Created by Hugsney

Glide Report Automation

In [ ]:

```
# Create a list with all refund id information
import pandas as pd
import os
import shutil
import xlsxwriter
from openpyxl import load_workbook
from openpyxl.utils.dataframe import dataframe_to_rows


# Reading the refund spreadsheet
glide_sheet = pd.read_excel('Newpark - Glide trasactions.xlsx')

#naming columns to have date fixed
date_tobefixed = ['Start Date', 'End Date']

#looping for the function with the dates columns
for col in date_tobefixed:
    #fix the dates to be added on Selenium
    glide_sheet[col] = glide_sheet[col].dt.strftime('%d/%m/%Y')

print(glide_sheet.head())
print(f'\nThere are {len(glide_sheet)} reports to be downloaded

# Create a list for each necessary column
link_ref = glide_sheet['WEB link'].tolist()
username_ref = glide_sheet['Username'].tolist()
password_ref = glide_sheet['Password'].tolist()
report_button_ref = glide_sheet['run report - 54'].tolist()
start_date_ref = glide_sheet['Start Date'].tolist()
end_date_ref = glide_sheet['End Date'].tolist()
site_name_ref = glide_sheet['Site'].tolist()
machine_name = glide_sheet['Machine Name'].tolist()
area_desc = glide_sheet['Area'].tolist()
```

Open a controlled chrome

In [ ]:

```
# Libraries imported from Selenium - automation
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as E
from selenium.webdriver.common.keys import Keys
import time

# download and install the latest version of ChromeDriver
#driver_path = ChromeDriverManager(version='115.0.835.40').inst
#driver_path = ChromeDriverManager().install()
driver_path= "chromedriver-mac-arm64/chromedriver"


# initialize the webdriver and open the webpage
service = Service(driver_path)
navegador = webdriver.Chrome(service=service)

# display or not display the robot working
#chrome_options = Options()
#chrome_options.add_argument("--headless")
#navegador = webdriver.Chrome(service=service, options=chrome_o
```

In [ ]:

```
# Location to receive the files downloaded
downloads_folder = "/Users/hugsneyf/Downloads"

# Location to save raw files
destination_folder = "/Users/hugsneyf/Downloads/GroupNexus/pyth

# Loop from all necessary column
for link, username, password, report_button, start_date, end_da
    username_ref, password_ref, report_button_ref, start_date_
    print(f"\nDonwloading report: {final_site} - Please wait")

    # Reads each report_manager link in the spreadsheet
    navegador.get(link)
    time.sleep(2)

    # Use the credentials to login the website
    input_user = navegador.find_element(By.ID, "MainContent_Log
    input_password = navegador.find_element(By.ID, "MainConten
    login_button = navegador.find_element(By.ID, 'MainContent_L
    navegador.execute_script("arguments[0].click();", login_but
    time.sleep(3)

    # Open report "54"
    report_button_54 = navegador.find_element(By.ID, report_but
```

```
    navegador.execute_script("arguments[0].click();", report_bu
    time.sleep(3)

    # Clean and fill the start date field
    date_start_box = navegador.find_element(By.ID, "MainConten
    date_start_box = navegador.find_element(By.ID, "MainConten
    time.sleep(1)

    # Select 00 start hour time
    dropdown_hour_00 = navegador.find_element(By.ID, "MainConte
    navegador.execute_script(f"arguments[0].selectedIndex = 0;'
    time.sleep(1)

    # Select 00 start minute time
    dropdown_min_00 = navegador.find_element(By.ID, "MainConter
    navegador.execute_script(f"arguments[0].selectedIndex = 0;'
    time.sleep(1)

    # Clean and fill the final date field
    date_start_box = navegador.find_element(By.ID, "MainConten
    date_start_box = navegador.find_element(By.ID, "MainConten
    time.sleep(1)

    # Select 23 final hour time
    dropdown_hour_23 = navegador.find_element(By.ID, "MainConte
    navegador.execute_script(f"arguments[0].selectedIndex = 23;
    time.sleep(2)

    # Select 59 final minute time
    dropdown_min_59 = navegador.find_element(By.ID, "MainConter
    navegador.execute_script(f"arguments[0].selectedIndex = 59;
    time.sleep(1)


    # Select Glide report from the dropdown and click ok
    dropdown_glide = navegador.find_element(By.ID, "MainConten
    navegador.execute_script(f"arguments[0].value = '{machine}'
    time.sleep(1)

    # Select area / site
    area_name = navegador.find_element(By.ID, "MainContent_Area
    navegador.execute_script(f"arguments[0].value = '{area}';",
    time.sleep(1)

    # Press ok buttom
    ok_button = navegador.find_element(By.ID, 'MainContent_OkBu
    navegador.execute_script("arguments[0].click();", ok_buttor
    time.sleep(5)

    # Select document format
    dropdown_xls = navegador.find_element(By.ID, "MainContent_E
    navegador.execute_script("arguments[0].value = 'XLS - No Fo
    time.sleep(1)

