Let’s slice & dice

Problem statement

In this problem we’ll create a micro-service to address some functionality which is useful to derive simplified summary statistics (mean, min, max) on a dataset. The dataset that you’ll be working with can be found later in the document and yes, it’s been kept very simple by design. In our view, depending on your speed and how elegantly you want to solve this problem, it could take you anywhere between **2 - 4 hours to implement this**.

NOTE: Whenever we mention <SS> we mean summary statistics which essentially means 3 values (mean, min, max)

For this assignment, we are looking for following functionality to be implemented:

1. An API to add a new record to the dataset.
2. An API to delete a new record to the dataset.
3. An API to fetch SS for salary over the entire dataset. You can ignore the currency (if not mentioned otherwise) of the salary and simply treat salary as a number.
4. An API to fetch SS for salary for records which satisfy `"on_contract": "true"`.
5. An API to fetch SS for salary for each department. This means that whatever you’ll do in Step 3, should be done for each department. The return of this API should have 1 SS available for each unique department.
6. An API to fetch SS for salary for each department and sub-department combination. This is similar to Case 5 but 1 level of nested aggregation.

Evaluation Criterion:

A few notes on implementation:

- Please have a `readme.md` at the root of your project, which should contain all the necessary steps for us to run your service. In addition your `readme.md` should have examples on running all the API end-points that you’ll implement.

- Please use docker to bootstrap all infrastructure for this project. This means a `docker-compose.yml` should be there at the root of your project and that a simple `docker-compose up` at the root of the project should be sufficient to run your service.

- 1 test case for every API that you will implement. The instructions to run the test should be part of your readme.

- For this exercise you can use an in-memory data structure to keep the entire data in-memory. This will help you avoid any DB setup. If you feel like it, you can use an in-memory DB implementation like H2 or a standalone binary implementation like SQLLite. Feel free to make a choice here.

- Implement basic authentication and authorization. For this part, you can create a dummy user (with any username and password) and then use those credentials to authenticate. As for authorization, you can use any token based mechanism available in your choice of stack. We'll test this functionality by first checking for a happy / failed case for authentication and then doing the same for authorization by trying to alter the authorization token and see if we are able to interact with the APIs.

- Error handling. For this part we expect at least following things:
    - The input payloads should be validated for their schema.
    - The use of proper error codes when authentication and authorization fails or when there's a logic error in your code.

Mode of submission:

You can zip your root folder and send it over the email. Please make sure that the zip file has its name in the following format **{first}-{last}.zip.** An example: **john-doe.zip**. If you are using version control, then zip the version control folder (.git or equivalent) as part of the zip.

Dataset (Json)

```
[
   {
       "name": "Abhishek",
       "salary": "145000",
       "currency": "USD",
       "department": "Engineering",
       "sub_department": "Platform"
   },
   {
       "name": "Anurag",
       "salary": "90000",
       "currency": "USD",
       "department": "Banking",
        "on_contract": "true",
        "sub_department": "Loan"
    },
    {
        "name": "Himani",
        "salary": "240000",
        "currency": "USD",
        "department": "Engineering",
        "sub_department": "Platform"
    },
    {
        "name": "Yatendra",
        "salary": "30",
        "currency": "USD",
        "department": "Operations",
        "sub_department": "CustomerOnboarding"
    },
    {
        "name": "Ragini",
        "salary": "30",
        "currency": "USD",
        "department": "Engineering",
        "sub_department": "Platform"
    },
    {
        "name": "Nikhil",
        "salary": "110000",
        "currency": "USD",
        "on_contract": "true",
        "department": "Engineering",
        "sub_department": "Platform"
    },
    {
        "name": "Guljit",
        "salary": "30",
        "currency": "USD",
        "department": "Administration",
        "sub_department": "Agriculture"
    },
    {
        "name": "Himanshu",
        "salary": "70000",
        "currency": "EUR",
        "department": "Operations",
        "sub_department": "CustomerOnboarding"
    },
    {
        "name": "Anupam",
        "salary": "200000000",
        "currency": "INR",
        "department": "Engineering",
        "sub_department": "Platform"
    }
]
```