TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

FINAL REPORT

Project

# WEB SCRAPING

## CUSTOMER REVIEW ANALYSIS

PROF. MANEL ABDELKADER

IT 360 2023 - 2024

FINAL REPORT

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

# PRESENTED BY:


ZEINEB TRABELSI

NOURHENE BEN LTAIEF

YESMINE CHALGHAM

ZEINEB HAJJI

MOHAMED AMINE ZOUAGHI


# WEB SCRAPING

## CUSTOMER REVIEW ANALYSIS

# Project Details :

This is our web scraping project report:
Scraping customer product reviews, prices, and brands for any product on any e-commerce website and comparing prices, quality, and comments. This scraped data will be used for further dashboarding and analysis where the customer can set their preferences concerning a certain range of products and receive an email containing the list of the suited products for them.

Web scraping is a technique used to extract large volumes of data from websites where data is publicly available programmatically.

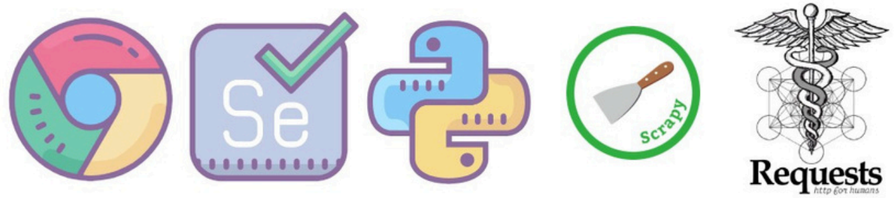
Our project will focus on **product and customer data  in e-commerce web pages such as star ratings, prices, and quality reviews ..to compare them and analyze customer inferences.**

This project will help as a tool for businesses to better understand the reach of their products through the analysis of customer reviews. It will also allow the customers to shop more comfortably. This way they can know about insightful trends of customers and fine-tune their products and services accordingly.

# TOPIC :

Project Overview

This project involves scraping product data from a website, cleaning and processing the data, analyzing and scoring the products based on user preferences, encrypting and decrypting the processed data, and finally sending an email with the top product recommendations. The main goal is to provide a user with the best product recommendations based on their specified criteria, while ensuring data security and integrity.




# 1. Reminder :

**In this project we will be using Python as a main tool to develop our solution .**
**The librairies needed are :**

## Requests

USED TO FETCH THE HTML CONTENT OF A WEBPAGE FROM THE WEB SERVER.

## Pandas

USED TO ORGANIZE AND MANIPULATE THE EXTRACTED DATA FROM THE WEBPAGE INTO A STRUCTURED FORMAT LIKE A DATAFRAME.

## Beautiful Soup

USED TO EXTRACT THE DESIRED INFORMATION FROM THE DOWNLOADED HTML CONTENT

## selenium

IDEAL FOR SCRAPING DYNAMIC WEBSITES THAT REQUIRE INTERACTION WITH JAVASCRIPT ELEMENTS. SELENIUM IS ALSO KNOWN TO HANDLING CAPTCHA SECURITY.

# 2- Development Phases:

- **Planning Phase:**

we defined our project objectives, scope, and target websites.
we identified the types of customer feedback and reviews to scrape.
we determined the data structure for storage and analysis.

- **Data Scraping Phase:**

we developed web scraping scripts using Python and relevant libraries( selenium, Beautiful Soup).
we extracted customer feedback, reviews, and prices from target websites( jumia).
we stored the scraped data in a csv file for further processing.

- **Data Cleaning and Preprocessing Phase:**

We cleaned the scraped data to remove duplicates, irrelevant information, and inconsistencies.
we normalized the data for analysis and dashboard creation.
we saved the exported data to a file and later exported it to the PowerBI for further analysis and  dashboard creation

# 2- Development Phases:

- **Data Analysis Phase:**

we analyzed the customer feedback and reviews to extract insights and trends.
Compare feedback across products and ratings.
we pointed out the top products with the lowest prices and better ratings.

- **Dashboard Creation Phase:**

we used PowerBi for the dashboard creation as we were taught in the Business Intelligence course.
we visualized the analyzed data in a user-friendly format.
we included key metrics, trends, and insights for easy interpretation.

- **Email :**

we used smtplib to allow our system to send an email with the suited product for the customer. The customer has to set the minimum and maximum price and the preferred brands (if existent).
This automated process simplifies finding and notifying users of the best products based on their criteria.

# 2- Development Phases:

- **Encryption /Decryption :**

we included the encryption/decryption part to add a security aspect to the project and put into practice what we have learned.
we used Python, specifically the cryptography library.
This phase encrypts a file using the Fernet encryption algorithm, ensuring data security. It generates an encryption key, reads data from a CSV file, encrypts it, and saves the encrypted data to a new file. Finally, it confirms successful encryption.

