1. Introduction

This assessment is designed for candidates to demonstrate their knowledge of good software and data engineering practices. Please take the time to read through this document before arriving at our office for your interview.

Please do not complete the assessment task at home, because we will treat it as a paired programming exercise.

You can use Python to generate the required sample data (instructions for installing python on your machine, are in the links below). However, you are free to use any language to solve the problem, whether that's Python, SQL, Scala - or any other language.

Here are Pythonic examples of a potential starting point:

• https://github.com/JSainsburyPLC/aspire-data-test-python

• https://github.com/JSainsburyPLC/aspire-data-test-python-pandas

So for example, should you choose to use Scala - use the instructions in one of the links above to generate the sample data before coming on site for your interview.

Before you arrive

- Clone one of the above repos
- Choose the language you'll use to solve the problem
- Build and setup your IDE for your chosen language
- Read this 5 page document and note down any questions you might like to ask when we meet

Test Instructions

The purpose of this exercise is to complete a small data pipeline that aggregates and transforms input data according to requirements driven by the data science team. The details of this are given in section 2.

The aim of this pairing exercise is to write code that processes existing data sources so that it meets requirements. Attention should be given to good engineering practises like automated testing as far as possible within the time available.

As part of solving this, we are looking to assess:

- Your problem-solving approach.
- Your ability to turn your solution into working code and choosing appropriate technology.
- How you structure and test your code and your ability to work with others.

The length of the pairing tech test is 1 hour, done as an on-site activity as part of our interview.

2. Pairing Tech Test

User Story

> As a data scientist I want to be able to consume a data source that contains information about how many times each of our customers buys our products in a given period, so that I can predict what they will buy next.

Customer Shopping Patterns

The task involves developing a data pipeline to complete the user story above using sample data sources that will be provided.

Our data science team has reached out to our data engineering team requesting we pre-process some of the data for them at scale so that they can make better use of it in their downstream algorithms. They would like us to deliver this data weekly.

The input data sources are comprised of customers (in CSV format), transactions (in JSON Lines format) and products (in CSV format). Their details are presented below:

Customers

Contains information about customers such as the customer id and the date when they joined:

customer_id	loyalty_score
C1	7

Transactions

Is an ever-increasing data source that currently contains 2 years of transactions.

Each transaction contains the customer id, details of what products they purchased and the date of purchase:

{"customer_id": "C1", "basket": [{"product_id": "P3", "price": 506}, {"product_id": "P4", "price": 121}], "date_of_purchase": "2018-09-01 11:09:00"}

Products

Contains information about products such as the product id, product description and category:

product_id	product_description	product_category
P100	red trousers	C

Acceptance Criteria

The output data source should contain information for every customer that has the following fields:

customer_id	loyalty_score	product_id	product_category	purchase_count
C1	7	P2	F	11
C1	7	P3	H	5
C2	4	P9	H	7

Further Implementation Details

The repo contains a starter project that includes the input data sources, a virtual environment with some dependencies you may find useful and some basic tests to ensure the environment is ready - but only for Python.

It is recommended that candidates bring their own laptop/IDE, and they should download the code and get it running on their machine ahead of the session to avoid losing time on the day.

The code and design should meet the above requirements, and should consider future extension or maintenance by different members of the team.