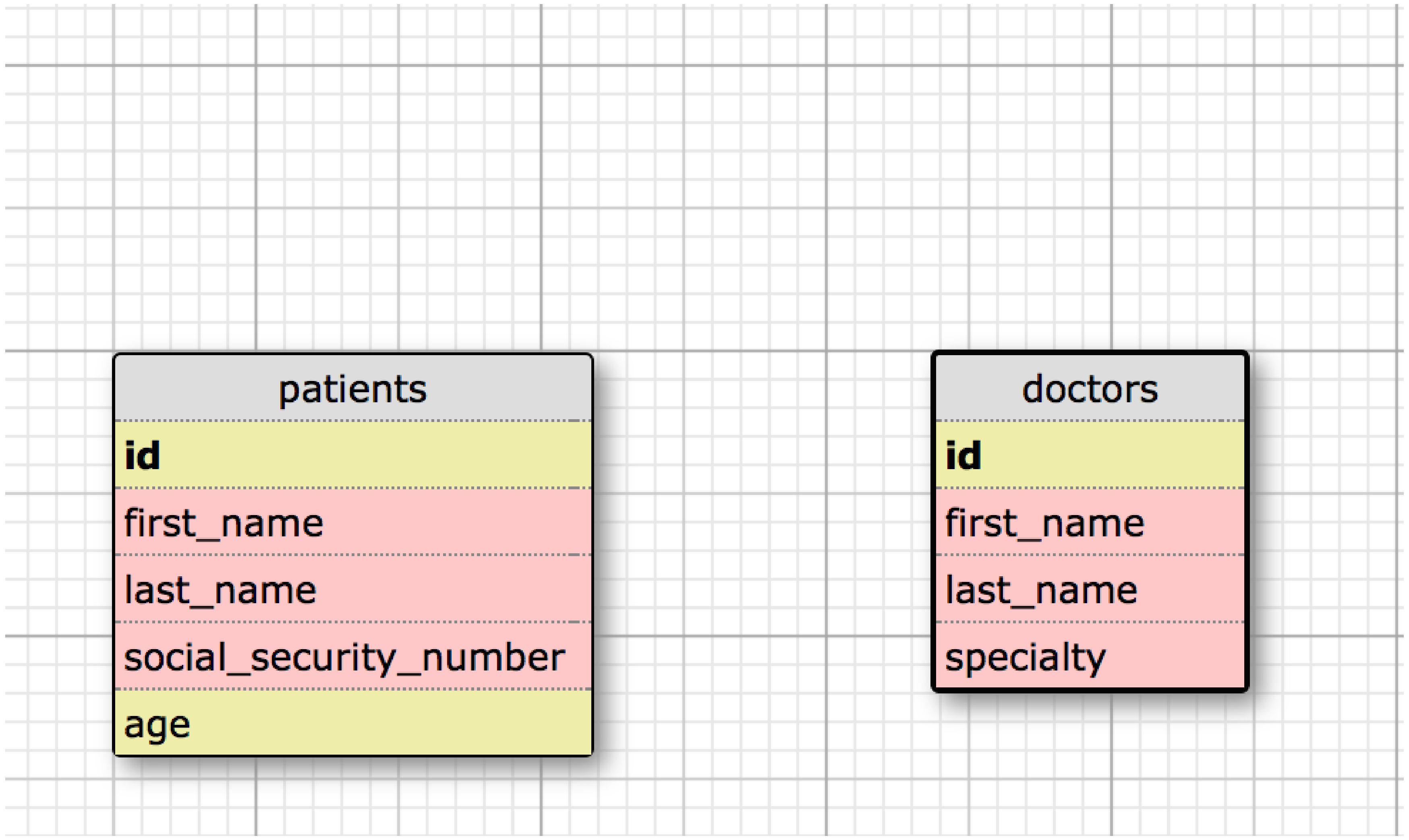
You can download this example: consultations.xslx

