# **ETL-PROJECT**

***Team Members: Sunwoo Kim***

***Elizabeth Odundo***

### Data Extraction

Original Michelin restaurant data was extracted from Kaggle (<https://www.kaggle.com/jackywang529/michelin-restaurants>).

The data includes one, two, three Michelin star restaurants from the following regions:

Austria, California, Chicago, Croatia, Czech Republic, Denmark, Finland, Greece, Hong Kong, Hungary, Iceland, Macau, Norway, New York City, Poland, Ireland, Rio de Janeiro, Sao Paulo, South Korea, Singapore, Sweden, Taipei, Thailand, Washington DC, and United Kingdom.

The following regions are not included in the dataset:

Belgium, France, Germany, Italy, Japan, Luxembourg, Netherlands, Portugal, China, Spain, and Switzerland.

We loaded the 3 CSV files into Jupyter notebook and generated the data frames for each of them. We then used the latitudes and longitude provided in the original CSVs to make API calls to Google Places and locate the Hotel/Lodgings that were closest to the restaurants. This would help tourists and visitors know where to stay during their trip. The data from the API calls were in JSON format, so we had to parse through the JSON response to access the hotel name. We added this information to the dataset.

Second, we used the URL paths provided in the original datasets to visit each restaurant’s webpage and scrape the information about the specialty dishes that they served. We used splinter and BeautifulSoup to read the HTML response from the data scrape. This information was also added to the datasets.

### Data Transformation

First, wemerged the three data frames (the original/raw, the hotel, and the specialties) so that each star ranking has one data frame with all relevant information that we got from Kaggle, from Google API, and from web scraping. We decided to keep our restaurants in separate data frames based on the Michelin stars ratings.

Next, we decided to drop some columns that we did not need and kept the following: name, city, region, cuisine, URL, hotel name, and specialties. Even though our specialties column contains some N/A values (since some restaurant websites did not list their special dishes), we decided to keep the N/A rows because there is still useful information about each restaurant without the special meals.

Finally, when exporting and reloading data frames, our specialties column was converted from a list to a string. So, we had to convert the string back to a list before uploading our data into MongoDB.

### Data Loading

We chose to load our data into a non-relational database over a relational database for two main reasons.

* First, our data did not require any relationships between tables.
* Second, since we wanted our specialties column to hold list type elements (so users of our database could easily access all special meals), we used MongoDB since SQL columns cannot hold list type elements.

Our final database in MongoDB is structured as follows:

* Database name: michelinDB
* Collections:
  + one\_star
  + two\_star
  + three\_star