# 3-Data Scraping Phase:

- **Target URLs:**

https://www.jumia.com.tn/

These are examples of the websites we can also be working with :



# Data scraping :

```python
import csv
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from lxml import html
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.action_chains import ActionChains
from itertools import zip_longest
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import logging

def scrape_product_data(start_url, num_pages):
    """Scrape product data from the start URL and subsequent pages and write to a CSV file."""
    logger = logging.getLogger(__name__)  # Creating a new logger object
    try:
        options = webdriver.ChromeOptions()
        options.add_argument('--no-sandbox')
        options.add_argument('--disable-dev-shm-usage')

        with webdriver.Chrome(options=options) as driver:
            product_data = []
            for page in range(1, num_pages + 1):
                url = f"{start_url}&page={page}#catalog-listing"
                driver.get(url)
                wait = WebDriverWait(driver, 10)
                wait.until(EC.presence_of_element_located((By.XPATH, "//h3[@class='name']")))
                html_content = driver.page_source
                tree = html.fromstring(html_content)
                product_names = tree.xpath("//h3[@class='name']/text()")
                product_prices = tree.xpath("//div[@class='prc']/text()")
                product_rates = tree.xpath("//div[@class='stars _s']/text()")
```

This script is designed to scrape product data from a specific URL, including product names, prices, and ratings, and then write this data to a CSV file. Additionally, it identifies the product with the best price from the scraped data.

```
        # Ensure equal length of lists
        zipped_data = zip_longest(product_names, product_prices, product_rates, fillvalue=u'')
        for name, price, rate in zipped_data:
            product_data.append({u'product name': name.strip(), u'product price': price.strip(), u'product rate': rate.strip()})
        logger.info(u"Number of products on page %d: %d", page, len(product_names))

    output_file_path = u'output.csv'
    with open(output_file_path, 'w', newline='', encoding='utf-8') as csvfile:
        fieldnames = [u'product name', u'product price', u'product rate']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for product in product_data:
            writer.writerow(product)

    logger.info(u"Output has been written to: %s", output_file_path)
```

# Detailed Steps of scraping code :

**1.Import Libraries:**

The script imports necessary libraries including csv and Selenium for web scraping, logging for logging messages, and other relevant modules.

**2. Scrape Product Data Function (scrape_product_data):**

- This function takes a start URL and the number of pages to scrape as input parameters.
- It sets up a Chrome WebDriver using Selenium, navigates to each page, and extracts product information.
- The product data (names, prices, and ratings) are collected and stored in a list of dictionaries.
- After scraping all pages, the data is written to a CSV file.

**3.Find Best Product Function (find_best_product):**

This function reads the CSV file containing product data and finds the product with the best price.

It iterates through each row of the CSV, compares prices, and identifies the product with the lowest price.

**4.Logging Setup:** Logging messages are used to provide information about the scraping process and any errors encountered. This helps in debugging and monitoring the script's execution.

**5.Example Usage:**

The script includes an example usage where it specifies the start URL (start_url) and the number of pages to scrape (num_pages), and then calls the scrape_product_data function with these parameters.

```
            # Read CSV file and find product with best price
            best_product = find_best_product(output_file_path)
            logger.info(u"Best product: %s", best_product)

    except Exception as e:
        logger.error("An error occurred: %s", e)

def find_best_product(file_path):
    """Find product with best price from CSV file."""
    best_price = float('inf')
    best_product = None

    with open(file_path, 'r', newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            price_str = row['product price']
            price_str = price_str.replace(' TND', '')
            price = float(price_str)
            if price < best_price:
                best_price = price
                best_product = row

    return best_product

# Example usage:
start_url = "https://www.jumia.com.tn/catalog/?q=ecran+solaire"
num_pages = 12
scrape_product_data(start_url, num_pages)
```

