Data Extraction and Analysis Project

1 Introduction

This document serves as a comprehensive guide for the data extraction project, which aims to gather company information from the web source https://industrie.de/firmenverzeichnis/. The task involves extracting valuable details about companies from webpages that include a specific section labeled **"Daten und Kontakte."** Links to these pages are provided on the main website mentioned above. The extracted data includes the company name, the URL to the webpage containing the **"Daten und Kontakte"** section, and the city where the company is located. Additionally, detailed information from the **"Daten und Kontakte"** section, such as contact details and key company information, is also collected.

2 Data Extraction Process

2.1 Webpage Structure

The webpage structure is analyzed to identify the relevant information. The link to webpage that contains "Daten und Kontake" is located within a div element with the CSS class "infoservice-entry-holder". The mentioned div also contains company name and city where the company is located, which is the source of Initial Extraction.

2.2 Initial Extraction

- Company Name
- URL to "Daten und Kontake" Section
- City

The above information is extracted directly from the parent div with the class "infoservice-entry-holder".

2.3 Extracting Information form “Daten und Kontake“ Section

To extract each company’s information, iteration through the list of links is used to extract detailed information from the "Daten und Kontakte" section. This section's data is structured using dl, dt, and dd HTML elements.

- dt contains icons representing data categories.
- dd contains the corresponding data values.
- Icon CSS class labels are extracted as data names for the values in dd elements and are later converted into human-readable names.

2.4 Address Handling

The address information consists of multiple parts separated by br HTML elements. Extracting this information without separator will have no space or any other separator between different parts of address. These parts are concatenated with spaces as separators.

2.5 Data Storage

The extracted data is stored in a CSV file named "companies_data.csv" for further processing and analysis.

3 Data Preprocessing

In certain columns, we encounter sparse data with a significant percentage of null values. To make the code work more efficient, the data processing function is only invoked when there's actual information present, not when it has null value.

3.1 Duplicate Removal

Duplicate records are removed from the dataset to ensure data accuracy.

3.2 Handling missing values:

To handle missing values in the dataset, they are filled in a Pandas DataFrame (df) with default null values based on column data types. When there's no reasonable way to calculate missing values from existing data, this step ensures data consistency by replacing them with null values.

3.3 Address Cleanup

Company names are removed from address fields when present, as some addresses include the company name.

3.4 Mobile and Fax Number Formatting

Non-numeric characters, except "+" are removed from mobile numbers. Numbers lacking the country code are assumed to be in Germany, because the website's domain is .de, therefore "+49" are prefixed with it for consistency.

3.5 Number of Employees and Year of Foundation

- Number of employees and the year of foundation are extracted from the respective text columns.
- The extracted data is cleaned and converted to integers and stored in respective columns and column data type is changed to integer.

3.6 Website Standardization

Website URLs are processed to remove the protocol (http/https) and ensure consistent formatting by adding "www" where not present.

3.7 Enhancements to 'net_assets' Column

Processing this column posed a challenge due to the presence of both year and net asset values, which are numerical data. Furthermore, it included numbers written out in words and various currency representations. To standardize the data, the approach employed was to initially extract the 4-digit year. Once the year is extracted, it is subsequently removed, enabling the isolation and extraction of the net asset value.

This column within our dataset contains diverse information related to a company's financial worth. In certain records, it includes details like the number of the company's customers, and for some records, it even specifies the year in which the net assets were calculated. To ensure data consistency and improve usability, the following tasks are executed:

i. Edge cases where the 'net_assets' value represents the number of customers are identified and removed.
ii. For the remaining cases, text-based values like '3 Million' are converted into numeric values, for instance, '3,000,000'. This ensures consistency and numerical accuracy.
iii. To standardize the currency format, unique representations are used. This reduces inconsistencies. The resulting data is stored in a new column called "net_assets_currency".
iv. To maintain transparency, a new column named 'net_assets_year' is created specifically to store the year in which the 'net_assets' information was calculated. This ensures that the temporal aspect of this data is well preserved and easily accessible for analysis.

