Project Report

Research Volunteer : **Ishan Palit** (palit.i@northeastern.edu)

Supervisor : **Prof. Srinivasan Radhakrishnan** (s.radhakrishnan@northeastern.edu)

Department : **Northeastern University, College of Engineering - Mechanical and Industrial Engineering Department**

Weather Data Collection, Storage, and Analysis

Project Task and Timeline Distribution

The Project Task Breakdown and Timeline is given below :

Project Task	**Week**	**Hours Per Week**
Project Planning and Scope	8th Jan 2024 - 13th Jan 2024	21
Data Model Design, Data Flow, Solution Architecture	15th Jan 2024 - 19th Jan 2024	21
Data Pipeline Implementation	22nd Jan 2024 - 26th Jan 2024	21
Data Pipeline Implementation	29th Jan 2024 - 2nd Feb 2024	21
Project Report and Documentation	5th Feb 2024 - 8th Feb 2024	21

Codebase

Github Link : https://github.com/palit-ishan/Weather-Data-Analysis

Introduction

The objective of this project was to design and implement a robust data pipeline for collecting, storing, and analyzing weather data for top cities in the United States. By leveraging the OpenWeatherMap API and SQLite database, the project aimed to provide valuable insights into weather patterns and trends over time.

OpenWeatherMap API

OpenWeatherMap API is a powerful and widely-used platform that provides access to a vast array of weather data and forecasts for locations around the world. It offers developers easy-to-use endpoints to retrieve current weather conditions, forecasts, historical weather data, and more.

Key Features

- **Current Weather Data:** Retrieve real-time weather conditions such as temperature, humidity, wind speed, and weather descriptions for any location globally.

- **Weather Forecasts:** Access 5-day and 16-day weather forecasts, including hourly forecasts, to plan activities and events in advance.

- **Historical Weather Data:** Obtain historical weather data for specific dates and locations, enabling trend analysis and historical comparisons.

- **Weather Maps:** Access various weather map layers, including temperature, precipitation, wind speed, and cloud cover, to visualize weather patterns and trends.

- **Weather Alerts:** Receive weather alerts and warnings for severe weather events such as thunderstorms, hurricanes, and blizzards, helping users stay informed and safe.

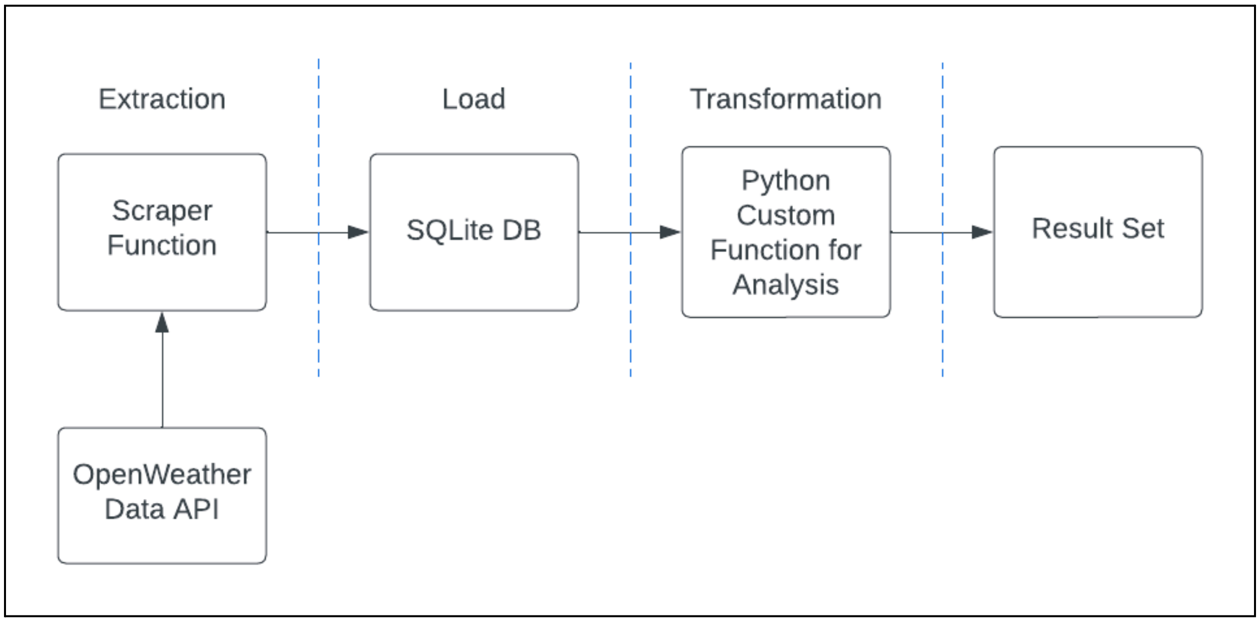
- **API Integration:** The API is easy to integrate into applications and websites, with comprehensive documentation, sample code, and SDKs available for popular programming languages.

- **Free and Paid Plans:** OpenWeatherMap offers both free and paid subscription plans, with the free plan providing limited access to certain features and data. Paid plans offer higher usage limits, additional features, and premium support.

