import datetime

from datetime import date, timedelta

#Change below inputs

search\_word = "HKD" #HKD or CNY

d1 = date(2024, 1, 1) # the start date

d2 = date(2024, 1, 31) # the end date

login\_id = "angela\_kw\_sze@hkma.gov.hk"

login\_pw = "Hkma1488!"

driver\_PATH = '/mnt/prototypehkmastorage1/DO-Data Science/utils/chromedriver1'

#import libraries

import time

import random

import glob

import urllib.parse as urllib

import pandas as pd

from time import gmtime, strftime

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.common.exceptions import TimeoutException

from selenium.webdriver.common.keys import Keys

from datetime import date, timedelta

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.action\_chains import ActionChains

#Global Variables:

error = ""

input\_table = pd.DataFrame()

output\_table = pd.DataFrame()

def Test\_Multiple\_Currencies():

#2

ActionChains(driver).send\_keys('USD').perform()

time.sleep(1)

ActionChains(driver).send\_keys(Keys.RETURN).perform()

#3

time.sleep(2)

ActionChains(driver).send\_keys('CNY').perform()

time.sleep(1)

ActionChains(driver).send\_keys(Keys.RETURN).perform()

#4

time.sleep(2)

ActionChains(driver).send\_keys('JPY').perform()

time.sleep(1)

ActionChains(driver).send\_keys(Keys.RETURN).perform()

#5

time.sleep(2)

ActionChains(driver).send\_keys('GBP').perform()

time.sleep(1)

ActionChains(driver).send\_keys(Keys.RETURN).perform()

def update\_DF():

global error, input\_table, output\_table

error = driver.find\_element("xpath","//\*[@id='error']").text

if error == "":

input\_table = pd.read\_html(driver.page\_source, attrs={'class': 'sortable'})[0]

output\_table = pd.concat([output\_table, input\_table])

#Program starts here

delta = d2 - d1

one\_day = datetime.timedelta(days=1)

one\_mo = datetime.timedelta(weeks=4)

one\_year = datetime.timedelta(weeks=48)

Cant\_use\_Method\_1 = False

warning = False

# Setting options

chrome\_options = webdriver.ChromeOptions()

chrome\_options.add\_argument("--headless") # Run Chrome in headless mode (without opening a browser window)

chrome\_options.add\_argument("--disable-gpu") # Disable GPU acceleration

chrome\_options.add\_argument("--start-maximized") # Start Chrome maximized

chrome\_options.add\_argument("--incognito") # Start Chrome in incognito mode

driver = webdriver.Chrome(service = Service(driver\_PATH), options = chrome\_options)

url = "https://my.euroclear.com/apps/en/bank-securities-search.html"

#wait = WebDriverWait(driver, 30)

#Accept cookies & Enter credentials

driver.get(url)

print("start website...")

time.sleep(random.uniform(3.5,4.5))

driver.find\_element("xpath","//\*[@id='js-modal-content']/div[1]/div[3]/div/div[1]/button/span").click()

driver.find\_element("xpath","//\*[@id='username']").send\_keys(login\_id)

driver.find\_element("xpath","//\*[@id='password']").send\_keys(login\_pw)

driver.find\_element("xpath","//\*[@id='password']").send\_keys(Keys.RETURN)

time.sleep(random.uniform(3.5,4.5))

driver.find\_element("xpath","//\*[@class='searchToggler \_moreOptions']").click()

time.sleep(random.uniform(3.5,4.5))

dloop = 0

retryno = 0

while dloop < (delta.days + 1):

dloop\_previous = pd.to\_numeric(dloop)

#reset

driver.find\_element("xpath","//\*[@type='reset']").click()

#Date Input

date = d1 + dloop\*one\_day

datestr = date.strftime('%d/%m/%y')

driver.find\_element("xpath","//\*[@id='firstclosingdate\_1\_container']").click()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(datestr).perform()

driver.find\_element("xpath","//\*[@id='firstclosingdate\_2\_container']").click()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(datestr).perform()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

driver.find\_element("xpath","//\*[@class='securitytype jsonDropdown']").click()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys("Debt: all").perform()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

#Currency Input

driver.find\_element("xpath","//\*[@class='currency jsonDropdown']").click()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(search\_word).perform()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4,5) + retryno\*30)

#Test\_Multiple\_Currencies()

now\_test = True #for testing purposes

#Results

error = driver.find\_element("xpath","//\*[@id='error']").text

if "Too many results" in error: #or now\_test == True:

#Check if it is workable to divide the maturity period into [<=1 year], [>1 year]

#Mature within 1 year

temp\_table1 = pd.DataFrame()

temp\_table2 = pd.DataFrame()

