**Name : Rabat Shahid**

**Roll Number : DS-W-14**

**Subject :  BDA**

**Submitted to : Sir Umer**

# Mid Term Report:

# ETL Pipeline for Weather Analytics

## 1. Introduction

This report outlines the implementation of an ETL (Extract, Transform, Load) pipeline designed for **weather analytics**. The system collects weather data from diverse sources, transforms it to maintain consistency and completeness, and loads the refined dataset into a **MongoDB Atlas NoSQL database** for trend analysis and future prediction tasks.

---

## 2. Pipeline Design

The pipeline consists of the following components:

1. **Extraction** from 5 different data sources
2. **Transformation** for unifying formats and data cleaning
3. **Loading** into MongoDB
4. **CI/CD automation** to validate, test, and deploy the ETL

## 3. Data Extraction (Sources of Data)

The ETL pipeline retrieves weather data from the following **five sources**:

### 1. Real-time Weather Data (WeatherAPI)

- **Source:** WeatherAPI
- **Format:** JSON
- **Method:** API request via Python's `requests`
- **Extracted:** Temperature, weather condition, location

### 2. Historical Weather Data (Open-Meteo API)

- **Source:** Open-Meteo
- **Format:** JSON
- **Method:** API request
- **Extracted:** Daily max temperatures

### 3. NOAA Climate Data

- **Source:** NOAA Climate (CSV file)
- **Format:** CSV
- **Method:** Direct download from NOAA
- **Extracted:** Hourly dry bulb temperatures

### 4. Google Drive CSV File

- **Source:** Google Drive (Simulated)
- **Format:** CSV
- **Method:** Download using `gdown`
- **Extracted:** Historic temperature records

### 5. MongoDB Atlas (Old Records)

- **Source:** MongoDB Atlas
- **Format:** JSON / Pandas DataFrame
- **Method:** `pymongo` connection
- **Extracted:** Previously stored records

---

## 4. Data Transformation

Post extraction, the pipeline performs multiple transformation steps:

### 1. Handling Missing Data

- Uses `ffill` and `bfill` to fill missing values

### 2. Unit Conversion

- Temperatures recorded in **Celsius** are converted to **Fahrenheit** using the formula:

$$\textbf{Temperature (°F)}=(\textbf{Temperature (°C)}\times 59)+32$$

### 3. Standardizing Date Formats

- Uses `pandas.to_datetime()` for uniformity

### 4. Removing Duplicates

- Drops duplicates using `drop_duplicates()`

### 5. Aggregation by Date

- Ensures one consolidated record per day

### 6. Weather Condition Mapping

- Normalizes conditions into categories: `Clear`, `Cloudy`, `Precipitation`, `Severe`

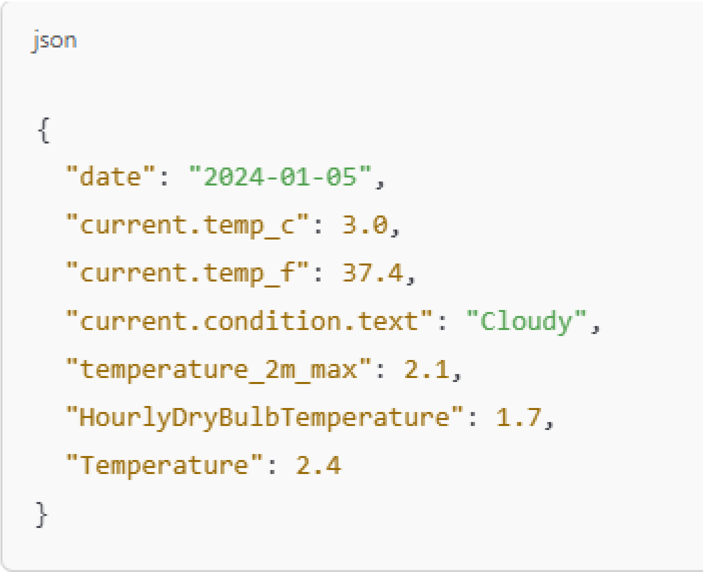
---

# 5. Data Loading (MongoDB Atlas)

Transformed data is loaded to **MongoDB Atlas** as follows:

1. Clear old data to avoid redundancy
2. Insert new cleaned records
3. Final dataset is printed and stored

OutPut:



```
{
  "date": "2024-01-05",
  "current.temp_c": 3.0,
  "current.temp_f": 37.4,
  "current.condition.text": "Cloudy",
  "temperature_2m_max": 2.1,
  "HourlyDryBulbTemperature": 1.7,
  "Temperature": 2.4
}
```

# 6. Justification for Technology Choices

| Component | Tool/Technology | Reason |
|---|---|---|
| **Programming** | Python | Versatile, has strong ETL libraries (pandas, requests, pymongo) |
| **Scheduling** | GitHub Actions / `schedule` lib | Automates daily ETL runs or on-commit execution |
| **Database** | MongoDB Atlas | Scalable NoSQL for semi-structured weather data |
| **Version Control** | GitHub | Code collaboration and integration with CI/CD |
| **CI/CD** | GitHub Actions | Automates testing, validation, deployment |

# 7. CI/CD Integration

### CI/CD Objectives:

- Automate testing and deployment of ETL script
- Ensure reliability with schema validation and unit testing

### GitHub Actions Pipeline Steps:

1. On push/PR to `main` branch:
    - Install dependencies
    - Run `pytest` unit tests
    - Validate schema (via `validate_schema.py`)
    - Run/trigger ETL (`weather_etl.py`)

**`.github/workflows/etl_ci_cd.yml`**

```
yaml
CopyEdit
name: ETL Pipeline CI/CD

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Code
      uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.10'

    - name: Install Dependencies
      run: |
        pip install -r requirements.txt

    - name: Run Unit Tests
      run: |
        pytest tests/

    - name: Validate Schema
      run: |
        python validate_schema.py

    - name: Simulate Deployment
      run: |
        echo "Running ETL..."
        python weather_etl.py
```

# 8. CI/CD Benefits & Reliability

| CI/CD Feature | Benefit |
|---|---|
| **Reduces Manual Errors** | Every push is tested and validated automatically |
| **Facilitates Fast Feedback** | Detects issues instantly on commit |
| **Improves Data Integrity** | Schema validations prevent dirty/incomplete data |
| **Accelerates Deployment** | Fully automated ETL runs and deployment to DB |

## 9. Conclusion

This weather ETL pipeline ensures clean, consistent, and integrated weather data using automation and modern tools. The use of CI/CD makes the system reliable and production-ready, reducing errors and accelerating updates. Final results are successfully loaded into **MongoDB Atlas**, ready for further trend analysis.