```

```
    # Download the report
    export_button = navegador.find_element(By.ID, 'MainContent_
    navegador.execute_script("arguments[0].click();", export_bu
    time.sleep(4)

    print(f"\n -Raw data of {final_site} donwloaded.")

    # Create a list of file on download folder
    files = os.listdir(downloads_folder)

    # Reads/Create a list of files in download folder starting
    filtered_files = [file for file in files if file.startswith

    # Save the file name in a variable file
    for file in filtered_files:
        # Starting the path file name in downloads
        source_file = os.path.join(downloads_folder, file)
        # Create a variable destination with the new path
        destination_file = os.path.join(destination_folder, fil
        # Move files from downaloads to file_raw
        shutil.move(source_file, destination_file)

        # Open raw report ignoring first row
        open_raw_data = pd.read_excel(destination_file, sheet_n

        # Delete the last 2 rows
        open_raw_data = open_raw_data.iloc[:-2]

        # Delete columns unnecessaries
        open_raw_data.drop(columns=open_raw_data.columns[11:17]

        # Determine the number of rows in the DataFrame
        num_rows = len(open_raw_data)

        # Fill last column with the site name
        open_raw_data['Site_Name'] = [final_site] * num_rows

        # Reset the index
        open_raw_data = open_raw_data.reset_index(drop=True)

        # Create a new path where it will save the formatted fi
        save_path = f'/Users/hugsneyf/Downloads/GroupNexus/pyth

        # Save the file
        open_raw_data.to_excel(save_path, index=False)

# Location with formatted files
formatted_folder = "/Users/hugsneyf/Downloads/GroupNexus/python

# Initialize a empty data frame to receive all the file combine
combined_data = []

# Loop for all the files in fomatted foler
for fomatted_file in os.listdir(formatted_folder):
```

```
    # Checking the type of document
    if fomatted_file.endswith('.xls'):
        # Starting the path file name in formatted folder
        file_path = os.path.join(formatted_folder, fomatted_fil
        # Create a variable destination for all the data with t
        open_raw_data = pd.read_excel(file_path)
        # Append all the data in combined empty data frame
        combined_data.append(open_raw_data)

# Finalizing the combination process in a normal spreadsheet
combined_df = pd.concat(combined_data, ignore_index=True)

# Save the combined DataFrame to a new Excel file
combined_df.to_excel('Glide_Combined_File.xlsx', index=False)


# Load data from an Excel file using pandas
glide_final = pd.read_excel('Glide_Combined_File.xlsx')

# Remove the newline character from the column name
glide_final.rename(columns={'AMOUNT\nPAID': 'Amount'}, inplace=

# Convert the AMOUNT_PAID column to float
glide_final['Amount'] = glide_final['Amount'].str.replace('£',


# Create a pivot table by Site_Name
pivot_table = glide_final.pivot_table(
    index='Site_Name',
    aggfunc={'VRM': 'count', 'Amount': 'sum'}
)

# Rename the VRM column to quantity
pivot_table = pivot_table.rename(columns={'VRM': 'Quantity'})

# Rearrange the columns in the pivot table
pivot_table = pivot_table[['Quantity', 'Amount']]

# Load the existing workbook
workbook = load_workbook('Glide_Combined_File.xlsx')

# Create a new worksheet for the pivot table
pivot_sheet = workbook.create_sheet(title='Pivot_table')
# Write the pivot table to the "Pivot_table" worksheet
for row, data in enumerate(dataframe_to_rows(pivot_table.reset_
    for col, value in enumerate(data, start=1):
        pivot_sheet.cell(row=row, column=col, value=value)

# Save the updated workbook
workbook.save('Glide_Combined_File.xlsx')

print("All report had been run and saved, please check: Glide_(
```

Download, format, combine and create a pivot table

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```