TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

traveltodo
les prix tout doux

SCRAPE & ESCAPE

A Traveltodo Web Scraping Journey

PROFESSOR

Dr.Manel abdelkader

Prepared By

Bel Haj Fraj Ahmed
Chabbouh Nour

Akrouti Sarra
Majoul Yassmine

TABLE OF CONTENTS

INTRODUCTION

1.TravelTodo Introduction

TravelTodo is a leading Tunisian travel platform that offers hotel bookings, flights, and travel packages, helping users plan and compare travel options easily across Tunisia and beyond.

2. Project Mission

In light of the recent surge in tourism to Tunisia and the viral appeal of the phrase "I'm in Tunisia and you're not," this project, titled **"TravelTodo: Scrape & Escape,"** seeks to explore the role of data-driven tools in enhancing the travel experience. The initiative focuses on securely **scraping** and **analyzing** publicly available hotel data from the TravelTodo platform **to uncover patterns** in pricing, services, and hotel quality across the country. By doing so, it aims to **provide actionable insights** for tourists while emphasizing ethical data collection practices and secure scraping methodologies in a real-world digital environment.
Team

3. Execution Team

Yassmine Majoul

Ahmed Belhadj

Nour Chabbouh

Sarra Akrouti

INTRODUCTION

4. Tools and Software Used

BeautifulSoup

CO

pandas

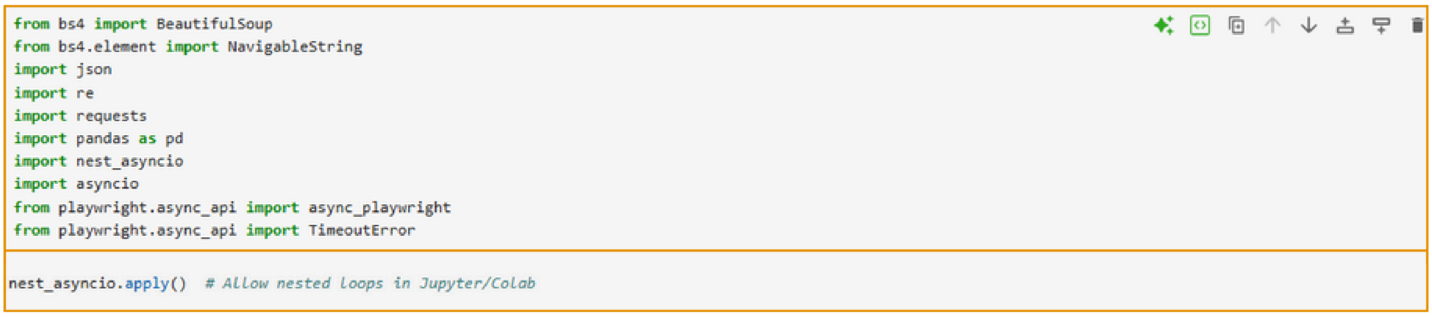
GitHub

jupyter

Power BI

Phase 1 : Data Scraping

Step 1: Setup / Imports

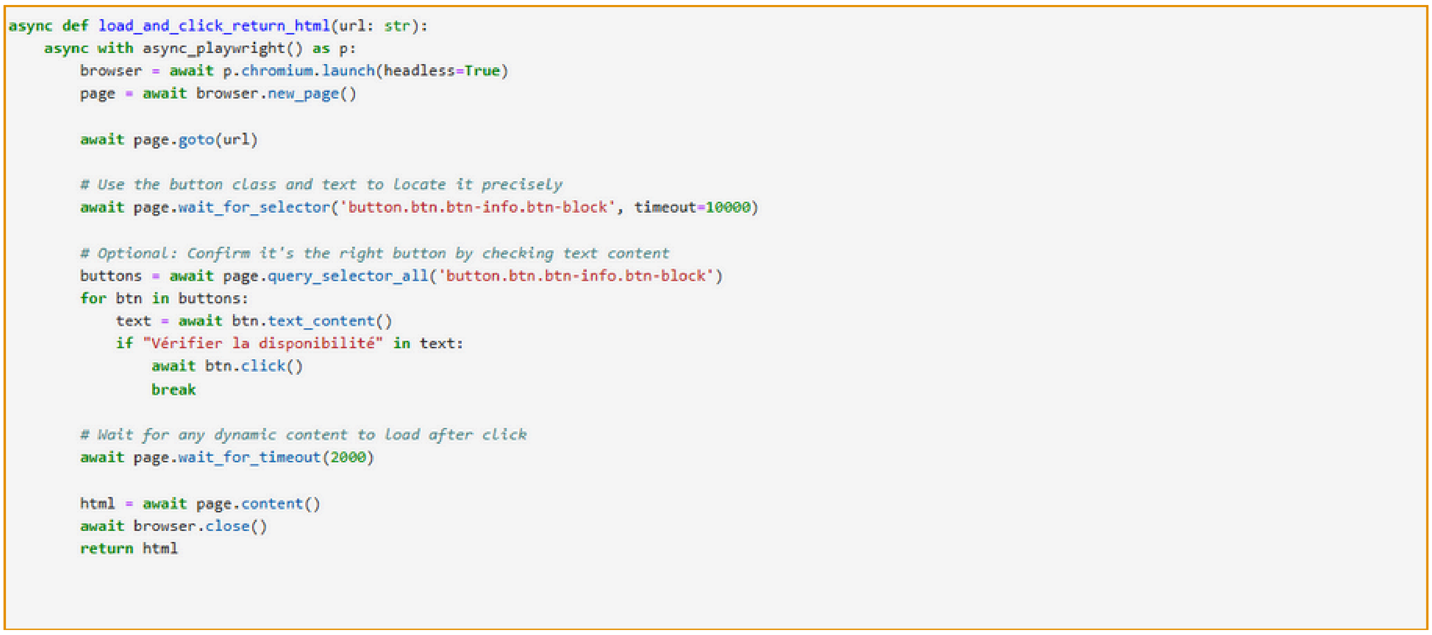


```python
from bs4 import BeautifulSoup
from bs4.element import NavigableString
import json
import re
import requests
import pandas as pd
import nest_asyncio
import asyncio
from playwright.async_api import async_playwright
from playwright.async_api import TimeoutError
```

```python
nest_asyncio.apply()  # Allow nested loops in Jupyter/Colab
```

Step 2: Initial Crawling

To handle dynamic content like **prices** and **availability**, a headless browser is used to load the hotel page, simulate a user click on the **"Check Availability"** button via XPath, wait for the content to render, and then return the complete HTML for **parsing**.

