Web Scraping Grocery Stores In New York From YellowPages.com

Project Submitted to the

IT VEDANT INSTITUTE, THANE

Data Science & Data Analytics With AI

Python-Web-Scrapping Project

BY

Mr. Aditya Patade

Under the Guidance of

Mr. Sameer Warsolkar

Overview

Grocery Stores In New York Web Scraping project involves the extraction of detailed information about grocery stores in New York City from the Yellow Pages website using web scraping techniques. The primary objective is to gather structured data including store names, types, addresses, and contact numbers. The extracted data is then cleaned and organized for further analysis or use in other applications.

Objectives:

- To extract structured information about grocery stores from the Yellow Pages.
- To clean and format the extracted data for usability.
- To provide a structured dataset that can be used for various analytical or business purposes.

Key Features:

Web Scraping Implementation:

HTML Content Fetching: Uses the requests library to send HTTP requests and retrieve HTML content from the Yellow Pages.

HTML Parsing: Utilizes BeautifulSoup to parse the retrieved HTML content, allowing for easy navigation and data extraction from the document object model (DOM).

Data Extraction:

Store Names: Extracts the names of grocery stores using specific CSS selectors to target relevant HTML elements.

Store Types: Captures the categories or types of grocery stores, applying regular expressions to ensure correct spacing and text formatting.

Street Addresses: Extracts the street addresses of the stores, cleanses the data using regular expressions to standardize the format.

Locality: Gathers information about the locality or neighbourhood where each store is located.

Contact Numbers: Extracts phone numbers and filters them using regular expressions, particularly focusing on identifying numbers that meet specific criteria, such as starting with a particular digit.

Data Cleaning and Validation:

Regular Expressions: Implements regex patterns to clean and validate the extracted data, ensuring consistency and removing unnecessary whitespace or formatting issues.

Consistent Data Formatting: Ensures all extracted information is uniformly formatted for ease of analysis and integration into other systems.

Data Organization:

Pandas Data Frame: Structures the cleaned data into a Pandas Data Frame, which facilitates easy manipulation, analysis, and exporting of the data to various formats.

Outline

From this site, we are going to grab the following information:

1. Store Name
2. Store Type
3. Street Address
4. Locality
5. Contact No

Steps

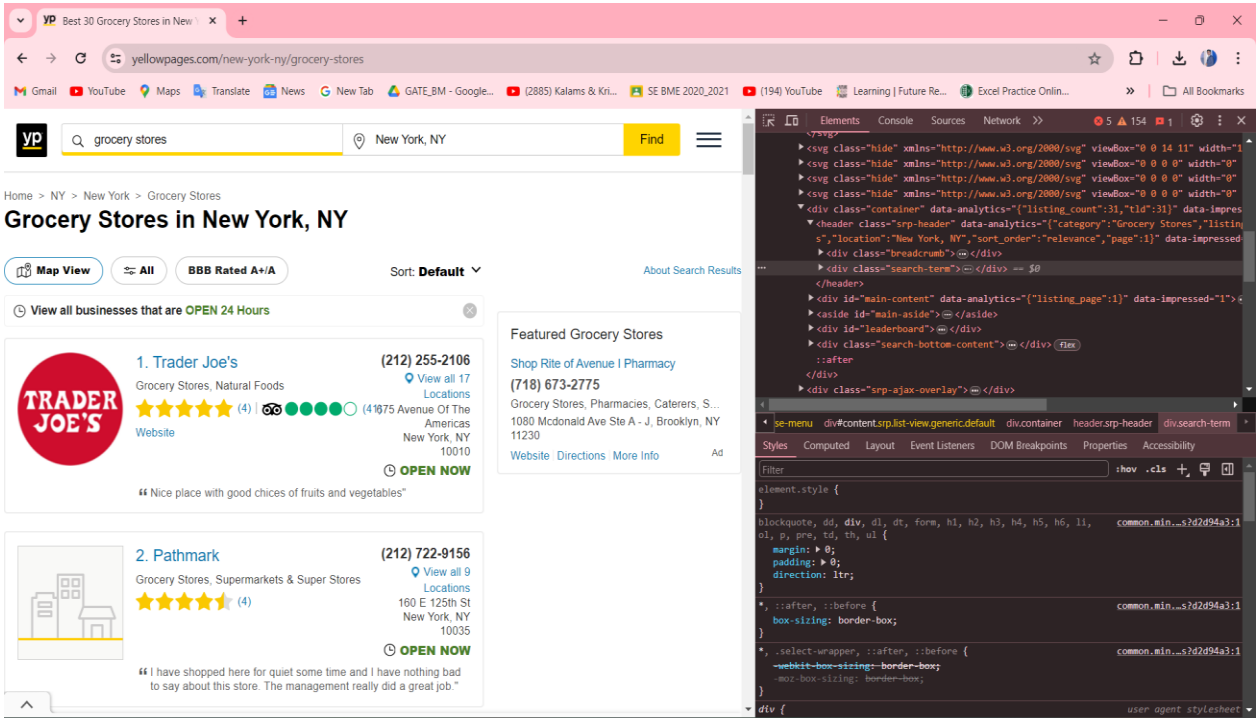
Choose the Website and Webpage URL:

