Software Design for Data Pre-processing Pipeline

Version 1.0

prepared by: ALROY CHIANG

organization: edpr sunseap

Date: 31-01-2023

Contents

[Revision History 2](#_Toc127519726)

[Introduction 3](#_Toc127519727)

[Project Folder Structure 3](#_Toc127519728)

[Software Design 4](#_Toc127519729)

[1. Raw data input 4](#_Toc127519730)

[2. Data quality 4](#_Toc127519731)

[3. Data transformation 5](#_Toc127519732)

[4. Meta data 6](#_Toc127519733)

[5. Publish data 6](#_Toc127519734)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
| Alroy Chiang | 31-01-2023 | First version | 1.0 |
|  |  |  |  |
|  |  |  |  |

# Introduction

The purpose of this document is to provide technical design for data pre-processing pipeline. The data pre-processing pipeline is a framework that allows users to process raw data files into data files ready for utilization. The pipeline includes the following components as shown in *Figure 1*:

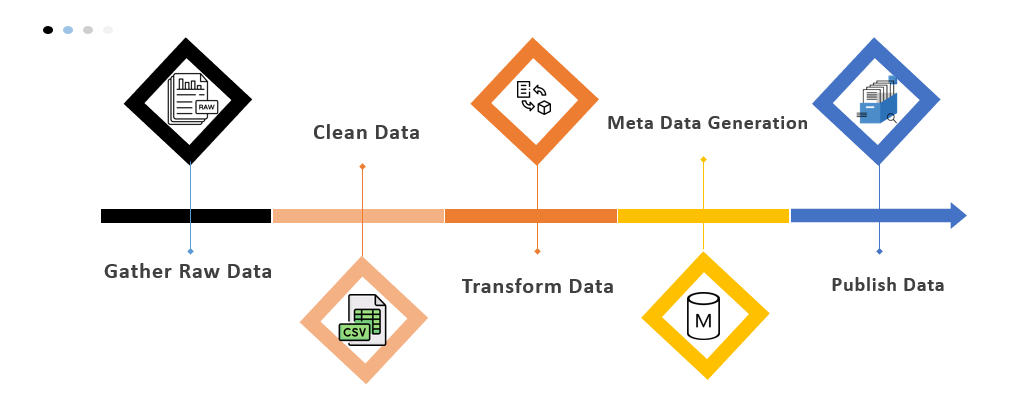


Figure Main components of Data pre-processing pipeline.

# Project Folder Structure

The overall default project folder structure is shown in *Figure 2*. “project” folder is the root folder of data pre-processing pipeline where all its scripts and configuration files are located. It also consists of “staging” and “processed” folders for raw data files and cleaned data files.

project



staging



processed

rejected



cleaned



rejected



Figure Overall project folder structure for data pre-processing pipeline.

# Software Design

## Raw data input

* 1. Use “staging” folder to place all raw data files to be processed.
  2. Raw data files are to be in CSV format.
  3. Raw data files will be deleted from “staging” folder once they have been processed.
  4. Raw data files with wrong extensions or file names that do not contain necessary information will not be processed and will be moved to “rejected” subfolder in “staging” folder.
  5. Raw data files will be processed sequentially.
  6. Raw data files do not necessarily contain a calendar month worth of data. Data files will be concatenated or segmented accordingly.
  7. Raw data files are to be named in the following format:

location\_\*\_load.csv

For example: Blk10ChaiCheeRd\_Sep2022\_Lighting.csv

File name format is case insensitive.

## Data quality

* 1. Each column will be checked for missing values. Any non-numbers such as NaN and NaT, will be treated as missing values. The number of missing values will be recorded in the meta data.
  2. Dataframe will be checked for any duplicate columns by comparing identical column headers and identical column values.
  3. Each column will be checked for outliers. Outlier is defined as a data point that satisfy either of the 2 following conditions:

*Less than Q1 – 1.5\*IQR*

*greater than Q3 + 1.5\*IQR*

where *Q1* is the first quartile, *Q3* is the third quartile, and *IQR* is the inter quartile range. The number of outliers in each column will be recorded in the meta data.

* 1. The column headings will be standardized to the headers tabulated below:

|  |  |  |
| --- | --- | --- |
| **SN** | **Standard Headers** | **Description** |
| 1 | TIMESTAMP | Timestamp |
| 2 | V1 | L1 Live-Neutral RMS voltage |
| 3 | V2 | L2 Live-Neutral RMS voltage |
| 4 | V3 | L3 Live-Neutral RMS voltage |
| 5 | I1 | L1 RMS current |
| 6 | I2 | L2 RMS current |
| 7 | I3 | L3 RMS current |
| 8 | P1 | L1 real power |
| 9 | P2 | L2 real power |
| 10 | P3 | L3 real power |
| 11 | Q1 | L1 reactive power |
| 12 | Q2 | L2 reactive power |
| 13 | Q3 | L3 reactive power |
| 14 | S1 | L1 apparent power |
| 15 | S2 | L2 apparent power |
| 16 | S3 | L3 apparent power |
| 17 | PF1 | L1 power factor |
| 18 | PF2 | L2 power factor |
| 19 | PF3 | L3 power factor |
| 20 | E1 | L1 energy |
| 21 | E2 | L2 energy |
| 22 | E3 | L3 energy |

