

整理成CSV檔



F-A0010-001.json

檔案

我要整理成csv檔



已分析 ^

```
import json, pandas as pd

# load json
with open('/mnt/data/F-A0010-001.json', encoding='utf-8') as f:
    data = json.load(f)

locs = data["cwaopendata"]["resources"]["resource"]["data"]
["agrWeatherForecasts"]["weatherForecasts"]["location"]

rows=[]
for loc in locs:
    name=loc["locationName"]
    wx=loc["weatherElements"]["Wx"]["daily"]
    maxt=loc["weatherElements"]["MaxT"]["daily"]
    mint=loc["weatherElements"]["MinT"]["daily"]
    for i in range(len(wx)):
        rows.append({
            "location": name,
            "date": wx[i]["dataDate"],
            "weather": wx[i]["weather"],
            "weatherid": wx[i]["weatherid"],
            "maxT": maxt[i]["temperature"],
            "minT": mint[i]["temperature"],
        })
df = pd.DataFrame(rows)
csv_path='/mnt/data/weather_forecast.csv'
```



Powered by [ChatGPT Exporter](#)

```
df.to_csv(csv_path, index=False, encoding='utf-8-sig')  
csv_path
```

結果

```
'/mnt/data/weather_forecast.csv'
```

已幫你把一週農業氣象預報整理成 **CSV** 檔 (地區、日期、天氣、最高溫、最低溫)。

👉 **下載你的 CSV :**

👉 [weather_forecast.csv](#)

資料來源 :  F-A0010-001

如果你還想把 **degreeDay / accumulatedTemperature**、或 **cropStatistics** (水稻積溫) 也整理成 CSV，我也可以幫你一起做！ </>



我要可預看的資料顯現



已分析 ^

```
import pandas as pd  
  
df = pd.read_csv('/mnt/data/weather_forecast.csv')  
df.head(10)
```

結果

	location	date	weather	weatherid	maxT	minT	
0	北部地區	2025-12-04	多雲時晴		3	21	15
1	北部地區	2025-12-05	多雲短暫雨		8	23	15
2	北部地區	2025-12-06	多雲時晴		3	24	15
3	北部地區	2025-12-07	晴時多雲		2	26	15
4	北部地區	2025-12-08	多雲短暫雨		8	23	16
5	北部地區	2025-12-09	多雲短暫雨		8	23	17
6	北部地區	2025-12-10	晴時多雲		2	26	16
7	中部地區	2025-12-04	晴時多雲		2	25	15



Powered by [ChatGPT Exporter](#)

8	中部地區	2025-12-05	晴時多雲	2	26	16
9	中部地區	2025-12-06	晴時多雲	2	26	16

這是你的 **CSV 檔前 10 筆資料預覽** (地區、日期、天氣、最高溫、最低溫) , 你可以確認格式與內容是否正確。

如果你還想 **顯示全部資料、篩選某地區、或做成表格視覺化**，我可以繼續幫你做喔！ </>



write python
1. web crawler to get this information
2. save data to sqlite3 database , data.db (you can help me to design database)
3. streamlit to show this information



已思考 34s >

好，我直接幫你把三個部分都寫好：

1 爬資料 → **2** 存進 SQLite → **3** 用 Streamlit 顯示。

下面你可以直接複製成三個檔案來用。

1. crawler.py (爬 CWA JSON + 存進 SQLite)

```
# crawler.py
import json
import requests
import sqlite3
from pathlib import Path

# ===== 你自己的設定 =====
# 若你用中央氣象署 OpenData , 通常長這樣：
# https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-A0010-001?Authorization=
你的API_KEY
API_URL = "https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-A0010-001"
API_KEY = "PUT_YOUR_API_KEY_HERE"    # TODO: 換成你自己的 key
```



```

DB_PATH = "data.db"

# ===== 建立 / 設計資料庫 =====
def init_db(db_path: str = DB_PATH):
    conn = sqlite3.connect(db_path)
    cur = conn.cursor()
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS weather_forecast (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            location TEXT NOT NULL,
            date TEXT NOT NULL,
            weather TEXT,
            weatherid INTEGER,
            maxT REAL,
            minT REAL,
            UNIQUE(location, date)
        )
        """
    )
    conn.commit()
    conn.close()

# ===== 解析 CWA JSON (依照你上傳的 F-A0010-001.json 結構) =====
def parse_cwa_json(data: dict):
    """
    回傳一個 list[tuple]
    (location, date, weather, weatherid, maxT, minT)
    """

    resource = data["cwaopendata"]["resources"]["resource"]
    locations = resource["data"]["agrWeatherForecasts"]["weatherForecasts"]
    ["location"]

    rows = []
    for loc in locations:
        loc_name = loc["locationName"]
        wx_daily = loc["weatherElements"]["Wx"]["daily"]
        max_daily = loc["weatherElements"]["MaxT"]["daily"]
        min_daily = loc["weatherElements"]["MinT"]["daily"]

