Data Processing file

1. join_dataframe_firmid(*data_frames: pd.DataFrame) -> pd.DataFrame:
    - This function takes a variable number of DataFrames as input, represented by *data_frames.
    - It merges these DataFrames on the "firm_id" column using an outer join, combining them into a single DataFrame.
    - Duplicate columns are removed to avoid redundancy.
    - The function then filters out businesses that are not incorporated in Minnesota ("MN").
    - It creates a new column called "Address" by combining address-related columns: "address_1", "address_2", "city", and "zip.".
    - Finally, it renames columns for consistency and returns the resulting DataFrame.
2. extract_data(file_path: str) -> pd.DataFrame:
    - This function reads data from a specified file path into a DataFrame using the pd.read_csv function.
    - If successful, it returns the DataFrame containing business information from the file; otherwise, it returns None.
3. concat_address(row: pd.Series) -> pd.Series:
    - This is a helper function that takes a row from a DataFrame as input.
    - It concatenates the "Address 1" and "city" columns to create an "Address" column, which represents the full address of the business.
    - If either the "Address 1" or "city" columns contain missing values (NaN), it returns np.nan for the "Address" column.
4. filter_dataframes(df: pd.DataFrame) -> (pd.DataFrame, pd.DataFrame):
    - This function filters a given data frame based on certain conditions:
        - Rows are considered invalid if the "BusinessName" column is null or empty.
        - Rows are considered valid if they have a non-empty "BusinessName" and at least one other non-empty column.
    - It returns two data frames: one containing valid rows and another containing invalid rows.
5. address_match_found_sos(historical_address, found_address):
    - This function compares historical and found addresses to determine if they match.
    - It returns 1 for a match, 2 for matching cities (even if zip codes don't match), and 0 for no match.
    - If found_address is not a string (e.g., if it's missing or None), it returns 0.
6. address_match_found(historical_address, found_address):
    - Similar to the previous function, this one compares historical and found addresses.
    - It returns 1 for a match, 2 for matching cities (even if zip codes don't match), and 0 for no match.
    - If found_address is not a string (e.g., if it's missing or None), it returns 0.

7. is_same_business(historical_name: str, new_name: str, threshold: int = 80, business_type_historical: str = None, business_type_new: str = None, is_SOS: bool = False) -> bool:
    - This function checks if a new business name is essentially the same as a historical one.
    - It uses fuzzy string matching to make this determination, and you can specify a threshold for the match (the default is 80).
    - It also considers business types and whether the names are from the Secretary of State (SOS). If business types don't match, it returns false.
    - The function returns True if any of the fuzzy matching ratios (token_sort_ratio, ratio, partial_ratio) exceed the specified threshold.
8. normalize_email(email: str) -> str:
    - This is a helper function that normalizes an email address to fit certain expectations:
        - It strips leading and trailing spaces from the email.
        - It converts all characters to lowercase.
        - Removes non-alphanumeric characters except for '.', '_', '-', and '@'.
        - Validates the email using the email-validator library and returns the normalized valid email.
9. get_valid_businesses_info(file_path: str) -> pd.DataFrame:
    - This function reads data from a specified file path into a DataFrame.
    - It filters the DataFrame to keep only rows where the "active" column is "true" (case-insensitive).
    - If there are no active businesses, it returns an empty DataFrame.
    - If an error occurs during reading or filtering, it returns None.
10. remove_duplicates_and_nans(row: pd.Series) -> pd.Series:
    - This function removes duplicates and NaN values from columns containing lists in a given row of a DataFrame.
    - It iterates through each column with list values (e.g., BusinessNameUpdate, PhoneUpdate) and removes duplicate items within each list.
    - It also removes any occurrences of "nan" (representing missing values) from the lists.
    - The cleaned row is then returned as a series.