➢ The first step is to select the website you want to scrape. We will try to extract data of Grocery Stores In New York from the YellowPages.com.

1. Inspect the website:

   Now the next step is to understand the website structure. Understand what the attributes of the elements that are of your interest are. Right click on the website

to select “Inspect”. This will open HTML code. Use the inspector tool to see the name of all the elements to use in the code.



2. Installing the important libraries:

   Python has several web scrapping libraries. We will use the following libraries:

   - Requests – for making HTTP requests to website
   - Beautiful Soup – for parsing the HTML code
   - Pandas – for storing the scraped data in data frame
   - In Python, the built-in re module provides support for regular expressions (regex). This module allows you to work with regular expressions for searching, matching, and manipulating text.

3. Write the Python source code:
   We’ll write the main python code. The code will perform the following steps:

   - Using requests to send an HTTP GET requests
   - Using Beautiful Soup to parse the HTML code

- Extracting the required data from the HTML code
- Regular expressions for searching, matching, and manipulating text.
- Store the information in a pandas Data Frame

4. Exporting the extracted data:

   We'll export the data as a CSV file. We will use the pandas library.

5. Benefits:
   - Access to valuable data for analysis or research.
   - Automation of data collection, saving time and effort.
   - Stay up to date with changes on the target websites.

6. Risk:
   - Legal issues related to web scraping.
   - Technical challenges due to website changes

Web Scraping Code:

```
import requests

from bs4 import BeautifulSoup

import re
```

import requests: This line imports the requests module, which is a popular Python library for making HTTP requests. With requests, you can easily send HTTP requests to web servers and work with their responses.

from bs4 import BeautifulSoup: This line imports the BeautifulSoup class from the bs4 module. BeautifulSoup is a Python library for parsing HTML and XML documents. It provides a convenient way to navigate and manipulate the parsed tree structure of HTML/XML documents.

import re: This line imports the built-in re module, which provides support for working with regular expressions in Python. Regular expressions are powerful tools for pattern matching and text manipulation.

page = requests.get("https://www.yellowpages.com/new-york-ny/grocery-stores")

soup = BeautifulSoup(page.content,'html.parser')

page = requests.get("https://www.yellowpages.com/new-york-ny/grocery-stores"):

This sends a GET request to the specified URL, "https://www.yellowpages.com/new-york-ny/grocery-stores", using the get() function from the requests module. It retrieves the HTML content of the web page.

soup = BeautifulSoup(page.content, 'html.parser'):

This creates a BeautifulSoup object called soup. It represents the parsed HTML content of the web page. The first argument, page.content, is the HTML content fetched from the web page. The second argument, 'html.parser', specifies the parser to use for parsing the HTML content. In this case, we are using the built-in HTML parser provided by BeautifulSoup.


```
def clean_text(text):
    text = re.sub(r'\s+', ' ', text).strip()
    return text


# Extract shop names
Shop_name_tags = soup.select(".business-name")
Shop_name = [clean_text(i.get_text()) for i in Shop_name_tags]
print(Shop_name[:30])
```

def clean_text(text)::

This line defines a function named clean_text that takes a string text as input.

text = re.sub(r'\s+', ' ', text).strip():

Inside the function, this line uses the re.sub() function from the re module to substitute one or more whitespace characters (\s+) with a single space ' '. It then calls the strip() method to remove leading and trailing whitespace from the resulting string. This effectively removes extra spaces, line breaks, and tabs from the text.

return text:

This line returns the cleaned text after removing extra spaces, line breaks, and tabs.

Shop_name_tags = soup.select(".business-name"):

This line selects all elements with the class "business-name" from the parsed HTML represented by the soup object. It uses the select() method provided by BeautifulSoup for this purpose.