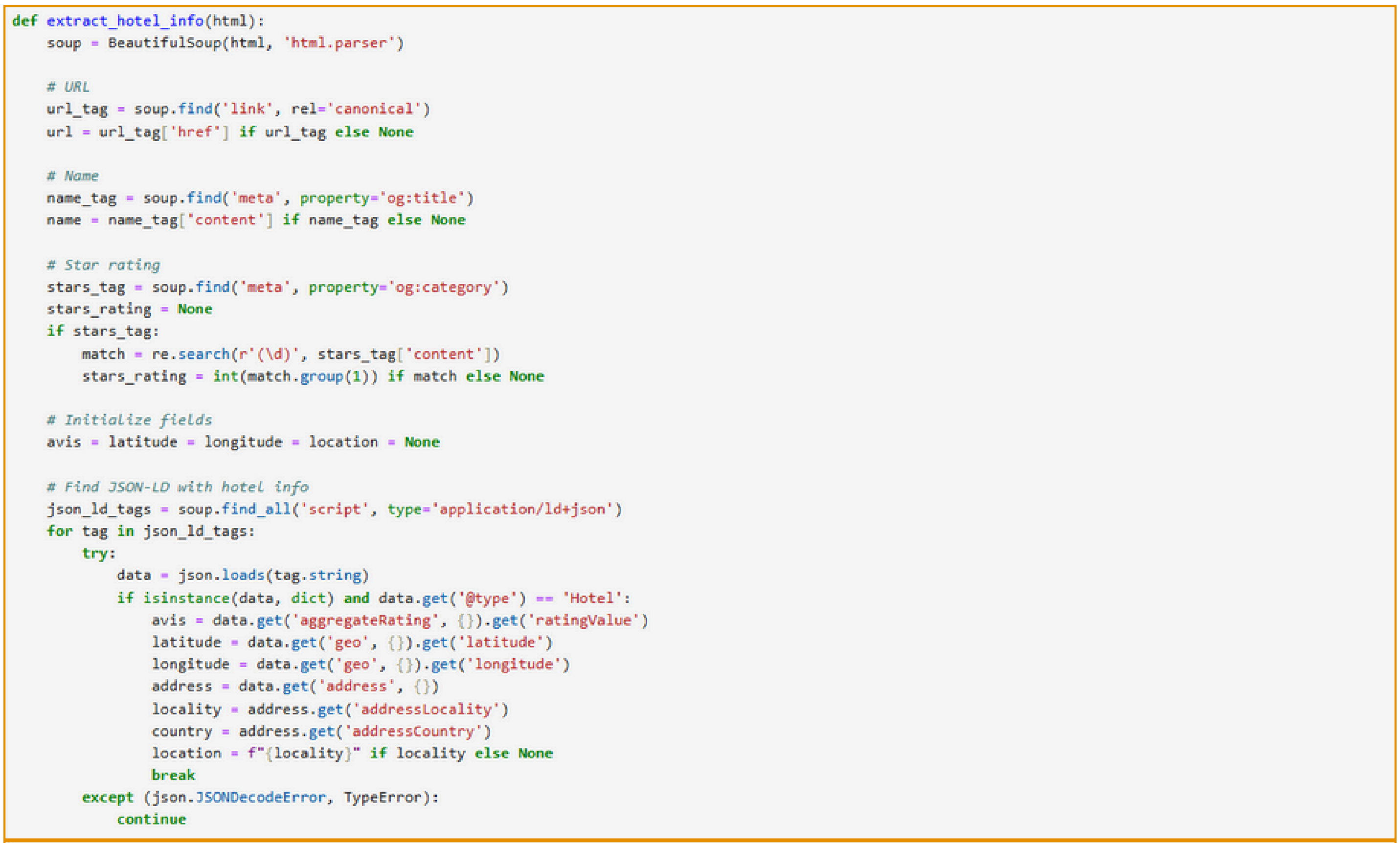


```python
async def load_and_click_return_html(url: str):
    async with async_playwright() as p:
        browser = await p.chromium.launch(headless=True)
        page = await browser.new_page()

        await page.goto(url)

        # Use the button class and text to locate it precisely
        await page.wait_for_selector('button.btn.btn-info.btn-block', timeout=10000)

        # Optional: Confirm it's the right button by checking text content
        buttons = await page.query_selector_all('button.btn.btn-info.btn-block')
        for btn in buttons:
            text = await btn.text_content()
            if "Vérifier la disponibilité" in text:
                await btn.click()
                break

        # Wait for any dynamic content to load after click
        await page.wait_for_timeout(2000)

        html = await page.content()
        await browser.close()
        return html
```

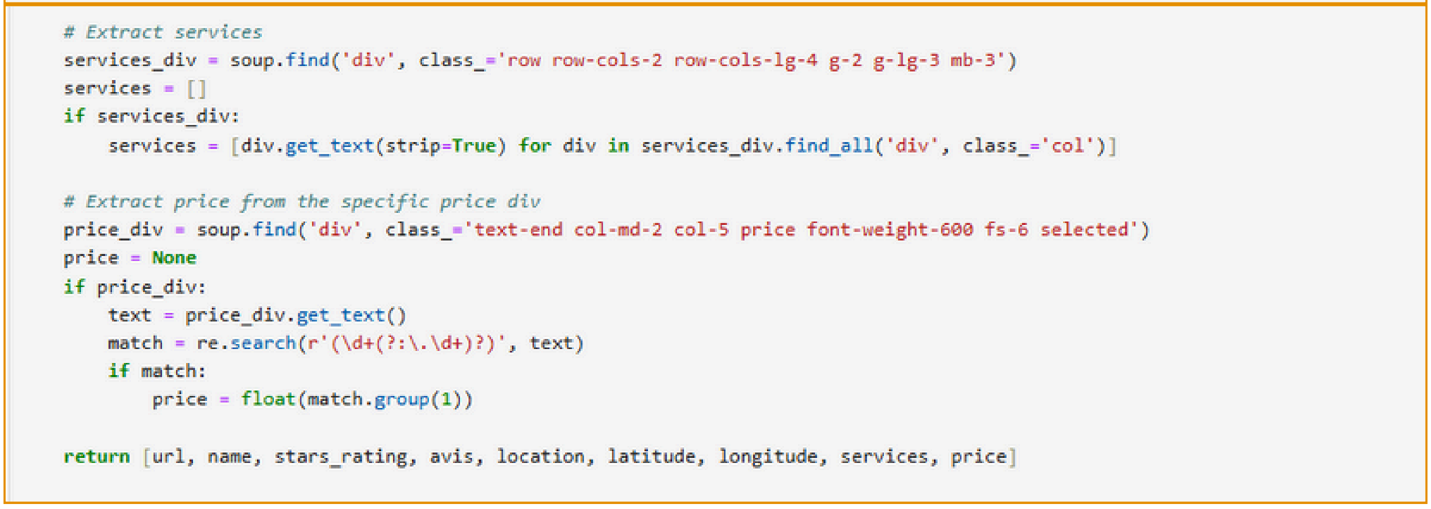
Phase 1 : Data Scraping

Step 3: Hotel Information Parser

After the dynamic content has loaded, this function extracts structured information from the HTML using **BeautifulSoup** and regular expressions. It looks for **hotel name**, **price** in TND, **services** offered (like Wi-Fi, pool, etc.), **review count**, **hotel star rating**, **location**.
—>This step turns raw HTML into usable, meaningful data.



```python
def extract_hotel_info(html):
    soup = BeautifulSoup(html, 'html.parser')

    # URL
    url_tag = soup.find('link', rel='canonical')
    url = url_tag['href'] if url_tag else None

    # Name
    name_tag = soup.find('meta', property='og:title')
    name = name_tag['content'] if name_tag else None

    # Star rating
    stars_tag = soup.find('meta', property='og:category')
    stars_rating = None
    if stars_tag:
        match = re.search(r'(\d)', stars_tag['content'])
        stars_rating = int(match.group(1)) if match else None

    # Initialize fields
    avis = latitude = longitude = location = None

    # Find JSON-LD with hotel info
    json_ld_tags = soup.find_all('script', type='application/ld+json')
    for tag in json_ld_tags:
        try:
            data = json.loads(tag.string)
            if isinstance(data, dict) and data.get('@type') == 'Hotel':
                avis = data.get('aggregateRating', {}).get('ratingValue')
                latitude = data.get('geo', {}).get('latitude')
                longitude = data.get('geo', {}).get('longitude')
                address = data.get('address', {})
                locality = address.get('addressLocality')
                country = address.get('addressCountry')
                location = f"{locality}" if locality else None
                break
        except (json.JSONDecodeError, TypeError):
            continue
```

