Purchase orders

Introduction

- This is a small application that provides an API to access the data from a csv file purchase-orders.csv.
- There are two endpoints in the api, one to *list all the purchase orders* (`/` or `/?action=list`) and another to *list average price of products per year* (`/?action=average-product-price-per-year`).
- Both endpoints support filtering by year, product or supplier by passing the `year` , `product` or `supplier` as a query parameter.
- In order to run this project,
  - install devenv and run `devenv up` in the project directory
    (if you're using Windows 10 or higher, then you need to install WSL 2 first).
  - if you're using some other OS that does not support `devenv` (like Windows 7) then you need to install PHP 8.1 and composer and run `composer install` in the project directory.
- The api is available at `http://127.0.0.1:8000/api.php` and the UI is available at `http://127.0.0.1:8000/` .

Task

If you're using PHP:

1. Implement the `averageProductPricePerTonnePerYear` method of the `PurchaseOrderService` class, which should calculate the weighted average of price per tonne for each product for each year.
   - Only price per tonne should be aggregated, other fields can be taken from the first row.
   - The result should be sorted by year and product name in ascending order.
   - `pricePerTonne` should be rounded to three decimal places.
   - It should be possible to filter the data by year, product, or supplier by passing the `year` , `product` , or `supplier` as a query parameter.
2. The list api (`/api.php?action=list&year=yyyy`) has a bug in year filter. It should return the purchase orders for the given year, but it returns an empty list. Fix the bug. **BONUS:** Write a test for the bug.

If you're using any other language (like Node.js, Python, Ruby, etc.):

- Create an api endpoint that accepts a GET request to `/` (or `/?action=list`) and returns all the purchase orders in JSON format.
- The result should be sorted by date and product name in ascending order.
- The result format should be the same as returned by php app (`http://127.0.0.1/api.php`)

- It should be possible to filter the data by year, product, or supplier by passing the `year`, `product`, or `supplier` as a query parameter.
- Serve the UI by copying `public/index.html` file to your app and changing the api path in the file (line 197).
- **BONUS TASK:** Make an API endpoint that responds to a GET request at /?action=average-product-price-per-year.
  - The response should be a weighted average of the price per ton of each product for each year, presented in JSON format.
  - The data should be sourced from the purchase-orders.csv file
  - Only price per tonne should be aggregated, other fields can be taken from the first row.
  - The result should be sorted by year and product name in ascending order.
  - `pricePerTonne` should be rounded to three decimal places.
  - It should be possible to filter the data by year, product, or supplier by passing the `year`, `product`, or `supplier` as a query parameter.

HINT: You can use sqlite database `db.sqlite` which is created upon first run of the api instead of the csv file `purchase-orders.csv`.

API

The structure of the api response should be similar to the following:

```
[
  {
    "date": "2020-05-25",
    "supplier": "Jones-Jenkins",
    "orderNumber": "S-1000001",
    "product": "UCOME",
    "pricePerTonne": 688000
  },
  {
    "date": "2020-05-25",
    "supplier": "Jones-Jenkins",
    "orderNumber": "S-1000002",
    "product": "UCOME",
    "pricePerTonne": 784000
  },
  {
    "date": "2020-05-25",
    "supplier": "Jones-Jenkins",
    "orderNumber": "S-1000003",
    "product": "UCOME",
    "pricePerTonne": 976000
  }
]
```

Caveats

The "price per tonne" field doesn't exist in the csv file, rather it has "Price per 100 grams" and "Quantity in kg". The average price per tonne needs to be calculated from these two fields. Average price per tonne is calculated by dividing the "total amount" by the "total quantity in tonnes" (aka weighted average).

```
1 tonne = 1000 kg
1kg = 1000 grams
```

Testing

Make sure the functional tests pass.

- Update the `API_URL` environment variable in `.env` file to point to your api. Create a `.env` file if it doesn't exist.
- Run `devenv shell` to open a shell in the container.
- If you've only implemented the list api, then run `composer test-list-api` to run the list api tests.
- If you've implemented the average price per tonne api as well, then run `composer test-api` to run the api tests.
- Results can also be compared with the working app which is hosted at https://tech-assessment.qbiltrade.com with the api at https://tech-assessment.qbiltrade.com/api.php.

Submission

You can submit the solution as a git repository, with a README file explaining how to run the solution or you can submit a zip file containing the solution and email to info@yarikul.com