Shop_name = [clean_text(i.get_text()) for i in Shop_name_tags]:

This line iterates over each element in Shop_name_tags and retrieves the text content of each element using the get_text() method. It then applies the clean_text() function to each text content to remove extra spaces, line breaks, and tabs. The cleaned text is stored in a list named Shop_name.

print(Shop_name[:30]):

This line prints the first 30 elements of the Shop_name list, containing the cleaned shop names extracted from the web page

```
[43]: def clean_text(text):
          # Remove extra spaces, line breaks, and tabs
          text = re.sub(r'\s+', ' ', text).strip()
          return text

      # Extract shop names
      Shop_name_tags = soup.select(".business-name")
      Shop_name = [clean_text(i.get_text()) for i in Shop_name_tags]
      print(Shop_name[:30])
```

```
["Trader Joe's", 'Pathmark', 'Whole Foods Market', 'Pioneer Supermarket', 'The Food Emporium', 'Morton Williams Supermarkets', 'Big Market', 'Pioneer Supermarket', 'Food City', 'New Lung Hing Market Incorpora', 'Tak Yee Fong Inc', 'Ammirati', 'Central Market', "D'agostino's", '765 Sixth Avenue Market', 'Well Green Market', 'Red Apple Supermarkets', 'Lb Deli Grocery', 'Fine Fare Supermarket', 'Manhattan Food Market', 'Urban Market', 'Fine Fare Supermarket', 'Chang Shun Market Inc', 'International Fine Fare Supermarkets', 'Jubilee Marketplace', 'Amish Market Tribeca', 'City Acres Market', "Smith's Food & Drug", 'Wegmans', 'Lifethyme']
```

```
Shop_types = soup.select(".categories")

Shop_type = []

for i in Shop_types:

    Shop_type.append(clean_text(i.get_text()))

print(Shop_type[:30])
```

Shop_types = soup.select(".categories"):

This line selects all elements with the class "categories" from the parsed HTML represented by the soup object.

Shop_type = []:

This initializes an empty list named Shop_type to store the extracted shop types.

for i in Shop_types::

This initiates a loop over each element in the Shop_types list, which contains elements with the class "categories".

Shop_type.append(clean_text(i.get_text())):

Inside the loop, this line retrieves the text content of each element using the get_text() method. It then applies the clean_text() function to remove extra spaces, line breaks, and tabs. The cleaned text is appended to the Shop_type list.

print(Shop_type[:30]):

Finally, this line prints the first 30 elements of the Shop_type list, which contain the cleaned shop types extracted from the web page.

```
address = soup.select(".street-address")

Street_address = []

for i in address:

    Street_address.append(clean_text(i.get_text()))

print(Street_address)
```

address = soup.select(".street-address"):

This line selects all elements with the class "street-address" from the parsed HTML represented by the soup object. It uses the select() method provided by BeautifulSoup to achieve this.

Street_address = []:

This initializes an empty list named Street_address to store the extracted street addresses.

for i in address::

This starts a loop over each element in the address list, which contains elements with the class "street-address".

Street_address.append(clean_text(i.get_text())):

Inside the loop, this line retrieves the text content of each element using the get_text() method. It then applies the clean_text() function to remove extra spaces, line breaks, and tabs from the text. The cleaned text is then appended to the Street_address list.

print(Street_address):

Finally, this line prints the Street_address list, which contains the cleaned street addresses extracted from the web page.

```
locality = soup.select(".locality")

locality

Locality_add = []

for i in locality:

    Locality_add.append(clean_text(i.get_text()))

print(Locality_add)
```

locality = soup.select(".locality"):

This line selects all elements with the class "locality" from the parsed HTML represented by the soup object. It uses the select() method provided by BeautifulSoup to accomplish this.

Locality_add = []:

This initializes an empty list named Locality_add to store the extracted localities.

for i in locality::

This initiates a loop over each element in the locality list, which contains elements with the class "locality".

Locality_add.append(clean_text(i.get_text())):

Inside the loop, this line retrieves the text content of each element using the get_text() method. It then applies the clean_text() function to remove extra spaces, line breaks, and tabs from the text. The cleaned text is appended to the Locality_add list.

print(Locality_add):

Finally, this line prints the Locality_add list, which contains the cleaned localities extracted from the web page.

```
[46]: locality = soup.select(".locality")
      locality
      Locality_add = []
      for i in locality:
          Locality_add.append(clean_text(i.get_text()))
      print(Locality_add)
```

```
['New York, NY 10010', 'New York, NY 10035', 'New York, NY 10007', 'New York, NY 10010', 'New York, NY 10019', 'New York, NY 10017', 'New York, NY 1001
8', 'New York, NY 10023', 'New York, NY 10025', 'New York, NY 10002', 'New York, NY 10002', 'New York, NY 10012', 'New York, NY 10004', 'New York, NY 100
14', 'New York, NY 10010', 'New York, NY 10019', 'New York, NY 10029', 'New York, NY 10029', 'New York, NY 10029', 'New York, NY 10027', 'New York, NY 10
036', 'New York, NY 10035', 'New York, NY 10002', 'New York, NY 10034', 'New York, NY 10038', 'New York, NY 10007', 'New York, NY 10005', 'New York, NY 1
0012', 'New York, NY 10003', 'New York, NY 10011']
```

```
phone = soup.select(".phones.phone.primary")

Phone_number= []

for i in phone:

    Phone_number.append(clean_text(i.get_text()))

print(Phone_number)
```

phone = soup.select(".phones.phone.primary"):