# Output :

| product name | product price | product rate |
|---|---|---|
| SVR SUN SECURE BLUR SPF 50+ 50 ML | 51.59 TND | 4.1 out of 5 |
| SVR SUN SECURE BLUR TEINTE SPF 50+ 50ML | 51.59 TND | 4 out of 5 |
| lirene Sun Protection - Emulsion SPF 50+ - 150+25 ML | 39.50 TND | 4.7 out of 5 |
| SVR SUN SECURE FLUIDE SPF50+ 50ML | 49.00 TND | 4.5 out of 5 |
| Cetaphil Ecran solaire - Spf50+ | 54.00 TND | 4.5 out of 5 |
| Daylong LAIT EXTREME 50+ 50ML | 49.50 TND | 4 out of 5 |
| SunSafeÂ® Ã‰cran Solaire Invisible SunSafeÂ® SPF 50+, Anti-Ã¢ge | 38.00 TND | 4.4 out of 5 |
| Isdin Foto ultra - Ecran invisible  anti-pigment - Active unify 50ml | 75.00 TND | 4.6 out of 5 |
| Dr Rashel CrÃ¨me solaire anti-Ã¢ge SPF 60 - 80 gr | 30.00 TND | 4.5 out of 5 |
| Isdin Fotoprotector - Transparent Spray - SPF 50 - 250ml | 58.00 TND | 4.6 out of 5 |
| Dr Rashel CrÃ¨me solaire anti-Ã¢ge SPF 75 - 80 gr | 30.00 TND | 3.9 out of 5 |
| Isdin Fotoprotector - Fusion Water - Fluide Solaire Invisible - SPF 50 - 50ml | 55.50 TND | 4.3 out of 5 |
| CYTOL NAT Pack  - Ecran invisible + Gel nettoyant + CrÃ¨me apaisante | 62.00 TND | 4.4 out of 5 |
| CYTOL NAT Cytolsun Max - Ecran Solaire Invisible spf50+ - 50ml | 31.00 TND | 4.1 out of 5 |
| Daylong Extreme -  Spf 50+ - Lait - 100ml | 87.41 TND | 4.5 out of 5 |
| Pharmaceris Fluid Sand SPF 50+ N:02- 30 Ml | 37.30 TND | 3 out of 5 |
| Doris Ecran Solaire Teinte Beige Claire  -50 Ml | 49.00 TND | 4.5 out of 5 |
| SVR Ecran solaire et anti-taches Clairial - CC CrÃ¨me light SPF 50+ 40ml | 68.88 TND | 3.4 out of 5 |
| Pharmaceris Fond de teint - SPF 50 - 02 SAND | 39.40 TND | 4.2 out of 5 |
| Isdin Pediatrics transparent spray pour enfants - Spf50 - 250ml | 64.00 TND | 4.8 out of 5 |
| Esth'Elle Sun protect Ã©cran  - Invisible | 27.90 TND | 4.5 out of 5 |
| Daylong Extreme -  Spf 50+ - Lait - 100 ml | 111.25 TND | 4.2 out of 5 |
| Ultrasun Ecran solaire - Antiage et anti-pigmentation - Spf50+ 40 ml | 62.50 TND | 3.9 out of 5 |
| Esth'Elle Ecran solaire - TeintÃ© beige - 50 ml - SPF 50+ | 27.90 TND | 5 out of 5 |
| CYTOL NAT Cytolsun Max - Ecran Solaire Invisible SPF50+ - 50ml | 39.00 TND | 5 out of 5 |

- We processed the scraped data, organized it into dictionaries, and stored it in a list for further processing.
- The data was then written to a CSV file using the csv.DictWriter module for easy storage and future analysis.

# Data Cleaning :

```python
import pandas as pd

# Provide the full path to the CSV file
file_path ='output.csv'

# Load the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Remove duplicates (if any)
df.drop_duplicates(inplace=True)

#Price column
# Extract numeric part from price column and convert to numeric
df['product price'] = df['product price'].str.replace('TND', '').astype(float)

#rate column
# Remove "out of 5" string from review column and convert to numeric
df['product rate'] = df['product rate'].str.replace('out of 5', '').astype(float)
# Replace blank values with 'None' as a string
df['product rate'].fillna('None', inplace=True)

#brand name column
# Extract brand name from product name column
df['brand'] = df['product name'].apply(lambda x: x.split()[0])

# Extract brand name from product name column
def extract_brand(product_name):
    # Split the product name by whitespace
    parts = product_name.split()
    brand = parts[0]
    if brand.lower() in ['fruit']:
        brand = 'Fruit Of The Wokali'
    # Check if the brand is in the specified list
    if brand.lower() in ['dr', 'milva', 'proderma', 'pro', 'la']:
        # If yes, add the next word as part of the brand name
        if len(parts) > 1:
            brand += ' ' + parts[1]
```

1. **Load CSV File into DataFrame:**
The script loads the CSV file located at the specified path into a Pandas DataFrame named df.

**2. Remove Duplicates:**
It removes duplicate rows from the DataFrame, if any, using the drop_duplicates() method.

**3. Data Cleaning:**
Price Column: It extracts the numeric part from the "product price" column and converts it to a float datatype, removing the "TND" string.
Rate Column: It removes the "out of 5" string from the "product rate" column and converts it to a float datatype. Blank values are replaced with 'None' as a string.
Brand Name Column: The script extracts the brand name from the "product name" column and creates a new column named "brand".

**4. Feature Engineering:**

Product Category Column: It classifies products based on keywords in the "product name" column, creating a new column named "product category".

Brand Category Column: Brands are classified as local or international based on a predefined list of local brands, creating a new column named "brand category".

Quantity Column: It extracts the quantity and unit from the "product name" column using regular expressions, creating a new column named "quantity". Rows with no quantity value are removed.