4 Data Analysis

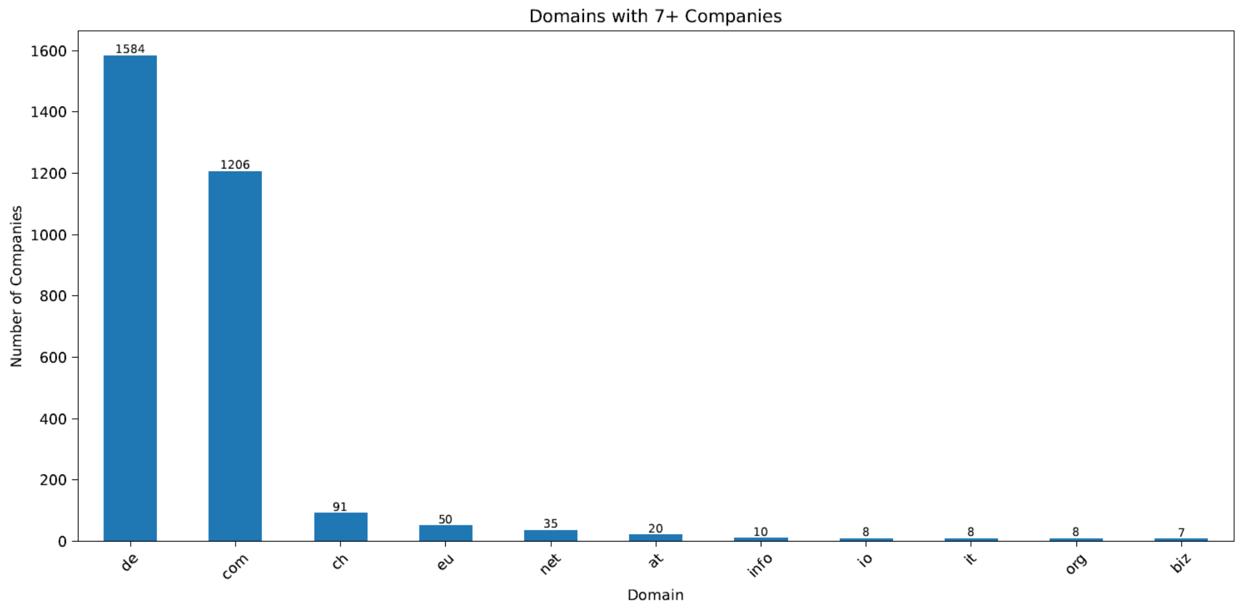
Consistency of net asset values is ensured by converting them into a common currency when calculating summary statistics. This conversion is determined by the exchange rate in effect on the date. This consistent currency conversion facilitates meaningful comparisons and analysis across the entire dataset, allowing us to derive accurate insights from the data.

4.1 Summary Statistics

A Python script called 'summary_stats.py' computes summary statistics for numeric columns ('year_founded,' 'employees,' 'net_assets,' 'net_assets_year') and it further reports the number of null/empty values. It is important to note that a significant percentage (97%+) of records in these columns contain null values. Therefore, any analysis and visuals may not provide insights for the entire dataset.

4.2 Company Distribution by Domain

A Python script named "count_per_domain_visual.py" calculates the number of companies per domain using the website column. Visualizations are generated using Matplotlib and saved as a PDF file named "companies_per_domain.pdf" in the visual directory.

