Why I chose V-Mart ?

- V-Mart Retail Limited is **chain Retail small size hypermarket** .
- 368 stores across 235 cities in 26 states and union territories in north, west, east and south India, with a total retail area of 3.2 Million sq.
- V-Mart win the trust of more than 3 crore satisfied customers every year.

Libraries used –

- **requests**: This library is used to make HTTP requests to a website and retrieve the HTML content of a web page.
- **pandas**: This library is used for data manipulation and analysis. It provides data structures for efficiently storing and analyzing data, as well as tools for importing and exporting data from various file formats.
- **re**: This is the built-in regular expression library in Python. It provides functions for working with regular expressions, which are patterns used to match and manipulate text.
- **BeautifulSoup**: This library is used for parsing HTML and XML documents. It provides tools for navigating and searching the structure of an HTML/XML document, and extracting specific elements and data from it.

General Approach-

- **Identify the website**: First, identify the website where you want to scrape the store location data.
- **Understand the structure of the website**: Check the HTML structure of the website, and identify the tags and attributes that contain the store location data. You can use the Inspect Element tool in your web browser to do this.
- **Choose a web scraping tool:** There are many web scraping tools available, including Beautiful Soup, Scrapy, and Selenium. Choose a tool that is best suited for your needs.
- **Write the code:** Once you have chosen a web scraping tool, write the code to extract the store location data. This may involve selecting the appropriate HTML tags and attributes and parsing the data into a usable format.
- **Test the code:** Test the code to make sure it is working correctly. You may need to tweak the code if it is not working as expected.
- **Scale up:** If you need to scrape data from multiple pages on the website, you can modify your code to automate the process. This may involve using loops and other control structures to iterate over multiple pages.
- **Store the data:** Once you have scraped the store location data, store it in a database or file format that is suitable for your needs. You may also want to clean and pre-process the data before storing it.

The Only Problem I Found Challenging was getting the co-ordinates-

```
<a target="_blank" rel="nofollow noopener" href="https://www.google.com/maps/dir/?api=1&origin=&destination=28.656525,77.436216" class="get_direction_click btn w-100 idx-info-card-str-dir-btn txt-color-3 bdr-color-6 bg-color-4 text-uppercase btn-hover-2" role="button"> flex
  <i class="flaticon-location-arrow align-middle map-icn mr-1">
    ::before
  </i>
  "Get Direction "
</a>
```

Getting Location links

```
line 1    tags = soup.find_all("i", {"class": "flaticon-location-arrow align-middle map-icn mr-1"})

    # Creating an empty list to store the href values
line 2    href_list = []

line 3    for tag in tags:
    # Getting the parent anchor tag of the <i> tag
line 4        parent_tag = tag.parent

    # Checking if the href attribute exists in the parent anchor tag
line 5        if "href" in parent_tag.attrs:
    # Append the href value to the list
line 6            href_list.append(parent_tag["href"])

    # Creating a Pandas DataFrame with the href values
line 7    location = pd.DataFrame({"href": href_list})

    # Print the DataFrame
line 8    print(location)
```

parent attribute: parent attribute in Beautiful Soup is used to access the parent tag of a given tag in an HTML document. It returns a Tag object that represents the parent tag in the parse tree.

The parent attribute is useful when you want to navigate the parse tree and access tags that are higher up in the hierarchy. For example, if you're scraping a web page and you want to extract information from a table, you might start by finding the table tag, and then use the parent attribute to access the parent tag (which might be a div or a section tag,)

Description-

- Line 1: **tags = soup.find_all("i", {"class": "flaticon-location-arrow align-middle map-icn mr-1"})** -This code gives the location links of the stores.
- Line 1: For this I needed to first find out the tag and class which are common in every location, which was class: **flaticon-location-arrow align-middle map-icn mr-1**, it is used to attach a location pin logo.
- Line 2 creating an empty list to store the output links
- Line 3: **for tag in tags**: - This starts a loop over a list of <i> tags.
- Line 4: **parent_tag** = tag.parent - This gets the parent tag of the current <i> tag in the loop. The parent attribute returns the parent tag of the current tag in the BeautifulSoup parse tree.
- Line 5: **if "href" in parent_tag.attrs**: - This checks if the href attribute exists in the attributes dictionary of the parent tag. If it does, that means the parent tag is likely an anchor (<a>) tag and contains a hyperlink.
- Line6: **href_list.append(parent_tag["href"])** - If the parent tag has an href attribute, this appends the value of the href attribute to a list called href_list. This list is likely being used to store all the hyperlinks found in the loop.
- Line 7: Stores the location into a dataframe.
- Line 8: Prints(shows) outputs –

