요청하신 대로 **Pandas**를 사용하여 PSV 파일을 읽고, SQLite3와 순수 SQL을 활용하며, **MVC** 구조를 따르는 모듈화된 FastAPI CRUD 애플리케이션의 전체 코드를 다시 한번 제출합니다.

1. 📁 프로젝트 디렉토리 구조

프로젝트 루트 디렉토리 **fastapi_db_app/** 아래에 파일과 디렉토리를 생성해야 합니다.

```
fastapi_db_app/
├── data/
│   ├── database.py       # SQLite3 연결 및 초기화 로직
│   └── psv_loader.py     # Pandas를 이용한 PSV 파일 로더
├── models/
│   ├── __init__.py       # (빈 파일)
│   ├── creature.py       # Pydantic Creature 모델
│   └── explorer.py       # Pydantic Explorer 모델
├── services/
│   ├── __init__.py       # (빈 파일)
│   ├── creature.py       # Creature CRUD 비즈니스 로직 (SQL)
│   └── explorer.py       # Explorer CRUD 비즈니스 로직 (SQL)
├── routers/
│   ├── __init__.py       # (빈 파일)
│   ├── creature.py       # Creature API 엔드포인트 (Controller)
│   └── explorer.py       # Explorer API 엔드포인트 (Controller)
├── main.py               # FastAPI 애플리케이션 진입점
├── creatures.psv         # 📜 Creature 초기 데이터 파일
└── explorers.psv         # 📜 Explorer 초기 데이터 파일
```

---

2. 📜 PSV 데이터 파일

프로젝트 루트에 이 두 파일을 정확히 생성해야 합니다.

creatures.psv

코드 스니펫

```
name|country|area|description|aka
Abaia|FJ||Lake eel|
Afanc|UK|CYM|Welsh lake monster|
Agropelter|US|ME|Forest twig flinger|
Akkorokamui|JP||Giant Ainu octopus|
Albatwitch|US|PA|Apple stealing mini Bigfoot|
Alicanto|CL||Gold eating bird|
Altamata-ha|US|GA|Swamp creature|Altie
```

explorers.psv

코드 스니펫

```
name|country|description
Claude Hande|UK|Scarce during full moons
Helena Hande-Basquette|UK|Dame with a claim to fame
Beau Buffette|US|Never removes his pith helmet
```

---

3. 🐍 Python 코드

A. data/psv_loader.py (Pandas 로더)

Python

```python
# data/psv_loader.py

import pandas as pd
from typing import List, Dict, Any
import numpy as np

def load_data_from_psv(file_path: str) -> List[Dict[str, Any]]:
    """
    Pandas를 사용하여 PSV 파일을 읽고 각 행을 딕셔너리 리스트 형태로 반환합니다.
    """
    try:
        # '|'를 구분자로 사용하여 파일 읽기
        df = pd.read_csv(file_path, sep='|')

        # Pandas의 NaN 값 (비어있는 셀)을 Python의 None으로 변환하여 SQLite NULL에 대응
        data_list = df.replace({np.nan: None}).to_dict('records')

        return data_list

    except FileNotFoundError:
        print(f"Error: PSV file not found at {file_path}")
        return []
    except Exception as e:
        print(f"Error reading or parsing PSV file with Pandas: {e}")
        return []
```