**5. Export Data:**

The cleaned and processed DataFrame is exported to both CSV and Excel formats at the specified output file paths.

```python
#product category column
# Function to classify products based on keywords in product name
def classify_product(product_name):
    if 'lait' in product_name.lower() or 'lotion' in product_name.lower():
        return 'lotion'
    elif 'spray' in product_name.lower():
        return 'spray'
    elif 'creme' in product_name.lower():
        return 'creme'
    elif 'Ecran' in product_name.lower() or 'sun' in product_name.lower() or 'solaire' in product_name.lower():
        return 'sunscreen'
    else:
        return 'other'

# Create a new column "product_category" by applying the classify_product function to the "product_name" column
df['product category'] = df['product name'].apply(classify_product)

#brand category column
# List of local brands
local_brands = ['Dermacare', 'SunSafe®', 'Proderma Dermalight', 'Milva Olcare', 'Fruit Of The Wokali', 'Arvea']

# Function to classify brands as local or international
def classify_brand(brand):
    if brand in local_brands:
        return 'local'
    else:
        return 'international'

# Create a new column "brand_category" by applying the classify_brand function to the "brands" column
df['brand category'] = df['brand'].apply(classify_brand)
```

# Main steps:

- Data Cleaning and Transformation: We utilized various Pandas methods and functions to clean and transform the data, such as string manipulation, regular expressions, and data type conversion.
- Feature Engineering: New features were created based on existing data to provide additional insights or improve model performance.
- Exporting Data: The cleaned data was exported to CSV and Excel formats for further analysis or use in other applications.

```
Create a new column "brand_category" by applying the classify_brand function to the "brands" column
['brand category'] = df['brand'].apply(classify_brand)

uantity
port re
Function to extract quantity and unit from product name
f extract_quantity_and_unit(product_name):
  # Regular expression pattern to extract the quantity and unit before 'ML' or 'GR'
  pattern = r'(\d+)\s*(ML|GR)'
  match = re.search(pattern, product_name, re.IGNORECASE)
  if match:
      quantity = int(match.group(1))
      unit = match.group(2).upper()  # Convert unit to uppercase (ML or GR)
      if unit == 'ML':
          return f'{quantity} ML'
      elif unit == 'GR':
          return f'{quantity} GR'
  else:
      return None

Create a new column "quantity" by applying the extract_quantity_and_unit function to the "product_name" column
['quantity'] = df['product name'].apply(extract_quantity_and_unit)
Remove rows where the quantity column has no value
 = df[df['quantity'].notna()]

Display the first few rows of the DataFrame to understand its structure
int(df.head(50))

Export the DataFrame to a CSV file
.to_csv('clean output.csv', index=False)
.to_excel('clean_output.xlsx', index=False)
```

# Output :

| 1 | product name | product price | product rate | brand | product category | brand category | quantity |
|---|---|---|---|---|---|---|---|
| 2 | SVR SUN SECURE BLUR SPF 50+ 50 ML | 51,59 | 4,1 | SVR | sunscreen | international | 50 ML |
| 3 | Pharmaceris S Sun Spectrum -protect- Crème écran très haute protection SPF50+ 50ML | 29,5 | 4,5 | Pharmaceris | sunscreen | international | 50 ML |
| 4 | SVR SUN SECURE BLUR TEINTE SPF 50+ 50ML | 51,59 | 4 | SVR | sunscreen | international | 50 ML |
| 5 | SVR SUN SECURE FLUIDE SPF50+ 50ML | 49 | 4,7 | SVR | sunscreen | international | 50 ML |
| 6 | Daylong LAIT EXTREME 50+ 50ML | 49,5 | 4 | Daylong | lotion | international | 50 ML |
| 7 | Isdin Foto ultra - Ecran invisible  anti-pigment - Active unify 50ml | 75 | 4 | Isdin | other | international | 50 ML |
| 8 | Isdin Fotoprotector - Fusion Water - Fluide Solaire Invisible - SPF 50 - 50ml | 55,5 | 4,6 | Isdin | sunscreen | international | 50 ML |
| 9 | Dr Rashel Crème solaire anti-âge SPF 75 - 80 gr | 30 | 4,4 | Dr Rashel | sunscreen | international | 80 GR |
| 10 | Dr Rashel Crème solaire anti-âge SPF 60 - 80 gr | 30 | 4,5 | Dr Rashel | sunscreen | international | 80 GR |
| 11 | Isdin Fotoprotector - Transparent Spray - SPF 50 - 250ml | 60 | 4,4 | Isdin | spray | international | 250 ML |
| 12 | CYTOL NAT Cytolsun Max - Ecran Solaire Invisible spf50+ - 50ml | 31 | 5 | CYTOL | sunscreen | international | 50 ML |
| 13 | Pharmaceris Fluid Sand SPF 50+ N:02- 30 Ml | 37,3 | 4,1 | Pharmaceris | other | international | 30 ML |
| 14 | SVR Ecran solaire et anti-taches Clairial - CC Crème light SPF 50+ 40ml | 68,88 | 4,2 | SVR | sunscreen | international | 40 ML |
| 15 | Ultrasun Ecran solaire - Antiage et anti-pigmentation - Spf50+ 40 ml | 62,5 | 4,5 | Ultrasun | sunscreen | international | 40 ML |
| 16 | lirene Sun Protection - Emulsion SPF 50+ - 150+25 ML | 39,5 | 4,8 | lirene | sunscreen | international | 25 ML |
| 17 | Doris Ecran Solaire Teinte Beige Claire  -50 Ml | 49 | 3,5 | Doris | sunscreen | international | 50 ML |
| 18 | Daylong Extrême - Lait solaire SPF50+ - 50ml | 49,9 | 5 | Daylong | lotion | international | 50 ML |
| 19 | CYTOL NAT Cytolsun Max - Ecran Solaire Invisible SPF50+ - 50ml | 34 | 3,9 | CYTOL | sunscreen | international | 50 ML |
| 20 | CYTOL NAT CYTOLSUN MAX  - Ecran Solaire Invisible SPF 50+ - 50 ML | 37,8 | 5 | CYTOL | sunscreen | international | 50 ML |
| 21 | CYTOL NAT CYTOLSUN ECRAN MINERAL SPPF50+ TEINTE BEIGE NATUREL 50ML | 29,8 | 3,9 | CYTOL | sunscreen | international | 50 ML |
| 22 | Esth'Elle Ecran solaire - Teinté beige - 50 ml - SPF 50+ | 27,9 | 3 | Esth'Elle | sunscreen | international | 50 ML |
| 23 | SVR Sun secure - Blur - SPF50+ - 50 ml | 63,12 | 5 | SVR | sunscreen | international | 50 ML |
| 24 | Milva Olcare - Ecran Solaire Teinté Beige claire - 50ML - SPF 50+ | 30 | 5 | Milva Olcare | sunscreen | local | 50 ML |
| 25 | Isdin Pediatrics transparent spray pour enfants - Spf50 - 250ml | 64 | 4 | Isdin | spray | international | 250 ML |
| 26 | Dr Rashel Crème solaire - anti-âge SPF100 +++ - 80 gr | 26,99 | 5 | Dr Rashel | sunscreen | international | 80 GR |
| 27 | Daylong Lotion lait solaire - Extreme SPF50+ - 100ml | 74,7 | None | Daylong | lotion | international | 100 ML |

