Integration project

The objective of this project is both to see if you're able to complete a feature very similar to what is expected on the job as well as to figure out whether this is something you would find enjoyable or not.

Summary

At Tap, we need to import data from over a hundred different vendors, each of them has a different data format, we need to import this data into our own database in a unified format. In this exercise, we will ask you to import data from an FTP server, process it appropriately for multiple data levels, and insert it into a database from a table structure we will provide.

Files provided

- ftp_credentials.txt, which will contain the credentials you need to use
- yashi_structure.sql, which will contain the stable structure we want you to fill
- Yashi_Advertisers.csv, within the FTP repository, will contain mappings to advertisers
- Data files per day (ex: Yashi_2016-04-27.csv), each will contain data for all advertisers, for a specific date

Uniqueness Key

The following combination of items will define a data row as unique in the different hierarchy levels

- **Cgn-level**: CampaignID
- **Order-level** [cgn_campaign_id] + OrderID
- **Creative-level** [cgn_order_id] + creative_id

Process

You need to fill the data in the tables we've provided based on the data on the FTP, each line represents a single row of data and each row of data contains a CampaignID, OrderID and CreativeID, each of those are data levels.

One data level contains two tables, the regular table (Ex: yashi_cgn) which will contain the level's information, such as the campaign_name, and a data table (ex: yashi_cgn_data) which will contain data **per day** according to the log_date. If you are still confused about the relationship between the tables, please take a look at the foreign keys in the included SQL script.

You will need to split the row into the three data levels yourself, refer to the unicity section of this document. Out of 10 rows, you may for example have 1 unique campaign, 5 unique orders and 10 unique creatives. Each of these will have X rows in their respective _data table where X = the number of days in which they have data. In this example, if you have 10 rows and 1 campaign, the data columns (ex: impression_count) in cgn_data for this respective campaign contain the sum of those 10 rows (because the campaign contains both the orders and the creatives)

Here is an overview of what needs to be done:

1) Download the mapping file and store the advertisers, you'll need them later.
2) Loop through the data files (they should be downloaded in the code, not manually) but **only the ones in the month of May** (ignore the others)
3) For each file, go through every row that have an advertiser **from the list stored in 1),** process these rows as you see fit and insert them into your database.
4) Ensure that your data makes sense, you should have more orders than campaigns, more creatives than orders, and the sum of a data columns in the creative_data should be equal to the sum of that data column in the cgn_data table for this respective campaign.

Tips

1) “Yashi_campaign_id” refers to the external ID found in the file, while “campaign_id” refers to the internal ID in our tables. This logic works for all other ID fields.
2) Make sure your log_date is in the right format (timestamp) and refers to the date column from the CSV file.
3) “Name” refers to the name of the entity, so in _cgn, name refers to the campaign name.
4) Avoid code repetition as much as possible (DRY)
5) The script would be able to be repeated multiple times, when it is repeated, it should not duplicate data.
6) Make it easy to test your code, the configuration should be simple
7) We evaluate code quality, but please keep the code simple, do not create your own framework or an extensive folder structure for this simple task