RELATIONAL DATABASE

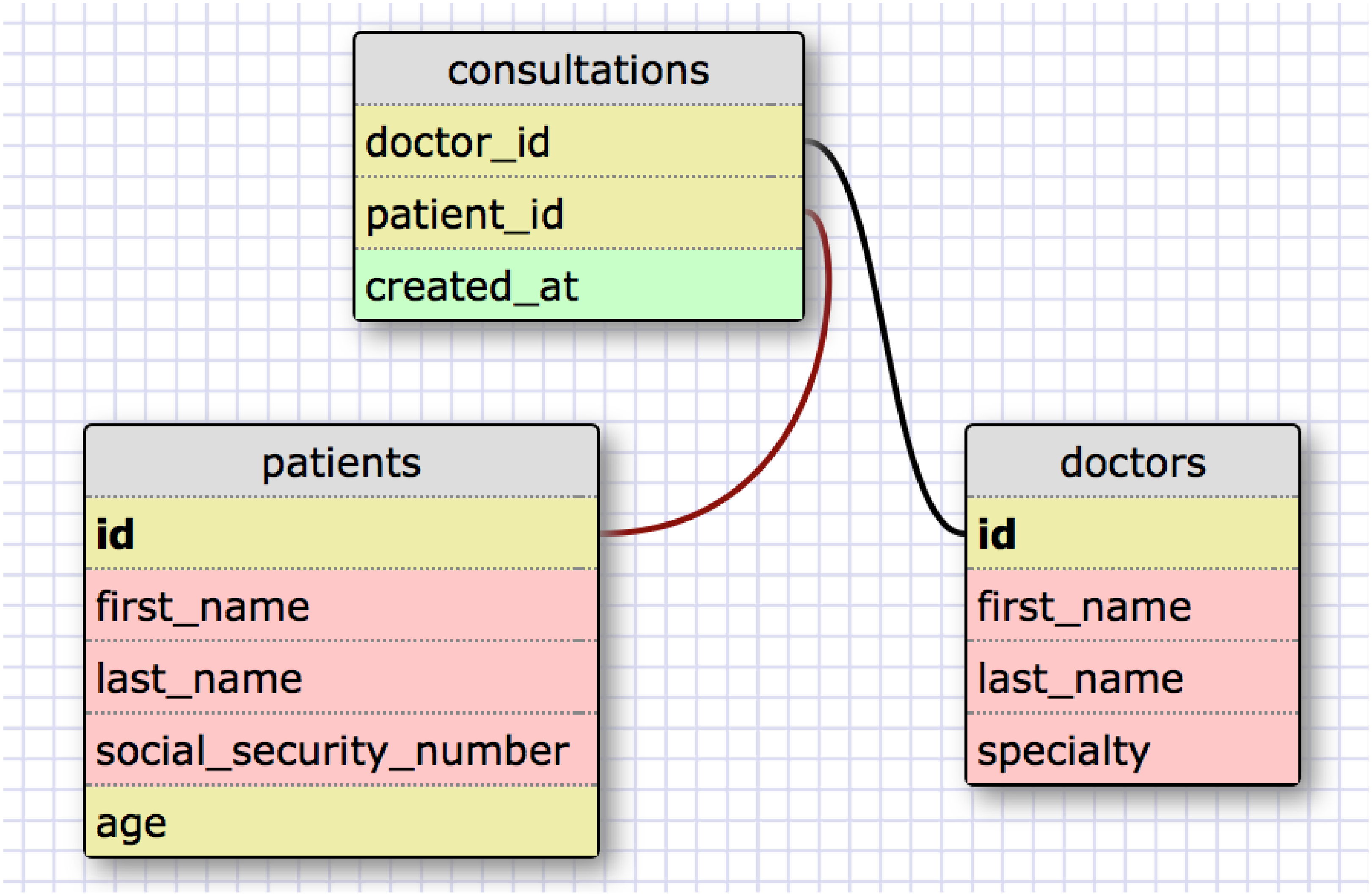
1:N


cities
id
name
inhabitants
id
city_id
first_name
last_name
age

N:N ?


patients
id
first_name
last_name
social_security_number
age
doctors
id
first_name
last_name
specialty

N:N


consultations
doctor_id
patient_id
created_at
patients
id
first_name
last_name
social_security_number
age
doctors
id
first_name
last_name
specialty

VOCABULARY

- A schema is composed of **tables**.
- Each table has a set of **columns**.
- When inserting data in a table, you create a **record** in a new **row**.

