import sqlite3

import matplotlib.pyplot as plt

from datetime import datetime

from bs4 import BeautifulSoup

import requests

import configparser

import re

# Load configuration settings

config = configparser.ConfigParser()

config.read('config.ini')

DATABASE = config['DEFAULT']['Database']

def connect\_db():

return sqlite3.connect(DATABASE)

def create\_table():

with connect\_db() as conn:

c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS products

(name text, brand text, price real, size real, date text)''')

conn.commit()

class Product:

def \_\_init\_\_(self, name, brand, price, size, date):

self.name = name

self.brand = brand

self.price = price

self.size = size

self.date = date

def save\_to\_db(self):

with connect\_db() as conn:

c = conn.cursor()

c.execute('INSERT INTO products VALUES (?, ?, ?, ?, ?)',

(self.name, self.brand, self.price, self.size, self.date))

conn.commit()

def validate\_input(prompt, input\_type=float):

while True:

user\_input = input(prompt)

try:

return input\_type(user\_input)

except ValueError:

print(f"Invalid input. Please enter a valid {input\_type.\_\_name\_\_}.")

def add\_product():

name = input("Enter product name: ")

brand = input("Enter brand: ")

price = validate\_input("Enter price: ")

size = validate\_input("Enter size: ")

date = datetime.now().strftime("%Y-%m-%d")

product = Product(name, brand, price, size, date)

product.save\_to\_db()

def detect\_shrinkflation():

with connect\_db() as conn:

c = conn.cursor()

c.execute('SELECT \* FROM products')

products = c.fetchall()

product\_dict = {}

for product in products:

key = (product[0], product[1])

if key not in product\_dict:

product\_dict[key] = product

else:

old\_product = product\_dict[key]

if old\_product[3] > product[3] and old\_product[2] <= product[2]:

print(f"Shrinkflation detected: {product[0]} {product[1]}")

print(f"Old size: {old\_product[3]}, New size: {product[3]}")

print(f"Old price: {old\_product[2]}, New price: {product[2]}")

def plot\_prices(product\_name):

with connect\_db() as conn:

c = conn.cursor()

c.execute('SELECT date, price FROM products WHERE name=?', (product\_name,))

data = c.fetchall()

dates = [datetime.strptime(date, "%Y-%m-%d") for date, \_ in data]

prices = [price for \_, price in data]

plt.plot(dates, prices)

plt.title(f"Price over Time for {product\_name}")

plt.xlabel("Date")

plt.ylabel("Price")

plt.show()

def validate\_url(url):

regex = re.compile(

r'^(?:http|ftp)s?://' # http:// or https://

r'(?:(?:A-Z0-9?\.)+(?:[A-Z]{2,6}\.?|[A-Z0-9-]{2,}\.?)|' # domain...

r'localhost|' # localhost...

r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|' # ...or ipv4

r'\[?[A-F0-9]\*:[A-F0-9:]+\]?)' # ...or ipv6

r'(?::\d+)?' # optional port

r'(?:/?|[/?]\S+)$', re.IGNORECASE)

return re.match(regex, url) is not None

def scrape\_product\_info(url):

if not validate\_url(url):

print("Invalid URL format.")

return

try:

page = requests.get(url)

page.raise\_for\_status()

soup = BeautifulSoup(page.content, 'html.parser')

product\_name = soup.find("h1", class\_="product-title").get\_text().strip()

product\_price = soup.find("span", class\_="price").get\_text().strip()

return {"name": product\_name, "price": float(product\_price[1:])}

except requests.exceptions.HTTPError as e:

print(f"HTTP error occurred: {e}")

except Exception as e:

print(f"An error occurred: {e}")

def main():

create\_table()

while True:

print("1. Add a new product")

print("2. Check for shrinkflation")

print("3. Plot product prices")

print("4. Scrape product info")

choice = input("Enter your choice: ")

if choice == '1':

add\_product()

elif choice == '2':

detect\_shrinkflation()

elif choice == '3':

product\_name = input("Enter the product name to plot: ")

plot\_prices(product\_name)

elif choice == '4':

url = input("Enter the URL of the product to scrape: ")

product\_info = scrape\_product\_info(url)

if product\_info:

print(f"Scraped product info: {product\_info}")

else:

print("Exiting the program.")

break

if \_\_name\_\_ == "\_\_main\_\_":

main()