USING PYTHON TO SOLVE REAL WORLD WORK PROBLEMS

Joao Paulo Campos Maia Pinto

DATA ANALYTICS BOOTCAMP Final Project

EXECUTIVE SUMMARY

The objective of this project is to demonstrate how I am able to apply the advanced techniques and skills learned throughout this course in developing scripts and solutions that help me solve a range of different problems at work.

I am grouping 4 different projects that collect and transform data inputs into deliverables to clients and impact the daily routines, immensely decreasing the time for performing certain tasks and greatly reducing risks of human error when sending information to clients.

It is also worth noting that this allows a better preparation for business growth as increase in volumes traded have marginal impact on the performance of the solutions.

PROJECT SCOPE

NI 24-101 Compliant Contact to Clients

OVERVIEW

1. Project Background and Description

The NI 24-101 is a norm established by the Ontario Securities Commission that regulates institutional trade matching and settlement and demands that participants have their trades matched in CDS by noon (12pm) the day prior to the settlement date. In order to achieve that, the Middle Office must search for all trades that are still unmatched in the morning and reach out to the clients requesting them to advise their custodians to match our instructions in CDS. The idea of the project is to create a python script that will investigate the trades that require action and will automatically group them by client and prepare an email with the trade details and client's email address in order to advise customers in a timely manner. This will reduce the time of execution of this task astonishingly and will diminish risk of not capturing all trades. It should also decrease the number of errors of sending incorrect information or even more important, sending the information to the wrong counterparty.

2. Project Scope

- Imports. Libraries required:
  - pandas
  - win32com.client
  - datetime
  - os.path

- Part 1. Reading and filtering data :
  - Read a csv file that is extracted using the system Arrow with filters that are currently in place.
  - Filter the dataframe's memo1 field to exclude etfs and other trades not covered by this team, removing items that start with "REF#", "REF #", "NO CLIENT", "YIELD" and "ETF TRADE"
  - Filter the dataframe's security field to exclude securities not covered by this team, removing items with code equals to "CA99997Z1099"
  - Filter the dataframe's date field to exclude today's trades
  - Cast column Client Account as string

- Part 2. Merging data with account information:
  - Read csv file generated daily that contains all institutional accounts information
  - Cast column containing account number as string
  - Left merge data with accounts dataframes on account number columns
  - Sort by status, client name and security
  - Move client name to first column
  - Save excel spreadsheet with information

- Part 3. Getting list of clients that need to be contacted:
  - Filter merged DF status column for unmatched trades only
  - Get list of unique clients from DF
  - For loop through client's DF to filter by client, convert table to html and append it a new list
  - Create dataframe with list of unique clients and html table
  - Read csv file containing contact information by client
  - Left merge the two DFs on client name

- Part 4. Preparing emails and finish task:
  - Dispatch outlook application
  - For loop through unique clients merged DF creating new email for each client
  - Dispatch excel with created spreadsheet

3. High-Level Requirements

- Permission to access Arrow
- Permission to access and save network I: drive
- Ability to run python scripts

4. Deliverables

- Email to each client that has unmatched trades
- Spreadsheet with all unmatched and DKed trades

PROJECT SCOPE

ETF Confirmation File Reader

OVERVIEW

1. Project Background and Description

The ETF desk generates an upload file that needs to be imported to the trading system in t+1. But, before uploading it, our team is responsible for comparing their information to what ETF providers have sent to us through excel confirmation files. The message files will be placed in a determined folder and the files should be automatically extracted and saved to a folder with the current date. The main issue with the confirmation files is that there is no standard format in which clients send us this information. The goal of this project is to automate the most as possible the process of gathering the required information from all these files in order to perform the comparison. It will need to find the ETF ticker, quantity, money and type (whether it is a subscription or a redemption).

2. Project Scope

- Imports. Libraries required:
  - win32com.client
  - os
  - shutil
  - pandas
  - numpy
  - datetime from datetime

- Part 1. Extracting Attachments from Messages:
  - Assign path for the folder where the messages are going to be saved to a variable.
  - Loop through files and append file names of files with the extension type ".msg" to a list
  - Create attachments folder with the current date
  - While looping through this list, open file dispatching Outlook in order to assign the item's attachments to a variable
  - Loop through this variable saving the attachments that are an excel spreadsheet or csv file.

- Part 2. Reading through Confirmation Files:
  - Assign path for the folder where the attachments are going to be saved to a variable
  - Loop through files inside that folder and read each one of them according to its file type (csv or excel)
  - Search for recurrent words from old confirmation files that refer to the ETF ticker and append to tickers list
  - Search for recurrent words from old confirmation files that refer to the quantity and append to quantities list
  - Search for recurrent words from old confirmation files that refer to the money and append to money list
  - Search for recurrent words from old confirmation files that refer to the type and append to type list
  - Add all lists to a dataframe
  - Make any necessary transformation
  - Move files from the current folder to the old confirms folder

- Part 3. Reading ETF Uploader file:
  - Read excel ETF uploader spreadsheet of the current date
  - Transform symbol column in order to match with tickers in the confirmation files dataframe
  - Filter for only the adjustment trades