DB SCHEMA COMPOSER

db.lewagon.com

You can save/load schemas. Try with patients-doctors.xml

QUERYING

Retrieve data using the schema.

SQL

Structured Query Language

GIVE ME ALL PATIENT NAMES

```
SELECT first_name, last_name FROM patients
```

GIVE ME ALL DOCTOR NAMES

```
SELECT first_name, last_name FROM doctors
```

GIVE ME ALL YOU GOT ABOUT PATIENTS

```sql
SELECT * FROM patients
```

GIVE ME ALL PATIENTS OF AGE 21

```
SELECT * FROM patients WHERE age = 21
```

GIVE ME ALL DOCTORS OF CARDIAC SURGERY SPECIALTY

```
SELECT * FROM doctors WHERE specialty = 'Cardiac Surgery'
```

GIVE ME ALL SURGERY DOCTORS

```sql
SELECT * FROM doctors WHERE specialty LIKE '%Surgery'
```

GIVE ME ALL CARDIAC SURGERY DOCTORS NAMED STEVE

```sql
SELECT * FROM doctors
WHERE specialty = 'Cardiac Surgery'
AND first_name = 'Steve'
```

GIVE ME ALL PATIENTS ORDERED BY AGE

```
SELECT * FROM patients ORDER BY age ASC
```

```
SELECT * FROM patients ORDER BY age DESC
```

HOW MANY DOCTORS DO I HAVE?

```
SELECT COUNT(*) FROM doctors
```

COUNT CARDIAC SURGERY DOCTORS

```
SELECT COUNT(*) FROM doctors WHERE specialty = 'Cardiac Surge
```

COUNT ALL DOCTORS PER SPECIALTY

```
SELECT COUNT(*), specialty
FROM doctors
GROUP BY specialty
```

