Tidio Data Engineer - DataOps

Background:

The analysts from your marketing department need data to analyse the efficiency of the marketing campaigns. The data will come from the URLs that our customers visit, the links are built as follows:

https://www.tidio.com/?utm_source=source1&utm_medium=cpc&utm_campaign=999999999&utm_content =9999999utm_term=+fake_term+chat&a_bucket=bucket1&a_type=type1&a_source=source1&a_v=2&a_g_ca mpaignid=999999999&a_g_keyword=+fake_term&a_g_adgroupid=999999999&a_g_creative=999999999

Below, you will find mappings between parts of the URL and the table fields:

URL part	**Column in table**
a_bucket	ad_bucket
a_type	ad_type
a_source	ad_source
a_v	schema_version
a_g_campaignid	ad_campaign_id
a_g_keyword	ad_keyword
a_g_adgroupid	ad_group_id
a_g_creative	ad_creative

Task:

Your task will be to create a simple ETL solution that parses the URL and puts the results in a data warehouse.
The data you will be working on is provided in the file data/raw_ulrs.csv.
The project is meant to be run as a docker-compose project. It runs a Python-based ETL service and a 'Data Warehouse' - PostgreSQL database.
The ETL service (found in the etl/ directory and docker-compose.yml file) is not fully implemented.
This is your task. To do this, you will also need to set up a table **customer_visits** in the Data Warehouse. You can do this inside the ETL service or anywhere else you feel is appropriate. If you decide to use any Python libraries beyond the standard library, please include them in a **requirements.txt** file.

We value:

- Simple but scalable solution
- Detailed documentation/readme with a guide on how to run a solution
- Bonus points for unit tests and logging
- Bonus points for “pythonic” coding style

How to submit a solution:

You can zip the solution up and send it to the recruiter.