Weather Data Retrieval and Analysis Project1

1. Project Overview

This project is a Python-based application that retrieves daily temperature data for a specific city using the OpenWeatherMap API. The script stores the data in a CSV file, processes it using Pandas, and calculates the daily average, minimum, and maximum temperatures. Finally, it generates an HTML report summarizing the findings.

The main objectives were:

- Automating data collection from a public API
- Storing and structuring weather data for analysis
- Performing statistical calculations on the data
- Generating a user-friendly summary report

2. Tools & Technologies

- Python – Core programming language
- Requests – To send API requests and fetch weather data
- Pandas – For data manipulation and aggregation
- Datetime – To handle and format timestamps
- CSV – For data storage
- HTML – For report generation

3. Implementation Workflow

1. Fetch data – The script connects to the OpenWeatherMap API using the city name and API key.
2. Extract & structure data – Temperature readings and timestamps are extracted and stored in a Pandas DataFrame.

3. Store in CSV – The DataFrame is exported to weather_data.csv for record keeping.
4. Data analysis – Daily average, minimum, and maximum temperatures are calculated.
5. Report generation – Results are saved in weather_report.html for easy readability.

4. Challenges Faced & Resolutions

Challenge 1: API Authentication & Key Handling

- Problem: The API call failed initially due to using it without getting validated or expired API keys.
- Resolution: Generated a valid API key from OpenWeatherMap and securely passed it to the request URL.

Challenge 2: Aggregating Data by Date

- Problem: The raw data contained multiple temperature records per day (every 3 hours).
- Resolution: Grouped the dataset by date (df.groupby("date")) and used Pandas aggregation functions (mean, min, max) to compute daily statistics.

Challenge 3: Converting Timestamps

- Problem: The timestamps returned by the API were in Unix format, not human-readable dates.
- Resolution: Used datetime.fromtimestamp(item["dt"]) to convert timestamps into a readable date format (YYYY-MM-DD).

5. Final Output

- CSV File: weather_data.csv containing raw weather data
- HTML Report: weather_report.html with daily temperature statistics

6. Conclusion

This project successfully demonstrates the process of integrating external APIs with Python for real-time data collection, cleaning, analysis, and reporting.
It also improved skills in data handling, error management, and report automation.