1 Introducción

Para la prueba se me ha dado una serie de archivos. El primero contiene datos sobre unas facturas simplificadas dadas por la empresa AGIKEY mientras que el segundo contenía las operaciones individuales realizadas, las cuales en su conjunto deben dar lugar a la información mostrada en las facturas simplificadas. Para ello se ha realizado un programa en Python para realizar la conciliación entre los dos conjuntos de datos.

En las facturas de AGIKEY, los datos se muestran por filas, siendo cada factura identificada por su número y se muestran datos referidos a cada una de ellas. Los datos de interés serán los catalogados como "Almacenamiento TVB", emitidos en abril de 2024 y con origen TVB, y de los cuales utilizaremos el número de identificación y el importe de la factura.

Por otro lado, en los datos de las actividades individuales los datos de nuevo se muestran por filas, estando cada actividad identificada por su número. Pero, a diferencia del caso con las facturas, cada actividad está conformada por un conjunto de subflows, aportando cada uno de ellos al importe final de la actividad. Por ello, será necesario tratar de forma distinta estas actividades al no disponer del importe total desde el principio.

2 Descripción del código

Para la realización de la conciliación, se ha realizado un programa de Python basado en el paquete Pandas, ya que facilita la carga de datos desde los diferentes archivos de Excel utilizados y simplifica el tratamiento de los datos a través de los dataframes. A continuación, detallaré cada parte del programa.

El programa comienza cargando los paquetes necesarios para su funcionamiento, siendo estos `Numpy` y `Pandas`.

```
import pandas as pd
import numpy as np
```

A continuación, el programa carga los datos del Excel que contiene los datos de las facturas de AGIkey utilizando la función de pandas `read_excel`. Fue necesario establecer que los datos de la columna `'FechaEstado'` fuesen cargados como cadenas de caracteres para evitar problemas con el formato de las fechas. Se cargan las facturas de interés dentro de la variable `facturas` filtrando los datos, tomando aquellas facturas que tengan `'Origen'` TVB, `'FechaEstado'` sea el 2 de abril de 2024 (al cargar los datos las fechas se pusieron en el formato yyyy-dd-mm) y que

'ServicioFacturado' fuese Almacenamiento TVB. Por último, se suman todos los importes de cada una de las facturas, donde fue necesario reemplazar la coma decimal por un punto para que Python pudiese convertir la cadena de caracteres en un número con formato `float`.

```
#   Carga facturas AGIKEY

factAGI = pd.read_excel('Archivos/AGIkey_facturas.xlsx',dtype={'FechaEstado': str})

facturas = factAGI.loc[(factAGI['Origen'] == 'TVB') &
                       (factAGI['FechaEstado'] == '2024-02-04 00:00:00') &
                       (factAGI['ServicioFacturado']=='Almacenamiento TVB')]


sumAGI = facturas['Importe'].str.replace(',', '.').astype(float).sum()
```

La siguiente fase es la carga de las actividades individuales del documento de Excel donde están almacenadas, de nuevo utilizando la función `read_excel`. Filtramos los datos totales utilizando la `'AggregatedKey'` de la actividad buscada y la almacenamos en la variable `deals`. Además, debemos cargar los datos de los subflows. Estos datos se encuentran en la columna `'Flows'` siguiendo la estructura {`[flow1],[flow2],...`}. Por ello, tomamos los datos de la columna `'Flows'` de las actividades extraídas antes y los convertimos en un array de `Numpy` para introducir los datos en un formato más manejable. Este array contendrá una única componente que consistirá en una cadena de caracteres con el contenido de la columna `'Flows'`. Por ello, utilizamos la función `eval` para convertir la cadena de caracteres en una expresión de Python y de esta forma obtenemos una lista cuyas componentes son los diferentes subflows de la actividad.

```
#   Carga Deals


cargadeals = pd.read_excel('Archivos/JVLNG_CTV_STOK_ABRIL_2024_06_18_13_34.xlsx')


#   Filtro deals

deals = cargadeals.loc[(cargadeals['AggregatedKey'] == '39980344,2024/06/13 16:51:32,STOK')]
subflows = eval(np.array(deals.loc[(deals['AggregatedKey'] == '39980344,2024/06/13
            16:51:32,STOK')]['Flows'])[0])
```

Los datos de los subflows están contenidos en una lista, por lo que debemos transformarlos en un dataframe de `Pandas`. Para ello, utilizamos un bucle `for` que tome cada componente

de la lista, la transforme en un dataframe y lo añada al dataframe `dfsubflow` que comienza vacío. De nuevo, filtramos los datos de los subflows de tal forma que solo usemos los subflows correspondientes al mes de marzo y los guardamos en la variable `dfsubflow` de nuevo, ya que el resto de datos no nos son útiles. A continuación, sumamos las cantidades asociadas a los subflows para encontrar el coste total de la actividad y lo guardamos en la variable `sumsubf`.

```
#   Creación dataframe con subflows

dfsubflow = pd.DataFrame()

for elem in subflows:
    df = pd.DataFrame([elem])
    dfsubflow=pd.concat([dfsubflow,df],ignore_index=True)


dfsubflow = dfsubflow.loc[dfsubflow[dfsubflow['Date'].str.startswith('2024-03')].index]
dfsubflow = dfsubflow.loc[dfsubflow['Amount']!=0]
sumsubf = dfsubflow['Amount'].sum()
```

