**Phase 1**

**ETL**

We merged the categories into 23 distinct values and normalized the data.

* For normalization, we split each row into multiple ones to avoid multiple values in single column (1NF)
* We created 4 tables out of the data:
  + yelp\_business.csv to yelp\_business and yelp\_location (separate location entity)
  + yelp\_review.csv to yelp\_review and yelp\_text (since the review’s text is really heavy data)
* Deleted all locations with length of state was not equal to 2 (removing non-US states as well)
* Deleted all locations with non-numeric zipcodes
* Deleted the businesses from yelp\_business table where business\_id was not in yelp\_location table. (i.e. location less businesses)
* Added year column in yelp\_review table to filter the year out of the date column
* Deleted the junk values

These operations were done using SQL Scripts and MS Excel.

**API**

API uses JDBC calls to fetch the data as per the filtration criteria and business needs. The data retrieved is converted to JSON which is sent to the UI to generate the tables. This will assist in future development with NoSQL DB, as all JSONs fetched can be sent to the UI to display the results I consistence with relational database’s results that we achieved in phase 1. Comments are added for explanation within code as well.

**UI**

Vanilla JS and JQuery along with embedded CSS in HTML has been used. The data is sent to rest controllers using POST request to the API and resulting data, in JSON format, is formatted to display tabular results. Comments are added for explanation within code as well.

**Flow**

AJAX request sent from UI to initialize - it loads the static data that is present in dropdown options.

Request reaches Spring-boot based controllers, which fetch the data and convert the result into JSON.

**Cluster**

Galera cluster is running on AWS with 3 nodes.

**Phase 2**

**ETL**

We merged all the tables from our relational structure into single aggregate: yelp\_reviews\_main to perform fetch operations for scenarios where we needed data from all tables. We created another aggregate called yelp\_business\_main that is same as the source business csv file and has merged data from yelp\_business and yelp\_location.

The data was converted to csv and uploaded to MongoDb using mongoimport command, and queries were made to fetch the data required for the business questions (commented the queries in NoSQLData.java).

**API**

Equivalent code for querying the data was written, along with queries for fetching static data for the UI. Helper methods were added to connect the application with MongoDb and to perform appropriate conversion for the framework to display data correctly and in sync with phase 1.

**Flow and UI remains same**

**No Clustering was done**