This line selects all elements with the class "phones.phone.primary" from the parsed HTML represented by the soup object. It uses the select() method provided by BeautifulSoup to accomplish this.

Phone_number = []:

This initializes an empty list named Phone_number to store the extracted phone numbers.

for i in phone::

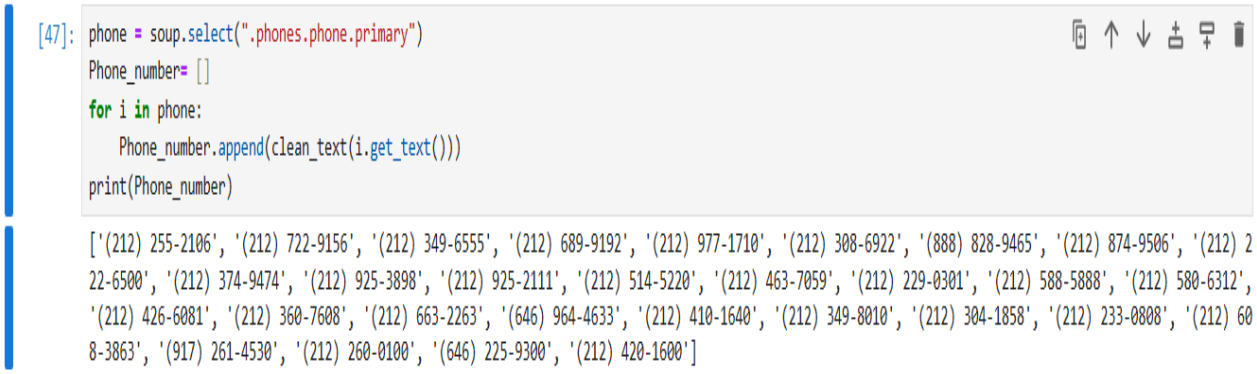
This starts a loop over each element in the phone list, which contains elements with the specified class.

Phone_number.append(clean_text(i.get_text())):

Inside the loop, this line retrieves the text content of each element using the get_text() method. It then applies the clean_text() function to remove extra spaces, line breaks, and tabs from the text. The cleaned text, representing the phone number, is appended to the Phone_number list.

print(Phone_number):

Finally, this line prints the Phone_number list, which contains the cleaned phone numbers extracted from the web page.



```
phone_numbers = [
   '(212) 255-2106', '(212) 722-9156', '(212) 349-6555', '(212) 689-9192', '(212) 977-1710',
   '(212) 308-6922', '(888) 828-9465', '(212) 874-9506', '(212) 222-6500', '(212) 374-9474',
   '(212) 925-3898', '(212) 925-2111', '(212) 514-5220', '(212) 463-7059', '(212) 229-0301',
   '(212) 588-5888', '(212) 580-6312', '(212) 426-6081', '(212) 360-7608', '(212) 663-2263',
   '(646) 964-4633', '(212) 410-1640', '(212) 349-8010', '(212) 304-1858', '(212) 233-0808',
   '(212) 608-3863', '(917) 261-4530', '(212) 260-0100', '(646) 225-9300', '(212) 420-1600'
]
pattern = re.compile(r'\(\d{3}\) 5\d{2}-\d{4}')
numbers_starting_with_5 = [number for number in phone_numbers if pattern.match(number)]
print(numbers_starting_with_5)
```

phone_numbers = [...]:

This initializes a list named phone_numbers containing various phone numbers in the format "(xxx) xxx-xxxx".

pattern = re.compile(r'\(\d{3}\) 5\d{2}-\d{4}'):

This compiles a regular expression pattern to match phone numbers where the local part (the part after the area code) starts with '5'. The pattern r'\(\d{3}\) 5\d{2}-\d{4}' breaks down as follows:

- \(and \): Matches literal parentheses around the area code.
- \d{3}: Matches exactly three digits inside the parentheses (the area code).
- 5\d{2}: Matches a '5' followed by exactly two digits.
- -\d{4}: Matches a hyphen followed by exactly four digits.

numbers_starting_with_5 = [number for number in phone_numbers if pattern.match(number)]:

This uses a list comprehension to create a new list named numbers_starting_with_5. It iterates over each number in the phone_numbers list and includes it in the new list if it matches the compiled regular expression pattern.

print(numbers_starting_with_5):

This prints the numbers_starting_with_5 list, which contains phone numbers where the local part starts with '5'.

