1. **Построить сервис, который забирает JSON данные с адреса** https://raw.githubusercontent.com/biggiko/nasa-dataset/refs/heads/main/y77d-th95.json и сохраняет их в базу данных. Он должен запускаться периодически по расписанию. Нужно предполагать, что возвращаемые JSON данные могут меняться (добавляться, удаляться, изменяться). Нужно обновлять данные в нашей базе с учетом этих изменений.

Что интересно в вашем решении:
- Как вы построите архитектуру сервиса
- Как будете забирать данные из стороннего сервиса (оптимальный метод, обработка ошибок и т.д.)
- Как вы построите схему базы данных (типы данных, индексы и т.д.)
- Как вы будете сохранять данные в базу (оптимальный метод)
- Стиль кода и следование гайдлайнам C#

2. **Создать API метод, который будет возвращать отфильтрованные, сгруппированные и отсортированные данные, которые сохранили в шаге 1.**
Фильтры: по диапазону лет (например с 1970 по 1980), по классу метеорита (recclass), по части названия метеорита (вхождение).
Группировка данных: по году.
Сортировка данных: по году, по количеству, по суммарной массе.

Что интересно в вашем решении:
- Валидация входящих параметров
- Выборка данных из базы (оптимальный метод)
- Кэширование
- Обработка ошибок

3. **Создать страницу, которая выведет данные из API метода, созданного на шаге 2, в табличном виде.** Можно использовать чистый HTML, JS, CSS либо один из frontend фреймворков (VueJS, React, Angular).
Фильтр: год с, по (выпадающие списки), класс метеорита (выпадающий список), часть названия метеорита (текстовое поле поиска).
Сортировка всех колонок по возрастанию и убыванию.

Год	Количество метеоритов	Суммарная масса
1980	12	100
1981	4	200
Total	16	300

Что интересно в вашем решении:

- Как вы построите архитектуру страницы
- Способ обращения к API, обработка ошибок
- Стиль кода и следование гайдлайнам HTML, CSS, JS/TS

Можно использовать любые инструменты и сторонние библиотеки, которые помогут вам быстрее, оптимальнее и лучше сделать ваш проект. Выложите готовый код на GitHub.