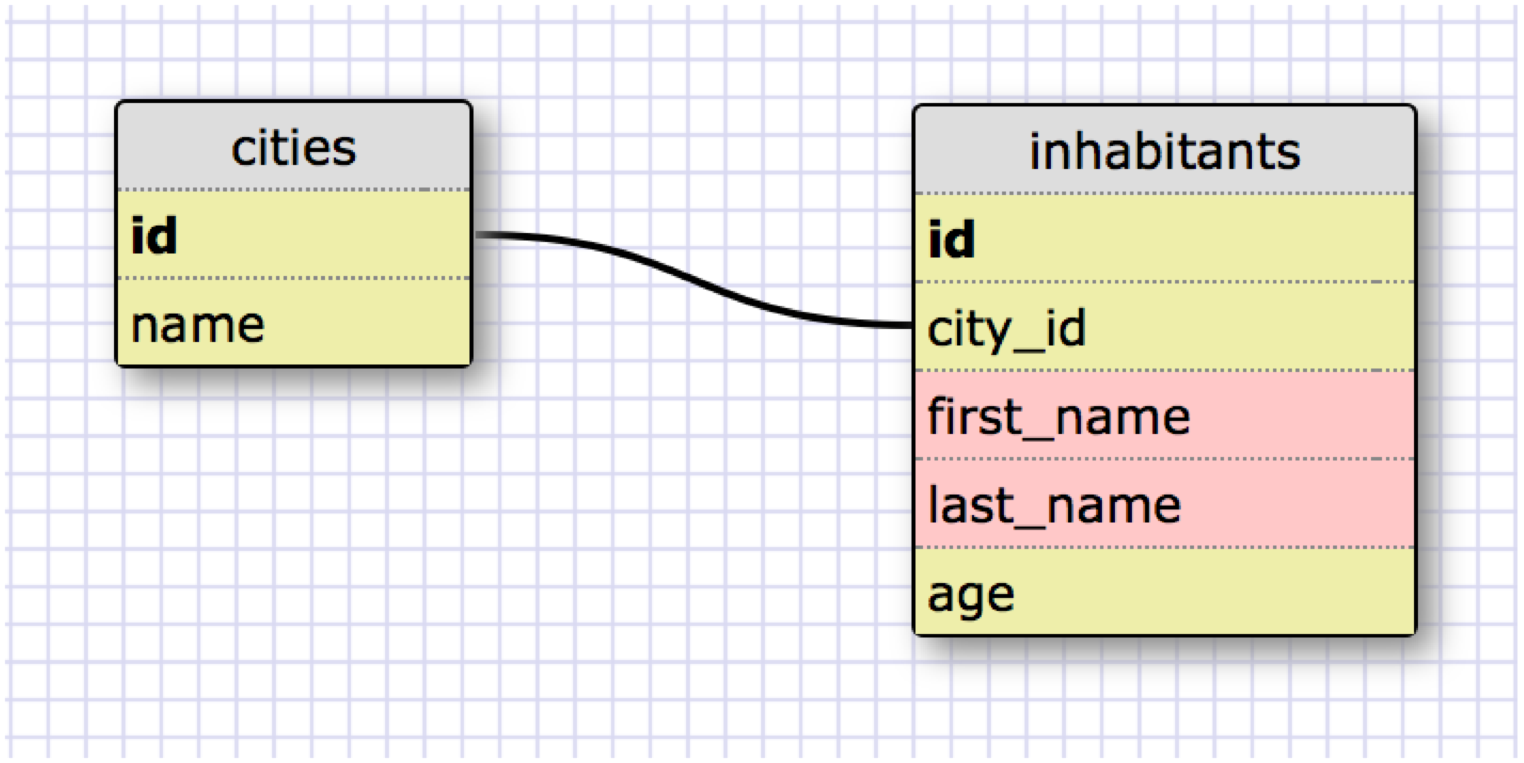
COUNT ALL DOCTORS PER SPECIALTY, ORDER BY SPECIALTY

You need to rename result column, with `AS`.

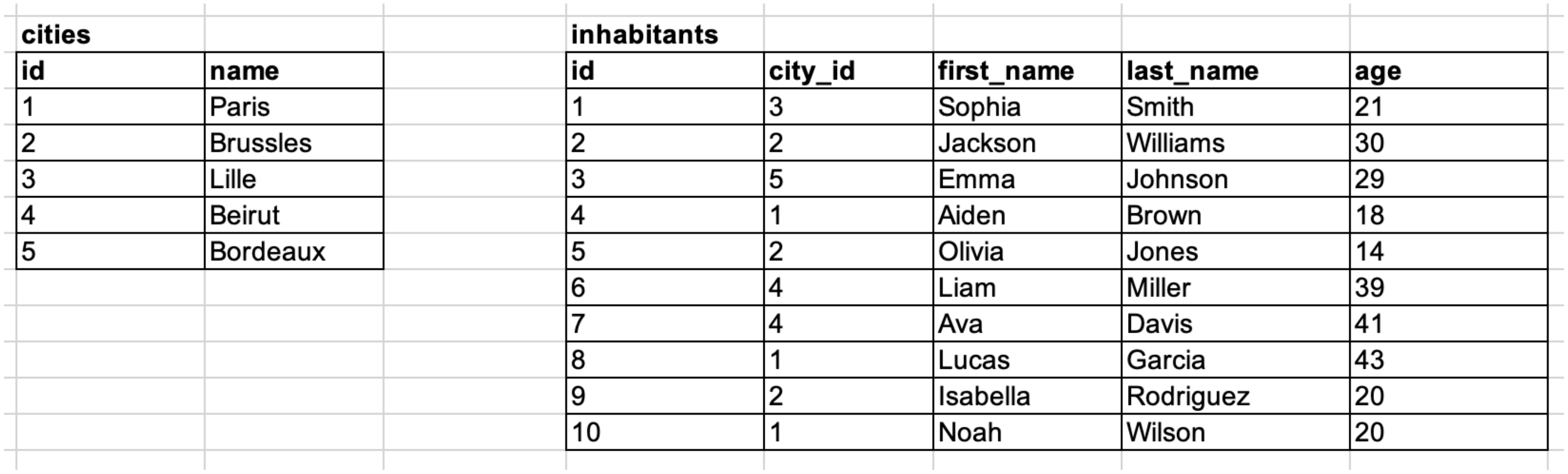
```sql
SELECT COUNT(*) AS c, specialty
FROM doctors
GROUP BY specialty
ORDER BY c DESC
```

