**Normalization**

normalize_email(emails):

- Purpose: This function is designed to normalize a list of email addresses.
- Steps:
  - Strip spaces from email addresses.
  - Convert email addresses to lowercase.
  - Remove non-alphanumeric characters except for '.', '_', '-', and '@'.
  - Validate the email addresses using a validation function from the email_validator package.
- Returns: A list of normalized and validated email addresses.

2. normalize_zipcode(zipcodes):
   - Purpose: This function normalizes a list of zipcodes for a business.
   - Steps:
     - Remove non-numeric characters from zipcodes.
     - Ensure that the resulting zipcodes have a length of 5 digits.
   - Returns: A list of normalized zipcodes.
3. normalize_dataframe(df):
   - Purpose: This function serves to normalize various columns within a given DataFrame, including "Email," "Phone Number," "Zipcode," "BusinessName," "Website," and "Address."
   - Steps:
     - It uses the previously defined normalization functions to process these columns.
   - Returns: The data frame after applying normalization to the specified columns.
4. normalize_us_phone_number(phones):
   - Purpose: Designed to process U.S. phone numbers, this function extracts all numeric digits from phone numbers and formats them according to the standard U.S. phone number format.
   - Returns: A list of normalized U.S. phone numbers.
5. standardizeName(names, is_sos=False):
   - Purpose: This function standardizes business names by:
     - Converting the entire name to lowercase.
     - Substituting '&' with 'and'.
     - Removing characters not found in the set [a-z, whitespace, '-'].
     - Trimming extra spaces.
   - Optional SOS (Secretary of State) Standardization: If the is_sos parameter is set to True, it performs additional standardization specific to Secretary of State purposes.
   - Returns: A list of standardized business names.
6. normalize_name(name):
   - Purpose: A helper function that replaces specific business-related abbreviations in a name, such as "llc."

7. normalize_address_i18n(addresses):
    - Purpose: Utilizes the i18naddress library to normalize and structure address data.
    - Returns: A list of structured and normalized addresses.
8. normalize_url(urls):
    - Purpose: This function normalizes URLs by:
        - Converting the entire URL to lowercase.
        - Eliminating spaces.
        - Appending 'http://' at the beginning if no protocol (http or https) is present.
    - Returns: A list of normalized URLs.

Additionally, the code includes the initialization of Pandarallel for parallel processing and sets up logging to capture detailed logs about warnings, errors, and other critical information.

These functions are designed to help standardize and clean various types of data commonly found in business records, making them more consistent and suitable for analysis or storage.

**1. Google Maps Class:**

- This is a Python class named google_maps, which appears to be designed to interact with the Google Places 2.0 API.
- It has class-level attributes api_key and place_ids to store the API key and a list of place IDs.
- The __init__ method is the constructor for this class. It takes an API key as a parameter and initializes the api_key attribute.

2. **find_place Method:**
    - This method is used to find places based on a provided input (e.g., the name of a business).
    - It sends a POST request to the Google Places API with the input and retrieves the place IDs associated with the input.
    - The place IDs are extracted from the API response and stored in the place_ids attribute.
    - If no places are found, it logs an error and returns None.
3. **details Method:**
    - This method is used to retrieve detailed information about places based on the place IDs obtained from the find_place method.
    - It iterates through the list of place IDs and sends a GET request to the Google Places API to retrieve details about each place.
    - The details include the name, address, phone number, and website URL.

    - The retrieved details are processed, normalized (e.g., address normalization), and stored in a dictionary.
    - The method returns a dictionary with the retrieved information.
    - If no place IDs are found or if there is an issue with the API request, it logs an error and returns None.
4. **google_validation Function:**
    - This function is designed to validate and enhance information in a Pandas DataFrame by using the Google Places API.
    - It expects a DataFrame with specific columns, such as "BusinessName," "Address," "Phone," "Website," and various other columns related to validation.
    - It first checks if the required columns exist in the data frame. If not, it logs an error and raises an exception.
    - It creates an instance of the google_maps class, passing the API key obtained from the environment variables.
    - The function then determines a search list based on the available information (e.g., "BusinessName" and "Address") and iterates through this list to find information using the Google Places API.
    - If information is found, it updates the DataFrame with the retrieved information.
    - It also performs some deduplication and checks for correctness in the retrieved data (e.g., matching business names).
    - The function returns the updated DataFrame.
5. **Main Block:**
    - The code in the main block is executed when the script is run directly (not imported as a module).
    - It initializes a sample DataFrame (sos_output) with mock data to demonstrate how the google_validation function works.
    - The google_validation function is applied to each row of the DataFrame using the apply method, and the updated DataFrame is printed.

a part of a data processing pipeline that interacts with the Google Places API to validate and enhance business information in a DataFrame. It finds detailed information about businesses based on their names, addresses, and other details and updates the DataFrame with the retrieved information. The code also includes error handling and logging for robustness.