B. data/database.py (SQLite 초기화)

Python

```python
# data/database.py

import sqlite3
import os
from typing import Generator
from .psv_loader import load_data_from_psv
```

```
DATABASE_NAME = "data/app_data.db"
CREATURES_FILE = "creatures.psv"
EXPLORERS_FILE = "explorers.psv"

def get_db_connection() -> Generator[sqlite3.Connection, None, None]:
    """FastAPI Dependency Injection을 위한 DB 연결 제공"""
    conn = sqlite3.connect(DATABASE_NAME)
    conn.row_factory = sqlite3.Row # 결과를 딕셔너리처럼 접근 가능하게 설정
    try:
        yield conn
    finally:
        conn.close()

def initialize_db() -> None:
    """Pandas로 PSV 파일을 읽고 데이터베이스 파일 생성 및 테이블, 데이터 삽입"""
    # 디렉토리 생성
    os.makedirs(os.path.dirname(DATABASE_NAME), exist_ok=True)

    conn = sqlite3.connect(DATABASE_NAME)
    cursor = conn.cursor()

    # 1. Creature 데이터 로딩 및 초기화
    creature_data = load_data_from_psv(CREATURES_FILE)
    if creature_data:
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS creature (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT NOT NULL UNIQUE,
                country TEXT,
                area TEXT,
                description TEXT,
                aka TEXT
            )
        """)

        for data in creature_data:
            try:
                cursor.execute("""
                    INSERT INTO creature (name, country, area, description, aka)
                    VALUES (?, ?, ?, ?, ?)
                    """,
                    (data.get('name'), data.get('country'), data.get('area'), data.get('description'),
data.get('aka'))
                )
```

```
            except sqlite3.IntegrityError:
                pass

    # 2. Explorer 데이터 로딩 및 초기화
    explorer_data = load_data_from_psv(EXPLORERS_FILE)
    if explorer_data:
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS explorer (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT NOT NULL UNIQUE,
                country TEXT,
                description TEXT
            )
        """)

        for data in explorer_data:
            try:
                cursor.execute("""
                    INSERT INTO explorer (name, country, description)
                    VALUES (?, ?, ?)
                    """,
                    (data.get('name'), data.get('country'), data.get('description'))
                )
            except sqlite3.IntegrityError:
                pass

    conn.commit()
    conn.close()
```

C. models/creature.py

Python

```
# models/creature.py

from pydantic import BaseModel, Field
from typing import Optional
```

```python
class CreatureBase(BaseModel):
    name: str = Field(..., example="Yeti")
    country: str = Field(..., example="NP")
    area: Optional[str] = Field(None, example="Himalayas")
    description: str = Field(..., example="Hairy bipedal creature")
    aka: Optional[str] = Field(None, example="Abominable Snowman")

class CreatureCreate(CreatureBase):
    pass

class CreatureUpdate(CreatureBase):
    pass

class Creature(CreatureBase):
    id: int

    class Config:
        from_attributes = True
```

D. models/explorer.py

Python

```python
# models/explorer.py

from pydantic import BaseModel, Field
from typing import Optional

class ExplorerBase(BaseModel):
    name: str = Field(..., example="Jane Goodall")
    country: str = Field(..., example="UK")
    description: str = Field(..., example="Primatologist")

class ExplorerCreate(ExplorerBase):
    pass
```

```python
class ExplorerUpdate(ExplorerBase):
    pass

class Explorer(ExplorerBase):
    id: int

    class Config:
        from_attributes = True
```

E. services/creature.py

Python

```python
# services/creature.py

import sqlite3
from typing import List, Optional
from models.creature import Creature, CreatureCreate, CreatureUpdate

class CreatureService:
    def get_all(self, conn: sqlite3.Connection) -> List[Creature]:
        """모든 Creature 조회 (R)"""
        cursor = conn.execute("SELECT * FROM creature")
        # sqlite3.Row 객체를 Pydantic 모델로 변환
        return [Creature.model_validate(row) for row in cursor.fetchall()]

    def get_by_id(self, conn: sqlite3.Connection, creature_id: int) -> Optional[Creature]:
        """ID로 Creature 조회 (R)"""
        cursor = conn.execute("SELECT * FROM creature WHERE id = ?", (creature_id,))
        row = cursor.fetchone()
        return Creature.model_validate(row) if row else None

    def create(self, conn: sqlite3.Connection, creature: CreatureCreate) -> Creature:
        """새 Creature 생성 (C)"""
        data = creature.model_dump()

        cursor = conn.execute(
```

```
            "INSERT INTO creature (name, country, area, description, aka) VALUES (?, ?, ?, ?, ?)",
            (data['name'], data['country'], data.get('area'), data['description'], data.get('aka'))
        )
        conn.commit()

        return self.get_by_id(conn, cursor.lastrowid)

    def update(self, conn: sqlite3.Connection, creature_id: int, creature_data: CreatureUpdate) ->
Optional[Creature]:
        """Creature 업데이트 (U)"""
        data = creature_data.model_dump()

        conn.execute(
            "UPDATE creature SET name=?, country=?, area=?, description=?, aka=? WHERE id=?",
            (data['name'], data['country'], data.get('area'), data['description'], data.get('aka'),
creature_id)
        )
        conn.commit()

        return self.get_by_id(conn, creature_id)

    def delete(self, conn: sqlite3.Connection, creature_id: int) -> bool:
        """Creature 삭제 (D)"""
        cursor = conn.execute("DELETE FROM creature WHERE id = ?", (creature_id,))
        conn.commit()
        return cursor.rowcount > 0
```

