No, it is NOT the same for all databases.

While the analogy I gave you works perfectly for **PostgreSQL** and its "Enterprise" cousins (like Oracle and SQL Server), other popular databases work quite differently.

Here is how the "Big Three" you will likely encounter as a developer differ:

1. MySQL (The "Relaxed" Cousin)

MySQL is very similar to PostgreSQL, but it simplifies the hierarchy.

- **PostgreSQL:** Server -> Database -> Schema -> Table
- **MySQL:** Server -> Database -> Table
- **The Difference:** In MySQL, **"Database" and "Schema" are synonyms.** They mean exactly the same thing. You don't have that extra "Floor" level. You just have the Tower (Database) and the Rooms (Tables) inside it.

2. MongoDB (The "NoSQL" Rebel)

This is completely different. It doesn't use the "Apartment/Room" structure at all. It works more like a **Warehouse**.

- **Analogy:** Instead of rigid rooms with built-in shelves (Columns) where everything must fit perfectly, MongoDB is a big open floor.

- **Structure:**

- **Database:** (The Warehouse)

- **Collection:** (A Pile of Boxes - replaces "Table")

- **Document:** (A Cardboard Box - replaces "Row")

- **The Key Difference:** In PostgreSQL, every row in the `users` table MUST have an `age` column. In MongoDB, one box can have an age, and the box next to it can have no age but have a "favorite_color" instead. It is **flexible**.

3. SQLite (The "Portable" One)

This is what your phone (Android/iOS) uses internally.

- **Analogy:** It is not a building. It is a **Single Suitcase**.
- **Structure:** There is **no Server**. There is **no Port 5432**.
- **How it works:** The entire database is just **one single file** (`data.db`) sitting in your folder. Your Python code just opens that file directly like a text document. It is extremely simple but not good for giant web apps like Amazon.

Summary Table

Database	Server (Building)?	Schema (Floors)?	Rigid Tables (Rooms)?	Best For...

Database	Server (Building)?	Schema (Floors)?	Rigid Tables (Rooms)?	Best For...
PostgreSQL	Yes	**Yes**	Yes	Complex, reliable apps (Amazon, Banking)
MySQL	Yes	**No** (Merged)	Yes	Web apps (WordPress, Facebook)
MongoDB	Yes	No	**No** (Flexible)	Fast, changing data (Real-time feeds)
SQLite	**No** (File)	No	Yes	Mobile apps, small tests

Since you are building an Amazon Clone: Stick with **PostgreSQL**. The rigid structure is exactly what you need to make sure an Order doesn't exist without a Customer!