# Getting the client's preferences and sending an email :

```
ort csv
ort smtplib
ort pandas as pd
m selenium import webdriver
m selenium.webdriver.common.by import By
m selenium.webdriver.support.ui import WebDriverWait
m selenium.webdriver.support import expected_conditions as EC
m lxml import html
m selenium.common.exceptions import NoSuchElementException
m selenium.webdriver.common.action_chains import ActionChains
m itertools import zip_longest
m email.mime.multipart import MIMEMultipart
m email.mime.text import MIMEText

ead CSV file
 read_csv(filename):
 data = []
 with open(filename, newline='') as csvfile:
     reader = csv.DictReader(csvfile)
     for row in reader:
         data.append(row)
 return data

unction to calculate score for each row
 calculate_score(row, price_range, brand_preference, brand_category_preference):
 score = 0

 # Price weight calculation
 price = float(row['product price'])
 if price >= price_range[0] and price <= price_range[1]:
     price_weight = 0.9  # High weight if price falls within user's preferred range
 else:
     price weight = 0.1  # Low weight if price falls outside user's preferred range
```

```
# Price weight calculation
price = float(row['product price'])
if price >= price_range[0] and price <= price_range[1]:
    price_weight = 0.9  # High weight if price falls within user's preferred range
else:
    price_weight = 0.1  # Low weight if price falls outside user's preferred range

# Product rate weight calculation
if row['product rate'] == 'None':
    rate_weight = 0.1
else:
    rate_weight = float(row['product rate']) / 5  # Assuming rate varies from 1 to 5, converted to p

# Brand weight calculation
if row['brand'] == brand_preference:
    brand_weight = 0.7  # Higher weight if the row's brand matches user's preference
else:
    brand_weight = 0.3  # Lower weight if the row's brand doesn't match user's preference

# Brand category weight calculation
if row['brand category'] == 'local':
    brand_category_weight = brand_category_preference[0]  # Use user's preference
elif row['brand category'] == 'international':
    brand_category_weight = brand_category_preference[1]  # Use user's preference for international
else:
    brand_category_weight = 0.5  # Default weight if brand category is not specified

# Calculate final score for the row
score = price_weight * rate_weight * brand_weight * brand_category_weight
return score
```

In these scripts, we have developed a system to recommend top products based on user preferences stored in a CSV file. Here's a summary of what the script does:

- Reading CSV File: The script reads product data from a CSV file containing information such as product name, price, rating, brand, and brand category.

- Calculating Scores: It calculates a score for each product based on user-defined preferences, including price range, preferred brand, and brand category. The scoring system assigns weights to these factors and computes a score for each product accordingly.

- Identifying Top Products: After calculating scores for all products, the script identifies the top 5 products with the highest scores.

- Sending Email: It composes an email containing the names of the top 5 products and sends it to the user's email address specified in the script.

