Marks & Spencer is a renowned British retailer known for its diverse product range, which includes high-quality clothing, premium food and groceries, home and living products, beauty and cosmetics, and a commitment to sustainability and social responsibility. This unique combination of offerings has made M&S a popular choice for consumers in the UK and beyond.

Here we will explore nightwear in lingerie and gain an understanding of the availability of different brands and the count of products in each brand, the most trusted and reviewed product brands, the top brand's average rating with respect to their count, and average discount on the brands on sale.

**Attributes Scraped**

The following data are extracted from each product present in Marks and Spencer women's lingerie

* Brand Name: The name of the product
* Title: The Category/Type of the product
* Product URL: The URL of the product
* Average rating: Average number of ratings available for the product
* Product Code: Product code specified for each product
* Reviews: Number of reviews each product received
* Selling Price: The current selling price of the product
* Original price: The maximum retail price of the product
* Discount: Difference between the original price and the selling price
* Sales status: The list of products that are on sale
* Composition: Inventory of materials employed in the production of the product

This Python script is designed to scrape data from a Marks & Spencer (M&S) website related to lingerie and nightwear products. It uses various libraries, such as requests, BeautifulSoup, pandas, regular expressions, and others, to extract, clean up, and export product information from web pages. Here's a plain English description of the code:

1. **Importing Libraries:**

The script begins by importing necessary libraries, including **requests** for making HTTP requests, **BeautifulSoup** for parsing HTML, **pandas** for data manipulation, **re** for regular expressions, **os** for file operations, and **RequestException** for handling request-related exceptions.

Here's a brief explanation of each library imported in the script and its usage:

1. **Requests** is a popular Python library for making HTTP requests to web pages.

- Usage in the script: It sends HTTP GET requests to URLs to retrieve the HTML content of web pages, which can then be parsed using BeautifulSoup.

2. **BeautifulSoup** is a library for parsing HTML and XML documents.

- Usage in the script: It is used to parse the HTML content of web pages obtained through requests. This allows the script to navigate and extract specific elements and data from the web pages.

3. **pandas** is a powerful data manipulation library in Python that provides data structures like DataFrames for handling and analyzing structured data.

- Usage in the script: It is used to create and manipulate DataFrames to store and organize the scraped product data. This makes it easier to work with and export the data to CSV format.

4. **re** (regular expressions) is a built-in Python library for working with regular expressions, which are used for pattern matching and text manipulation.

- Usage in the script: It is used to extract specific information from text strings obtained from web pages. For example, it is used to extract the number of reviews from a string.

5. **os** is a built-in Python library for interacting with the operating system, including file and directory operations.

- Usage in the script: It is used to check if a file already exists and to manage the CSV file where the scraped data is stored. It helps in creating, replacing, or appending to the CSV file as needed.

6. **RequestException** (from requests.exceptions) is an exception class provided by the requests library for handling exceptions related to HTTP requests.

- Usage in the script: It is used to catch and handle exceptions that may occur when making HTTP requests, such as network errors or timeouts. This ensures that the script can handle errors gracefully and continue execution if possible.

In summary, these libraries are essential for various aspects of web scraping, data extraction, data manipulation, and error handling in the script.

**2. Defining Data Storage:**

Several lists are initialized to store information about each product, including its URL, company, product name, product code, average rating, reviews, selling price, original price, saved price, color, sales status, styles, and composition.

**3. Functions:**

The script defines several functions for specific tasks, such as extracting HTML content from a URL, extracting the number of pages to scrape, extracting product URLs from pages, and extracting various product details like company, product name, etc.

**4. main Function:**

* The main function is the entry point of the script.
* It defines the base URL and headers for making HTTP requests to the M&S website.
* It starts by extracting the HTML content of the main product page using the extract\_href\_values function.
* It then calculates the number of pages of products available and generates the URLs for these pages.
* Next, it extracts the individual product URLs using the extract\_product\_url function.
* The script then proceeds to scrape product details using the fetch\_product\_details function.

**5. Scraping Product Details:**

Inside the fetch\_product\_details function, the script iterates through each product URL, sends an HTTP request to the product page, and extracts details such as the company, product name, product code, average rating, reviews, prices, color, sales status, and composition. These details are stored in the respective lists.

**6. Exporting Data:**

* After scraping all the product details, the script creates a DataFrame using pandas to organize the data.
* It checks if a file named 'mas.csv' already exists. If it does, it replaces it; otherwise, it creates a new CSV file.
* The final DataFrame is saved to the 'mas.csv' file.

**7. Execution:**

The script checks whether it is being executed directly (not imported as a module) and, if so, calls the main function to start the scraping process.

In summary, this Python script scrapes information about lingerie and nightwear products from the M&S website, extracts various details about each product, and exports the data to a CSV file for further analysis or storage.

**Extracting Product Information**

Several functions are defined for extracting different information from product pages, brand name, product names, prices, ratings, the count of ratings, colors, and various details. Here's an explanation of each of the functions :

1. `extract\_href\_values(url, headers, max\_retries=3)`:

* This function takes a URL, headers, and an optional maximum retry count as input.
* It sends a GET request to the provided URL with the specified headers.
* If the request fails due to network issues or other problems, it retries up to three times.
* It returns the parsed HTML content of the response using Beautiful Soup.

2. `extract\_number\_of\_pages(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for a specific element in the HTML (a span with a particular class) that contains information about the number of pages.
* It extracts and returns the maximum number of pages as an integer.

3. `extract\_product\_url(baseUrl, pageUrls, headers)`:

* This function takes a base URL, a list of page URLs, and headers as input.
* It iterates through each page URL, sends requests, and extracts product URLs from the HTML content of each page.
* It returns a list of product URLs.

4. `extract\_company(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It attempts to find the brand or company name in the HTML using specific classes.
* If it finds the name, it returns it as a string. Otherwise, it returns "Not available."

5. `extract\_product\_name(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for an element containing the product name in the HTML.
* If it finds the name, it returns it as a string. Otherwise, it returns "Not available."

6. `extract\_product\_code(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It searches for an element that contains the product code.
* It extracts and returns the product code as a string if found; otherwise, it returns "Not available."

7. `extract\_avg\_rating(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It searches for an element containing the average rating.
* It extracts and returns the average rating as a float if found; otherwise, it returns None.

8. `extract\_reviews(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It searches for an element that contains the number of reviews.
* It extracts and returns the number of reviews as an integer if found; otherwise, it returns None.

9. `extract\_selling\_price(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for an element that contains the selling price of a product.
* It extracts and returns the selling price as a float, removing the currency symbol if present.

10. `extract\_original\_price(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for an element that contains the original price of a product.
* If found, it extracts and returns the original price as a float. If not found, it falls back to using the selling price.

11. `extract\_saved\_price(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It searches for an element containing information about the saved price (discount).
* It extracts and returns the saved price as a float if found; otherwise, it returns None.

12. `extract\_color(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for an element containing the product color information.
* It returns the color as a string if found; otherwise, it returns "Not available."

13. `extract\_sales\_status(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It searches for an element that indicates the sales status (e.g., "In stock" or "Out of stock").
* It returns the sales status as a string if found; otherwise, it returns "Not available."

14. `extract\_composition(soup)`:

* This function takes the parsed HTML content (soup) as input.
* It looks for an element containing information about the product's composition or materials.
* It returns the composition as a string if found; otherwise, it returns "Not available."

**Storing product information**

Python script scrapes information about lingerie and nightwear products from the M&S website, extracts various details about each product, and exports the data to a CSV file for further analysis or storage.