Outputs –

	href
0	https://www.google.com/maps/dir/?api=1&origin=...
1	https://www.google.com/maps/dir/?api=1&origin=...
2	https://www.google.com/maps/dir/?api=1&origin=...
3	https://www.google.com/maps/dir/?api=1&origin=...
4	https://www.google.com/maps/dir/?api=1&origin=...
5	https://www.google.com/maps/dir/?api=1&origin=...
6	https://www.google.com/maps/dir/?api=1&origin=...
7	https://www.google.com/maps/dir/?api=1&origin=...
8	https://www.google.com/maps/dir/?api=1&origin=...

Extracting coordinates from the links-

Code –

```
      # Define a regular expression pattern to extract the latitude and
       longitude values
   line 1   pattern = r"destination=([\d\.]+),([\d\.]+)"

      # Define a function to extract the latitude and longitude values
      from a link
   line 2   def extract_coords(link):
   line 3   match = re.search(pattern, link)
   line 4   if match:
                  latitude = match.group(1)
                  longitude = match.group(2)
                  return f"{latitude},{longitude}"
   line 5   else:
                  return None

# Applying the extract_coords function to the href column to extract th
e coordinates
   Line 6    location["href"] = location["href"].apply(extract_coords)
```

Description -

pattern = r"destination=([\d\.]+),([\d\.]+)" - This defines a regular expression pattern to match latitude and longitude values in a URL. The pattern uses capturing groups to extract the latitude and longitude values from the URL.

def extract_coords(link): - This defines a function called extract_coords that takes a single argument, link. The purpose of this function is to extract the latitude and longitude values from a URL.

match = re.search(pattern, link) - This uses the re.search function to search for the regular expression pattern defined in step 1 in the link argument. The result is a Match object that contains information about the matched string.

if match: - This checks if a match was found in the previous step.

latitude = match.group(1) - This extracts the first capturing group (the latitude value) from the matched string.

longitude = match.group(2) - This extracts the second capturing group (the longitude value) from the matched string.

return f"{latitude},{longitude}" - This returns a string containing the latitude and longitude values separated by a comma.

else: - This is part of the if statement from step 4. If no match was found, execution jumps to this line.

return None - This returns None to indicate that no latitude and longitude values were found in the URL.

location["href"] = location["href"].apply(extract_coords) - This applies the extract_coords function to the href column of a Pandas DataFrame called location. The result is a new column called href that contains latitude and longitude values extracted from the original href column. The apply method applies the function to each element of the column.

Final Output

	address	area	contact_no	coordinates	timing	directions
0	No D 3, Block 1, Central Market 2, Lajpat Naga...	Lajpat Nagar, South East Delhi	7872478724	28.568548,77.242454	10:00 AM to 10:00 PM (Closing Soon)	https://www.google.com/maps/dir/?api=1&origin=...
1	Shop No 610 And 611, Ravi Das Marg, Main Bazar...	Laxmi Nagar, New Delhi	0844 8982 285	28.6365793,77.2790923	10:00 AM to 10:00 PM (Closing Soon)	https://www.google.com/maps/dir/?api=1&origin=...
2	C 7, Jyoti Nagar West, North Chajjupur, Block ...	Shahdara, New Delhi	0844 8982 285	28.689468,77.2880388	10:00 AM to 10:00 PM (Closing Soon)	https://www.google.com/maps/dir/?api=1&origin=...
3	No E/561/A & E/561B, Palam Extension, Dwarka, ...	Dwarka, New Delhi	0931 3699 995	28.583891,77.071784	10:00 AM to 10:00 PM (Closing Soon)	https://www.google.com/maps/dir/?api=1&origin=...
4	Virendar Nagar, Block B, Sant Nagar, 100 feet ...	Burari, North Delhi	0931 3699 995	28.7427944,77.1980152	10:00 AM to 10:00 PM (Closing Soon)	https://www.google.com/maps/dir/?api=1&origin=...