1. Слияние логов

Имеется два файла с логами в формате **JSONL**, пример лога:

```
…
{"timestamp": "2021-02-26 08:59:20", "log_level": "INFO", "message": "Hello"}
{"timestamp": "2021-02-26 09:01:14", "log_level": "INFO", "message": "Crazy"}
{"timestamp": "2021-02-26 09:03:36", "log_level": "INFO", "message": "World!"}
…
```

Сообщения в заданных файлах упорядочены по полю timestamp в порядке возрастания.

Требуется написать скрипт, который объединит эти два файла в один.
При этом сообщения в получившемся файле тоже должны быть упорядочены в порядке возрастания по полю timestamp.

К заданию прилагается вспомогательный скрипт на python3, который создает два файла "log_a.jsonl" и "log_b.jsonl".

Командлайн для запуска:
log_generator.py <path/to/dir>

Ваше приложение должно поддерживать следующий командлайн:
<your_script>.py <path/to/log1> <path/to/log2> -o <path/to/merged/log>

Решение:

https://github.com/tuimazy2008/MergeLogs/blob/main/logs_merger.py

2. Миграция базы данных

Есть база данных и два типа сервисов **А** и **Б:**

- Сервисы типа **А** добавляют в базу записи в формате: **id**, **name**, **status**, **timestamp**.
- Сервисы типа **Б** читают эти данные для агрегации и прочих нужд.

В какой-то момент выясняется, что строковые имена в этих записях занимают слишком много памяти, и при этом часто повторяются. Поэтому целесообразно вынести их в отдельную табличку.

В результате данные должны будут добавляться в следующем формате: **id**, **name_id**, **status**, **timestamp**. Где **name_id** внешний ключ к новой таблице с полями: **id**, **name**.

Задача заключается в том, что бы составить пошаговый план миграции базы и сервисов на новый формат данных, при этом не разломав работоспособность системы.

ВАЖНО:
1. Нельзя остановить и обновить все сервисы разом (т.е. нельзя остановить сразу все сервисы типа А, либо все сервисы типа Б, обновление сервисов происходит по одному за раз).
2. В базе данных атомарно можно делать только следующие запросы:
  - добавить колонку
  - удалить колонку
- переименовать колонку

Решение:

1) Состояние БД до миграции:

Id	Name	Status	timestamp
1	Иван	а	1
2	Василий	б	2
3	Иван	в	3

1) Состояние БД после миграции:

Id	Name_id	Status	timestamp
1	#1	а	1
2	#2	б	2
3	#1	в	3

Id	name
1	Иван
2	Василий

Последовательность действий:

1) По одному обновляем экземпляры сервиса А таким образом, чтобы запись велась и в старом и в новом виде.
Таким образом добавляемая запись в табл. содержит и name и name_id, при этом для сервиса типа Б ничего не поменялось.
В какой-то момент времени когда будут обновлены все сервисы типа А в таблицу не добавляются новые записи которые не содержат name_id.

2) Далее используя доступные операции для работы с БД дополняем старые записи из БД до нового формата

3) На текущий момент БД имеет возможность выдавать записи в новом формате поэтому мы можем начать обновление сервисов типа Б чтобы выполнять чтение в новом формате

4) После того как мы обновили все сервисы типа Б мы можем изменить экземпляры сервиса типа А таким образом чтобы больше не добавлялось записей в старом формате.

5) После этого мы можем изменить оставшиеся в БД записи используя доступные операции для работы с БД, удаляем устаревшую колонку name