Implementation, Limitations, and Next Steps

Implementation:

The code for the newsletter API is stored in callnewsletter.py. The code is organized into a Python class, which in turn can be broken into 7 functions:

1. Sales Messages
2. Media Messages
3. Analytics Messages
4. Handle
5. Get Campaign
6. Get Emails
7. Send Newsletter

Functions 1 - 3 call existing viewsets within the backend code, in order to populate the newsletter. These viewsets include client_manger, api_manager and insights_manager. Each week, functions 1 - 3 randomly selects and runs one of these viewsets. It then reformats the information provided by the viewset, to ensure that the output can be easily read by the end user.

Functions 4 –7 are responsible for retrieving other pieces of information needed to send the newsletter.

Function 4, Handle, sets the date range of the newsletter, comparing data from Monday to Sunday of the previous week, to data from Monday to Sunday 2 weeks before.

Function 5, get campaigns, retrieves the ad campaigns each user has access too.

Function 6, get emails, retrieves the emails of all users subscribed to the newsletter.

Function 7, Send Newsletter, sends the newsletter to the user.

The newsletter API is designed to be called once every week, using a CRON Job.

Limitations:

The main limitation of the newsletter is that much of information provided by it must be hardcoded. These include the images and page links of the newsletter. A good next step would be to dynamically generate these fields.

In addition, the newsletter API takes a substantial amount of time to run. So as long as the runtime does not exceed CRON Job's time limit however, this should not be a major issue.

Finally, the CRON Job call must be written so that the newsletter API is called exclusively on Monday. Otherwise, the data in the newsletter will be based on an incorrect date range.

Next Steps:

The main modification that can be made to the newsletter is to add more information to it. This can be done by calling more viewsets, in addition to the existing viewsets called by the function in callnewsletter.py

In addition, an interface allowing users to subscribe and unsubscribe to the newsletter should be created. This interface can be included in Merln admin, once Merln admin has been completed.

A final modification that can be made is to allow users to select the categories of information they receive each week. For example, in a particular week, a user might be particular interested in sales data. As of right now, the user does not have the option to create a newsletter with exclusively sales data. Instead, the categories of information within the newsletter are hardcoded.