- Part 4. Merging Dataframes:
  - Merge dataframes on the symbol/ticker columns
  - Make additional transformations for positive or negative values
  - Zero the money for trades conditioned to the type of the trade
  - Create column with net quantity
  - Create column with net money
  - Export dataframe to excel
  - Show dataframe

3. High-Level Requirements

- Permission to access and save network G: drive
- Ability to run python scripts

4. Deliverables

- Spreadsheet with comparison between confirmation files and information provided by the trading desk

PROJECT SCOPE

Statements Encryption

OVERVIEW

1. Project Background and Description

It is a norm in the bank that all account statements sent to clients through email must be password protected. Our team is responsible to send monthly statements and additional ones by client request. The files are received in pdf format through email but they are not encrypted. Currently, the members of the team need to open each of these files and save with a standard password that varies from client to client. After that, the files for each account must be aggregated in one email and forwarded to the respective client. The password must also be sent out in a different email. This project aims to automate this process reducing the time taken to just a few seconds.

2. Project Scope

- Imports. Libraries required:
  - os
  - shutil
  - datetime
  - pandas
  - numpy
  - win32com.client
  - PdfFileReader, PdfFileWriter from PyPDF2

- Part 1. Aggregate database files:
  - Read the current date's csv file that contains all client accounts covered by the team.
  - Read email and password excel database file
  - Merge both dataframes on client codename

- Part 2. Extract and Transform PDF files:
  - Loop through files saved in the designated folder and append the PDF type files to a list
  - Loop through the list of PDF files
  - Rename the file so we can use the old file name further ahead
  - Open input PDF file as readable
  - Write new PDF file with input file information to a variable
  - Extract account number from file name
  - Locate the account number in our database dataframe and get the password related to that account
  - Account for error in case account cannot be found in the database
  - Encrypt output with password
  - Account for error in case account doesn't have a password set up
  - Open PDF file with old name as writable and write output file
  - Append information to a new dataframe
  - Delete renamed (not encrypted) statements from folder

- Part 3. Sending Emails:
  - Get unique client names from dataframe
  - Loop through client name list
  - Dispatch outlook application

- Prepare an email with statements
- Prepare an email with the password

3. High-Level Requirements

- Permission to access and save to network G: drive
- Ability to run python scripts

4. Deliverables

- Email to each client with statements for all of their accounts
- Email to each client with the password for opening those statements

PROJECT SCOPE

Allocations Amendments for .U and .V Stocks

OVERVIEW

1. Project Background and Description

The regular case for stocks is that stocks that trade in the Canadian exchange are traded in Canadian dollars and settle at CDS, while stocks that trade in American exchanges are traded in US dollars and settle at DTC. However, there are a number of exceptions for that case, some stocks that trade in the Canadian market, settle at CDS but are traded in US dollars. These stocks can be identified by their symbol that ends in ".U" or ".V". Our system and also some client's systems do not know how to interpret that difference and many times, the allocations for these stocks are assigned to DTC instead to CDS. This creates breaks to the settlement process and is very time consuming for the team to keep checking if a trade was assigned to the wrong account and manually amend it to the correct one. It gets even tougher for the team when the trade is allocated to a large number of accounts, a number that could reach beyond a hundred. This project goal is to create a daily routine where every morning the script will read the previous day's trading records and find all ".U" and ".V" names that were traded that day and will check each of these stocks for any wrong allocations, making the amendments when necessary.

2. Project Scope

- Imports. Libraries required:
  - pyautogui
  - pandas
  - time
  - datetime, timedelta from datetime
  - cv2
  - GetKeyState from win32api
  - VK_CAPITAL from win32con
- Part 1. Get List of Names:
  - Read the previous date's csv file that contains all trades covered by our team.
  - Filter for symbols ending with a ".U" or ".V"
  - List unique names and print it for user visualization
- Part 2. Search for Allocations:
  - Check status of Caps Lock and press key to turn off in case it is turned on
  - Move mouse to Fidessa window and click to select it
  - Navigate through system using keyboard shortcuts
  - Select yesterday's date and press enter to filter trades
  - Loop through symbols list and input each symbol at a time to filter trades
  - Using keyboard shortcuts select all and copy selection
  - Read clipboard into a dataframe
  - Using the dataframe as a guide, use the keyboard to run line by line avoiding specific clients or services that shouldn't be amended
  - If ok to amend, use keyboard to open allocations
- Part 3. Amend Allocations:
  - Move mouse to select allocation window

- Using keyboard shortcuts select all and copy selection
- Read clipboard into a dataframe
- Using the dataframe as a guide, use the keyboard to run line by line searching for allocations to DTC accounts and status different than “ACAN” (which means “cancelled”)
- Print symbol, client and account that is being amended
- Using keyboard shortcuts, amend the account
- Move to next line to continue loop
- Move back to the first window when done
- Print a message warning user that it finished running the script

3. High-Level Requirements

- Permission to access network G: drive
- Access to Fidessa
- Ability to run python scripts

4. Deliverables

- Print with all “.U” and “.V” names traded on the specified date
- Print with all the amendments made