This is the single most powerful line in your startup code. It is the "Construction Crew" that builds your database infrastructure.

Here is the breakdown of the syntax:

1. `models`

This is simply the file you imported (`from . import models`).

- **What it holds:** It contains your Python classes like `class User(Base)` and `class Post(Base)`.
- **Why we need it:** We need to access the "blueprint" of your tables.

2. `.Base`

Inside your `database.py` (which `models` uses), you created a special variable: `Base = declarative_base()`.

- **What it does:** Think of `Base` as a **Registry** or a "Catalog".
- **How it works:** Every time you write `class User(Base):`, that class automatically registers itself in this Catalog.
- **The Result:** By the time `main2.py` runs, `Base` knows about *every single table* you defined in your code.

3. `.metadata`

This is the internal information stored inside `Base`.

- It holds the specific details for SQL: "User table has a column named 'email' which is a String(200)".
- It converts your Python code (classes) into SQL Schema data.

4. `.create_all(...)`

This is the command to take action.

- **The Logic:** It looks at the database and asks, "Do these tables exist?"
- If **No**: It runs `CREATE TABLE users ...`.
- If **Yes**: It does **nothing**. (It is smart enough not to delete or overwrite existing data).

5. `(bind=engine)`

This tells SQLAlchemy *where* to build the tables.

- **The Engine:** Remember `engine` contains your URL (`postgresql://...`).
- **Translation:** "Take all the blueprints found in `Base`, and go build them inside the database connected to this `engine`."

Summary in Plain English

> "Hey **Models**, look at your **Base** catalog where all the tables are registered. Take their **metadata** (blueprints) and **create all** of them inside the database connected to this **engine**."