Milking our data…

Backend Engineering Assignment

At Connecterra, we integrate and process data from various types of management systems to make sense out of what is going on at the farm and provide valuable insights to farmers and their advisors.

These different systems do not always integrate nicely with each other on the farm, and on some dairies the initial data entry still happens by pen and paper as digitalization is often not a top priority in the day-to-day of a farmer. Which results in late entry, sometimes mistakes and some 'fun' challenges for us at Connecterra!

Better data entry means better data quality and as a result, more accurate and higher value insights that we can generate on top of this data, and therefore our data pipeline aims to 'repair' inaccuracies and add context where it is missing.

In this assignment, we have provided 2 datasets (exported from MongoDB collections) as Json, which contain:

- 'Production': These come from the farm's milking system, which just takes care of identifying the animals by an electronic tag and record their yield. This system does not have any of the contextual information about the animals that we need for making aggregations on top of the data to analyse specific 'segments' of the herd.
- 'Snapshots': These come from a daily upload from the "Dairy Herd Management System" (DHMS) which keeps track of the status and location of all the animals on the farm. This information can be used to know if an animal was pregnant on a certain day, giving milk or dry, in which physical location ('Pen') they reside and more.

The first step for you is to import these datasets so you can perform some data joining operations on them in the next steps of the assignment. At Connecterra we commonly use MongoDB aggregation pipelines for data processing, but you are free to choose whatever tools you feel comfortable with for making the assignment as long as you can explain your decision.

⚠️ If you choose to import the data in Mongo, some possible ways to get your system up and running to work with these datasets is to:

- install mongodb community server,
- or use the official Docker image
- or signup for the free tier of MongDB Atlas and then import the collections, i.e. using Mongo Compass. You can also use another tool, or simply read it in memory with a small program if that suits you better.

Exercise questions:

1. Write a program (script or data pipeline) that aggregates the production dataset, which could be used to give a farmer insight in the average yield per cow per day on his farm. (i.e. could be shown in a table or plotted on a grid with a time and 'yield' axis.
   - Please note that the datasets contain data from 2 different 'Farms' (as differentiated by their distinct farmIDs).
2. Farmers are often interested in different sub-groups of their animals. Therefore, alter the script to calculate the average yield per Cow for each pen per day.

3. Finally, consider only the animals that are in their third lactation or higher. (i.e. they must have calved 3 times in their lifetime).
   Now that you (hopefully) got a bit of a feel for what the data looks like, envision you were tasked to exposing the above query results to our frontend application.
4. Propose an interface for what the new endpoint should look like so that the frontend team can use it to obtain the results. Please think about what information needs to be exchanged between frontend and backend, such as the query parameters, headers, body, and response format, and consider how the interface would progress if other 'potentially interesting' milking related data would be requested or when other ways for querying the data would need to be added.
5. Write a sample application that exposes this new endpoint and can run the query and (loosely) adheres to the proposed interface in step 4. Note: We say loosely because we do not expect a full-fledged web application, and you may even simplify your program by keeping it just a simple CLI application or read input options from a file location.

Deliverables:

- The scripts to run questions 1 and 2 and 3.
- A text-based document (< 1 page) explaining the design of 4.
- The source code of your program for 5.

We do not necessarily need to run the application ourselves, but please make sure you will be able to demonstrate it to us when we discuss the assignment (please bring a laptop if you can!).

You may send it in your solution by sharing an archived file, or link to a (public) git repo from which we can clone your solution and respond to igor.schouten@datamars.com to let us know we received your input.

Please also reach out on the above email address in case you have any questions that prevent you from finishing the assignment.

We hope you will find some fun in this challenge, and good luck!