# Data Ingest Engineer - Technical Exercise

Daniel Outten

## Introduction

This document is for the Panaseer technical exercise for the Data Ingest Engineer. Understanding a variety of data sources that can be ingested from is key as well as being able to demonstrate data transformation techniques and justify them.

This task has been completed in Python as well as R, a common data science programming language. Both scripts are in the `src/` directory and are self contained. The README contains some useful commands to set-up a virtual environment for python and install the required packages using pip.

Both scripts will process the incoming data from the api call as well as save a copy of the raw and transformed data as an output.

## Objectives

- Ingest data from a publicly available API.
- Transform raw data into a common information model.
- Handle edge cases and data inconsistencies.
- Make informed decisions when faced with different options.
- Consider backward compatibility in an ETL (Extract, Transform, Load) process.
- Demonstrate your problem-solving approach and communication skills

## Python - Overview

Python is a popular choice for data science tasks such as this due to its simplicity, readability, and extensive libraries. It supports easy data manipulation, analysis, and visualisation as well as easy integrations with other tools.

## R - Overview

R is widely used in data science for statistical analysis, data visualisation, and machine learning. R excels in handling complex data manipulations, producing high-quality plots, and applying statistical models, making it a powerful tool for data exploration and insights.

## Data Ingestion

Data ingestion from an api is simple in python can can be done in a few lines of code. In `src/cocktailDB.py` this is done in function `fetch_cocktail_data()` where. using the requests library, an api call is made to

the *TheCocktailDB* random api. This function checks the status code for a 200 and will raise an exception if it doest get this.

The random api was used to make sure that each time the code was run it would work in the majority of cases on as much variation of the data as possible. The status code exception allows the script to be more robust.

This could be improved further with a retries system so that it can try two more times before raising the exception.

## Data Transformation

Data transformation involves converting raw data into a format suitable for analysis. It includes cleaning, normalizing, aggregating, encoding, or reshaping data to enhance its quality and structure. This process improves accuracy, consistency, and usability, making it easier to derive insights and apply statistical or machine learning models.

These script contains three functions that preform data transformations: *transform_cocktail_data()*, *standardise_units()*, and *normalise_string()*.

Within the *transform_cocktail_data()* function a mapping of the api json output data is made to names that are easier to understand and use. Inside this mapping it calls the use of the two other functions:

*standardise_units()* function will attempt to convert the units for each cocktails into SI units (ml) while the *normalise_string()* function will clean text based data by removing any special characters and white space characters.

Data transformation is a key component in the pipeline to make sure that the raw data being ingested is in a usable form that can be further processed and analysed later on.

The unit conversions, while they work could be more accurate to more decimal points as well as including other conversions that might be seen within the data, e.g. “a dash”. This would be an ideal place for a natural language processing (NLP) model could be used to make an assumption on the ml amount.

## Edge Cases

Edge cases refer to situations that occur at the extreme boundaries of input data, conditions, or usage scenarios. Testing for edge cases ensures robustness, preventing failures in uncommon or unexpected situations during real-world use.

The *handle_edge_cases()* function shows an example of these by removing ingredients with no matching measurements or vice versa. Simply put it will match up ingredient 1 with the corresponding measurement 1.

This is only one edge case and a lager data set or more running the script over a longer period of time could highlight other potential edge cases that need to be handled. They too would have their own functions and this script would need maintained while in production use.

## Main loop

This is the core of the script and where the request to the api is made and where each of the above sections are applied to the raw incoming data. This script ultimately chooses to process the data at ingestion (real-time processing). This is key for streams of data that needs actions quickly or data that requires an immediate response.

However, the scripts include a section to save both the raw data ingested from the api call and the transformed data. This allows for future batch processing of the data, further analysis on the transformed data and/or verification of the original raw data.

## Conclusion

Both script in R and Python do the same task and this was more of a showcase of the skills needed for each. When it comes to a data science pipe like they both excel at this and have their merits. At its core these script complete these task however further features could be added to improve the reliability and robustness of these scripts, as already mentioned with the api retires. These scripts also only process the data into a normalised and standardised data set. Further analysis and testing can be performed and visualisations could be made out of this data as required.

## Final words

Finally, this report was written in R markdown a mark-up language. This was chosen for the integration with the R langue but it also has great support for python as well as many other langues.

It also includes a preamble tex file. This is mainly used in the creation of the pdf document specifying the layout, font choice and size, headers and footers and more.

This was chosen for the repeatability in formatting and standardisation in the report. Every report made using this will have the same formatting.