Data Engineering Test

Introduction

Don’t stress about the two exercises too much – they are supposed to be fun and a way for us to get a feel for your level of experience.

Here are some tips and guidelines:

- We don’t expect you to spend more than 2-3 hours on this challenge;
- If you don’t have time to fully complete the exercises, please still send them in and indicate what your next steps would be;
- These are not pass or fail exercises, we just want to get an idea of how you approach solving problems;
- KISS - Keep it Seriously Simple. Don’t overthink or over engineer it;
- Put your code on a public source repository (such as GitHub) and give us the URL;

Python

Problem: Where should we go?

The Problem
The Time Out team loves to socialise, some of us are also really fussy! In order to spend less time deciding where to go we'd like a program that decides for us. All members of the team will want both food and drink so if a member of staff cannot eat anything, or no drinks are served that they like, the team will not visit the venue.

The Input
There are two JSON feeds, one of which is a list of the team members with what they do drink and do not eat, the other feed contains venues with the food and drink options for that venue.
The person using the app needs to be able to enter which team members will be attending.

The Output
The output should return which places are safe to go to, and if applicable why the team should avoid the other places.

Sample Input

https://gist.github.com/benjambles/ea36b76bc5d8ff09a51def54f6ebd0cb

Sample Output

```
{
  "places_to_visit": [
    "The Diner",
    "The cambridge",
    "The spice of life"
  ],
  "places_to_avoid": [
    {
      "name": "Loch Fine",
      "reason": [
        "There is nothing for Ben to eat.",
        "There is nothing for Far to drink."
      ]
    },
    {
```

```
      "name": "El Cantina",
      "reason": [
        "There is nothing for Ben to drink."
      ]
    }
  ]
}
```

Things to keep in mind for the Python Exercise

- Please take special care to ensure that the submission outputs are to spec;
- We're looking for code that is easy read and focussed on the main problem;
- Where applicable, we expect tests for the main logic;
- We should be able to run your code without any crazy steps;
- Secret tip: Make sure you fetch the sample data at the url above and don't forget data sources can be messy.

SQL

An hypothetical database has 2 tables:

- Prices: contains a record of price changes for a list of products.
- Sales: contains a record of sales, by product/day in quantities.

Prices

product	price_effective_date	price
product_1	01/01/2018	50
product_2	01/01/2018	40
product_1	03/01/2018	25
product_2	05/01/2018	20
product_1	10/01/2018	50
product_2	12/01/2018	40

As an example, *product_1* started at £50 and had a 50% off (£25) between 03/01/2018 and 09/01/2018. On 10/01/2018 it was back to full price (£50).

Sales

product	sales_date	quantity
product_1	01/01/2018	10
product_2	02/01/2018	12
product_1	04/01/2018	50
product_2	06/01/2018	70
product_1	12/01/2018	8
product_2	15/01/2018	9

As an example, *product_1*, on 04/01/2018, sold 50 units (at a price of £25 each).

Question 1: Based on the model and data above, what is the total revenue value?
Question 2: Write a SQL query which would determine the total revenue value.