[36]:
```
phone_numbers = [
    '(212) 255-2106', '(212) 722-9156', '(212) 349-6555', '(212) 689-9192', '(212) 977-1710',
    '(212) 308-6922', '(888) 828-9465', '(212) 874-9506', '(212) 222-6500', '(212) 374-9474',
    '(212) 925-3898', '(212) 925-2111', '(212) 514-5220', '(212) 463-7059', '(212) 229-0301',
    '(212) 588-5888', '(212) 580-6312', '(212) 426-6081', '(212) 360-7608', '(212) 663-2263',
    '(646) 964-4633', '(212) 410-1640', '(212) 349-8010', '(212) 304-1858', '(212) 233-0808',
    '(212) 608-3863', '(917) 261-4530', '(212) 260-0100', '(646) 225-9300', '(212) 420-1600'
]

# Regular expression to match phone numbers where the local part starts with '5'
pattern = re.compile(r'\(\d{3}\) 5\d{2}-\d{4}')

# Filter phone numbers that match the pattern
numbers_starting_with_5 = [number for number in phone_numbers if pattern.match(number)]

print(numbers_starting_with_5)
```

```
['(212) 514-5220', '(212) 588-5888', '(212) 580-6312']
```

Importing Pandas and creating Data Frame:

import pandas as pd

df=pd.DataFrame({"StoreName":Shop_name[:30],"StoreType":Shop_type[:30],"Stree
t Address":Street_address,"Locality":Locality_add,"Contact No":Phone_number})

df

import pandas as pd:

This imports the pandas library and assigns it the alias pd. pandas is a powerful library for data manipulation and analysis.

df = pd.DataFrame():

This creates a pandas DataFrame named df using the pd.DataFrame constructor. It combines multiple lists into a structured table format.

{"StoreName":Shop_name[:30],"StoreType":Shop_type[:30],"Street Address":Street_address,"Locality":Locality_add,"Contact No":Phone_number}) }:

This dictionary specifies the columns and their corresponding data for the DataFrame:

- **"StoreName": Shop_name[:30]**: The "StoreName" column contains the first 30 elements of the Shop_name list.
- **"StoreType": Shop_type[:30]**: The "StoreType" column contains the first 30 elements of the Shop_type list.
- **"Street Address": Street_address**: The "Street Address" column contains all elements from the Street_address list.
- **"Locality": Locality_add**: The "Locality" column contains all elements from the Locality_add list.
- **"Contact No": Phone_number**: The "Contact No" column contains all elements from the Phone_number list.

df:

This line outputs the DataFrame df, displaying the structured data in a table format with columns "StoreName", "StoreType", "Street Address", "Locality", and "Contact No"

We need to ensure that all lists (Shop_name, Shop_type, Street_address, Locality_add, Phone_number) have the same length, particularly for the first 30 entries, to avoid issues when creating the DataFrame.

[37]:

	Store Name	Store Type	Street Address	Locality	Contact No
0	Trader Joe's	Grocery StoresNatural Foods	675 Avenue Of The Americas	New York, NY 10010	(212) 255-2106
1	Pathmark	Grocery StoresSupermarkets & Super Stores	160 E 125th St	New York, NY 10035	(212) 722-9156
2	Whole Foods Market	Grocery StoresFruit & Vegetable MarketsHealth ...	270 Greenwich St	New York, NY 10007	(212) 349-6555
3	Pioneer Supermarket	Grocery Stores	311 E 23rd St	New York, NY 10010	(212) 689-9192
4	The Food Emporium	Grocery StoresSupermarkets & Super Stores	810 8th Ave	New York, NY 10019	(212) 977-1710
5	Morton Williams Supermarkets	Grocery StoresFish & Seafood MarketsMeat Markets	908 2nd Ave	New York, NY 10017	(212) 308-6922
6	Big Market	Grocery Stores	555 8th Ave	New York, NY 10018	(888) 828-9465
7	Pioneer Supermarket	Grocery StoresSupermarkets & Super Stores	289 Columbus Ave	New York, NY 10023	(212) 874-9506
8	Food City	Grocery StoresSupermarkets & Super Stores	705 Columbus Ave	New York, NY 10025	(212) 222-6500
9	New Lung Hing Market Incorpora	Grocery Stores	51 E Broadway Frnt 5	New York, NY 10002	(212) 374-9474
10	Tak Yee Fong Inc	Grocery Stores	106 Canal St	New York, NY 10002	(212) 925-3898
11	Ammirati	Grocery Stores	584 Broadway Rm 808	New York, NY 10012	(212) 925-2111
12	Central Market	Grocery StoresFlorists	4 South St	New York, NY 10004	(212) 514-5220
13	D'agostino's	Grocery StoresOnline & Mail Order ShoppingSupe...	666 Greenwich St Frnt 1	New York, NY 10014	(212) 463-7059
14	765 Sixth Avenue Market	Grocery StoresSupermarkets & Super Stores	765 Avenue Of The Americas	New York, NY 10010	(212) 229-0301
15	Well Green Market	Grocery StoresSupermarkets & Super Stores	1413 Avenue Of The Americas	New York, NY 10019	(212) 588-5888
16	Red Apple Supermarkets	Grocery Stores	1891 3rd Ave	New York, NY 10029	(212) 580-6312
17	Lb Deli Grocery	Grocery StoresConvenience Stores	349 E 109th St	New York, NY 10029	(212) 426-6081
18	Fine Fare Supermarket	Grocery StoresSupermarkets & Super Stores	1718 Madison Ave	New York, NY 10029	(212) 360-7608
19	Manhattan Food Market	Grocery StoresSupermarkets & Super Stores	507 Manhattan Ave	New York, NY 10027	(212) 663-2263