USING 2 OR MORE TABLES AT ONCE

GIVEN THIS CITIES/INHABITANTS SCHEMA...



… AND THIS DATA



cities

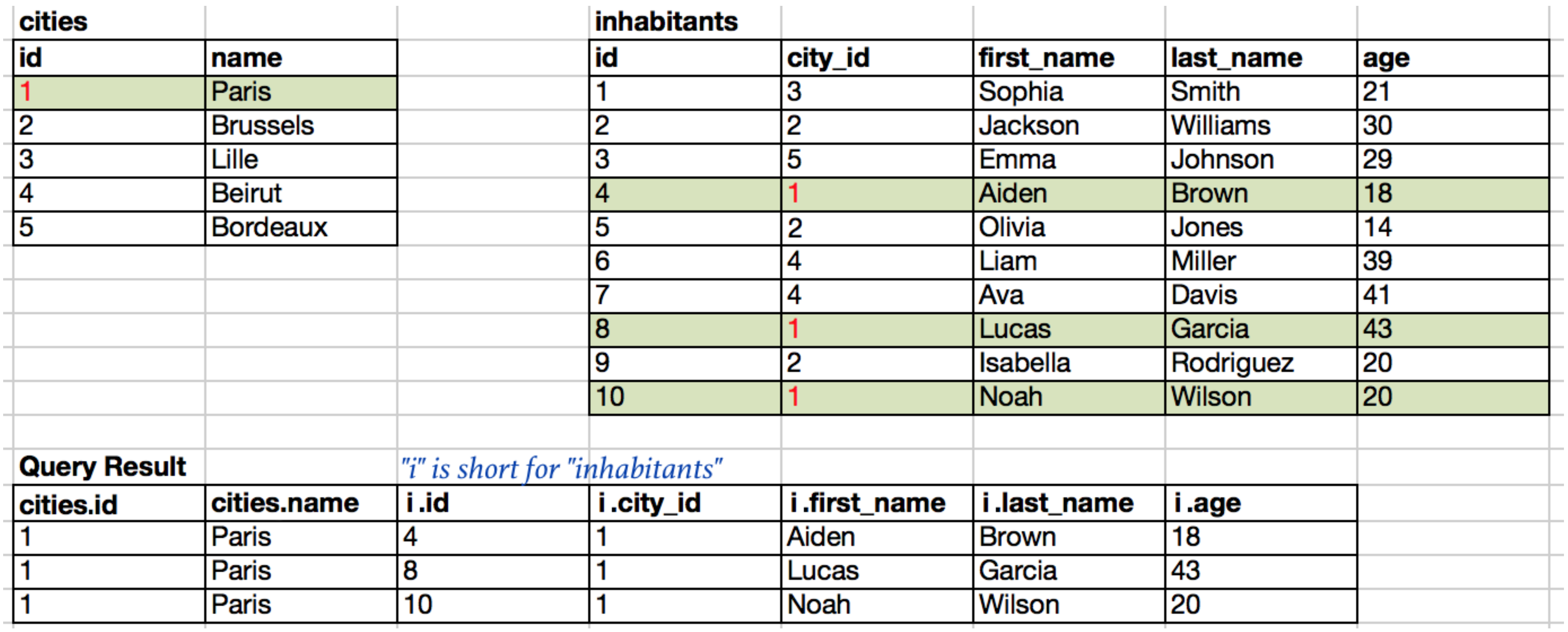
id	name
1	Paris
2	Brussles
3	Lille
4	Beirut
5	Bordeaux

