Weather sensor data storage

The problem

You have been tasked with building a system to store data generated by a collection of sensors set up worldwide. The sensors send a CSV payload of data which they collect over a period of time. The schema of the data which the sensors send will be identical everytime, the only difference will be how much data is sent with each request.

Below is an example of the CSV that will be sent, including a small collection of the data that can be expected.

```
Unset
timestamp,temperature,rainfall,humidity,wind_speed,visibility
1690967790,14.1,6.11,20,23,M
1690999756,16.2,4.23,30,12,M
1691012723,15.7,3.56,20,11,G
1691032353,17.6,2.19,40,18,VG
1691054751,19.5,1.20,50,7,E
```

In the data we have six columns:

1. Timestamp - an epoch timestamp, in seconds
2. Temperature - the temperature in celsius
3. Rainfall - the amount of rainfall during the day, in mm
4. Humidity - the humidity, as a relative humidity %
5. Windspeed - the average wind speed in mph
6. Visibility - the visibility as a scale with options, VP, P, M, G, VG and E (very poor to excellent)

The system

The service will need to have three major endpoints implemented:

1. POST “/api/login” - authentication endpoint, the body of requests to this endpoint should be JSON, with an email and password, e.g.
   `{ “email”: “admin@admin.com”, “password”: “pass” }`
   The request should respond with a cookie that can be used for subsequent requests.

2. POST “/api/sensors/upload” - this endpoint will receive data from the sensors. It should be authenticated utilizing the cookie from the login request. The CSV data will be sent in the POST body, which should then be stored in a way that allows for it to be queryable.

3. POST “/api/sensors/search” - this endpoint will be used to retrieve data and again should be authenticated utilizing the cookie from the login request. This request can perform three key functions:

- Grouping the data of a sensor
- Filtering the data of a sensor
- Sorting the data of a sensor

The endpoint should return all of the data which matches given the above parameters, the request body will be in the form of a JSON object, like such:

```
Unset
{
  "filters": {
    "temperature": { gte: 20 },
  },
  "sort": {
    "column": "temperature",
    "order": "descending",
  },
  "aggregate": {
    "column": "visibility",
    "operator": "COUNT",
  }
}
```

The following options are available for the various search keys:

1. Filters, the following filter operators should be supported: “gte”, greater than or equal, “lte”, less than or equal and “eq”, equals, returned results must match all of the provided filters.
2. Sort - the column can be any of the columns in the CSV, the order can be either “descending” or “ascending”.
3. Aggregate - the column can be any of the data columns in the CSV (not timestamp) and should support the aggregate functions of COUNT, MAX, MIN, SUM and AVG.

Technical requirements

This assignment should be completed using Node.js, either in JavaScript or TypeScript. Latest Node versions should be used (Node 18+).

If any dependencies are required these should be provided using Docker.

The completed project should be returned via a Github repo which can be shared with the team. Send your completed private Github repo link to techtest@budibase.com and invite 2 users:

- shogunpurple
- mike12345567