Converting and Storing Data Frame in the form of a CSV file and opening the file in application:

df.to_csv("Grocery Stores In NewYork.csv",index = False)

df = pd.read_csv("Grocery Stores In NewYork.csv")

df.to_csv():

This method saves the pandas Data Frame df to a CSV (Comma-Separated Values) file.

"Grocery Stores In NewYork.csv":

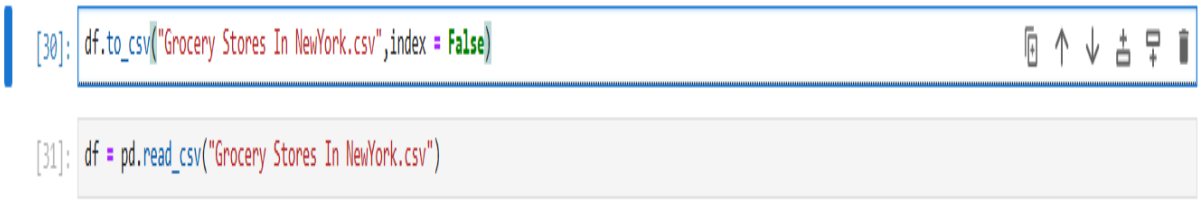
This is the filename for the CSV file where the DataFrame will be saved. The file will be named "Grocery Stores In NewYork.csv" and will be stored in the current working directory.

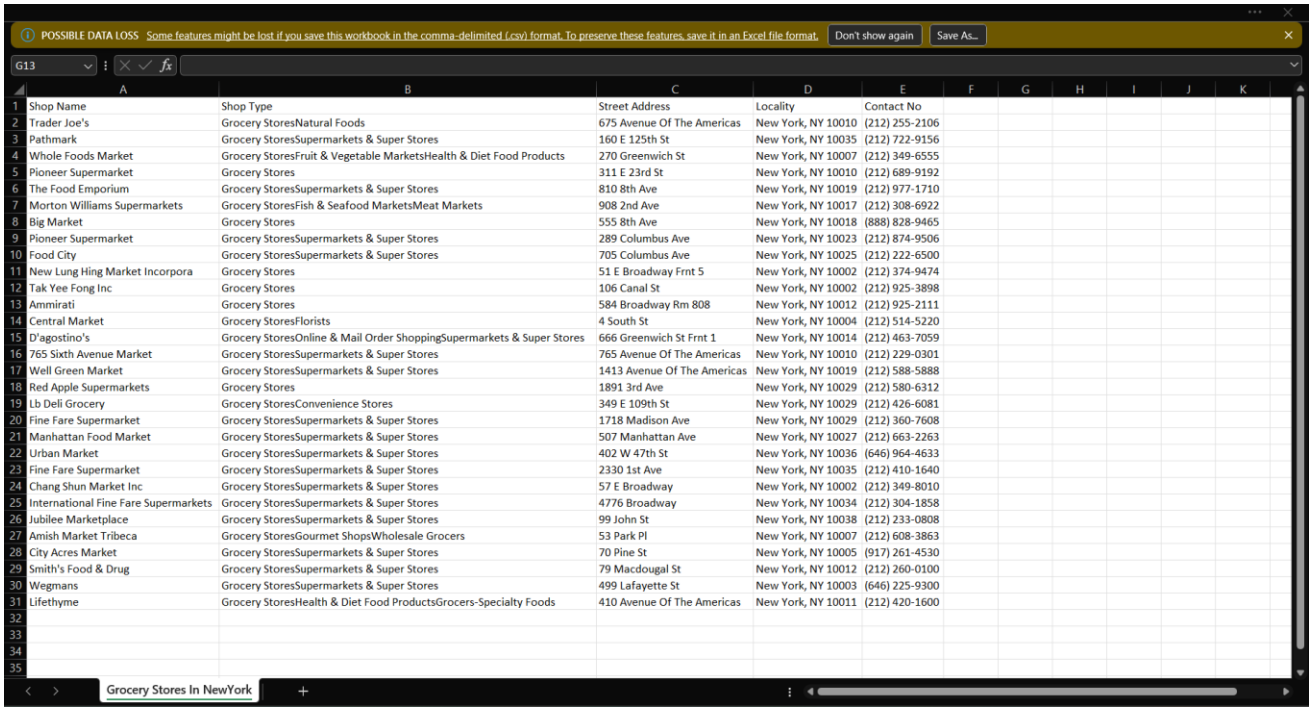
index=False:

This argument specifies that the Data Frame's index (row labels) should not be included in the CSV file. If index=True or if this parameter is omitted, the index will be saved as the first column in the CSV file.

pd.read_csv(...):

This function reads a CSV (Comma-Separated Values) file into a pandas DataFrame. It is used to import data from a CSV file into a structured pandas DataFrame for data manipulation and analysis.





	A	B	C	D	E
1	Shop Name	Shop Type	Street Address	Locality	Contact No
2	Trader Joe's	Grocery StoresNatural Foods	675 Avenue Of The Americas	New York, NY 10010	(212) 255-2106
3	Pathmark	Grocery StoresSupermarkets & Super Stores	160 E 125th St	New York, NY 10035	(212) 722-9156
4	Whole Foods Market	Grocery StoresFruit & Vegetable MarketsHealth & Diet Food Products	270 Greenwich St	New York, NY 10007	(212) 349-6555
5	Pioneer Supermarket	Grocery Stores	311 E 23rd St	New York, NY 10010	(212) 689-9192
6	The Food Emporium	Grocery StoresSupermarkets & Super Stores	810 8th Ave	New York, NY 10019	(212) 977-1710
7	Morton Williams Supermarkets	Grocery StoresFish & Seafood MarketsMeat Markets	908 2nd Ave	New York, NY 10017	(212) 308-6922
8	Big Market	Grocery Stores	555 8th Ave	New York, NY 10018	(888) 828-9465
9	Pioneer Supermarket	Grocery StoresSupermarkets & Super Stores	289 Columbus Ave	New York, NY 10023	(212) 874-9506
10	Food City	Grocery StoresSupermarkets & Super Stores	705 Columbus Ave	New York, NY 10025	(212) 222-6500
11	New Lung Hing Market Incorpora	Grocery Stores	51 E Broadway Frnt 5	New York, NY 10002	(212) 374-9474
12	Tak Yee Fong Inc	Grocery Stores	106 Canal St	New York, NY 10002	(212) 925-3898
13	Ammirati	Grocery Stores	584 Broadway Rm 808	New York, NY 10012	(212) 925-2111
14	Central Market	Grocery StoresFlorists	4 South St	New York, NY 10004	(212) 514-5220
15	D'agostino's	Grocery StoresOnline & Mail Order ShoppingSupermarkets & Super Stores	666 Greenwich St Frnt 1	New York, NY 10014	(212) 463-7059
16	765 Sixth Avenue Market	Grocery StoresSupermarkets & Super Stores	765 Avenue Of The Americas	New York, NY 10010	(212) 229-0301
17	Well Green Market	Grocery StoresSupermarkets & Super Stores	1413 Avenue Of The Americas	New York, NY 10019	(212) 588-5888
18	Red Apple Supermarkets	Grocery Stores	1891 3rd Ave	New York, NY 10029	(212) 580-6312
19	Lb Deli Grocery	Grocery StoresConvenience Stores	349 E 109th St	New York, NY 10029	(212) 426-6081
20	Fine Fare Supermarket	Grocery StoresSupermarkets & Super Stores	1718 Madison Ave	New York, NY 10029	(212) 360-7608
21	Manhattan Food Market	Grocery StoresSupermarkets & Super Stores	507 Manhattan Ave	New York, NY 10027	(212) 663-2263
22	Urban Market	Grocery StoresSupermarkets & Super Stores	402 W 47th St	New York, NY 10036	(646) 964-4633
23	Fine Fare Supermarket	Grocery StoresSupermarkets & Super Stores	2330 1st Ave	New York, NY 10035	(212) 410-1640
24	Chang Shun Market Inc	Grocery StoresSupermarkets & Super Stores	57 E Broadway	New York, NY 10002	(212) 349-8010
25	International Fine Fare Supermarkets	Grocery StoresSupermarkets & Super Stores	4776 Broadway	New York, NY 10034	(212) 304-1858
26	Jubilee Marketplace	Grocery StoresSupermarkets & Super Stores	99 John St	New York, NY 10038	(212) 233-0808
27	Amish Market Tribeca	Grocery StoresGourmet ShopsWholesale Grocers	53 Park Pl	New York, NY 10007	(212) 608-3863
28	City Acres Market	Grocery StoresSupermarkets & Super Stores	70 Pine St	New York, NY 10005	(917) 261-4530
29	Smith's Food & Drug	Grocery StoresSupermarkets & Super Stores	79 Macdougal St	New York, NY 10012	(212) 260-0100
30	Wegmans	Grocery StoresSupermarkets & Super Stores	499 Lafayette St	New York, NY 10003	(646) 225-9300
31	Lifethyme	Grocery StoresHealth & Diet Food ProductsGrocers-Specialty Foods	410 Avenue Of The Americas	New York, NY 10011	(212) 420-1600