inhabitants

id	city_id	first_name	last_name	age
1	3	Sophia	Smith	21
2	2	Jackson	Williams	30
3	5	Emma	Johnson	29
4	1	Aiden	Brown	18
5	2	Olivia	Jones	14
6	4	Liam	Miller	39
7	4	Ava	Davis	41
8	1	Lucas	Garcia	43
9	2	Isabella	Rodriguez	20
10	1	Noah	Wilson	20

QUESTION: GIVE ME ALL THE INHABITANTS FROM PARIS

```
SELECT * FROM inhabitants
JOIN cities ON cities.id = inhabitants.city_id
WHERE cities.name = 'Paris'
```



cities

id	name
1	Paris
2	Brussels
3	Lille
4	Beirut
5	Bordeaux

inhabitants

id	city_id	first_name	last_name	age
1	3	Sophia	Smith	21
2	2	Jackson	Williams	30
3	5	Emma	Johnson	29
4	1	Aiden	Brown	18
5	2	Olivia	Jones	14
6	4	Liam	Miller	39
7	4	Ava	Davis	41
8	1	Lucas	Garcia	43
9	2	Isabella	Rodriguez	20
10	1	Noah	Wilson	20

Query Result

"i" is short for "inhabitants"

cities.id	cities.name	i.id	i.city_id	i.first_name	i.last_name	i.age
1	Paris	4	1	Aiden	Brown	18
1	Paris	8	1	Lucas	Garcia	43
1	Paris	10	1	Noah	Wilson	20

QUESTION: GIVE ME ALL THE ADULTS LIVING IN PARIS

```sql
SELECT * FROM inhabitants
JOIN cities ON cities.id = inhabitants.city_id
WHERE inhabitants.age >= 18
AND cities.name = 'Paris'
```

GIVEN THIS CONSULTATIONS SCHEMA

QUESTION: FOR EACH CONSULTATION, GIVE ME ITS DATE, PATIENT AND DOCTOR NAMES

```sql
SELECT c.created_at, p.first_name, p.last_name, d.first_name,
FROM consultations c
JOIN patients p ON c.patient_id = p.id
JOIN doctors d ON c.doctor_id = d.id;
```

GOING FURTHER

You can read more about `INNER` (the default one), `LEFT OUTER`, `RIGHT OUTER` or `FULL OUTER JOIN` here, here and there.

SQLITE

It is a simple database storing everything in **one** file. Great to quickly test, but not suited for production.

INSTALLATION

On OSX, run this:

```
brew install sqlite
```

On Ubuntu, run this:

```
sudo apt-get install sqlite3 libsqlite3-dev
```

QUICK START

Create a new folder, and go into it.

Create a DB and start typing SQL queries:

```
sqlite3 db.sqlite
```

It will create the `db.sqlite` file.

TABLE CREATION

```
CREATE TABLE `cities` (
  `id` INTEGER PRIMARY KEY AUTOINCREMENT,
  `name` VARCHAR
);
```

INSERTING AND QUERYING

```
sqlite> INSERT INTO cities ('name') VALUES ('Paris');
sqlite> INSERT INTO cities ('name') VALUES ('London');
```

```
sqlite> .mode column
sqlite> .headers on
```

```
sqlite> SELECT * FROM cities;
id          name
----------  ----------
1           Paris
2           London
```

HELP

```
sqlite> .help
```

```
.exit                    Exit this program
.header(s) ON|OFF        Turn display of headers on or off
.read FILENAME           Execute SQL in FILENAME
.schema ?TABLE?          Show the CREATE statements
                           If TABLE specified, only show tables
                           LIKE pattern TABLE.
.show                    Show the current values for various se
.tables ?TABLE?          List names of tables
                           If TABLE specified, only list tables
                           LIKE pattern TABLE.
[...]
```

IN 2 WEEKS: POSTGRESQL