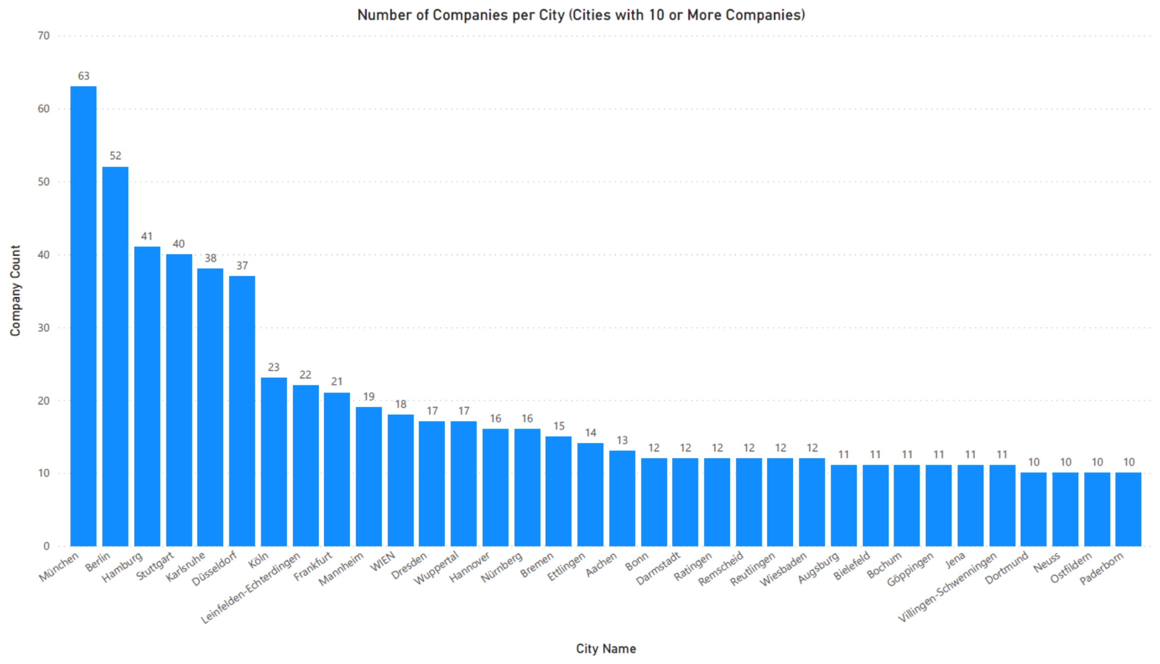


4.3 Additional Visualizations

Further visualization is done using Power BI, the visuals are included as a single PDF in the visual directory. They are as following:

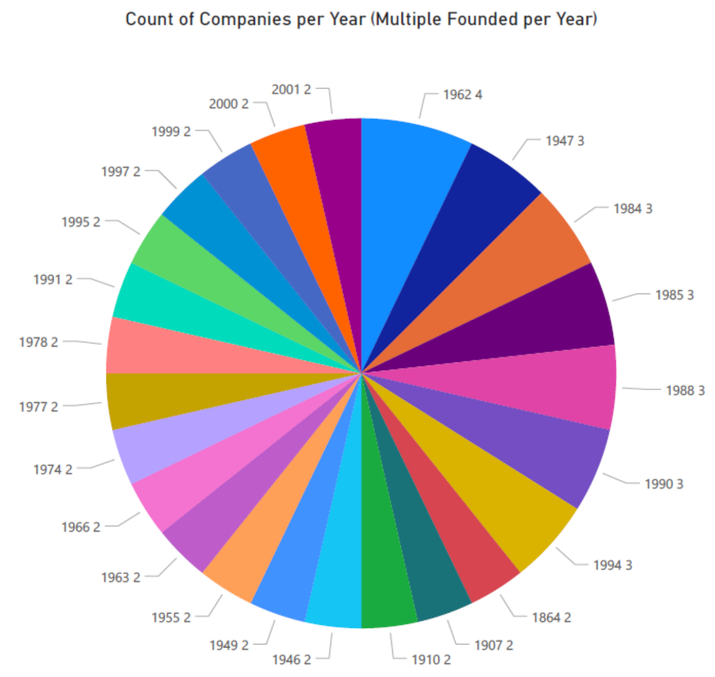
4.3.1 Cities with Many Companies:

We've visualized which cities host a significant number of companies, specifically those with ten or more. This helps us understand where businesses tend to cluster, which can be handy for regional business planning.