Accessing first five rows of DataFrame Using head():

```
[48]: df.head()
```

[48]:

	Store Name	Store Type	Street Address	Locality	Contact No
0	Trader Joe's	Grocery StoresNatural Foods	675 Avenue Of The Americas	New York, NY 10010	(212) 255-2106
1	Pathmark	Grocery StoresSupermarkets & Super Stores	160 E 125th St	New York, NY 10035	(212) 722-9156
2	Whole Foods Market	Grocery StoresFruit & Vegetable MarketsHealth ...	270 Greenwich St	New York, NY 10007	(212) 349-6555
3	Pioneer Supermarket	Grocery Stores	311 E 23rd St	New York, NY 10010	(212) 689-9192
4	The Food Emporium	Grocery StoresSupermarkets & Super Stores	810 8th Ave	New York, NY 10019	(212) 977-1710

Accessing Last five rows of DataFrame Using tail():

```
[54]: df.tail()
```

[54]:

	Store Name	Store Type	Street Address	Locality	Contact No
25	Amish Market Tribeca	Grocery StoresGourmet ShopsWholesale Grocers	53 Park Pl	New York, NY 10007	(212) 608-3863
26	City Acres Market	Grocery StoresSupermarkets & Super Stores	70 Pine St	New York, NY 10005	(917) 261-4530
27	Smith's Food & Drug	Grocery StoresSupermarkets & Super Stores	79 Macdougal St	New York, NY 10012	(212) 260-0100
28	Wegmans	Grocery StoresSupermarkets & Super Stores	499 Lafayette St	New York, NY 10003	(646) 225-9300
29	Lifethyme	Grocery StoresHealth & Diet Food ProductsGroce...	410 Avenue Of The Americas	New York, NY 10011	(212) 420-1600

Using the describe() function to get information about numerical columns:

```
[55]: df.describe()
```

[55]:

	Store Name	Store Type	Street Address	Locality	Contact No
count	30	30	30	30	30
unique	28	10	30	20	30
top	Pioneer Supermarket	Grocery StoresSupermarkets & Super Stores	675 Avenue Of The Americas	New York, NY 10010	(212) 255-2106
freq	2	16	1	3	1

Using info() to get information about the DataFrame:

```
[56]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Store Name      30 non-null     object
 1   Store Type      30 non-null     object
 2   Street Address  30 non-null     object
 3   Locality        30 non-null     object
 4   Contact No      30 non-null     object
dtypes: object(5)
memory usage: 1.3+ KB
```

Future Work

1. **Automation**: Automate the scraping process to update the data regularly.
2. **Extended Data**: Scrape additional details, such as operating hours, customer reviews, and services offered.
3. **Geospatial Analysis**: Use the address data for geospatial analysis to study the distribution of grocery stores across New York City.
4. **Data Integration**: Combine this data with other datasets (e.g., demographic or economic data) for more comprehensive analysis.

Conclusion

This web scraping project successfully extracted, cleaned, and organized data on grocery stores in New York City. The resultant dataset provides valuable insights into the grocery landscape of the city and serves as a foundational resource for further analysis and decision-making.

By automating and expanding the scope of the scraping process, more extensive and dynamic datasets can be maintained, enabling more sophisticated applications and insights in the future.

Thank You