# The dataset consumption

## Description

Domain.com is providing access to their partners to a new API that can be used to retrieve the transactions users have made using each partner's services.
As a partner, we want to be able to digest that data to understand how our users are using our transaction system throughout the 27 countries we operate in.
The endpoint of such an API is *domain.com/new_api*. This API returns a CSV file, base64-encoded, with a maximum size of 5Gb but the total size of the data can be much bigger. For this, the API supports pagination and we can request a given page via the *page* parameter in the request body (a JSON request body).

The API response includes the base64-encoded CSV file and a *next_page* value that can be used to retrieve the next page. If *next_page* is null it means there are no more pages to load.

The dataset has the following columns:

| Column Name | Description |
|---|---|
| id | Transaction ID (int) |
| country | Country where the transaction took place (string) |
| status | Status of the transaction (string with values *pending, completed* and *failed)* |
| amount | Transaction amount (float) |

Keep in mind that the CSV returned has been manually filled by our users and can have missing values. We want to digest the entire dataset (all pages) and produce a resulting dataset that includes the following columns:

| Column name | Description |
|---|---|
| country | String representing a given country |
| average_outstanding | Average amount of pending transactions per country |
| total_completed | Total amount of completed transactions per country |
| critical_rate | Error rate (failed transactions over the total number of transactions) with an amount greater than $1M per country |
| error_rate | Error rate (failed transactions over the total number of transactions) with an amount smaller or equal than $1M per country |

## You should:

Create a system that digests the entire dataset supplied by the API and processes it producing the resulting dataset described above.

The resulting dataset should be stored as requested via a command line option. The option --output-type can have three values: local, s3 and pg. These values represent local storage as a CSV, S3 as a CSV or in a postgres database respectively.

We also want to be able to measure the execution time of every method.

Build a set of unit tests that test the behavior of the system.

## Notes

- You can assume there is a method *retrieve_dataset(page)* that makes the api call for you for a given page and returns the JSON response. You do not need to implement the request.
- You can assume there is a method *save_to_s3(file_key)* that handles writing into an S3 key. You do not need to implement the writing into S3 logic.
- You do not need to run the code to ensure it is working or have unit tests passing. The design choices and structure will be evaluated and no working example is needed.
- You can use any third party libraries you feel comfortable using.