Documentation: Geographical Data Processing

## Objective

This document outlines the process of geospatial data processing for geographical data related to food tech companies retrieved from Crunchbase. The process involves matching and filling missing or inaccurate postal codes, latitude, and longitude values using a combination of exact and partial matching techniques.

## Inputs

*Main Files*

1. **foodtech.csv** or **CB**
   1. output of create\_foodtech.py
   2. contains the agro-foodtech universe of organizations
      1. Crunchbase bulk download file *organizations.csv* filtered for entries classified with industries or category\_groups\_list
         * Agriculture and Farming
         * Food and Beverage
   3. Relevant columns
      1. uuid
      2. country\_code
      3. state\_code
      4. region
      5. city
      6. address
      7. postal\_code.
2. Mapping Files
   1. External database
      1. **geonames.json** or CODES
         * JSON file with geographical data including postal codes, latitude, and longitude
   2. Previous mapping work by the team
      1. **asia\_cities\_mapped.csv**
         * uses Startup Asia legacy database **merged data v.xlsx**
      2. **world\_cities\_mapped.csv**
         * uses **food tech city data cleaned.xlsx**

*Supporting Files*

* + - * Countries and ISO Codes
        1. **isocodes.csv**: contains ISO2 and ISO3 country codes.
        2. **countries.csv**: includes country names and their corresponding ISO3 codes.

## Processes and Steps

PART 1: FINDING EXACT MATCHES

* Using the 3 input mapping files, exact matches of entries in the city column of the Crunchbase dataframe are matched

|  |  |  |
| --- | --- | --- |
| **CRUNCHBASE (CB)** |  | **CODES** |
| **Changdong District** | **Changdong District** |
|  |  |  |
| **Changdong** |  | **Changdong District** |

*Database 1: geonames.csv*

Crunchbase city and region names are are directly mapped to three combinations of city-region column pairs from the geonames database and these pairs’ corresponding postal code, latitude, and longitude for each matching entry.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CRUNCHBASE (CB)** | | |  | **CODES** | | | | |
| **City** | **Region** | **Postal Code** | **Place\_Name** | **Admin\_Name1** | **Postal Code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Admin\_Name2** | **Admin\_Name1** | **Postal Code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Admin\_Name3** | **Admin\_Name1** | **Postal Code** | **Latitude** | **Longitude** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MERGED DATABASE** | | | | | |
| **City** | **Region** | **PC\_CB** | **PC\_Filled** | **Latitude** | **Longitude** |

*Database 2: asia\_cities\_mapped.csv*

Crunchbase city names are first mapped to city names from Startup Asia database. The newly mapped city names are then mapped to the geonames database.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STEP 1** | | | | | **STEP 2** | | | | | |
| **CRUNCHBASE** | | |  | **ASIA CITIES** | |  | **CODES** | | | |
| **City** | **Region** | **Postal Code** | **Old\_city** | **New\_city** |  | **Place\_Name** | **Postal code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Old\_city** | **New\_city** |  | **Admin\_Name2** | **Postal code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Old\_city** | **New\_city** |  | **Admin\_Name3** | **Postal code** | **Latitude** | **Longitude** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MERGED DATABASE** | | | | | |
| **City** | **New\_city** | **PC\_CB** | **PC\_Filled** | **Latitude** | **Longitude** |

*Database 3: world\_cities\_mapped.csv*

Crunchbase city names are first mapped to city names from World cities database. The newly mapped city names are then mapped to the geonames database. Entries that were mapped to World Cities but not to geonames are then used to fill data on longitude and latitude.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STEP 1** | | | | | **STEP 2** | | | | | | | |
| **CRUNCHBASE** | | |  | **WORLD CITIES** | | | |  | **CODES** | | | |
| **City** | **Region** | **Postal Code** | **Old\_city** | **City\_mapped** | **LAT** | **LONG** |  | **Place\_Name** | **Postal code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Old\_city** | **City\_mapped** | **LAT** | **LONG** |  | **Admin\_Name2** | **Postal code** | **Latitude** | **Longitude** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| **City** | **Region** | **Postal Code** |  | **Old\_city** | **City\_mapped** | **LAT** | **LONG** |  | **Admin\_Name3** | **Postal code** | **Latitude** | **Longitude** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STEP 3** | | | | | | |
| **MERGED DATABASE** | | | | | **WORLD CITIES** | |
| **City** | **City\_mapped** | **PC\_Filled** | **Latitude** | **Longitude** | **LONG** | **LAT** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MERGED DATABASE** | | | | |
| **City** | **City\_mapped** | **PC\_Filled** | **Lat\_Filled** | **Long\_Filled** |

2. Partial Matches:

- Identify and select the best partial matches for city names.

- Fill missing postal codes, latitude, and longitude values using partial matching.

3. Existing Postal Codes:

- Use existing postal codes from the dataset to fill missing postal code values.

4. Saving and Reporting:

- Save the processed data as CSV files.

- Generate reports on the count of missing postal codes and latitude/longitude values per country.

---

---

Output:

1. Processed CSV Files: Saved in the '0508' directory with country-specific filenames.

2. NaN Counts CSV Files: Reports on the count of missing postal codes and latitude/longitude values per country.

---

Execution:

1. Load input data: Read CSV files and JSON file into pandas DataFrames.

2. Perform exact matching: Match company data to geonames data using exact matching techniques.

3. Perform partial matching: Identify and fill missing postal codes, latitude, and longitude values using partial matching.

4. Fill missing data with existing postal codes: Utilize existing postal codes from the dataset to fill missing values.

5. Save processed data: Save the processed data and generate reports on missing data counts.

---

Conclusion:

This geospatial data processing workflow effectively matches and fills missing or inaccurate geographical data for food tech companies, ensuring improved data accuracy and completeness for further analysis and use cases.

---

Note:

- Ensure all input files are present in the specified directories before executing the code.

- Review the saved CSV files and reports for insights into missing data and processed data quality.