Report for the Plista GmbH task challenge

Tihomir Vachkov

April 18, 2018

1 Instroduction

This report describes my deliveries for the code challenge given to me by the FRI:DAY company. In the attachments of this report you can find the files with the resulting code. In the ZIP file there are three folders - one for each exercise.

My system information:

- OS - Windows 10 Pro
- PHP - 7.1.9 (cli)
- MySQL - 5.7.19
- Eclipse Oxygen 2 for PHP
- Docker - Version 17.12.0-ce-win47 (15139)

2 Task MySQL

For this task I had to make several assumptions. I preferred to explain my assumptions here and to leave the technical explanations as comments in the files. Hence, they can be reviewed in place.

2.1 Database structure

First assumption. Before compiling a query, there should be a defined structure of the tables in the database. In the folder `1_MySQL` you will find a file `db_create.sql`. In this file the database schema is defined. The foreign keys constraints imply the many-to-one relationship between a `campaign` and `ads`, `advertiser` and `campaign`, and the implicit many-to-one constraint between an `ad` and `advertiser`.

These foreign key are referencing the name of the campaign/advertiser which is considered to be unique. I made this choice because doing so, in the result we will see a pretty readable name rather than a number of an id. I know that in practice, referencing an id number reduces the query time which has big impact in a large data set. In a real-world work environment I would refer to an id but just for the purpose of a test exercise, I thought it would be more convenient to show a name rather than a number.

The choice to create three tables instead of one is for performance reasons. This way, the database is normalized and we avoid redundancy and many null attributes.

2.2 Query 1

Second assumption is about the meaning of `advertiser #100 that have more than 50 ads`. At first glance it seems an `id` but if I assume this, it still remains ambiguous and did not make much sense to me. Instead, I assumed that there are many advertisers who have many ads. So we want to make a classification of the top advertisers, i.e. the top advertiser is the one who has the most ads and at least 50. So, my interpretation was, in such a classification, pick the hundredth advertiser. I left some comments in the `db_query_01.sql` file to clarify better how I thought to achieve this.

2.3 Query 2

If such a question is asked, means that the system permits somehow that there exist registered advertisers without an ad. This could be the case when an ad is expired and removed automatically or the advertiser removed his ads. Because we have a table only for advertisers, it is very convenient to try to find those advertisers who appear in the proper table but not in the ads table. The query can be found in `db_query_02.sql` inside the `1_MySQL` folder.

3 Task API

This task is straight forward and I did not have to take any assumptions or had any doubts. I followed the basic rules of REST APIs calls. You can find a `API_calls.txt` file with the calls in the `2_API` subfolder.

4 Task PHP

For this task I found a PHP library with a MIT license by Kevin Warren. There are PHP files to be used directly. I tested it with Opera, Chrome and Microsoft Edge browsers for PC and with an Android emulator for phone and tablet devices. My assumption here is about the output of the small project. By opening to the `index.php` file in a browser, the device type and the browser information are output on the page. I thought that for a simple classification this is enough as a demonstration and the actions to be taken later having these `strings` are beyond the scope of the exercise. In the subfolder `3_PHP` you can find the `Dockerfile` and the folder with the PHP code which will run on the docker.

5 Conclusion

Thank you very much for your time! I hope I described sufficiently my delivery. If you have any questions, do not hesitate to contact me - tivachkov@gmail.com.
I am looking forward to hear from you soon.