- SMTP Configuration: The script utilizes the SMTP protocol to send emails using the smtplib library, configuring the sender's email address, receiver's email address, and login credentials.

```
ain function
main():
# Read CSV file
filename = 'clean output.csv'
data = read_csv(filename)

# Get user preferences
min_price = float(input("Enter minimum price: "))
max_price = float(input("Enter maximum price: "))
brand_preference = input("Enter preferred brand (or leave blank if no preference): ")
brand_category_preference = input("Enter preferred brand category (local/international) (or leave blank if no preference): ").lowe
if brand_category_preference == 'local':
    brand_category_weight = (0.8, 0.2)
elif brand_category_preference == 'international':
    brand_category_weight = (0.2, 0.7)
else:
    brand_category_weight = (0.5, 0.5)

# Calculate weights
price_range = (min_price, max_price)

# Calculate scores for each row
scores = []
for row in data:
    score = calculate_score(row, price_range, brand_preference, brand_category_weight)
    scores.append((row['product name'], score))

# Create dataframe from scores
df_scores = pd.DataFrame(scores, columns=['Product Name', 'Product Score'])

# Sort dataframe by score in descending order
df_scores = df_scores.sort_values(by='Product Score', ascending=False)

# Display top 5 products based on scores
top_5_products = df_scores.head(5)
```

# Output :

## Top 5 Products Based on Score

Inbox ×

**trabelsizeineb@tbs.u-tunis.tn**

to me

The best 5 products found based on your preferences are:

CYTOL NAT CYTOLSUN ECRAN MINERAL SPPF50+ TEINTE BEIGE NATUREL 50ML
Esth'Elle Ecran solaire - TeintÃ© beige - 50 ml - SPF 50+
Dr Rashel CrÃ¨me solaire anti-Ã¢ge SPF 75 - 80 gr
INODERMA Aloha Huile de monoi de Tahiti - 100ml
Dr Rashel CrÃ¨me solaire anti-Ã¢ge SPF 60 - 80 gr

Reply Forward

# Encrypting :

```python
from cryptography.fernet import Fernet # type: ignore

# Generate a key
key = Fernet.generate_key()

# Save the key to a file
with open('encryption_key.key', 'wb') as key_file:
    key_file.write(key)

# Create a Fernet symmetric key
cipher = Fernet(key)

# Read the contents of the clean output CSV file
input_file_path = u'clean output.csv'
with open(input_file_path, 'rb') as file:
    data = file.read()

# Encrypt the data
encrypted_data = cipher.encrypt(data)

# Write the encrypted data to a new file
output_file_path = u'encrypted_output.csv'
with open(output_file_path, 'wb') as file:
    file.write(encrypted_data)

print("File encrypted successfully.")
```

In this script, we demonstrate how to securely encrypt a CSV file using the cryptography library. Here's a summary of what we did:

1. Generate a Key: We generate a symmetric encryption key using Fernet.generate_key(). This key is necessary for both encrypting and decrypting the data.
2. Save the Key: The generated key is saved to a file named encryption_key.key. This ensures that we can later decrypt the data using the same key.
3. Create a Cipher Object: We create a Fernet cipher object with the generated key. This object will be used to encrypt and decrypt the data.
4. Read the CSV File: The script reads the contents of the CSV file named clean_output.csv from the specified directory (D:\clean_output.csv). The file is opened in binary read mode.
5. Encrypt the Data: The read data is then encrypted using the cipher.encrypt(data) method. This method ensures that the CSV file contents are securely transformed into an encrypted format.
6. Write Encrypted Data to a New File: The encrypted data is written to a new file named encrypted_output.csv in the specified directory (D:\encrypted_output.csv). The file is opened in binary write mode to handle the encrypted content properly.
7. Confirmation Message: Finally, the script prints a confirmation message indicating that the file has been successfully encrypted.

# Encrypted Output :

| B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|
| _jzW5PHMvUWmJYDRRwLGdgkucYPGdYPnT-rU4sfiSOePkHKAXb0CzKMzuxBA5Y1jeUN | | | | | | |
| sR2GH9LvAkjY2M5_eMkSPV3lOXg7d_4-DB7MwF4PGsFksuT_341ORSYTpa8C_a2Oz03L | | | | | | |

# Decrypting :

In this script, we demonstrate how to decrypt a previously encrypted CSV file using the cryptography library. Here's a summary of what we did:

1. Load the Key: We load the previously saved symmetric encryption key from the file encryption_key.key. This key is essential for decrypting the encrypted data.
2. Create a Cipher Object: We create a Fernet cipher object using the loaded key. This object will be used to decrypt the data.
3. Read the Encrypted File: The script reads the contents of the encrypted CSV file named encrypted_output.csv from the specified directory (D:\encrypted_output.csv). The file is opened in binary read mode.
4. Decrypt the Data: The read encrypted data is then decrypted using the cipher.decrypt(encrypted_data) method. This method transforms the encrypted content back into its original, readable form.
5. Write Decrypted Data to a New File: The decrypted data is written to a new file named decrypted_output.csv in the specified directory (D:\decrypted_output.csv). The file is opened in binary write mode to handle the decrypted content properly.
6. Confirmation Message: Finally, the script prints a confirmation message indicating that the file has been successfully decrypted.

```python
from cryptography.fernet import Fernet # type: ignore

# Load the key (replace 'key.txt' with the path to your key file)
with open('encryption_key.key', 'rb') as key_file:
    key = key_file.read()

# Create a Fernet symmetric key
cipher = Fernet(key)

# Read the contents of the encrypted file
input_file_path = u'encrypted_output.csv'
with open(input_file_path, 'rb') as file:
    encrypted_data = file.read()

# Decrypt the data
decrypted_data = cipher.decrypt(encrypted_data)

# Write the decrypted data to a new file
output_file_path = u'decrypted_output.csv'
with open(output_file_path, 'wb') as file:
    file.write(decrypted_data)

print("File decrypted successfully.")
```