Project Overview

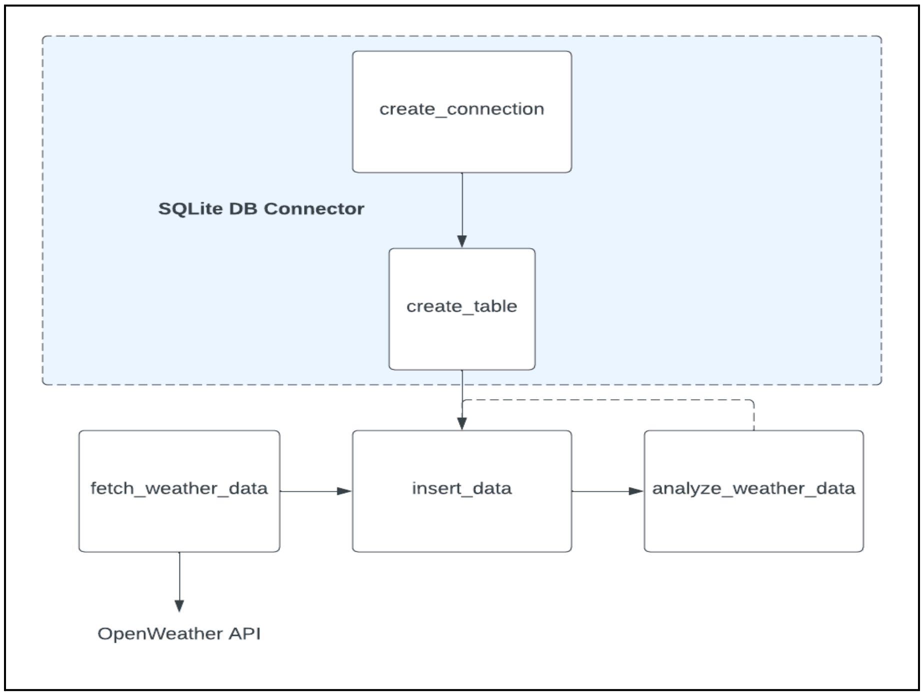
- **Data Collection:** The project involved the retrieval of current weather data from the OpenWeatherMap API for a predefined list of top cities in the US.
- **Data Storage:** The collected weather data was stored in an SQLite database, providing a centralized repository for easy access and retrieval.
- **Data Analysis:** Basic analysis techniques were employed to calculate key metrics such as average temperature and humidity for each city, enabling the identification of weather trends and patterns.
- **Scheduler:** The pipeline is scheduled to run every 6 hours to continually scrape weather data, store it and perform analysis.

Data Flow



The data flow consists of 3 segments before the result set - they are the Extraction layer, the Load layer and the Transformation layer.

Solution Architecture



Components

- **API Client (fetch_weather_data):** This function fetches weather data for multiple cities using the OpenWeatherMap API. It takes an API key and a list of cities as input and returns a list of weather data for each city.

- **Database Storage (create_connection, create_table, insert_data):** These functions handle database operations using SQLite. They create a database connection, create a table to store weather data if it doesn't exist, and insert weather data into the database.

- **Analysis (analyze_weather_data):** This function performs analysis on the collected weather data. It calculates the average temperature and humidity for each city and prints the results.

- **Main Script (__main__):** This part of the script initializes variables, establishes a connection to the database, and orchestrates the data collection, storage, and analysis processes. It continuously fetches weather data at regular intervals using the API client, stores it in the database, and analyzes the collected data.

Implementation

- **API Integration:** Python scripts were developed to interface with the OpenWeatherMap API, enabling the retrieval of current weather data for multiple cities simultaneously.
- **Database Management:** SQLite was chosen as the database management system for its lightweight nature and ease of integration with Python scripts. A database schema was designed to store weather data efficiently.
- **Scripting:** Python scripts were utilized to handle data retrieval, storage, and analysis tasks. Error handling mechanisms were implemented to ensure robustness and reliability.
- **Scheduled Execution:** A cron job was configured to schedule the execution of the data collection script at regular intervals (every 6 hours), ensuring the continuous updating of weather data.

Results and Findings

- **Data Collection:** The project successfully retrieved current weather data for the specified list of top cities in the US from the OpenWeatherMap API. API rate limits were carefully managed to avoid exceeding usage thresholds.
- **Data Storage:** The collected weather data was stored in the SQLite database, organized by city and timestamp. This facilitated easy access and retrieval of historical weather data for analysis purposes.
- **Data Analysis:** Basic analysis techniques, such as calculating average temperature and humidity for each city, were applied to the collected data. The analysis revealed insights

into seasonal variations, temperature trends, and humidity patterns across different cities.

Challenges

- **API Rate Limits:** Ensuring compliance with API rate limits posed a challenge, requiring careful management of API requests to avoid exceeding usage quotas.
- **Data Quality:** Handling inconsistencies or missing data in the collected weather data presented challenges during analysis. Robust error handling mechanisms were implemented to address data quality issues.

Future Enhancements

- **Advanced Analysis Techniques:** Implement more sophisticated analysis techniques, such as trend analysis, correlation analysis, and predictive modeling, to uncover deeper insights into weather patterns and trends.
- **Visualization:** Develop interactive visualizations using libraries like Matplotlib or Plotly to present weather data trends and patterns more effectively.
- **Alerting System:** Implement an alerting system to notify stakeholders of significant weather events or anomalies, enabling proactive decision-making.

Conclusion

In conclusion, the project successfully demonstrated the implementation of a comprehensive data pipeline for weather data collection, storage, and analysis. By leveraging the OpenWeatherMap API and SQLite database, valuable insights into weather patterns and trends for top US cities were obtained. Future enhancements could further enhance the utility and effectiveness of the system, enabling more informed decision-making in various domains.

Acknowledgments

I would like to thank Prof. Sri Radhakrishnan for valuable guidance and support during the entire research project duration. It would not be possible to have this project without his support and encouragement.

Special thanks to OpenWeatherMap for providing access to their weather data API, which was instrumental in the completion of this project. Additionally, thanks to the Python and SQLite communities for their invaluable tools, resources, and support throughout the project lifecycle.