**Extract Retail Stores Details**

**Objective:** Identify any retail stores in India using web scraping techniques to extract the store names, addresses, timings, coordinates, and phone numbers from the website.

**Description:**

The order to extract the required information, the **BeautifulSoup** library. This library is used to scrape data from the website or url.

In this particular case, we used [***https://stores.vmartretail.com/***](https://stores.vmartretail.com/) to extract the data. The attached code extracts the name, address, timing, coordinates, and phone number of retail stores in India and saves the data in a CSV file named "output.csv".

The scraping module "scrape\_page(url)" takes a URL as input, sends an HTTP request to the URL using the **requests** library, and extracts the desired data from the HTML response using BeautifulSoup selectors. Additionally, it stores the extracted data into a CSV file.

**The first challenge of finding coordinates**

To locate an HTML anchor tag (<a>) with a specific set of classes.

*href="https://www.google.com/maps/dir/?api=1&origin=&destination=12.960966,77.507281*

['href']: Once the find() method has located the desired anchor tag, the code accesses the value of the href attribute by using square brackets ([]). This extracts the value of the href attribute from the anchor tag and assigns it to the variable direction.

direction.split('/')[5]: This line splits the value of the direction variable into a list of strings using the forward-slash **(/)** character as a delimiter. It then extracts the sixth element (index 5) from the resulting list and assigns it to the variable destination.

destination.split('=')[3]: Finally, this line splits the value of the destination variable into a list of strings using the equals sign **(=)** as a delimiter. It then extracts the fourth element (index 3) from the resulting list and assigns it to the variable coordinates.

**Second challenge of scraping multiple pages**

To scrape data from multiple pages, there are a few key steps to follow:

1. Identify the base URL: This ***'https://stores.vmartretail.com/?page={page\_no}'*** URL is common to all pages we want to scrape. It should include the part of the URL that is constant across all pages.
2. Determine the number of pages: we need to know how many pages there are to scrape. we can either manually count the number of pages or use a program to automatically determine this. I used it automatically in this code.
3. Loop through each page: Use a loop to visit each page and scrape the data. we can use the base URL and the page number to construct the URL for each page.
4. Store the data: As we scrape data from each page, store it in a data structure like a list. Once we have scraped all the pages, we can combine the data into a single data structure like a Pandas DataFrame.
5. Save the data: Once we have combined it, save it to a file using a CSV format.

**Step-by-step process of the code:**

1. Import the required libraries: BeautifulSoup, requests, and pandas
2. Define a list to store the details of the retail store, such as store name, address, timing, coordinates, and phone number.
3. Define the output file name as 'output.csv'
4. Define the base URL as 'https://stores.vmartretail.com/?page='
5. Define a function named 'scrape\_page' to scrape data from each page of the given URL.
6. Use the requests module to get the response from the URL and BeautifulSoup to extract data from the page.
7. Use the BeautifulSoup selectors to extract the details of each retail store from the page and append it to the respective list.
8. Create a pandas DataFrame using the above lists and save it as a CSV file.
9. Define the total number of pages to scrape by getting the maximum number of pages from the website.
10. Loop through each page and call the 'scrape\_page' function to extract data from each page and save it to the output file.