1. I would create a web application, along with a Restful Web Service to receive the http requests webhook. The web application would be responsible for parsing the JSON response from the post to the webhook. I would use Zend PHP to implement a Restful Application, which would parse the response from the webhook, saving the values to a relation MySQL database. Important fields such as the message type, campaign\_id, customer\_id,

Delivery method, from the message event response type, could be stored in the MySQL database. This would be handled in the model, where the message event fields could be stored, along with other fields related to the mailing, such as the date of the monthly mailing, success or failure based on the message event type, along with other types of information such as number of delivered messages, number of failed message delivery (bounced, spam complaint). The controller would be set up to handle the various actions associated with a mail application. The endpoint URL for the Web Service would be set in the app.ini file, and Go allows you to set up the proper routing in its own config files, and allows you to set up a Restful service to handle the actions around the Sparkpost service.

**Which fields seem like they would be important for customer analytics? Which might be important for other reasons? What reasons?**

I feel like the delivery method, and the message event Type are especially important for tracking active members, inactive members, and addresses which bounce the message back, as well as addresses which report the message as span. These are useful for keeping a running tab on active and inactive members. You could also set up accompanying fields in your database related to the mailings. Dates, number of success/failures, reasons, which mailings the user has received. You would be able to easily set up a robust Mailing application using the design I just explained.

1. I would use Object Interfaces to allowing me to define methods between unrelated classes, and Abstract classes to implement default behavior as well shared methods between the different session handling services.
2. It might make sense to add and index to the type and city columns, as these fields will be searched frequently, according to the select statement. However, since type can only 5 different values, many of the rows will be identical, which is definitely a con to creating an index on the city,type fields.

Assuming there are no other related tables or different querying scenarios, do you think mysql is an optimal approach here? Why or why not and what might an alternative be?

I don’t think mysql is the optimal choice. It might make more sense to look into a NoSQL solution, possibly an Object Oriented approach as there are no other related tables, and the table schema for ‘buildings’ looks like it would lend itself well to Object Oriented approach. Each building type would then become an inherited property of building, living in a separate table, depending on your approach. This implementation could speed up searches like the one described in the first question.

1. See “newer\_parser.php” included within the file submission