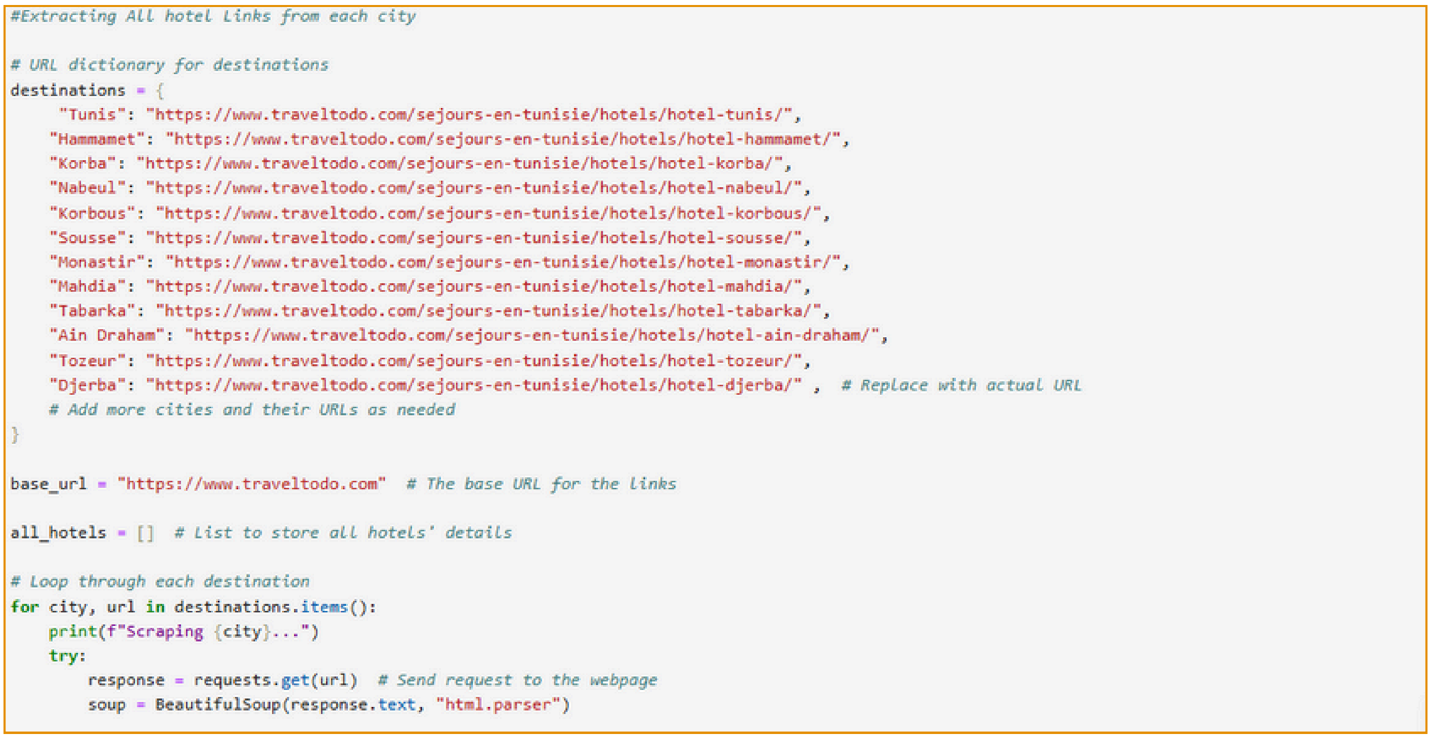


```python
    # Extract services
    services_div = soup.find('div', class_='row row-cols-2 row-cols-lg-4 g-2 g-lg-3 mb-3')
    services = []
    if services_div:
        services = [div.get_text(strip=True) for div in services_div.find_all('div', class_='col')]

    # Extract price from the specific price div
    price_div = soup.find('div', class_='text-end col-md-2 col-5 price font-weight-600 fs-6 selected')
    price = None
    if price_div:
        text = price_div.get_text()
        match = re.search(r'(\d+(?:\.\d+)?)', text)
        if match:
            price = float(match.group(1))

    return [url, name, stars_rating, avis, location, latitude, longitude, services, price]
```

Phase 1 : Data Scraping

Step 4: City Destination Setup

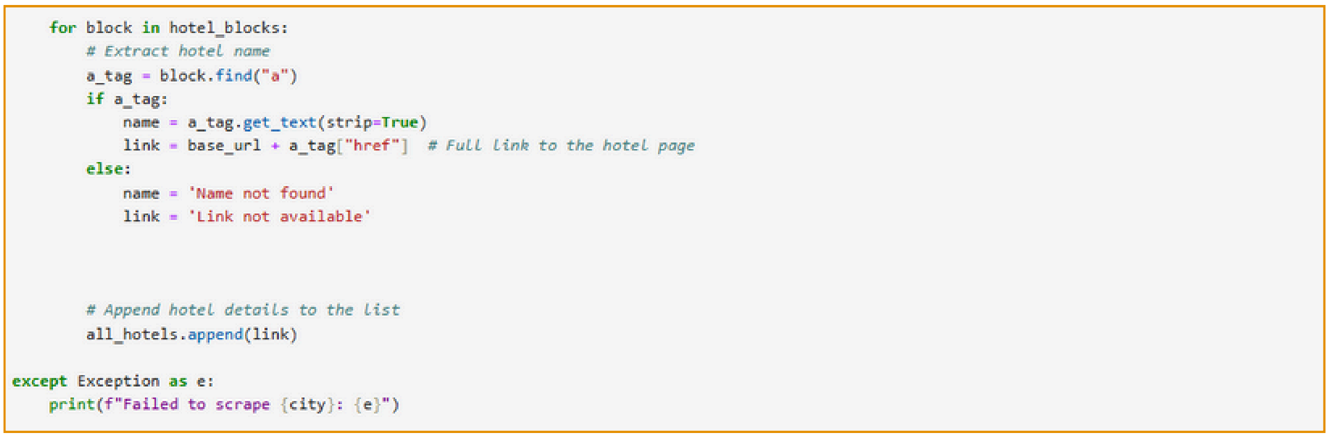
In this step, we defined our scraping scope, which cities(regions) we want to scrape hotel data from. Each **key-value pair** maps a city name to its **respective URL** on the website. This provides **the crawler** with the starting points for each city's hotel list.



```
#Extracting All hotel Links from each city

# URL dictionary for destinations
destinations = {
     "Tunis": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-tunis/",
    "Hammamet": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-hammamet/",
    "Korba": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-korba/",
    "Nabeul": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-nabeul/",
    "Korbous": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-korbous/",
    "Sousse": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-sousse/",
    "Monastir": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-monastir/",
    "Mahdia": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-mahdia/",
    "Tabarka": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-tabarka/",
    "Ain Draham": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-ain-draham/",
    "Tozeur": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-tozeur/",
    "Djerba": "https://www.traveltodo.com/sejours-en-tunisie/hotels/hotel-djerba/" ,  # Replace with actual URL
    # Add more cities and their URLs as needed
}

base_url = "https://www.traveltodo.com"  # The base URL for the links

all_hotels = []  # List to store all hotels' details

# Loop through each destination
for city, url in destinations.items():
    print(f"Scraping {city}...")
    try:
        response = requests.get(url)  # Send request to the webpage
        soup = BeautifulSoup(response.text, "html.parser")
```

Step 5: Crawling for all hotels pages

For each city an **HTTP request** is made to fetch the city's hotel listings page, **BeautifulSoup** is used to parse the HTML and extract **individual** hotel links using specific HTML tags (h3.text-capitalize), each link is **cleaned** and **stored** in **a list** for later access.