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn']").click()

time.sleep(random.uniform(0.3,0.5))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn\_o\_0']").click()

time.sleep(random.uniform(0.3,0.5))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").clear()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_2\_date']").clear()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").click()

time.sleep(random.uniform(1,1.2))

ActionChains(driver).send\_keys(date.strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(1,1.2))

mday0 = date + 1\*one\_year

driver.find\_element("xpath","//\*[@id='maturitydate\_2\_date']").click()

time.sleep(random.uniform(1,1.2))

ActionChains(driver).send\_keys(mday0.strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(1,1.2))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4.5,5))

if "Too many results" in driver.find\_element("xpath","//\*[@id='error']").text:

Cant\_use\_Method\_1 = True

print(1)

else:

if "not been found" not in driver.find\_element("xpath","//\*[@id='error']").text:

temp\_table1 = pd.read\_html(driver.page\_source, attrs={'class': 'sortable'})[0]

#Mature after than 1 year or more

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn']").click()

time.sleep(random.uniform(0.3,0.5))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn\_o\_2']").click()

time.sleep(random.uniform(0.3,0.5))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").clear()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").click()

time.sleep(random.uniform(0.6,0.8))

ActionChains(driver).send\_keys(mday0.strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(1,1.2))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4.5,5))

if "Too many results" in driver.find\_element("xpath","//\*[@id='error']").text:

Cant\_use\_Method\_1 = True

print(1)

else:

if "not been found" not in driver.find\_element("xpath","//\*[@id='error']").text:

temp\_table2 = pd.read\_html(driver.page\_source, attrs={'class': 'sortable'})[0]

if Cant\_use\_Method\_1 == False:

output\_table = pd.concat([output\_table, temp\_table1, temp\_table2])

else:

#Method 2: [>2 years], [every 3-month in the remaining period]

#Capture all entities maturing after 2 years or more

time.sleep(random.uniform(1,1.2))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn']").click()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn\_o\_2']").click()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").clear()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").click()

time.sleep(random.uniform(0.5,0.6))

mday1 = date + 2\*one\_year

ActionChains(driver).send\_keys(mday1.strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(1.1,1.2))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4.5,5))

update\_DF()

if error == 'Too many results':

warning = True

#break

#Capture all entities maturing on the ceiling day

time.sleep(random.uniform(1,1.2))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn']").click()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn\_o\_3']").click()

time.sleep(random.uniform(0.5,0.6))

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").click()

time.sleep(random.uniform(0.5,0.6))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4.5,5))

update\_DF()

if error == 'Too many results':

warning = True

#break

#Capture all entities maturing within every 1-month in the remaining period

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn']").click()

time.sleep(random.uniform(0.2,0.3))

driver.find\_element("xpath","//\*[@id='modifier\_maturitydate\_chzn\_o\_0']").click()

while mday1 - 3\*one\_mo >= date:

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").clear()

driver.find\_element("xpath","//\*[@id='maturitydate\_2\_date']").clear()

time.sleep(random.uniform(0.95,1.1))

driver.find\_element("xpath","//\*[@id='maturitydate\_2\_date']").click()

time.sleep(random.uniform(0.2,0.3))

ActionChains(driver).send\_keys((mday1-one\_day).strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(0.95,1.1))

mday1 = mday1 - 3\*one\_mo

driver.find\_element("xpath","//\*[@id='maturitydate\_1\_date']").click()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(mday1.strftime('%d/%m/%y')).perform()

time.sleep(random.uniform(0.95,1.1))

ActionChains(driver).send\_keys(Keys.RETURN).perform()

time.sleep(random.uniform(4.5,5))

update\_DF()

if error == 'Too many results':

warning = True

#break

if warning == True:

print("!!! unable to scrap data for ", datestr, " please input data manually")

else:

print(datestr, " Done. has too many results, please double check manually") # has too many results, please double check manually

elif error == "":

update\_DF()

print(datestr, " Done.")

elif "not been found" in error:

retryno += 1

if date.weekday() < 5:

if retryno <=3:

print(datestr, " No results. Weekday, retrying...(added ", retryno\*30, " sec waiting time.)")

dloop -= 1

else:

print(datestr, " No results. Weekday, Please double check manually")

else:

print(datestr, " No results.")

else:

print(datestr, " ERR. Check code")

dloop += 1

if dloop != dloop\_previous:

retryno = 0

#Save File

filename = "Euroclear\_" + search\_word + "\_" + str(d1.strftime("%y%m%d")) + "\_" + str(d2.strftime("%y%m%d"))

output\_table.to\_csv(filename + '.csv', columns=['Name','ISIN','Common code','Rate','Nominal currency','First Closing','Payment date','Record date','Market','Instrument','Last update'], index=False)