Project: Test Project – Bank Import Files

Summary:

* Vidrio performs cash reconciliations for client portfolios
* Vidrio receives files containing client portfolio bank activity. These files are received daily and can contain more than 1 bank account
* Data in the bank activity files must be extracted and formatted into a new file that can be imported into the Vidrio platform
* The purpose of this code is to create the import files for each bank account.
* Also, the code should keep track of issues and report issues in an exception file.
* Some of the bank accounts may invest in Money Market (MM) instruments. A Money Market instrument is an over night deposit that earns a higher rate of interest than leaving cash in the bank account. The interest is paid into the bank account after the month end. Vidrio clients do not want to see their MM activity in Vidrio so this activity should not be in the import files and should be excluded from all balances in Vidrio. MM activity contains ‘STIF’ in the Description.

Steps to create python script:

1. Create a subfolder: Mapping, in the location for your python code. Save the mapping file provided in the Mapping subfolder. The Mapping file will remain in this location.
2. Create a subfolder: Output, in the location for your python code. The Exception file and import files your code creates will be saved in the Output folder
3. Ask the user the location of the Bank Activity file
4. Load both Mapping file and Bank Activity file using Pandas into individual Dataframes
5. Replace any Nan fields in the Bank Activity file with a blank string (i.e. ‘’)
6. Create a blank DataFrame to hold exceptions
7. Create a Boolean variable to keep track of whether there are any exceptions. Set the initial value to False.
8. In the Bank Activity Dataframe, add the following columns:

* ‘Bank Reference ID’ and set equal to ‘Reference Number’ column from the original file
* ‘Post Date’ and set equal to ‘Cash Post Date’ column from the original file. Be sure to format as Date
* ‘Value Date’ and set equal to ‘Cash Value Date’ column from the original file. Be sure to format as Date
* ‘Amount’ and set equal to ‘Transaction Amount Local’ column from the original file
* ‘Description’ and set equal to the combination of the following columns from the original file: ‘Transaction Description 1’, ‘Transaction Description 2’, ‘Transaction Description 3’, ‘Transaction Description 4’, ‘Transaction Description 5’, ‘Transaction Description 6’, 'Detailed Transaction Type Name', and 'Transaction Type'.
* ‘Bank Account’ and set equal to ‘Cash Account Number’ column from the original file
* ‘Closing\_Balance’ and set equal to ‘Closing Balancer Local’ column from the original file
* ‘Filename’ and set equal ‘Cash Account Number’ + the current date and time + ‘.csv. and format as a string

1. Create a dataframe to hold all the Bank Ref ID from the Mapping file
2. Create a single dataframe to hold all the Bank Ref ID and Starting\_Balance values from the Mapping file
3. Using a for loop, loop through the Bank Ref IDs in step 9. For each Bank Ref ID, do the following:
4. Get the Starting\_Balance value from 10
5. Create an Output Dataframe that contains all the columns from the Bank Activity dataframe where the "Cash Account Number" equals the Bank Ref ID
6. Create a MM dataframe that contains all MM activity in the Output Dataframe for the current Bank Ref ID
7. Remove all MM activity from the Output Dataframe
8. Create a write\_file Dataframe from the Output Dataframe containing all rows and the following columns:

'Bank Reference ID',

'Post Date',

'Value Date',

'Amount',

'Description',

'Bank Account',

'Closing\_Balance'

1. Remove all rows from the write\_file DataFrame is the value is NA in the column: 'Bank Reference ID'
2. If the write\_file DataFrame is empty print the 'Bank Reference ID' + “ has no activity”

Else,

1. Look up the bank\_closing\_balance from the Bank Activity DataFrame for the given Bank Reference ID from the “Closing\_Balance” column.
2. Calculate the overnight MM investment from the MM dataframe.
3. Add the starting balance row to the write\_file DataFrame, populate the columns with the following:

'Bank Reference ID' = "Starting Balance"

'Post Date' = '2020-01-01'

'Value Date' = '2020-01-01'

'Amount' = balance from step 12

'Description' = "Starting Balance"

'Bank Account' = the given Bank Reference ID

'Closing\_Balance' = 0

1. Calculate the closing balance from the write\_file dataFrame by summing the “Amount” column. Save the value in the variable called: calc\_closing\_balance
2. Compare the calc\_closing\_balance to the sum of the overnight MM value (calculated in step 20) and the bank\_closing\_balance (calculated in step 19). If the values are not equal, add a row to the exception DataFrame (from step 6) indicating the Bank Reference ID, the bank\_closing\_balance (from step 19) + MM value (from step 20) and the Calculated Closing Balance (from step 22). Also, set the Boolean variable indicating I there are exceptions (from step 7) to True
3. Save the write\_file DataFrame to the Output subfolder as an excel file with the file name: Bank Reference ID + current date and time +”.xlsx” and a tab name “Bank Transactions”
4. Update the mapping Dataframe for the relevant Bank Reference ID with the calculated closing balance (from step 22)
5. Save the mapping Dataframe to excel, replacing the existing mapping file
6. If there are exceptions recorded in the exception DataFrame, then write the exception DataFrame to excel in the Output subfolder.

Results:

Please save your project to a public repository on Github and send us the URL to review your code.