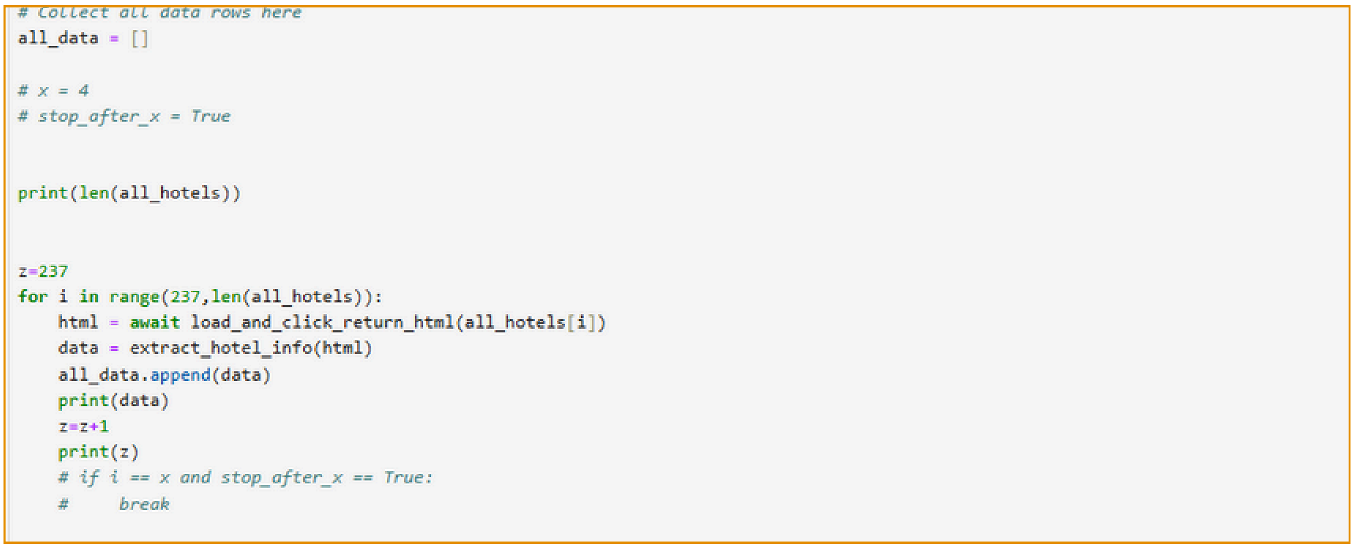


```
    for block in hotel_blocks:
        # Extract hotel name
        a_tag = block.find("a")
        if a_tag:
            name = a_tag.get_text(strip=True)
            link = base_url + a_tag["href"]  # Full link to the hotel page
        else:
            name = 'Name not found'
            link = 'Link not available'



        # Append hotel details to the list
        all_hotels.append(link)

except Exception as e:
    print(f"Failed to scrape {city}: {e}")
```

Phase 1 : Data Scraping

Step 6: Page Fetcher and Data Extractor Loop

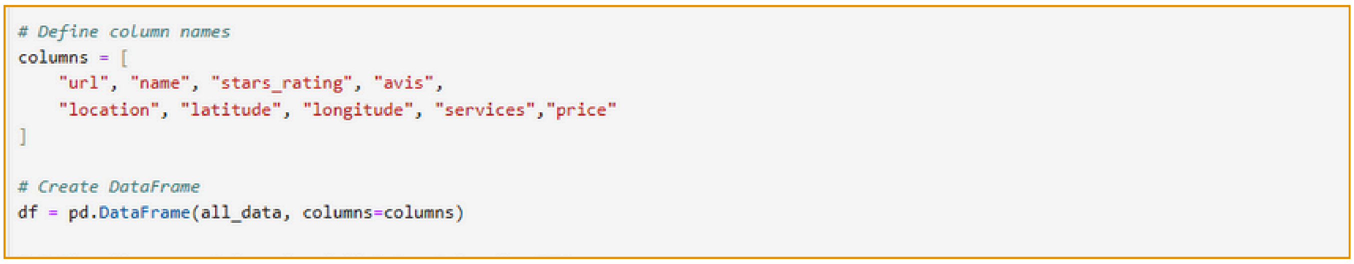
This loop visits every hotel link collected earlier and appends the parsed data to a list.



```
# Collect all data rows here
all_data = []

# x = 4
# stop_after_x = True


print(len(all_hotels))


z=237
for i in range(237,len(all_hotels)):
    html = await load_and_click_return_html(all_hotels[i])
    data = extract_hotel_info(html)
    all_data.append(data)
    print(data)
    z=z+1
    print(z)
    # if i == x and stop_after_x == True:
    #     break
```

Step 7: Data Processing

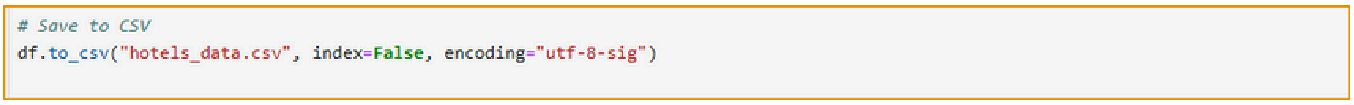
Once all the raw data is collected, we **convert** it into a **structured Data Frame** using pandas. This makes the dataset easier to **analyze**, **filter**, or **export** later. Each row corresponds to a hotel, and each column to an attribute.



```
# Define column names
columns = [
    "url", "name", "stars_rating", "avis",
    "location", "latitude", "longitude", "services","price"
]

# Create DataFrame
df = pd.DataFrame(all_data, columns=columns)
```

Step 8: Export Data to CSV

This final step saves the initial DataFrame as a **CSV file** (hotels_data.csv).

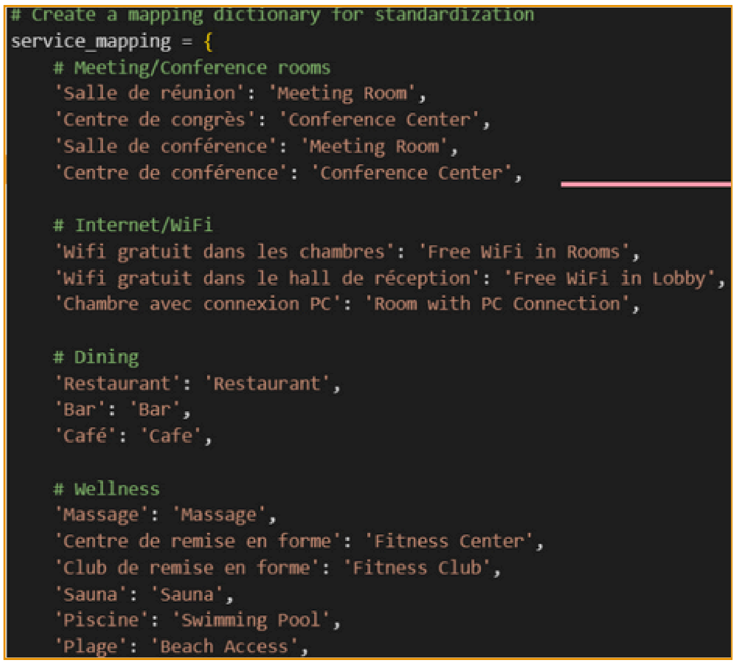


```
# Save to CSV
df.to_csv("hotels_data.csv", index=False, encoding="utf-8-sig")
```

