**Spark Cognition – Renewables**

This code implements a **data pipeline** to process, analyse, and visualize renewable energy data using **Apache Spark**. It consists of several modular components (Extraction, Transformation, Loading, and Visualization) to streamline ETL (Extract, Transform, Load) processes. The pipeline logs all operations, handles errors gracefully, and ensures modularity for scalability.

**Python Libraries**

- Apache Spark

- Matplotlib

- Logging

**Installation**

1. Install Apache Spark

2. Use Python 3.10 or later.

3. Install required Python libraries: **pip install pyspark matplotlib**

**Scripts**

* extract.py
* transform.py
* load.py
* visualization.py
* logger.py
* main.py

**1. Main Script (main.py)**

Acts as the central controller for the ETL and visualization pipeline.

* Sets up a local Spark cluster for distributed data processing.
* Input path: Where CSV datasets are located.
* Output paths:
* Aggregated data (with and without nulls).
* Visualization plots.

**ETL Workflow**:

* **Extract**: Reads data from CSV files into a Spark DataFrame.
* **Transform**:
* Handles data both with and without null values.
* Applies windowed aggregations (average, min, max, standard deviation, etc.) over a specified time duration (10 minutes).
* **Load**: Saves the processed data into partitioned CSV files by device.
* **Visualize**: Generates line plots for each device and saves them to the output directory.

**Logging**: Logs progress and errors for all stages of the pipeline.

**2. Extraction Module (extract.py)**

Extracts and loads raw data into a Spark DataFrame.

* **Schema**:
  + timestamp: Time of data collection.
  + variable: Type of measurement.
  + value: Observed value.
  + device: Device identifier.

**3. Transformation Module (transform.py)**

Prepares and aggregates the data for further analysis.

**Functions:**

1. **without\_nulls**: Removes duplicate rows and drops records with missing values.
2. **with\_nulls**: Removes duplicate rows but retains records with missing values.
3. **aggregations**:
   * Groups data by device and time-based windows (e.g., 10 minutes).
   * Also, groups data based on the window given in main.py
   * Calculates average, minimum, maximum, standard deviation, last observed of value.
   * Converts timestamps to a specific timezone (America/Los\_Angeles).

**4. Loading Module (load.py)**

* Displays and saves the processed data.

**Functions:**

1. **display\_data**: Prints the schema and the top 20 rows of a DataFrame.
2. **save\_data**: Saves data to the specified path in a **partitioned CSV format** (partitioned by device).

**5. Visualization Module (visualization.py)**

* Generates visual insights from the processed data.

**Visualization Logic**:

* Plots the time series of average values (WTUR1\_W\_Avg) for each device.
* Titles and labels the plot with details about the device and null handling.
* Rotates x-axis labels for better readability.

**Saving Plots:**

* Saves visualizations as PNG files in the specified directory.

**6. Logging Utility (logger.py)**

* Provides a reusable logging setup for all modules.
* Logs messages to both the console and a file.
* Supports structured logging with timestamps and log levels.