Non-Rubyists test assignment

Introduction

The purpose of this assignment is to test developers who have no or little professional experience with Ruby.

Hence we don’t expect the candidate to demonstrate expertise in Ruby frameworks (e.g. Rails), nor that (s)he knows all the idioms in the Ruby programming language.

However, it is essential that the candidate can show that (s)he is able to adapt to the Ruby environment quickly and produce readable and correct solutions, as well as demonstrating his/her problem-solving skills.

Task

You will be creating a web-application to serve banners for an advertising agency.

The web-application should be smart enough to render banners based on their revenue-performance.

You are given a set of CSV files, the set contains the following files:

- *impressions.csv* [:banner_id, :campaign_id]
- *clicks.csv* [:click_id, :banner_id, :campaign_id]
- *conversions.csv* [:conversion_id, :click_id, :revenue]

Based on these data, you should be able to determine how well a banner performs for a campaign based on the revenue (which you can find in the *conversions.csv*)

Then there are a few possible scenarios:
(x = amount of banners with conversions within a campaign)

Scenario:	**Requirements**:
x > 10	Show the Top 10 banners based on revenue within that campaign.
5 < x <= 10	Show the Top x banners based on revenue within that campaign.
0 < x <= 5	Your collection of banners should consists of 5 banners, containing: - The Top x banners based on revenue within that campaign. - Banners with the most clicks within that campaign to make up a collection of 5 unique banners when needed.
x == 0	Your collection of banners should consists of 5 banners, containing: - The Top-5 banners based on clicks within that campaign. - If there are less than 5 banners with clicks within that campaign, then add random banners within that campaign to make up a collection of 5 unique banners.

We expect you to be able to create a web-application (be it Rails, Sinatra or Rack), and when a request hits your Campaigns-URL, (for example: http://localhost:3000/campaigns/{campaign_id}), it should somehow render or redirect to one of your top-x banners.

The banners to be served should be simple html page with "<h1><%= banner_id %></h1>".

To avoid saturation, we also believe that the top banners being served should not follow an order based on its performance, but they should appear in a random sequence.

You should also avoid that a banner will be served twice, before the sequence has finished for a unique visitor.

Requirements

- Correct, working, simple and understandable solution.
- It is completely up to you which tools you will use, but please explain all your choices.
- Code pushed to Github.
- Reproducible completely isolated Vagrant / Docker development environment (see Acceptance).

Acceptance

We will perform the following steps to test your application.

1. $ git clone <your git repository> .
2. <your command to setup the environment>
    - => Assume that only Vagrant / Docker exist on the host.
3. $ <your command to run the test>
    - => All the tests should be green.
4. $ <your command to start the application>
    - => http://localhost:3000/campaigns/{campaign_id} should display a banner.