## Data transformation

* 1. Transform all other date-time formats into the following format:

**DD-MM-YYYY HH:mm:ss** (24 hours format)

* 1. Calculate readings in log-scale from linear scale. The flag to perform this operation is set in the config.json file with *“LOG\_SCALE”: [columns to be in log scale]* entry. Default entry is set to an empty list []. New columns will be generated for each of the columns specified. Below is a sample entry for the config file:

“*LOG\_SCALE*”: [ “A”, “B”, “C” ]

The alphabets “A”, “B”, “C” represents example columns in the raw data file to be converted into log-scale.

* 1. Compute reading difference in time. The flag to perform this operation is set in the config.json file with *“TIME\_**DIFFERENCING”: [columns to be time differenced]* entry. Default entry is set to an empty list []. The time difference to use for this operation is determined by the number of rows. Below is a sample entry for the config file:

“TIME\_*DIFFERENCING*”: [[“A”, 1], [“A”, 2], [“B”, 2]]

Here, [“A”, 1] represents an example column “A” where the difference between consecutive ‘1’ number of rows is used calculate the time difference in an adjacent separate column.

## Meta data

* 1. A meta data file in text format will be generated for each of the cleaned data file. The meta file will be co-located with the data file.
  2. The meta data file will be named similarly to the data file.
  3. The meta data file within will contain the file’s name itself, in the format

postal\_code\_YYYY\_MM\_load

A sample of the file name would be:

467010\_2020\_09\_Agg

* 1. The meta data fille will contain the number of entries in the dataset
  2. The meta data file will contain the location information where the dataset was collected.
  3. The meta data file will contain the time resolution of its associated data file.
  4. The meta data file will contain the time gap statistics of the dataset.
  5. The meta data file will contain the time period, the start and the end time of the dataset.
  6. The meta data file will contain the reading statistics of each column within the dataset.
  7. The meta data file will contain the missing values count of each column within the dataset.
  8. The meta data file will contain the outliers count of each column within the dataset.

## Publish data

* 1. Data pipeline uses the output folder structure as shown in *Figure 3* to allow a systematic approach to locate a file of interest. Data files will be placed into their respective folders according to location and the year in which the data is collected.

cleaned



postal code 1



postal code n

2022

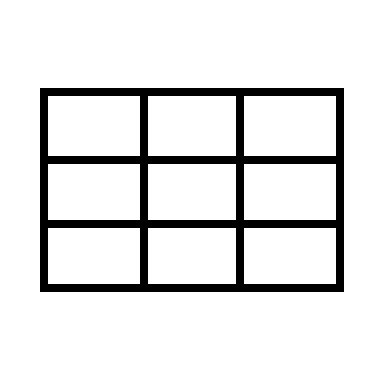


2023



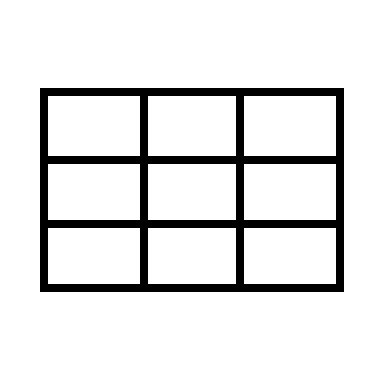
. . .

. . .



oct\_data.csv

nov\_data.csv



oct\_meta.txt



nov\_meta.txt



. . .

. . .

Figure Output folder structure for processed data files.

* 1. Each data file will contain only data collected in a same calendar month.
  2. All cleaned data files will only use ‘,’ as CSV delimiter.
  3. An address legend file address\_legend.csv will be created to translate all postal codes into their full address for readability. This legend will be created and maintained manually and placed in the cleaned folder.
  4. Sample entries for the legend files are shown below:

119074, “5 Lower Kent Ridge Rd, Singapore 119074”

142043, “43 Tanglin Halt Rd, Singapore 142043”

* 1. Cleaned data files will use the following filename format:

postal\_code\_YYYY\_MM\_load.csv

The filename consists of 6 digits postal code, 4 digits year data is collected, 2 digits calendar month data is collected, and the type of load, a sample cleaned data filename would be:

822255\_2022\_12\_WaterPump.csv

* 1. Data files with time gaps larger than a user defined threshold will be placed in the ***rejected***folder. Time gap threshold is defined in the config.json file in unit of seconds. The default threshold is set to 3600 seconds.