**📊 ETL Pipeline Project Report**

**Project Title: Weather Data ETL Pipeline**

**Author**: Shaikh Muhammad Muddassir (DS-032-2024)  
**Course**: Big Data Analytics  
**Date**: April 2025

**🔧 Objective**

The objective of this project is to build a fully automated **ETL (Extract, Transform, Load)** pipeline to process weather-related datasets collected from multiple sources. The goal is to clean, standardize, enrich, and store the final data for further analytics and visualization.

**🛠️ Tools & Technologies Used**

| **Component** | **Tool/Tech** |
| --- | --- |
| Programming | Python 3.12.8 |
| Data Handling | pandas, json, requests |
| APIs | OpenWeatherMap API |
| Cloud Storage | Google Sheets |
| Database | MongoDB Atlas (NoSQL, cloud-hosted) |
| Integration | gspread, pymongo, oauth2client |

**📥 Data Sources**

| **Source** | **Format** | **Description** |
| --- | --- | --- |
| sample\_data.csv | CSV | Static historical weather records |
| sample\_weather.json | JSON | Local file with weather details |
| Google Sheet | Spreadsheet | Crowd-sourced weather entries (real-time) |
| OpenWeather API | REST (JSON) | Live weather data for London |
| MongoDB Atlas | NoSQL | Previously stored records (for re-processing) |

**🔄 ETL Pipeline Flow**

**1️⃣ Extraction (extract\_\*() functions)**

* Read CSV from data/sample\_data.csv
* Normalize JSON using json.load() + pandas.json\_normalize()
* Connect to **Google Sheets** using OAuth2 to pull structured tabular data
* Fetch real-time data from **OpenWeather API** (London)
* Pull previously inserted records from **MongoDB Atlas**

**2️⃣ Transformation (transform\_data(df) function)**

Performed the following operations:

**✅ Data Cleaning**

* Handled missing values using forward-fill
* Removed exact duplicate rows
* Flattened nested columns (e.g., list or dict) into strings

**✅ Unit Conversion**

* Converted temperature from Fahrenheit to Celsius using:  
  ‘tempc=(tempf−32)∗5/9‘`temp\_c = (temp\_f - 32) \* 5/9`

**✅ Timestamp Standardization**

* Converted all timestamp fields to ISO 8601 format in UTC:
* 2025-04-05T08:00:00Z

**✅ Feature Engineering**

* Introduced a new column: weather\_score  
  Based on average of:
  + Temperature (in Celsius)
  + Humidity
  + Wind Speed
* weather\_score = (temp\_c + humidity + wind\_speed) / 3

**✅ Data Formatting**

* Dropped MongoDB \_id fields before re-insertion
* Rounded weather\_score to two decimal places for readability

**3️⃣ Load (load\_to\_mongo(df) function)**

* Configured a cloud-based **MongoDB Atlas cluster** (task)
* Created database: weathere
* Created collection: etl\_weather
* Inserted cleaned data into MongoDB
* Verified insertion via Atlas UI

**📤 Export & Output**

* Final processed data exported to:
* output/final\_cleaned\_data.csv
* Fields include:
  + id, name, timestamp, temp\_c, humidity, wind\_speed, weather\_score

**📊 Sample Output (Transformed Data)**

| **id** | **name** | **timestamp** | **temp\_c** | **humidity** | **wind\_speed** | **weather\_score** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Station A | 2025-04-05T08:00:00Z | 23.88 | 65 | NaN | NaN |
| 2 | Station B | 2025-04-05T08:10:00Z | 26.67 | 70 | NaN | NaN |
| 2643743 | London | 2025-04-05T08:00:00Z | 27.78 | 71 | 4.63 | 34.47 |

**🔐 Security & Configuration**

* MongoDB credentials managed via config/db\_config.json
* Google Sheets access secured using OAuth2 (client\_secret.json)
* MongoDB IP Whitelisting configured via Network Access settings

**🚀 Possible Extensions**

* Add error logging and exception handling
* Create a **real-time dashboard** using Streamlit or Dash
* Schedule ETL job with cron or Airflow
* Integrate SMS/email alerts for extreme weather (weather\_score > X)
* Predict future weather score using ML models (regression)

**✅ Conclusion**

This project demonstrates the successful implementation of a robust and automated ETL pipeline using Python, integrating multiple heterogeneous data sources (files, APIs, cloud DB, and sheets), transforming and enriching data, and storing it in a scalable NoSQL database.

All steps — **extract, transform, load, and export** — are automated and modular, allowing for easy future scaling and adaptation.