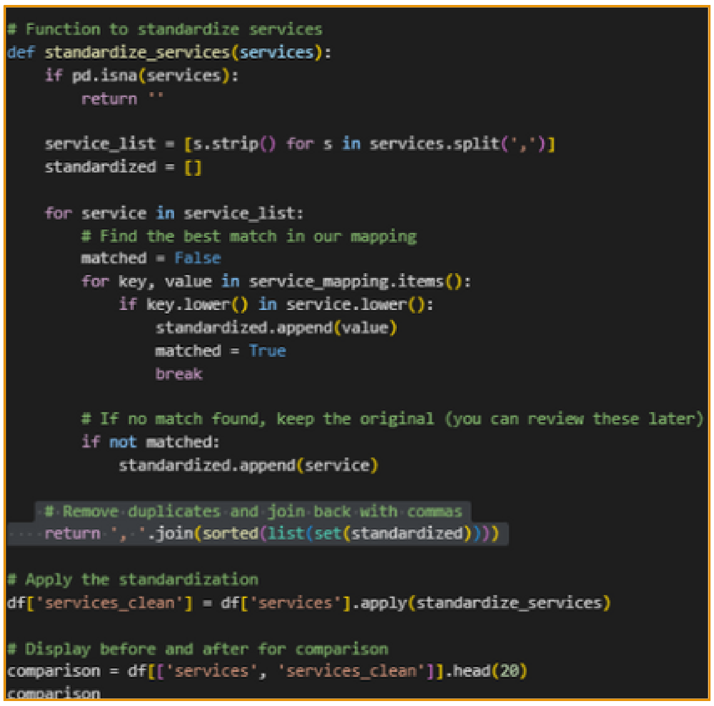
Phase 2 : Data cleaning

Step1 :Data Cleaning

Once the data was extracted, the next step focused on transforming it into a usable format. Special attention was given to cleaning the "services" field, where variations like "salle de conférence" and "salle meeting" were standardized under a unified label such as "meeting room".



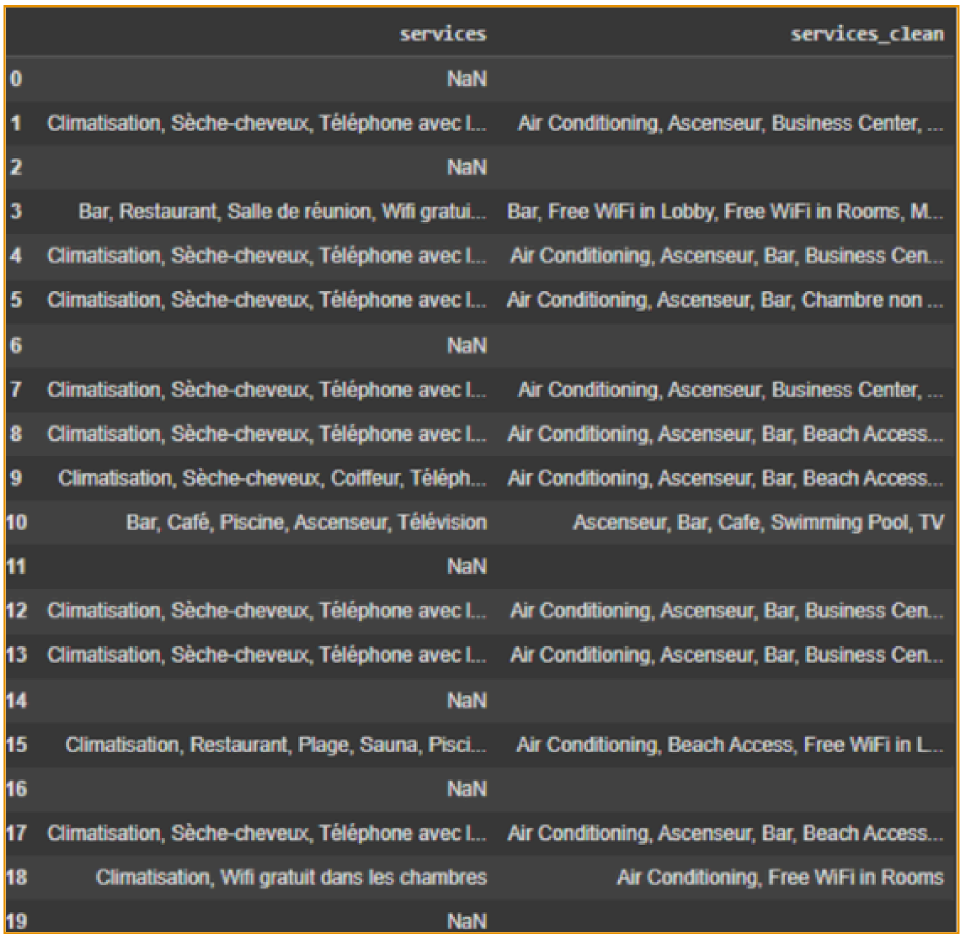
```python
# Create a mapping dictionary for standardization
service_mapping = {
    # Meeting/Conference rooms
    'Salle de réunion': 'Meeting Room',
    'Centre de congrès': 'Conference Center',
    'Salle de conférence': 'Meeting Room',
    'Centre de conférence': 'Conference Center',

    # Internet/WiFi
    'Wifi gratuit dans les chambres': 'Free WiFi in Rooms',
    'Wifi gratuit dans le hall de réception': 'Free WiFi in Lobby',
    'Chambre avec connexion PC': 'Room with PC Connection',

    # Dining
    'Restaurant': 'Restaurant',
    'Bar': 'Bar',
    'Café': 'Cafe',

    # Wellness
    'Massage': 'Massage',
    'Centre de remise en forme': 'Fitness Center',
    'Club de remise en forme': 'Fitness Club',
    'Sauna': 'Sauna',
    'Piscine': 'Swimming Pool',
    'Plage': 'Beach Access',
```



```python
# Function to standardize services
def standardize_services(services):
    if pd.isna(services):
        return ''

    service_list = [s.strip() for s in services.split(',')]
    standardized = []

    for service in service_list:
        # Find the best match in our mapping
        matched = False
        for key, value in service_mapping.items():
            if key.lower() in service.lower():
                standardized.append(value)
                matched = True
                break

        # If no match found, keep the original (you can review these later)
        if not matched:
            standardized.append(service)

    # Remove duplicates and join back with commas
    return ', '.join(sorted(list(set(standardized))))

# Apply the standardization
df['services_clean'] = df['services'].apply(standardize_services)

# Display before and after for comparison
comparison = df[['services', 'services_clean']].head(20)
comparison
```

