Managing Data Using CSV and SQL

Michael Borck

Table of contents

Today

- Learn how to manage weather data using CSV files
- Understand how to use SQL to store and retrieve weather data
- Practice reading, writing, and processing data with pandas and SQLite

OpenWeatherMap API (Homework)

- Navigate OpenWeatherMap API endpoints.

- Customise requests with parameters.
- Understand API response formats.
- Handle common API errors.

Project Overview

- Web app displays weather trends.
- Uses OpenWeatherMap API data.
- Shows current and forecast weather.
- Location selection and unit switch.
- Handles API errors gracefully.

Setup Project

- Create GitHub repository.
- Clone repository locally.
- Structure project folders.
- Create Python environment.
- Install dependencies via requirements.txt.

Breakout Room Activity

- Setup Project Folder and Environment

Managing Data with CSV Files

- Reading data from a CSV file
- Processing data by converting temperature from Kelvin to Celsius
- Writing data to a new CSV file

Read data

```
import pandas as pd

df = pd.read_csv('data/raw/weather_data.csv')
df
```

Processing data

```
def convert_temp_kelvin_to_celsius(temp_k):
    return temp_k - 273.15

df['Temperature (C)'] = df['Temperature (K)'].apply(convert_temp_kelvin_to_celsius)
df.drop(columns=['Temperature (K)'], inplace=True)
df
```

Saving data

```
df.to_csv('data/processed/processed_weather_data.csv', index=False)
```

Managing Data with SQL

- Creating a SQLite database and table
- Inserting data into the table
- Querying data from the table

Create database

```
import sqlite3

conn = sqlite3.connect('data/weather_data.db')
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS weather (
```

```
        location TEXT,
        temperature_c REAL,
        humidity INTEGER,
        weather_description TEXT
    )
''')
conn.commit()
```

inserting into table

```
for index, row in df.iterrows():
    cursor.execute('''
        INSERT INTO weather (location, temperature_c, humidity, weather_description)
        VALUES (?, ?, ?, ?)
    ''', (row['Location'], row['Temperature (C)'], row['Humidity (%)'], row['Weather']))
conn.commit()
```

Querying form table

```
cursor.execute('SELECT * FROM weather')
rows = cursor.fetchall()
for row in rows:
    print(row)

df_sql = pd.DataFrame(rows, columns=['Location', 'Temperature (C)', 'Humidity (%)', 'Weather
df_sql
```

Combining CSV and SQL Data Management

- Reading additional weather data from a CSV file
- Inserting data into the SQLite table
- Querying combined data from the SQLite table

Code snippet:

```
additional_data = pd.read_csv('data/raw/additional_weather_data.csv')
for index, row in additional_data.iterrows():
    cursor.execute('''
        INSERT INTO weather (location, temperature_c, humidity, weather_description)
        VALUES (?, ?, ?, ?)
    ''', (row['Location'], row['Temperature (C)'], row['Humidity (%)'], row['Weather']))
conn.commit()
```

Verify

```
cursor.execute('SELECT * FROM weather')
combined_rows = cursor.fetchall()
df_combined = pd.DataFrame(combined_rows, columns=['Location', 'Temperature (C)', 'Humidity
df_combined
```

Homework

- Complete Understanding the Weather Dashboard Project: WeatherVista worksheet
- Write `fetch_data.py` and `test_fetch_data.py`
- Practice reading and writing data to CSV files with different weather datasets
- Explore additional SQL queries to retrieve specific subsets of the weather data

Summary

- manage weather data using CSV files and SQL.
- practice reading, writing, and processing data with pandas and SQLite
- combined both techniques to manage our weather data effectively.

Next Session

- we will focus on basic data visualisation with Matplotlib.