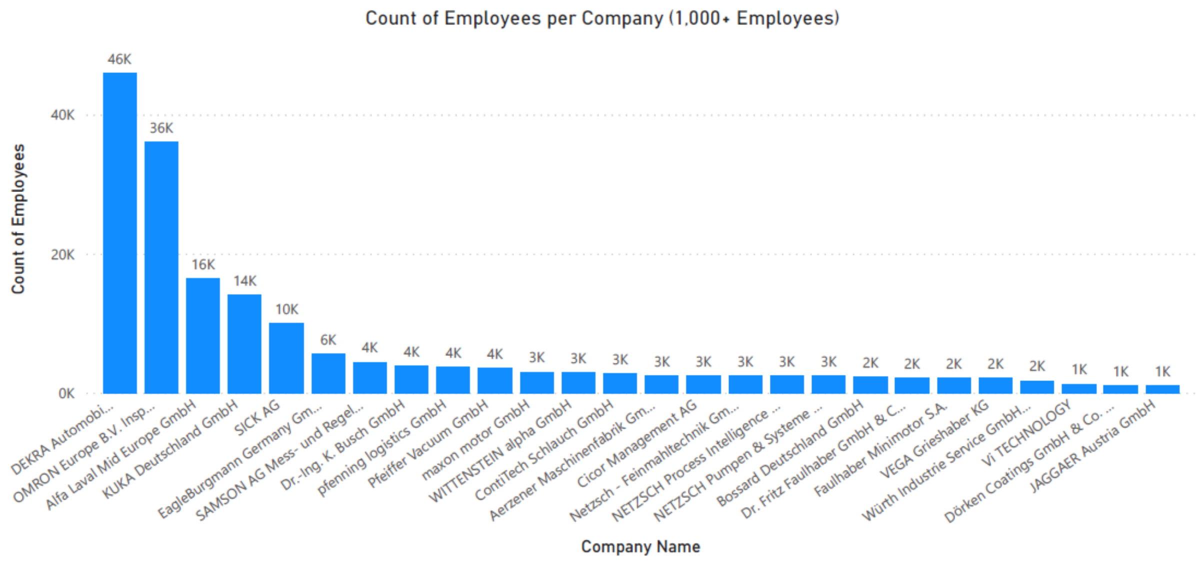
4.3.2 Company Founding Years:

This visualization shows the years when companies were founded, highlighting when multiple companies share the same founding year. It's a great way to see historical trends in business creation.



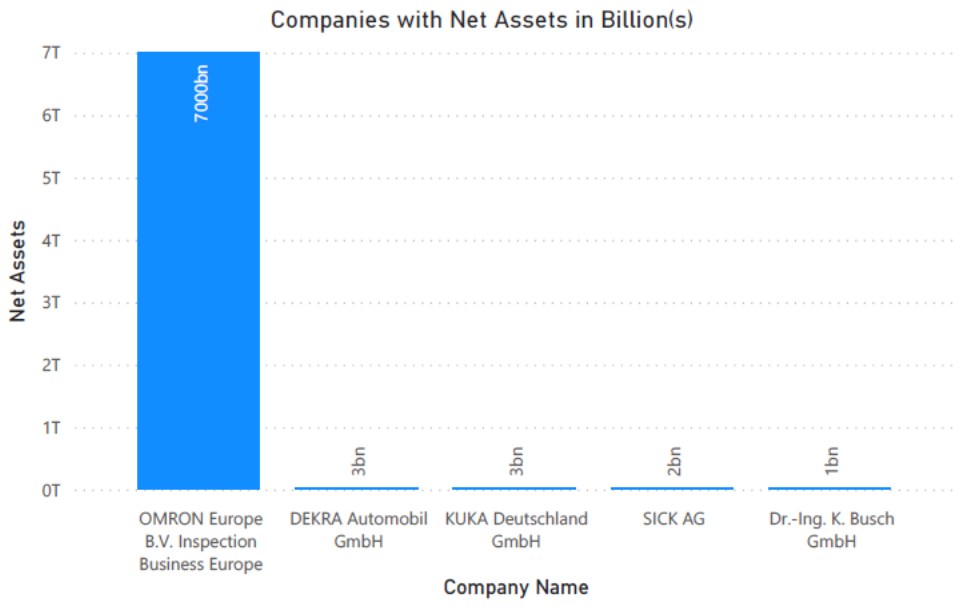
4.3.3 Larger Companies:

I have looked at companies with over 1,000 employees to see how many fall into this category. It helps us gauge the presence of larger corporations in our dataset, which can be important for various analyses.



