Обработка csv файла

Что нужно сделать?

Нужно написать скрипт для обработки CSV-файла, поддерживающий операции:

- фильтрацию с операторами «больше», «меньше» и «равно»
- агрегацию с расчетом среднего (avg), минимального (min) и максимального (max) значения

Собираем прототип, поэтому всё по простому. Фильтрацию и агрегацию делаем по одной любой колонке. Делать фильтрации с составными условия, например с and или or, а также по нескольким колонкам одновременно не нужно. Фильтрация поддерживает любые колонки, то есть с текстовыми и числовыми значениями, а агрегация только числовые. Гарантируется что входные файлы валидны, например если в колонке числа, то там все значения числа. Чтобы сфокусироваться на функционале и не отвлекаться на рутинные задачи (обработка аргументов скрипта, чтение файла и форматированный вывод), можно использовать стандартную библиотеку argparse и csv, а для красивого отображения в консоли установить библиотеку tabulate.

Пример файла csv:

```
name,brand,price,rating
iphone 15 pro,apple,999,4.9
galaxy s23 ultra,samsung,1199,4.8
redmi note 12,xiaomi,199,4.6
poco x5 pro,xiaomi,299,4.4
```

Пример запуска скрипта:

```
(.venv) $ python3 main.py --file products.csv
+------------------+---------+---------+----------+
| name             | brand   |   price |   rating |
|------------------+---------+---------+----------|
| iphone 15 pro    | apple   |     999 |      4.9 |
| galaxy s23 ultra | samsung |    1199 |      4.8 |
| redmi note 12    | xiaomi  |     199 |      4.6 |
| poco x5 pro      | xiaomi  |     299 |      4.4 |
+------------------+---------+---------+----------+
(.venv) $ python3 main.py --file products.csv --where "rating>4.7"
+------------------+---------+---------+----------+
| name             | brand   |   price |   rating |
|------------------+---------+---------+----------|
| iphone 15 pro    | apple   |     999 |      4.9 |
| galaxy s23 ultra | samsung |    1199 |      4.8 |
+------------------+---------+---------+----------+
(.venv) $ python3 main.py --file products.csv --where "brand=apple"
+---------------+---------+---------+----------+
| name          | brand   |   price |   rating |
|---------------+---------+---------+----------|
| iphone 15 pro | apple   |     999 |      4.9 |
+---------------+---------+---------+----------+
(.venv) $ python3 main.py --file products.csv --aggregate "rating=avg"
+-------+
|   avg |
|-------|
|  4.67 |
+-------+
(.venv) $ python3 main.py --file products.csv --where "brand=xiaomi" --aggregate "rating=min"
+-------+
|   min |
|-------|
|   4.4 |
+-------+
(.venv) $
```

Для фильтрации используем where, для агрегации aggregate, значение передаются как “column=value”. Не меняем интерфейс скрипта, например не разбиваем параметр aggregate на два параметра aggregate-column и aggregate-value.

Какие функциональные требования?

- можно передать путь к файлу
- можно указать условие фильтрации
- можно указать условие агрегации
- в консоль выводится таблица с результатами выборки или агрегации

Какие не функциональные требования?

- для всего кроме тестов и красивого вывода в консоль, можно использовать только стандартную библиотеку, например:

    - для работы с параметрами скрипта нельзя использовать click, но можно использовать argparse
    - для чтения файлов нельзя использовать pandas, но можно использовать csv
- код соответствует:
    - общепринятым стандартам написания проектов на python
    - общепринятому стилю

Как получить дополнительные баллы за тестовое?

Это дополнительные требования, их не обязательно реализовывать, но они помогут проявить себя. Если эти требования будут реализованы, хотя бы частично - это даст дополнительные баллы:

- в архитектуру заложена возможность быстрого добавления новых видов агрегации или даже команд, например, если захотим добавить медиану или order by по колонке типа --order-by “brand=desc” или “brand=asc”, то это можно будет сделать не переписывая пол проекта
- в коде используются аннотации

Дополнительные требования

- Входные файлы всегда в формате csv
- Не использовать pandas, он не входит в стандартную библиотеку, можно использовать csv.
- Нужно учесть случаи, когда пользователь при запуске скрипта ввёл что-то не то или совершить опечатку.
- Скрипт не должен привязывать к колонкам из тестового файла.