# Output :

| product name | product price | product rate | brand | product category | brand category | quantity |
|---|---|---|---|---|---|---|
| SVR SUN SECURE BLUR SPF 50+ 50 | 51.59 | 4.1 | SVR | sunscreen | international | 50 ML |
| SVR SUN SECURE BLUR TEINTE SPF | 51.59 | 4 | SVR | sunscreen | international | 50 ML |
| lirene Sun Protection - Emulsion SPI | 39.5 | 4.7 | lirene | sunscreen | international | 25 ML |
| SVR SUN SECURE FLUIDE SPF50+ 5( | 49 | 4.5 | SVR | sunscreen | international | 50 ML |
| Daylong LAIT EXTREME 50+ 50ML | 49.5 | 4 | Daylong | lotion | international | 50 ML |
| Isdin Foto ultra - Ecran invisible ant | 75 | 4.6 | Isdin | other | international | 50 ML |
| Dr Rashel CrÃ¨me solaire anti-Ã¢ge | 30 | 4.5 | Dr Rashel | sunscreen | international | 80 GR |
| Isdin Fotoprotector - Transparent Sp | 58 | 4.6 | Isdin | spray | international | 250 ML |
| Dr Rashel CrÃ¨me solaire anti-Ã¢ge | 30 | 3.9 | Dr Rashel | sunscreen | international | 80 GR |
| Isdin Fotoprotector - Fusion Water - | 55.5 | 4.3 | Isdin | sunscreen | international | 50 ML |
| CYTOL NAT Cytolsun Max - Ecran So | 31 | 4.1 | CYTOL | sunscreen | international | 50 ML |
| Daylong Extreme - Spf 50+ - Lait - 1( | 87.41 | 4.5 | Daylong | lotion | international | 100 ML |
| Pharmaceris Fluid Sand SPF 50+ N:( | 37.3 | 3 | Pharmaceris | other | international | 30 ML |
| Doris Ecran Solaire Teinte Beige Cla | 49 | 4.5 | Doris | sunscreen | international | 50 ML |
| SVR Ecran solaire et anti-taches Cla | 68.88 | 3.4 | SVR | sunscreen | international | 40 ML |
| Isdin Pediatrics transparent spray p( | 64 | 4.8 | Isdin | spray | international | 250 ML |
| Daylong Extreme - Spf 50+ - Lait - 1( | 111.25 | 4.2 | Daylong | lotion | international | 100 ML |
| Ultrasun Ecran solaire - Antiage et a | 62.5 | 3.9 | Ultrasun | sunscreen | international | 40 ML |
| Esth'Elle Ecran solaire - TeintÃ© bei | 27.9 | 5 | Esth'Elle | sunscreen | international | 50 ML |
| CYTOL NAT Cytolsun Max - Ecran So | 39 | 5 | CYTOL | sunscreen | international | 50 ML |
| CYTOL NAT CYTOLSUN ECRAN MINE | 29.8 | 5 | CYTOL | sunscreen | international | 50 ML |
| CYTOL NAT CYTOLSUN MAX - Ecran | 37.8 | 4.5 | CYTOL | sunscreen | international | 50 ML |
| Daylong ExtrÃªme - Lait solaire SPF | 49.9 | 3 | Daylong | lotion | international | 50 ML |
| Milva Olcare - Ecran Solaire TeintÃ© | 30 | 3.5 | Milva Olcare | sunscreen | local | 50 ML |

# Steps Further Explained :

1. **Web Scraping**:
   - **Purpose**: Gather product data from an e-commerce website (Jumia in our case).
   - **How We Did It**:
     - We used Selenium to automate web browsing and scrape product names, prices, and ratings.
     - Data was extracted from multiple pages of search results and stored in a CSV file (**output.csv**).
2. **Data Cleaning and Processing**:
   - **Purpose**: Prepare the scraped data for analysis by removing duplicates, standardizing formats, and extracting relevant information.
   - **How We Did It**:
     - Loaded the CSV data into a pandas frame.
     - Removed duplicate entries.
     - Converted price strings to numeric values.
     - Standardized and extracted ratings, filling in missing values with 'None'.
     - Extracted brand names and categorized products based on keywords (e.g., lotion, spray, sunscreen).
     - Classified brands as either local or international and extracted product quantities from product names.
     - The cleaned data was then saved to a new CSV file (**clean_output.csv**).
3. **Data Encryption**:
   - **Purpose**: Secure the processed data to ensure confidentiality and prevent unauthorized access.
   - **How We Did It**:
     - Generated an encryption key using the **cryptography** library and saved it to a file (**encryption_key.key**).
     - Encrypted the contents of the cleaned CSV file (**clean_output.csv**) and saved the encrypted data to a new file (**encrypted_output.csv**).