Phase 2 : Data cleaning

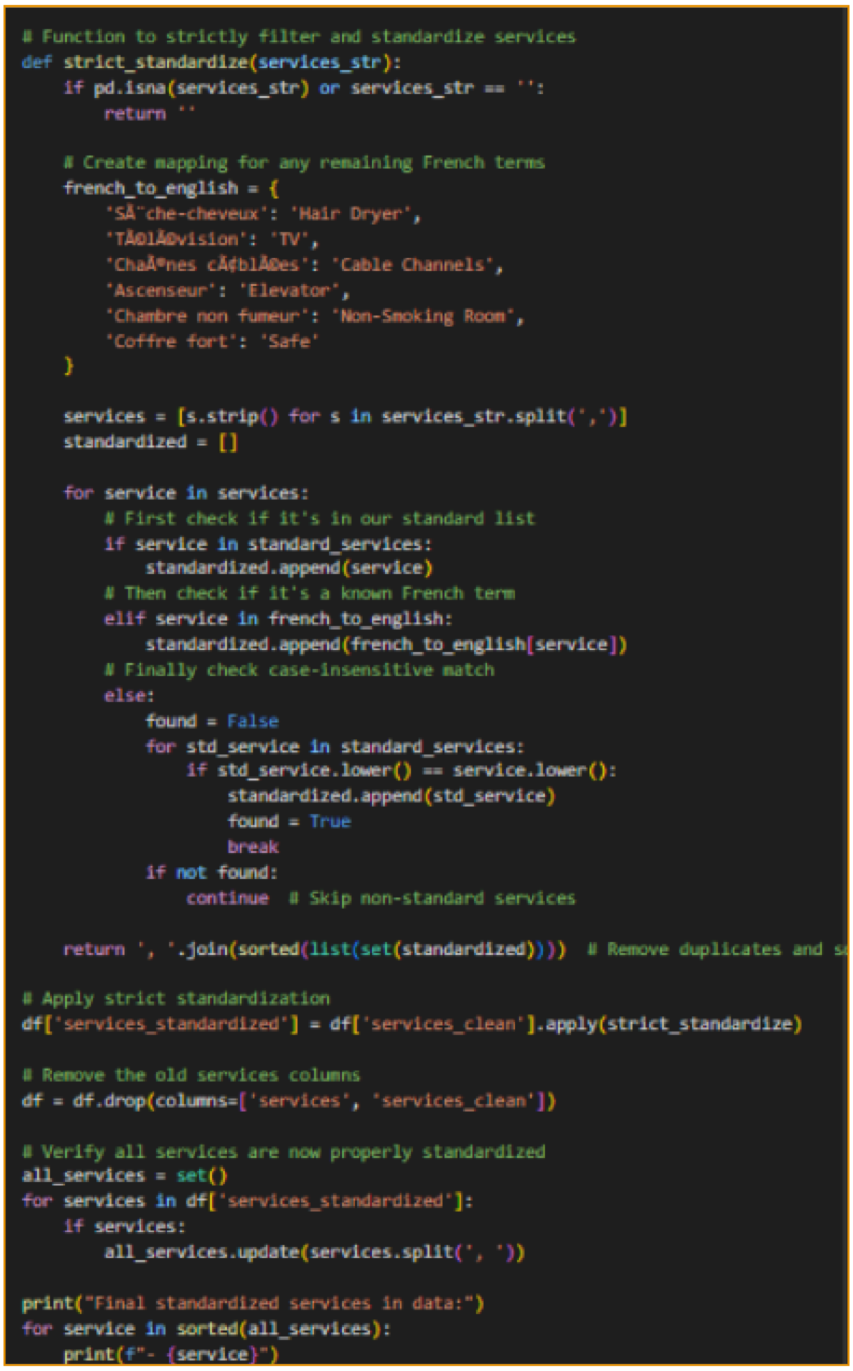
Here is an example of the output :



	services	services_clean
0	NaN	
1	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Business Center, ...
2	NaN	
3	Bar, Restaurant, Salle de réunion, Wifi gratui...	Bar, Free WiFi in Lobby, Free WiFi in Rooms, M...
4	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Business Cen...
5	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Chambre non ...
6	NaN	
7	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Business Center, ...
8	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Beach Access...
9	Climatisation, Sèche-cheveux, Coiffeur, Téléph...	Air Conditioning, Ascenseur, Bar, Beach Access...
10	Bar, Café, Piscine, Ascenseur, Télévision	Ascenseur, Bar, Cafe, Swimming Pool, TV
11	NaN	
12	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Business Cen...
13	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Business Cen...
14	NaN	
15	Climatisation, Restaurant, Plage, Sauna, Pisci...	Air Conditioning, Beach Access, Free WiFi in L...
16	NaN	
17	Climatisation, Sèche-cheveux, Téléphone avec l...	Air Conditioning, Ascenseur, Bar, Beach Access...
18	Climatisation, Wifi gratuit dans les chambres	Air Conditioning, Free WiFi in Rooms
19	NaN	

Phase 2 : Data cleaning

We also addressed issues with empty values and removed duplicate entries to ensure data consistency and accuracy for analysis.

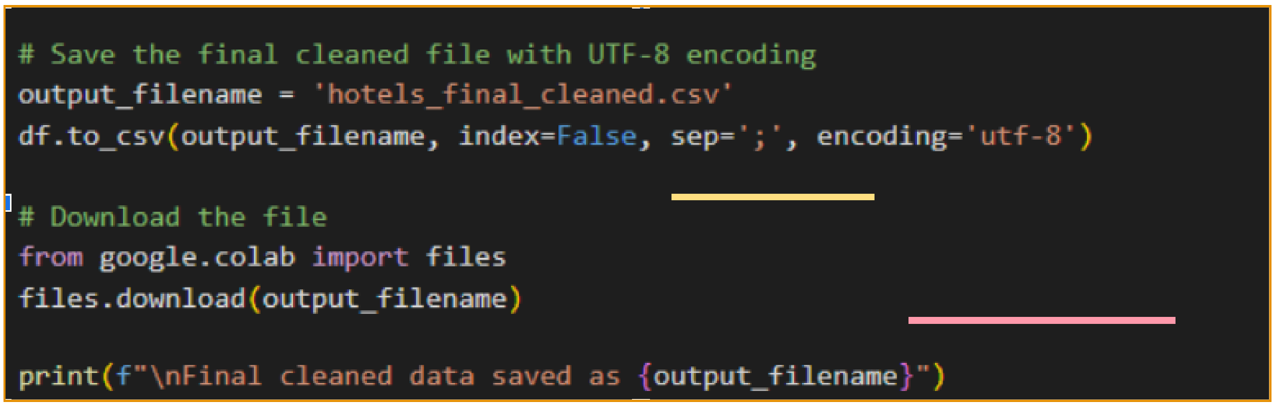


```python
# Function to strictly filter and standardize services
def strict_standardize(services_str):
    if pd.isna(services_str) or services_str == '':
        return ''

    # Create mapping for any remaining French terms
    french_to_english = {
        'SÃ¨che-cheveux': 'Hair Dryer',
        'TÃ©lÃ©vision': 'TV',
        'ChaÃ®nes cÃ¢blÃ©es': 'Cable Channels',
        'Ascenseur': 'Elevator',
        'Chambre non fumeur': 'Non-Smoking Room',
        'Coffre fort': 'Safe'
    }

    services = [s.strip() for s in services_str.split(',')]
    standardized = []

    for service in services:
        # First check if it's in our standard list
        if service in standard_services:
            standardized.append(service)
        # Then check if it's a known French term
        elif service in french_to_english:
            standardized.append(french_to_english[service])
        # Finally check case-insensitive match
        else:
            found = False
            for std_service in standard_services:
                if std_service.lower() == service.lower():
                    standardized.append(std_service)
                    found = True
                    break
            if not found:
                continue  # Skip non-standard services

    return ', '.join(sorted(list(set(standardized))))  # Remove duplicates and s

# Apply strict standardization
df['services_standardized'] = df['services_clean'].apply(strict_standardize)

# Remove the old services columns
df = df.drop(columns=['services', 'services_clean'])

# Verify all services are now properly standardized
all_services = set()
for services in df['services_standardized']:
    if services:
        all_services.update(services.split(', '))

print("Final standardized services in data:")
for service in sorted(all_services):
    print(f"- {service}")
```

