Today is **very important** because this is where your project becomes a *real backend*, not just a running API.

---

# 🚀 DAY 2 — Database Design, Models & Core Data Layer

## 🎯 Day 2 Goal

By the end of today, you will have:

✔ A clean database schema
✔ SQLAlchemy models for all core entities
✔ Relationships between tables
✔ Database session dependency
✔ Tables created correctly in PostgreSQL
✔ Ready foundation for authentication & CRUD

---

# 🧠 1. Database Design (Think Like a Backend Engineer)

We will create **4 tables**:

### Tables

1. `users`
2. `categories`
3. `tasks`
4. `habits`

### Relationships

- A **user** has many tasks
- A **user** has many habits
- A **task** belongs to a user and a category
- A **habit** belongs to a user

---

# 📐 2. Database Schema (Conceptual)

### users

| column | type |
|---|---|
| id | integer (PK) |
| email | string (unique) |
| hashed_password | string |
| created_at | datetime |

---

### categories

| column | type |
|---|---|
| id | integer (PK) |
| name | string |
| user_id | integer (FK → users.id) |

---

### tasks

| column | type |
|---|---|
| id | integer (PK) |
| title | string |
| description | string |
| completed | boolean |
| due_date | datetime |
| user_id | integer (FK) |
| category_id | integer (FK) |

---

### habits

| column | type |
|---|---|

| | |
|---|---|
| id | integer (PK) |
| name | string |
| frequency | string (daily / weekly) |
| streak | integer |
| user_id | integer (FK) |

---

# 🧠 Concept → Reality Mapping

You already have:

## Conceptual tables

- users
- categories
- tasks
- habits

We will now **create them manually in pgAdmin 4** using SQL (the professional way).

---

# ✅ STEP 1 — Open pgAdmin 4 & Your Database

1. Open **pgAdmin 4**

Left panel:

```
Servers
```

```
└── PostgreSQL 16 (or your version)
    └── Databases
        └── myappdb (your database name)
```

2.
3. Click your database
4. Click **Query Tool** (top toolbar or right-click → Query Tool)

This is where you write SQL.

---

# ✅ STEP 2 — Create `users` Table

Paste this **exact SQL** into the Query Tool:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    hashed_password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Click ▸ **Execute**.

✔ You just created your first table.

---

## ✅ STEP 3 — Create `categories` Table

```
CREATE TABLE categories (

    id SERIAL PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    user_id INTEGER NOT NULL,

    CONSTRAINT fk_user

        FOREIGN KEY(user_id)

        REFERENCES users(id)

        ON DELETE CASCADE

);
```

✔ This links categories → users.

---

## ✅ STEP 4 — Create `tasks` Table

```
CREATE TABLE tasks (
```

```
id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

description TEXT,

completed BOOLEAN DEFAULT FALSE,

due_date TIMESTAMP,

user_id INTEGER NOT NULL,

category_id INTEGER,



CONSTRAINT fk_user

    FOREIGN KEY(user_id)

    REFERENCES users(id)

    ON DELETE CASCADE,



CONSTRAINT fk_category

    FOREIGN KEY(category_id)

    REFERENCES categories(id)

    ON DELETE SET NULL
```

```
);
```

✔ Task belongs to a user
✔ Task optionally belongs to a category

---

# ✅ STEP 5 — Create `habits` Table

```
CREATE TABLE habits (

    id SERIAL PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    frequency VARCHAR(50) NOT NULL,

    streak INTEGER DEFAULT 0,

    user_id INTEGER NOT NULL,



    CONSTRAINT fk_user

        FOREIGN KEY(user_id)

        REFERENCES users(id)

        ON DELETE CASCADE
```

```
);
```

✔ Habit belongs to a user

---

# ✅ STEP 6 — Verify Tables in pgAdmin

In left panel:

```
myappdb
  └── Schemas
      └── public
          └── Tables
```

1. 
2. Right-click **Tables** → **Refresh**

You should see:

- users
- categories
- tasks
- habits

🎉 **Your conceptual schema is now REAL in PostgreSQL**

---

# 🔍 STEP 7 — Visualize Relationships (Optional but Cool)

1. Right-click a table (e.g. `tasks`)
2. Click **Properties**
3. Go to **Constraints → Foreign Keys**

You'll see:

- `user_id → users(id)`
- `category_id → categories(id)`

This confirms your backend design is correct.