```



```

for i in range(len(wx_daily)):
    rows.append(
        (
            loc_name,
            wx_daily[i]["dataDate"],
            wx_daily[i]["weather"],
            int(wx_daily[i]["weatherid"]),
            float(max_daily[i]["temperature"]),
            float(min_daily[i]["temperature"]),
        )
    )
return rows

# ===== 將資料寫進 SQLite =====
def save_to_db(rows, db_path: str = DB_PATH):
    conn = sqlite3.connect(db_path)
    cur = conn.cursor()
    cur.executemany(
        """
        INSERT OR REPLACE INTO weather_forecast
        (location, date, weather, weatherid, maxT, minT)
        VALUES (?, ?, ?, ?, ?, ?, ?)
        """,
        rows,
    )
    conn.commit()
    conn.close()

# ===== 從網路抓資料 =====
def fetch_from_api():
    params = {"Authorization": API_KEY}
    resp = requests.get(API_URL, params=params, timeout=10)
    resp.raise_for_status()
    return resp.json()

# ===== 也可以從本地 JSON 測試 (你上傳的 F-A0010-001.json) =====
def fetch_from_local(file_path: str):
    with open(file_path, "r", encoding="utf-8") as f:

```



```

    return json.load(f)

if __name__ == "__main__":
    # 1. 建資料庫 & table
    init_db()

    # 2a. ---- 從官方 API 抓資料 ----
    # data = fetch_from_api()

    # 2b. ---- 或先用本地檔案測試 ----
    data = fetch_from_local("F-A0010-001.json")

    # 3. 解析 JSON
    rows = parse_cwa_json(data)

    # 4. 存進 SQLite
    save_to_db(rows)

    print(f"已寫入 {len(rows)} 筆資料到 {DB_PATH}")

```

測試流程：

1. 把 `F-A0010-001.json` 放在同一個資料夾
2. 在終端機執行：`python crawler.py`
3. 會產生 `data.db`，裡面有 `weather_forecast table`。

2. app.py (Streamlit 顯示 SQLite 內容)

```

# app.py
import sqlite3
import pandas as pd
import streamlit as st

DB_PATH = "data.db"

def get_conn():
    return sqlite3.connect(DB_PATH)

```



```

def load_locations(conn):
    return pd.read_sql_query(
        "SELECT DISTINCT location FROM weather_forecast ORDER BY location",
        conn
    )["location"].tolist()

def load_date_range(conn):
    df = pd.read_sql_query(
        "SELECT MIN(date) AS min_date, MAX(date) AS max_date FROM
weather_forecast",
        conn,
    )
    return df.loc[0, "min_date"], df.loc[0, "max_date"]

def load_data(conn, location=None, start_date=None, end_date=None):
    query = "SELECT location, date, weather, weatherid, maxT, minT FROM
weather_forecast WHERE 1=1"
    params = []

    if location and location != "全部地區":
        query += " AND location = ?"
        params.append(location)

    if start_date:
        query += " AND date >= ?"
        params.append(start_date)

    if end_date:
        query += " AND date <= ?"
        params.append(end_date)

    query += " ORDER BY location, date"
    return pd.read_sql_query(query, conn, params=params)

def main():
    st.title("一週農業氣象預報資料瀏覽")

    conn = get_conn()

```

```

# 側邊欄篩選
with st.sidebar:
    st.header("篩選條件")

locations = ["全部地區"] + load_locations(conn)
selected_loc = st.selectbox("選擇地區", locations)

min_date, max_date = load_date_range(conn)
date_range = st.date_input(
    "選擇日期區間",
    value=(pd.to_datetime(min_date), pd.to_datetime(max_date)),
)
if isinstance(date_range, tuple):
    start_date, end_date = date_range
else:
    start_date = end_date = date_range

start_date = start_date.strftime("%Y-%m-%d")
end_date = end_date.strftime("%Y-%m-%d")

df = load_data(conn, selected_loc, start_date, end_date)

st.subheader("資料表格")
st.dataframe(df, use_container_width=True)

st.subheader("氣溫折線圖 (最高 / 最低) ")
if not df.empty:
    # 只畫一個地區比較清楚，若全部地區就先選其中一個
    loc_for_chart = (
        selected_loc if selected_loc != "全部地區" else
df["location"].iloc[0]
    )
    chart_df = df[df["location"] == loc_for_chart].copy()
    chart_df["date"] = pd.to_datetime(chart_df["date"])
    chart_df = chart_df.set_index("date")[["maxT", "minT"]]
    st.line_chart(chart_df)
    st.caption(f"顯示地區 : {loc_for_chart}")
else:
    st.info("目前條件下沒有資料。")

```



```
conn.close()

if __name__ == "__main__":
    main()
```

執行 Streamlit

在終端機 / CMD 裡：

```
pip install streamlit requests pandas
streamlit run app.py
```

瀏覽器會自動打開一個頁面，你就可以：

- 左邊選 地區 / 日期
- 中間看到 表格預覽 (可捲動)
- 下面有 最高 / 最低溫 折線圖

如果你要再加上第二張表（例如 `degreeDay` / `accumulatedTemperature`），我也可以幫你：

- 擴充 SQLite schema
- 更新 crawler.py 一起寫進 DB
- 在 Streamlit 多加一個 tab 顯示。