Phase 2 : Data cleaning

Step 2 : Data Storage

We finally obtained this final output



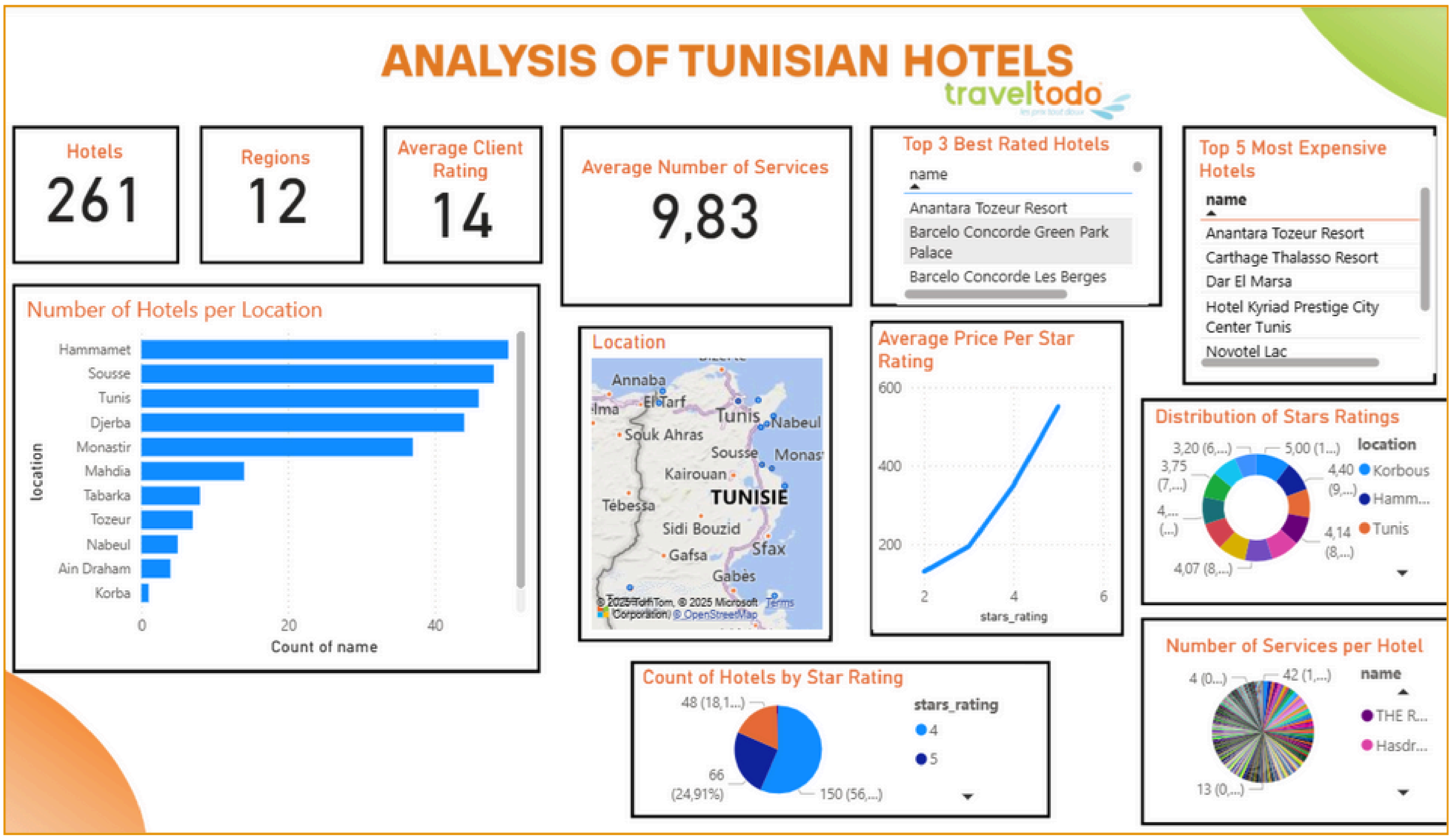
```
# Save the final cleaned file with UTF-8 encoding
output_filename = 'hotels_final_cleaned.csv'
df.to_csv(output_filename, index=False, sep=';', encoding='utf-8')

# Download the file
from google.colab import files
files.download(output_filename)

print(f"\nFinal cleaned data saved as {output_filename}")
```

—>The data has been cleaned, transformed, and loaded, making it ready for the next phase.

Phase 3 : Data Visualization

Now we move to **Power BI** to bring our cleaned Traveltodo data to life and create insightful visuals to better understand trends in pricing, ratings, and distribution across regions.

Drawing insights from **TRAVELTODO** Dashboard :



Some measures were performed using **DAX** (Data Analysis Expressions)

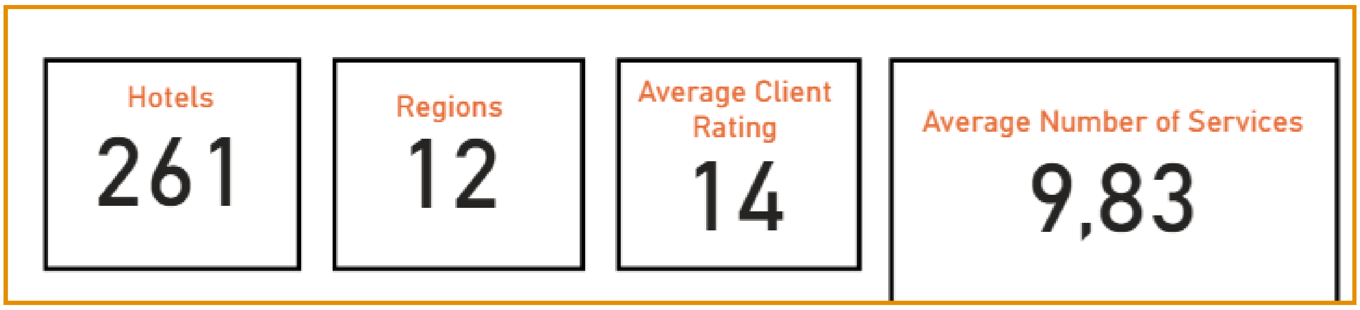
```
PrixMoyenParNote = AVERAGE('hotels_final_cleaned'[price_number])
```

```
MoyenneNombreServices = AVERAGE('hotels_final_cleaned'[NombreServices])
```

Phase 3 : Data Visualization

1. Overall Performance Metrics:

This section offers a high-level view of the hotel landscape in **Tunisia**, reflecting strong service availability across regions .

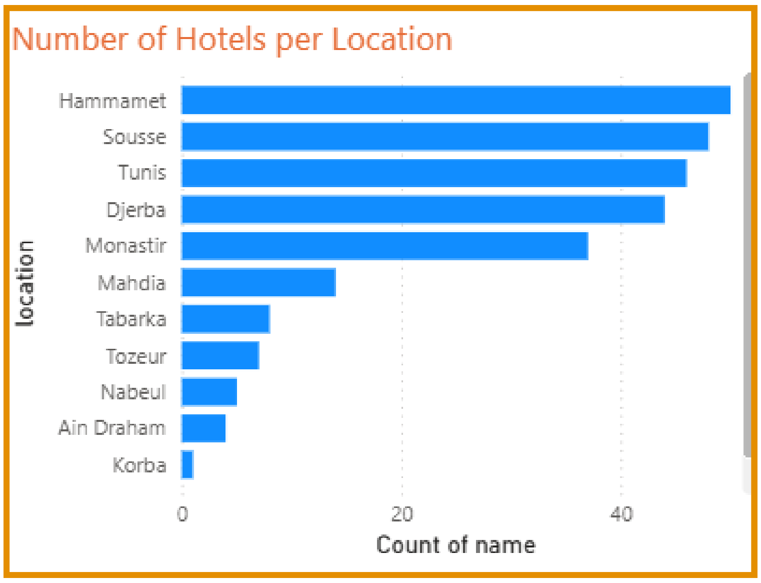


Hotels	Regions	Average Client Rating	Average Number of Services
261	12	14	9,83

2.Insights

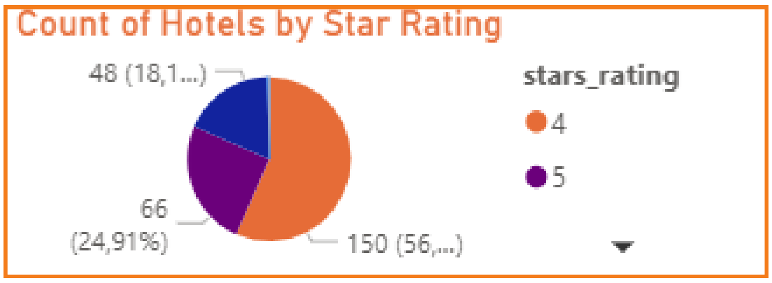
Hotels Distribution by City

From this graph we can conclude that Hammamet, Sousse, and Tunis have the highest hotel counts, indicating they are top tourist destination



Star Ratings

A majority of hotels are rated 4 stars (56.6%), followed by 3 stars (24.91%) suggesting a focus on mid to high end hospitality.