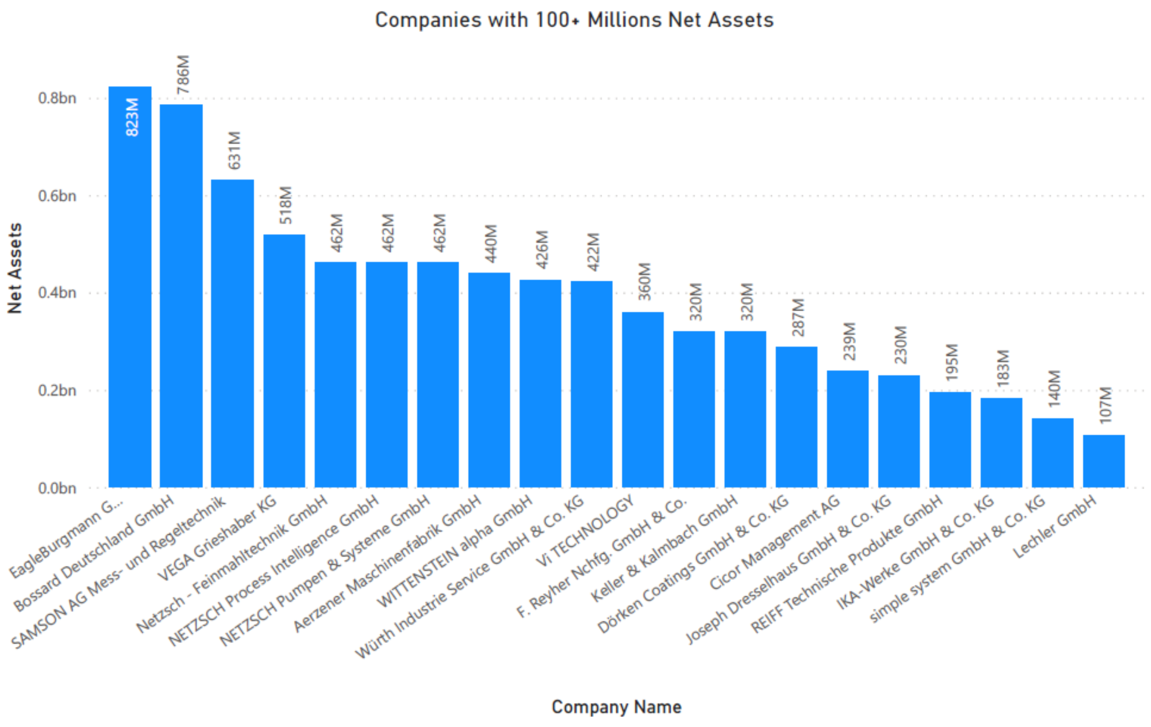
4.3.4 Billion-Dollar Assets:

This chart identifies companies with net assets in the billion(s) range. It's useful for spotting financially robust companies, which might interest investors or partners.