# 🧱 3. Backend Folder Structure (Updated)

Your backend should now look like this:

```
backend/
  .env
  venv/
  app/
    main.py
    database.py
    models/
      __init__.py
      user.py
      category.py
      task.py
      habit.py
```

⚠️ **IMPORTANT:**
Create `__init__.py` inside `models/` (empty file).

---

# 🧩 4. SQLAlchemy Models (FULL CODE)

---

## 4.1 User Model

### `app/models/user.py`

```
from sqlalchemy import Column, Integer, String, DateTime
from sqlalchemy.orm import relationship
from datetime import datetime
from app.database import Base

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    email = Column(String, unique=True, index=True, nullable=False)
    hashed_password = Column(String, nullable=False)
    created_at = Column(DateTime, default=datetime.utcnow)

    tasks = relationship("Task", back_populates="user")
    habits = relationship("Habit", back_populates="user")
    categories = relationship("Category", back_populates="user")
```

---

## 4.2 Category Model

### `app/models/category.py`

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from app.database import Base

class Category(Base):
    __tablename__ = "categories"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    user_id = Column(Integer, ForeignKey("users.id"))

    user = relationship("User", back_populates="categories")
    tasks = relationship("Task", back_populates="category")
```

---

## 4.3 Task Model

### app/models/task.py

```
from sqlalchemy import Column, Integer, String, Boolean, DateTime, ForeignKey
from sqlalchemy.orm import relationship
from datetime import datetime
from app.database import Base

class Task(Base):
    __tablename__ = "tasks"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, nullable=False)
    description = Column(String)
    completed = Column(Boolean, default=False)
    due_date = Column(DateTime)

    user_id = Column(Integer, ForeignKey("users.id"))
    category_id = Column(Integer, ForeignKey("categories.id"))

    user = relationship("User", back_populates="tasks")
    category = relationship("Category", back_populates="tasks")
```

---

## 4.4 Habit Model

### app/models/habit.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from app.database import Base

class Habit(Base):
    __tablename__ = "habits"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    frequency = Column(String, nullable=False)
    streak = Column(Integer, default=0)

    user_id = Column(Integer, ForeignKey("users.id"))

    user = relationship("User", back_populates="habits")
```

---

# 🔗 5. Update `main.py` to Register Models

### `app/main.py`

```
from fastapi import FastAPI
from sqlalchemy import inspect
from app.database import Base, engine

from app.models.user import User
from app.models.task import Task
from app.models.habit import Habit
from app.models.category import Category

app = FastAPI()

Base.metadata.create_all(bind=engine)

@app.get("/")
def root():
    return {"message": "FocusFlow API running 🚀"}

@app.get("/debug/tables")
def debug_tables():
    inspector = inspect(engine)
    return inspector.get_table_names()
```

---

# 🧪 6. Test Day 2 (VERY IMPORTANT)

### Run the server:

```
uvicorn app.main:app --reload
```

### Check PostgreSQL (pgAdmin):

- You should now see tables:
  - users
  - tasks
  - habits
  - categories

If tables appear → **Day 2 SUCCESS** ✅

---

# 📌 7. Git Commit for Day 2

git add .
git commit -m "Day 2: database schema and SQLAlchemy models"

---

# 🎉 DAY 2 COMPLETED — YOU NOW HAVE

✔ Professional database design
✔ Proper relationships
✔ Scalable SQLAlchemy models
✔ PostgreSQL fully integrated
✔ Strong backend foundation

## 👌 NEXT STEP (Day 2)

**Authentication (JWT, password hashing, secure routes)** — this is where recruiters really start paying attention.

---