Por último, el programa crea el archivo con formato `.txt` donde se mostrarán los resultados de la conciliación, llamado `Resultados.txt`. Se ha utilizado una declaración `with` para simplificar el trabajo con el archivo de texto. El programa comienza mostrando las diferentes facturas de AGIkey que contienen los importes a conciliar, ya obtenidos en fases previas. Para ello, se utiliza un bucle `for` que recorra los índices del dataframe `facturas` y escriba en el archivo el número de factura, su importe y su fecha de emisión. A continuación, se escriben las diferentes actividades y sus flows que contribuyen a las facturas de interés, usando el mismo método que con las facturas a través de un bucle `for`. Como último paso, el programa escribe en el archivo los resultados de la conciliación, mostrando el importe total de todas las facturas de AGIkey, el total de todas las actividades y la diferencia entre ellas.

A continuación, se muestra el código utilizado en su totalidad:

```
import pandas as pd
import numpy as np


#   Carga facturas AGIKEY

factAGI = pd.read_excel('Archivos/AGIkey_facturas.xlsx',dtype={'FechaEstado': str})

facturas = factAGI.loc[(factAGI['Origen'] == 'TVB') &
                       (factAGI['FechaEstado'] == '2024-02-04 00:00:00') &
                       (factAGI['ServicioFacturado']=='Almacenamiento TVB')]
```

```
sumAGI = facturas['Importe'].str.replace(',', '.').astype(float).sum()


#   Carga Deals


cargadeals = pd.read_excel('Archivos/JVLNG_CTV_STOK_ABRIL_2024_06_18_13_34.xlsx')


#   Filtro deals

deals = cargadeals.loc[(cargadeals['AggregatedKey'] == '39980344,2024/06/13 16:51:32,STOK')]
subflows = eval(np.array(deals.loc[(deals['AggregatedKey'] == '39980344,2024/06/13
            16:51:32,STOK')]['Flows'])[0])

#   Creación dataframe con subflows

dfsubflow = pd.DataFrame()

for elem in subflows:
    df = pd.DataFrame([elem])
    dfsubflow=pd.concat([dfsubflow,df],ignore_index=True)


#   Cálculo de la conciliación

dfsubflow = dfsubflow.loc[dfsubflow[dfsubflow['Date'].str.startswith('2024-03')].index]
dfsubflow = dfsubflow.loc[dfsubflow['Amount']!=0]
sumsubf = dfsubflow['Amount'].sum()

#   Creación del archivo txt con los resultados

with open('Resultados.txt', 'w',encoding='utf-8') as archivo:
    archivo.write('RESULTADOS DE LA CONCILIACIÓN\n')
    archivo.write('\n')
    archivo.write('  Facturas de AGIKEY:\n')

    for i in facturas.index.tolist():
        archivo.write('      Factura: ' + str(facturas['NumeroFactura'][i]) +
                      ', Importe: ' + str(facturas['Importe'][i]) +
                      ', Fecha de emisión: ' + str(facturas['FechaFactura'][i]) +
                                               '\n')
    archivo.write('\n')
```

```
archivo.write('  Deals:\n')
archivo.write('      Deal: ' + str(deals['AggregatedKey'][26])[0:7] +
              ', Cantidad: ' + str(sumsubf) +
              ', Start delivery date: ' + str(deals['StartDeliveryDate'][26]) +
              ', End delivery date: ' + str(deals['EndDeliveryDate'][26]) +
              ', Procedencia: ' + 'JVLNG_CTV_STOK\n')
for i in dfsubflow.index.tolist():
    archivo.write('          Sublow: ' + str(dfsubflow['Id'][i]) +
                  ', Date: ' + str(dfsubflow['Date'][i]) +
                  ', Subcantidad: ' + str(dfsubflow['Amount'][i]) + '\n')

archivo.write('RESULTADO DE LA CONCILIACIÓN\n')
archivo.write('    Total facturas AGIKEY: ' + str(sumAGI) + '\n')
archivo.write('    Total deals: ' + str(sumsubf) + '\n')
archivo.write('    Total facturas AGIKEY - Total deals = ' + str(sumAGI-sumsubf) +
                                                          '\n')
```

3 Resultados

La conciliación da como resultado que las facturas de AGIKEY contienen 13984.62 € que no aparecen en las actividades indivuales. Es posible que una diferencia tan grande venga del hecho de que los datos de los subflows no estaban completos en el archivo Excel, ya que la celda había alcanzado el máximo de caracteres, dejando flows sin mostrar. Este problema puede evitarse de varias formas, como evitar agrupar tantos subflows en una misma actividad, modificar el documento de Excel de tal forma que cada subflow tenga su celda individual. Otra solución, más radical, pero que podría dar mejores resultados, es utilizar bases de datos de Access en vez de hojas de cálculo, ya que estas permiten más dinamismo a la hora de relacionar datos entre sí.