F. services/explorer.py

Python

```
# services/explorer.py

import sqlite3
from typing import List, Optional
from models.explorer import Explorer, ExplorerCreate, ExplorerUpdate
```

```python
class ExplorerService:
    def get_all(self, conn: sqlite3.Connection) -> List[Explorer]:
        """모든 Explorer 조회 (R)"""
        cursor = conn.execute("SELECT * FROM explorer")
        return [Explorer.model_validate(row) for row in cursor.fetchall()]

    def get_by_id(self, conn: sqlite3.Connection, explorer_id: int) -> Optional[Explorer]:
        """ID로 Explorer 조회 (R)"""
        cursor = conn.execute("SELECT * FROM explorer WHERE id = ?", (explorer_id,))
        row = cursor.fetchone()
        return Explorer.model_validate(row) if row else None

    def create(self, conn: sqlite3.Connection, explorer: ExplorerCreate) -> Explorer:
        """새 Explorer 생성 (C)"""
        data = explorer.model_dump()

        cursor = conn.execute(
            "INSERT INTO explorer (name, country, description) VALUES (?, ?, ?)",
            (data['name'], data['country'], data['description'])
        )
        conn.commit()

        return self.get_by_id(conn, cursor.lastrowid)

    def update(self, conn: sqlite3.Connection, explorer_id: int, explorer_data: ExplorerUpdate) ->
Optional[Explorer]:
        """Explorer 업데이트 (U)"""
        data = explorer_data.model_dump()

        conn.execute(
            "UPDATE explorer SET name=?, country=?, description=? WHERE id=?",
            (data['name'], data['country'], data['description'], explorer_id)
        )
        conn.commit()

        return self.get_by_id(conn, explorer_id)

    def delete(self, conn: sqlite3.Connection, explorer_id: int) -> bool:
        """Explorer 삭제 (D)"""
        cursor = conn.execute("DELETE FROM explorer WHERE id = ?", (explorer_id,))
        conn.commit()
        return cursor.rowcount > 0
```

G. routers/creature.py (Controller)

Python

```python
# routers/creature.py

import sqlite3
from fastapi import APIRouter, Depends, HTTPException, status
from typing import List
from models.creature import Creature, CreatureCreate, CreatureUpdate
from services.creature import CreatureService
from data.database import get_db_connection

router = APIRouter(prefix="/creatures", tags=["Creatures"])
service = CreatureService()

DBCONN = Depends(get_db_connection)

@router.get("/", response_model=List[Creature])
def list_creatures(conn: sqlite3.Connection = DBCONN):
    """모든 Creature 조회 (R)"""
    return service.get_all(conn)

@router.get("/{creature_id}", response_model=Creature)
def get_creature(creature_id: int, conn: sqlite3.Connection = DBCONN):
    """ID로 특정 Creature 조회 (R)"""
    creature = service.get_by_id(conn, creature_id)
    if not creature:
        raise HTTPException(status_code=404, detail="Creature not found")
    return creature

@router.post("/", response_model=Creature, status_code=status.HTTP_201_CREATED)
def create_creature(creature: CreatureCreate, conn: sqlite3.Connection = DBCONN):
    """새로운 Creature 생성 (C)"""
    try:
        return service.create(conn, creature)
```

```
    except sqlite3.IntegrityError:
        raise HTTPException(status_code=400, detail="Creature with this name already exists")


@router.put("/{creature_id}", response_model=Creature)
def update_creature(creature_id: int, creature_data: CreatureUpdate, conn: sqlite3.Connection = DBCONN):
    """특정 Creature 정보 업데이트 (U)"""
    updated_creature = service.update(conn, creature_id, creature_data)
    if not updated_creature:
        raise HTTPException(status_code=404, detail="Creature not found")
    return updated_creature

@router.delete("/{creature_id}", status_code=status.HTTP_204_NO_CONTENT)
def delete_creature(creature_id: int, conn: sqlite3.Connection = DBCONN):
    """특정 Creature 삭제 (D)"""
    if not service.delete(conn, creature_id):
        raise HTTPException(status_code=404, detail="Creature not found")
    return
```

