For each source, ensure:

* We have all available data for each zip/state.
  + External data not added(time constraint, will explore)
* No non-operator data is included, such as data from clothing stores.
  + Used a local script to remove records associated with non-operator terms, can be improved using ML models.
  + Labeling a subset of the data to train a network to recognize restaurant- related keywords with context/pre-existing trained models
* All duplicates are merged or removed. Merging should combine the same attributes, for instance, types of restaurants (e.g., café, fast food, pizzeria) into a single category.
  + All tables were deduplicated. Checked for key dependency to find a unique field and removed duplication in snowflake.
  + Little confused about combining the same attributes? Aggregated some fields across different tables/records to prevent loss of data

2. Once each source is verified to be correct and complete:

* Merge all data sources and create a unique ID for each restaurant. [Issue - Determine the method for merging data and creating identifiers. Should general logic or ML techniques be applied?]
  + Merged the rhode island details table, grubhub details table, and doordash details table together using name and address hashed as unique IDs, and checked for data discrepancies.
  + Did not merge the menu info file and the review files due to not finding common fields to make composite keys across tables.. For records without an address, will use other unique attributes (e.g., phone number, latitude/longitude) in combination with the name to generate a unique ID.
  + Will test out non-key dependencies and Check functional dependencies in mega table and convert to 3 NF.
* Create new fields and merge the common fields as necessary, ensuring no loss of relevant information.
  + Aggregated some field data in case of duplicate entries, but not merged common fields yet (not checked field dependencies/common. Will do before merging the remaining tables.

For menu data:

* Extract menu data from the link provided in the menu field on Google Maps or from the official restaurant websites. We may need to identify the correct page on the website and extract data from HTML, images, or PDFs.
  + (time) will explore later
* Alternatively, extract the menu data from the images provided by reviewers of the paper menu.
  + Used tesseract with pytesseract in python and requests to scrape text data from images in the menu\_pic\_info file and add the unstructured text into a new field.
  + Will try to structure the menu\_extract field in a similar way to the Menu field in the Store Detail file from doordash and grubhub .(without the image)
  + Will extract text data from the Menu URL field in the details folder from rhode island and then establish consistent structure for menu data.(aggregate with images data extracted to prevent data loss)
  + Then combine menu data from all four files (menu\_pic\_info file from rhode island, menu url field extraction from the rhode island details file, and grubhub’s menu and doordash’s.)
* Combine different menus to create a hierarchy and achieve standardization.
  + My plan:

Menu Data Classification and Hierarchy:

* + - Create a Menu Items Table:
      * MenuItemID: A unique identifier for each distinct menu item.
      * ItemName: The name of the menu item.
      * ItemDescription: A description of the menu item, if available.
      * ItemCategoryID: A foreign key linking to a Menu Categories table.
    - Create a Menu Categories Table:
      * ItemCategoryID: A unique identifier for each menu category (e.g., Appetizers, Entrees, Desserts).
      * CategoryName: The name of the category.
      * ParentCategoryID: Allows for the creation of a hierarchy (null if top-level).
    - Create a Restaurant Menu Table (intermediate table to represent the many-to-many relationship):
      * RestaurantID/uniqueID(in mega table): A foreign key linked to the Restaurant/mega table.
      * MenuItemID: A foreign key linked to the Menu Items table.
      * ItemPrice: The price of the menu item at that particular restaurant.
      * ItemCustomDescription: Any restaurant-specific descriptions for the menu item.
      * ItemImageURL: The URL to the restaurant's image of the item.