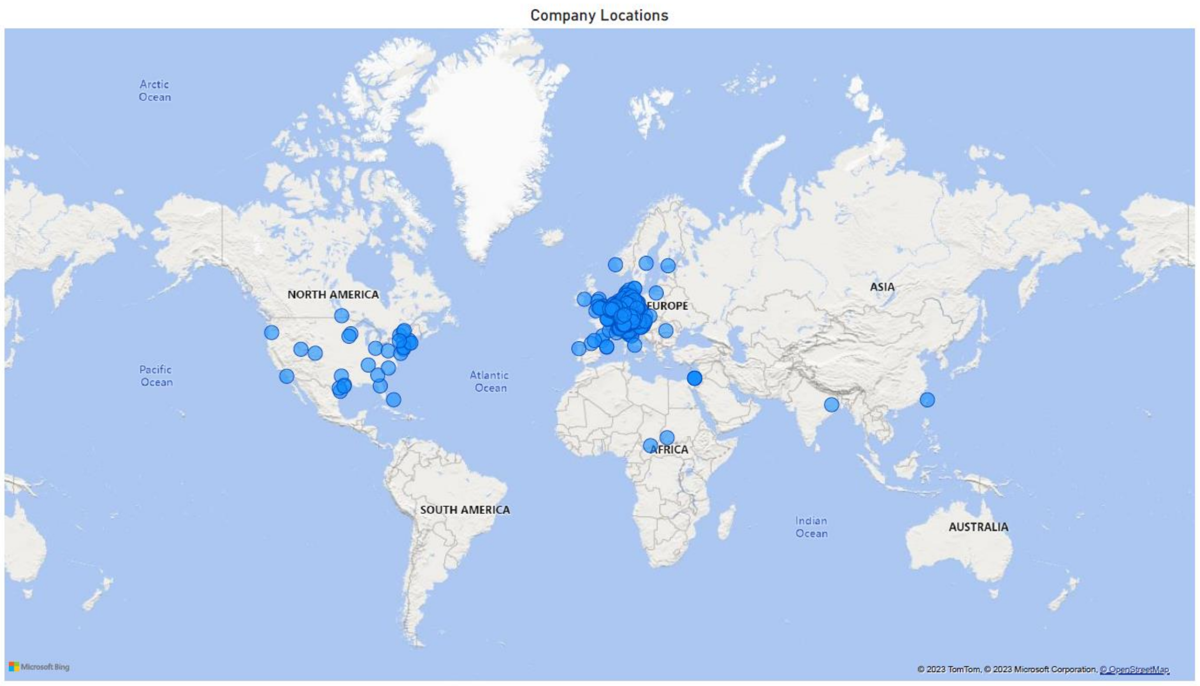
4.3.5 Million-Dollar Assets:

Similar to the previous one, but focused on companies with net assets exceeding 100 million. It gives a broader perspective on financially strong companies for various stakeholders.



4.3.6 Global Company Distribution Map

In map visuals, the locations of companies are displayed on a world map. Since we have city names as our data, these visuals allow us to see the concentration of companies in different parts of the world. By mapping out the cities, we gain insights into where most of the companies in our dataset are located, helping us understand their geographical distribution.



These visualizations add valuable insights to our dataset, making it easier to understand the distribution and characteristics of the companies in the dataset.

5 Running the project:

- Install Libraries: Use *"pip install -r requirements.txt"* to install the necessary Python libraries.
- Extract Data: Run 'extract_data.py' to get data in 'companies_data.csv'.
- Clean Data: Use 'preprocess_data.py' for data cleaning. Find results in 'cleaned_companies_data.csv'.

For more information on files and scripts, check the "File Structure" section.

6 File Structure

Inside the main folder, "web_scraping_task," everything is neatly organized to make our work smoother. Here's a closer look at what's in there:

6.1 requirements.txt:

This file lists all the required Python libraries and their specific versions. It ensures consistency and reproducibility in the development environment, making it easier for others to set up their environments.

6.2 scraping_functions.py:

This Python script contains a collection of functions designed to extract links and information from webpages efficiently. These functions are essential components of the web scraping process.

6.3 extract_data.py and companies_data.csv:

The "extract_data.py" script is responsible for the data extraction from web sources. The extracted data is stored in "companies_data.csv," a CSV file that serves as the primary dataset for further processing.

6.4 preprocess_functions.py:

This script contains a number of functions that are used for data preprocessing tasks. These functions are essential for cleaning and preparing the data for analysis.

6.5 preprocess_data.py and cleaned_companies_data.csv:

"preprocess_data.py" is the script responsible for performing the data preprocessing steps described in the documentation. The cleaned and refined data is stored in "cleaned_companies_data.csv". This file is used for further analysis.

6.6 summary_stats.py and summary_statistics.csv:

"summary_stats.py" is a Python script that calculates and generates summary statistics for the dataset. The results are stored in "summary_statistics.csv". They provide valuable insight into the numeric columns of the dataset.

6.7 count_per_domain_visual.py:

This script calculates the number of companies per domain by analyzing the website column. It uses Matplotlib to create visualizations, and the resulting graphs are saved as a PDF file named "companies_per_domain.pdf" in the visual directory.

6.8 Visual Directory:

This directory stores the visual outputs generated during the analysis phase. Visualizations produced by "count_per_domain_visual.py" are stored here. The visuals that are created using Power BI are also stored as a single PDF file in this directory.

This organized structure makes it easier for everyone to work together, find what they need, and keep everything running smoothly.