# 4-Steps Further Explained :

**4.User Preference Analysis and Scoring:**

- **Purpose**: Analyze the cleaned data based on user input and score products to find the best matches.
- **How We Did It**:
  - Loaded the cleaned data and gathered user preferences for price range, preferred brand, and brand category (local or international).
  - Calculated a score for each product based on how well it matched the user's preferences.
  - Sorted the products by score and selected the top 5 products.

5.**Sending Email with Recommendations:**

- **Purpose**: Communicate the top product recommendations to the user via email.
- **How We Did It**:
  - Created an email message using the **smtplib** and **email** libraries.
  - Configured the email settings and server details.
  - Sent an email to the user with the top 5 product recommendations based on their preferences.

**6.Data Decryption:**

- **Purpose**: Provide a way to securely decrypt and access the encrypted data if needed.
- **How We Did It**:
  - Loaded the encryption key.
  - Decrypted the encrypted CSV file (**encrypted_output.csv**) and saved the decrypted data to a new file (**decrypted_output.csv**).
-

# DASHBOARD :

## Best Sunscreen products

| product name | product price | product rate | brand |
|---|---|---|---|
| La Roche-Posay laboratoires Ecran solaire teinté anti brillance SPF 50+ , 50ml | 115.00 | 5.00 | La Roche-Posay |
| Pharmaceris Pack - Crème - protection- solaire- spf100+ + mousse - nettoyante- 150ml offerte | 90.00 | 5.00 | Pharmaceris |
| Ultrasun Ecran solaire ANTI-AGEING & ANTI-PIGMENTATION SPF 50+, 50ML | 133.65 | 5.00 | Ultrasun |

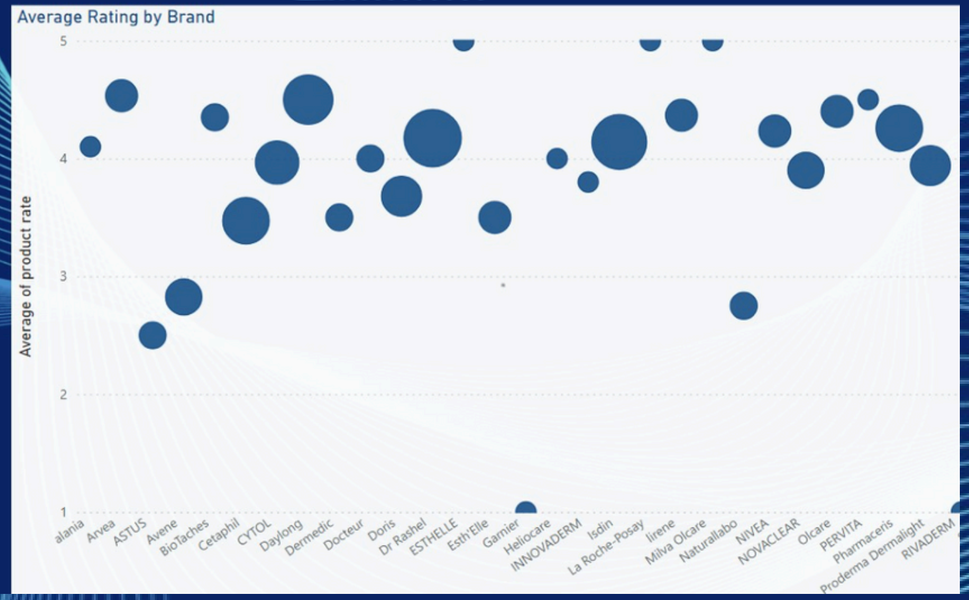
126
Different products

55.31
Average Price in TND

36
Different brands

Number of products by Category


Average Rating by Brand




## Best Spray products

| product name | product price | product rate | brand |
|---|---|---|---|
| Isdin Pediatrics Transparent Spray Pour Enfants Spf50, 250ml | 62.00 | 5.00 | Isdin |
| Heliocare 360 360 Invisible Spray Spf50+ 200 Ml | 81.00 | 4.00 | Heliocare |
| Isdin Pediatrics transparent spray pour enfants - Spf50 - 250ml | 64.00 | 4.00 | Isdin |

## Best Lotion products

| product name | product price | product rate | brand |
|---|---|---|---|
| lirene Pack solaire crème 40ml +lait 175ml + huile sèche + lait bébé | 175.00 | 5.00 | lirene |
| Daylong Actinica Lotion - 80ml | 94.00 | 4.40 | Daylong |
| Daylong Extreme - Spf 50+ - Lait - 100 ml | 111.25 | 4.30 | Daylong |

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

FINAL REPORT

# THANK YOU

# WEB SCRAPING

## CUSTOMER REVIEW ANALYSIS