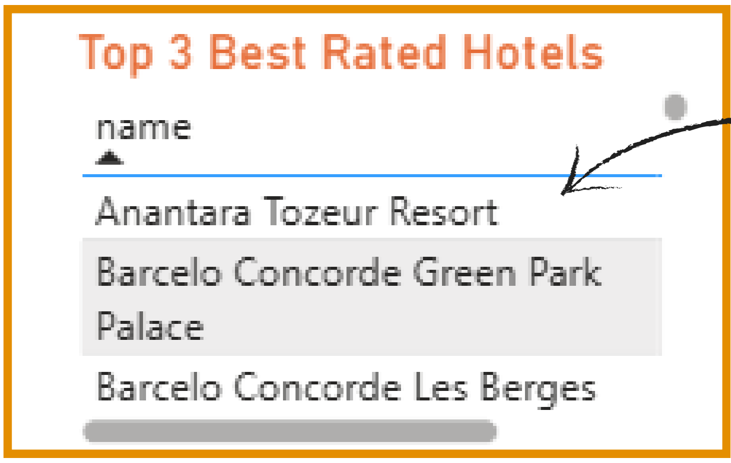


Phase 3 : Data Visualization

2.Insights 2/2

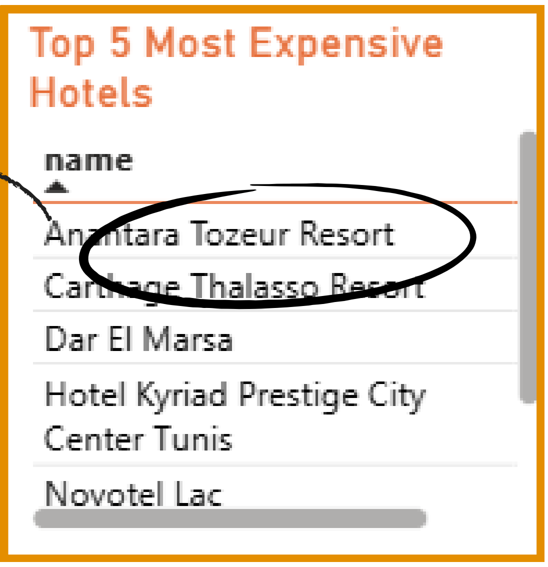
Top-Rated Hotels

Anantara Tozeur Resort appears in both top-rated and most expensive lists, implying a **correlation** between price and perceived quality.



Top 3 Best Rated Hotels

name
Anantara Tozeur Resort
Barcelo Concorde Green Park Palace
Barcelo Concorde Les Berges

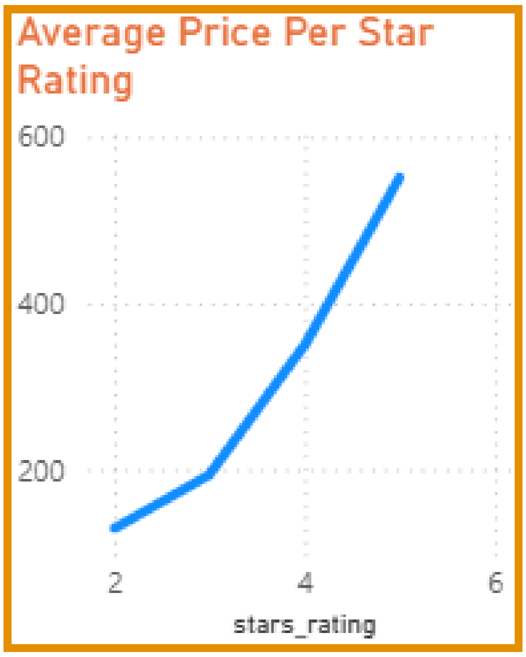


Top 5 Most Expensive Hotels

name
Anantara Tozeur Resort
Carthage Thalasso Resort
Dar El Marsa
Hotel Kyriad Prestige City Center Tunis
Novotel Lac

Average Price per Star Rating

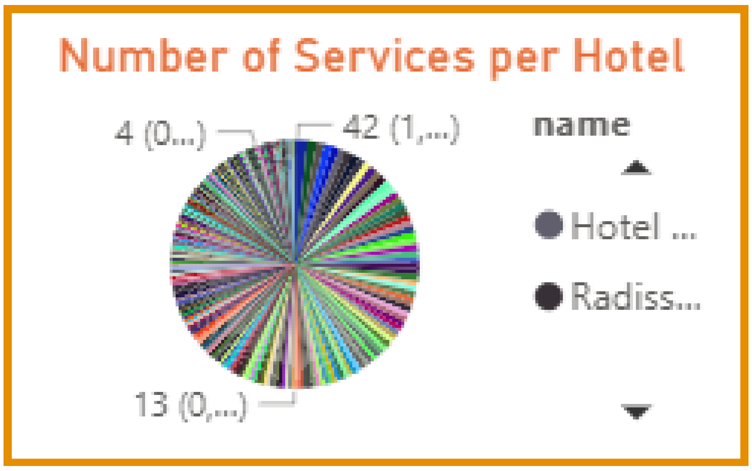
Prices increase steadily with star rating, confirming expected pricing behavior based on hotel class



Phase 3 : Data Visualization

Service Offering

Some hotels, like **Radisson**, offer over 26 services, indicating a strong focus on guest experience.



3.Recommendations for the tourists

Based on the analysis, here are strategic suggestions for travelers:

Best Experience: For luxury seekers, destinations like Tozeur, Tunis, or Hammamet offer top-rated hotels with extensive services

Value for Money: 3 and 4 star hotels in cities like Sousse or Nabeul may provide excellent service quality at more affordable rates

Plan by Region: Coastal cities are more equipped for tourists, while southern and inland regions may offer more authentic, less crowded experiences especially for nature lovers and culture seekers.

Travel Off-Season for Better Deals : Tunisia's tourism peaks in summer. For lower prices and less crowded hotels, consider visiting in spring (April–June) or early fall (September–October)

CONCLUSION

This data analytics project offered a structured view of Tunisia’s hotel landscape by extracting and analyzing data from the Traveltodo platform. Through **web scraping,** cleaning, and insightful Power BI dashboards, we highlighted popular tourist destinations, pricing dynamics, hotel quality distribution, and service offerings. The results not only identify trends in hospitality but also serve as a practical decision making tool for travelers seeking both luxury and value.

In a broader sense, this project reinforces the importance of secure and efficient data collection processes in travel intelligence. Our findings contribute to a better informed, safer travel planning experience aligning with the overarching goal of creating smart, secure, and user-centric tourism insights.