H. routers/explorer.py (Controller)

Python

```
# routers/explorer.py

import sqlite3
from fastapi import APIRouter, Depends, HTTPException, status
from typing import List
from models.explorer import Explorer, ExplorerCreate, ExplorerUpdate
from services.explorer import ExplorerService
from data.database import get_db_connection

router = APIRouter(prefix="/explorers", tags=["Explorers"])
service = ExplorerService()
DBCONN = Depends(get_db_connection)
```

```python
@router.get("/", response_model=List[Explorer])
def list_explorers(conn: sqlite3.Connection = DBCONN):
    """모든 Explorer 조회 (R)"""
    return service.get_all(conn)

@router.get("/{explorer_id}", response_model=Explorer)
def get_explorer(explorer_id: int, conn: sqlite3.Connection = DBCONN):
    """ID로 특정 Explorer 조회 (R)"""
    explorer = service.get_by_id(conn, explorer_id)
    if not explorer:
        raise HTTPException(status_code=404, detail="Explorer not found")
    return explorer

@router.post("/", response_model=Explorer, status_code=status.HTTP_201_CREATED)
def create_explorer(explorer: ExplorerCreate, conn: sqlite3.Connection = DBCONN):
    """새로운 Explorer 생성 (C)"""
    try:
        return service.create(conn, explorer)
    except sqlite3.IntegrityError:
        raise HTTPException(status_code=400, detail="Explorer with this name already exists")

@router.put("/{explorer_id}", response_model=Explorer)
def update_explorer(explorer_id: int, explorer_data: ExplorerUpdate, conn: sqlite3.Connection =
DBCONN):
    """특정 Explorer 정보 업데이트 (U)"""
    updated_explorer = service.update(conn, explorer_id, explorer_data)
    if not updated_explorer:
        raise HTTPException(status_code=404, detail="Explorer not found")
    return updated_explorer

@router.delete("/{explorer_id}", status_code=status.HTTP_204_NO_CONTENT)
def delete_explorer(explorer_id: int, conn: sqlite3.Connection = DBCONN):
    """특정 Explorer 삭제 (D)"""
    if not service.delete(conn, explorer_id):
        raise HTTPException(status_code=404, detail="Explorer not found")
    return
```

I. main.py (FastAPI Entry Point)

Python

```python
# main.py

from fastapi import FastAPI
from routers import creature, explorer
from data.database import initialize_db

# 데이터베이스 초기화 (PSV 파일 로드 및 SQLite3 테이블 생성)
initialize_db()

app = FastAPI(
    title="Creature and Explorer Management API",
    version="1.0.0"
)

# 라우터 모듈 포함
app.include_router(creature.router)
app.include_router(explorer.router)

@app.get("/", tags=["Root"])
def read_root():
    return {"message": "API is ready. Check /docs for endpoints."}

# 실행 명령어: uvicorn main:app --reload
```

🚀 실행 안내

1. 패키지 설치:
   Bash
   pip install fastapi uvicorn pydantic pandas

2. **FastAPI** 실행:
   Bash
   uvicorn main:app --reload

3. 테스트: http://127.0.0.1:8000/docs에서 생성된 API를 확인하고 테스트할 수 있습니다.