Ervaringwijzer Programming Assignment

Assignment

Context

An Energy Supplier platform receives readings from metering points for all of its customers everyday. For this assignment, we are going to assume that such metering points are either measuring the usage of gas or the usage of electricity and report a new reading value every 15 minutes. We can convert these readings to usage intervals for every 15 minutes by using the formula: $usage = reading_{i+1} - reading_i$ .Your task in this assignment is to calculate the total cost for all usages. However, metering points are sensitive to errors, thus a metering point could return incorrect reading values. Readings are considered incorrect whenever a usage for a 15 minute interval is either negative or greater than 100. Therefore, you should filter incorrect readings before calculating the cost. For this assignment you can estimate the incorrect reading values by assuming that the missing usage has been consumed linearly.

In short, filtering means skipping the incorrect reading values entirely, and linearly consumed means the usage between two correct readings is divided equally per 15 minutes.

Input

The input for this assignment is provided as a CSV file containing readings for both electricity and gas usage. You can assume that the CSV file is ordered on metering point ids and ascending date. The CSV file consists of 4 columns:

- **metering_point_id**: the id of a metering point
- **type**: the type of reading; ‘1’ Electricity or ‘2’ Gas
- **reading**: the raw value from the metering point. The readings for electricity are provided as Wh and the reading for gas are provided as m3
- **created_at:** contains a unix timestamp for when the measurement was done

Cost

The readings are in Wh and m3, to obtain kWh the following rules apply:

- Wh to kWh: $kWh = Wh / 1000$
- m3 to kWh: $kWh = m_3 * 9,769$

There are different prices for electricity and gas based on the time in which they are read. These different prices are described in the table below:

Type	**23:00 - 07:00**	**07:00 - 23:00**
Elec (type = 1) Mo t/m Fr	0,18/kWh	0,20/kWh

Elec (type = 1) Weekend	0,18/kWh	0,18/kWh
Gas (type = 2)	0,06/kWh	0,06/kWh

Output

The objective is to process the CSV and output a CSV file containing the **total_price** per **metering_point_id**

So input:

metering_point_id,type,reading,created_at
1,1,166606,1415963700
1,1,166694,1415964600
1,1,166714,1415965500

Should output:

id,cost
1,0.02

Correctness

Your solution will be validated against a test set to validate correctness.

Performance

Your solution will be tested against large CSV files(a few GB's).

Code Quality

Your code will be analyzed to see if you are able to apply the Go conventions. Furthermore, we value readability and the correct use of design patterns.

To ensure your code is properly structured please use a linter. A recommended tool for linting your solution is golangci-lint: h ttps://github.com/golangci/golangci-lint

Unit testing

We think it's really important to write unit tests for our software. Therefore we ask you to write unit tests for your solution. We highly recommend using the testify library to do so: https://github.com/stretchr/testify

Deliverable

1. Your source code shared as zip or via a private Github repository
2. A README explaining the general idea and the architecture of your solution
3. A clear description of assumptions you have made
4. A clear description on how to run your solution
5. A clear description for running your tests & linter