```python
# ================================
# IMPORTS
# ================================
# Import required libraries for web scraping, browser automation, and data handling
import requests
from bs4 import BeautifulSoup
from playwright.sync_api import sync_playwright
import pandas as pd
import time
import os


def scrape_nawy_properties(
    url="https://www.nawy.com/search?page_number=1&category=property",
    scroll_count=100,
    scroll_wait=5,
    scroll_distance=1500,
    container_selector="div.sc-88b4dfdb-0.cgVQXi",
    property_class="sc-100c08da-0 eeBcMz",
    output_path="real_estate_properties.csv"
):
    """
    Scrapes real estate property data from Nawy.com using Playwright and BeautifulSoup.

    Parameters:
    - url (str): The URL of the Nawy search page to scrape.
    - scroll_count (int): Number of times to scroll down to load more properties.
    - scroll_wait (float): Time to wait (in seconds) between each scroll.
    - scroll_distance (int): Pixels to scroll down per iteration.
    - container_selector (str): CSS selector for the scrollable container.
    - property_class (str): Class name of individual property listing elements.
    - output_path (str): File path to save the resulting CSV.

    Returns:
    - pd.DataFrame: DataFrame containing scraped property data.
    """

    # ================================
    # INITIAL SETUP
    # ================================
    # Make an initial GET request (kept for reference, not used in final scraping)
    response = requests.get(url)


    # ================================
    # LAUNCH BROWSER WITH PLAYWRIGHT
    # ================================
    with sync_playwright() as p:
        # Launch the browser in non-headless mode (visible window)
        browser = p.chromium.launch(headless=True)

        # Open a new browser page
        page = browser.new_page()

        # Navigate to the target URL
        page.goto(url)

        # Wait until the scrollable container (which holds the property listings) is loaded
        page.wait_for_selector(container_selector)

        # ================================
        # INFINITE SCROLL SIMULATION
        # ================================
        # Scroll down inside the scrollable container multiple times to load all properties
        for _ in range(scroll_count):
            page.evaluate(f"""
                () => {{
                    const container = document.querySelector('{container_selector}');
                    if (container) {{
                        container.scrollBy(0, {scroll_distance});  // Scroll down by specified pixels
                    }}
                }}
            """)
            time.sleep(scroll_wait)  # Wait for content to load

        # ================================
        # EXTRACT FULL PAGE HTML AFTER SCROLLING
        # ================================
        # After scrolling, retrieve the complete page HTML (now includes dynamically loaded content)
        html = page.content()

        # Parse the HTML using BeautifulSoup for easier data extraction
        soup = BeautifulSoup(html, "html.parser")

        # Find all property listing elements using the specified class name
        Properties = soup.find_all('div', class_=property_class)


        # ================================
        # DATA STORAGE INITIALIZATION
        # ================================
        # Initialize empty lists to store extracted property data
        location_list = []
        name_list = []
        description_list = []
        area_list = []
        bed_list = []
        bath_list = []
        price_list = []


        # ================================
        # EXTRACT DATA FROM EACH PROPERTY
        # ================================
        # Loop through each property element found on the page
        for property in Properties:
            # Extract basic textual information using CSS selectors
            location = property.select_one('div.area')                      # Location of the property
            name = property.select_one('div.name')                          # Name/title of the property
            description = property.select_one('h2.sc-4b9910fd-0.hyACaB') # Description headline
            price = property.select_one('div.price-container span.price') # Price of the property

            # Append text content if element exists; otherwise, append empty string
            location_list.append(location.text.strip() if location else "")
            name_list.append(name.text.strip() if name else "")
            description_list.append(description.text.strip() if description else "")
            price_list.append(price.text.strip() if price else "")

            # Initialize default values for area, beds, and baths
            area_val = ""
            bed_val = ""
            bath_val = ""

            # ================================
            # EXTRACT FEATURE BLOCKS (AREA, BEDS, BATHS)
            # ================================
            # Some properties display additional details in labeled feature blocks
            feature_blocks = property.select("div.sc-234f71bd-0.bbWDeD")  # Select all feature blocks

            # Loop through each feature block (e.g., "m2", "beds", "baths")
            for block in feature_blocks:
                label = block.select_one("span.label")   # Label (e.g., "m2", "beds")
                value = block.select_one("span.value")   # Value (e.g., "120", "3")

                # If both label and value exist, process them
                if label and value:
                    label_text = label.text.strip().lower()  # Normalize label to lowercase
                    value_text = value.text.strip()

                    # Match label to appropriate field and assign value
                    if label_text == "m2":
                        area_val = value_text
                    elif label_text == "beds":
                        bed_val = value_text
                    elif label_text == "baths":
                        bath_val = value_text

            # Append extracted feature values to their respective lists
            area_list.append(area_val)
            bed_list.append(bed_val)
            bath_list.append(bath_val)

        # ================================
        # CLOSE BROWSER
        # ================================
        # Close the browser after scraping is complete
        browser.close()


    # ================================
    # CREATE PANDAS DATAFRAME
    # ================================
    # Combine all data lists into a structured DataFrame
    df = pd.DataFrame({
        'Location': location_list,
        'Name': name_list,
        'Description': description_list,
        'Area': area_list,
        'Beds': bed_list,
        'Baths': bath_list,
        'Price': price_list
    })


    # ================================
    # DEBUG OUTPUT: PRINT SCRAPED DATA
    # ================================
    # Print all collected data to verify successful scraping
    print("Locations:", location_list)
    print("Names:", name_list)
    print("Descriptions:", description_list)
    print("Prices:", price_list)
    print("Areas:", area_list)
    print("Beds:", bed_list)
    print("Baths:", bath_list)


    # ================================
    # DISPLAY DATAFRAME INFO
    # ================================
    # Print basic information about the scraped
```